

**GOVERNMENT OF TAMILNADU
DIRECTORATE OF TECHNICAL EDUCATION
CHENNAI – 600 025**

STATE PROJECT COORDINATION UNIT

Diploma in Computer Engineering

**Course Code: 1052
M – Scheme**

**e-TEXTBOOK
on
MOBILE COMPUTING
for
VI Semester DCOMP**

Convener for COMPUTER ENGINEERING Discipline:

Mrs.A.Ghousia Jabeen,
Principal,
Thanthai Periyar E.V.Ramasamy Govt. Polytechnic College ,
Vellore – 632 002

Team Members for Mobile Computing:

Mr. T.Muthamil Selvam.,
Lecturer/(SG) Computer Engg.
Seshasayee Institute of Technology,
Trichy – 620 010

Ms.C.Rita Roselin,
Lecturer / Computer Engg.
Seshasayee Institute of Technology
Trichy – 620 010

Mr.M.Thirumurugan,
Lecturer/ Computer Engg.
Arasan Ganesan Polytechnic College,
Sivakasi - 626 130

Validated By

Mrs. K.Sivagama Sundari,
Lecturer(Senior Grade) /Computer Engg.
Ayyanadar Janakiammal Polytechnic College,
Sivakasi – 626 189

STATE BOARD OF TECHNICAL EDUCATION & TRAINING,
TAMILNADU.

DIPLOMA IN COMPUTER
ENGINEERING M- SCHEME

(to be implemented to the student Admitted from the Year 2015-2016 on wards)

Course Name : Diploma in Computer Engineering.

Subject Code : 35262

Semester : VI

Subject title : MOBILE COMPUTING

TEACHING & SCHEME OF EXAMINATION:

No. of weeks per Semester 15 Weeks

Subject	Instructions		Examination			Duration
	Hours /	Hours / Semeste	Internal Assessmen	Board Examinatio	Total	
MOBILE COMPUTIN	5	75	25	75	100	3 Hrs

Topics and Allocation of Hours:

Sl.No	Topic	Time (Hrs)
1	Introduction to Mobile Computing , WiFi , Bluetooth	11
2	Introduction to GSM , SMS ,GPRS , Mobile OS	11
3	Introduction to ANDROID	15
4	VIEWs	14
5	Location Based Service and SQLite	14
	TEST AND REVISION	10
	TOTAL	75

Rationale:

Knowing the details of Mobile and their working principle are need of the every common man. Mobile Application development is the very hot business domain. Majority of the corporate have a separate division for the development of mobile applications. It is imperative that students must know the way to apply advanced data communicating methods and networking protocols for wireless and mobile devices.

Students must utilize and employ application frameworks for developing mobile applications including under disconnected and weakly connected environment They should be in a position to

select components and networks for particular application , creatively analyze mobile and wireless networks and critically analyze security issues of mobile and wireless computing systems

Objectives:

- To introduce the characteristics, basic concepts and systems issues in mobile Computing
- To illustrate architecture and protocols in Mobile computing and to identify the trends and latest development of the technologies in the area
- To understand the network protocols governing the mobile communication
- To know the different kinds of mobile OS prevailing in the market
- To know Android OS in detail
- To understand the components of a Mobile App.
- To give practical experience in the area through the development of Mobile apps
- To design successful mobile computing applications and services
- To evaluate critical design tradeoffs associated with different mobile technologies, architectures, interfaces and business models and how they impact the usability, security, privacy and commercial viability of mobile and pervasive computing services and applications
- To know the development of Mobile apps using database

DETAILED SYLLABUS

UNIT -I Introduction to Mobile Computing , WiFi , Bluetooth		
1.1	Introduction : Evolution of Mobile Computing – Important terminologies – Mobile computing functions – Mobile computing Devices – Networks: Wired , Wireless , Adhoc - Comparison of wired and wireless mechanism - Various types of wireless communication technologies used in Mobiles, Antennas	3 Hrs
1.2	Architecture : Architecture of Mobile Computing – 3- Tier Architecture – Presentation(Tier-1), Application (Tier -2), Data (Tier – 3)	4 Hrs
1.3	Mobile computing through Telephony: Evolution through telephony	1 Hrs
1.4	Wireless LAN: Introduction - Applications of WLAN – Infrared versus Radio transmission – Features of WI-FI and WI-MAX – Bluetooth : Introduction and application	3 Hrs

UNIT-II Introduction to GSM , SMS ,GPRS , Mobile OS		
2.1	Global System for Mobile Communication (GSM): Introduction – GSM Architecture – GSM Entities (Basics only) – Introduction to CDMA	2 Hrs
2.2	Short Message Service (SMS): Mobile computing over SMS – Short Message Service – Strength of SMS – SMS Architecture – Value added services through SMS – VAS Examples	3 Hrs
2.3	General Packet Radio Service (GPRS): Introduction – GPRS Packet data Network : Applications for GPRS : Generic Applications, GPRS Specific Applications – Limitations of GPRS – Features of 3G and 4G Data Service	3 Hrs
2.4	Mobile Operating Systems : Evaluation of Mobile Operating System-Handset Manufactures and their Mobile OS- Mobile OS and their features. Linux Kernel based Mobile OS	3 Hrs
UNIT-III Introduction to ANDROID		
3.1	ANDROID : Android Versions – Features of Android – Architecture of Android –Android Market – Android Runtime (Dalvik Virtual Machine)	4 Hrs
3.2	ANDROID SDK & ADT : Android SDK – Android Development Tool (ADT) – Installing and configuring Android – Android Virtual Device (AVD)	3 Hrs
3.3	ACTIVITIES & INTENTS : Understanding Activities – Linking activities and intents –Calling built-in applications using intents – Fragments Displaying Notifications	4 Hrs
3.4	User Interface : Views and Viewgroups – Layouts – Display Orientation – Action Bar – Listening for UI Notifications	4 Hrs
UNIT-IV VIEWS		
4.1	Basic Views : Textview, Button, Image Button, EditText, CheckBox, ToggleButton, RadioButton and RadioGroup Views, ProgressBar View, Auto Complete Text View	4 Hrs
4.2	Advanced Views : Time Picker View and Date Picker View – List Views – Image View – Menus – Analog and Digital View – Dialog Boxes	4 Hrs
4.3	Displaying Pictures & Menus with Views: Image View – Gallery View – ImageSwitcher – GridView - Creating the Helper Methods – Options Menu – Context Menu	4 Hrs
4.4	SMS, Phone: Sending SMS – Receiving SMS – Making phone call	2 Hrs
UNIT V Location Based Service and SQLite		
5.1	Location Based Services : Obtaining the Maps API Key- Displaying the Map – Zoom Control – Navigating to a specific location – Adding Marker – Geo Coding and reverse Geo coding	5 Hrs

5.2	Content Provider : Sharing data – view contacts – Add contacts – Modify contacts –Delete Contacts	3 Hrs
5.3	Storage : Store and Retire data's in Internal and External Storage – SQLite - Creating and using databases	2 Hrs
5.4	Android Service : Consuming Web service using HTTP , downloading binary Data –Downloading Text Content – Accessing Web Service	4 Hrs

TEXT BOOK

Sl.No.	Title	Author	Publisher
1.	Beginning Android 4 Application Development	Wei-Meng Lee	Wiley India Edition
2.	Android Apps for Absolute Beginners	Jackson	Apress
3	Mobile Computing	Computing Asoke K Talukder, Hasan Ahmed, Roopa R Yavagal	TMGH
4	Mobile communications	Jochen schiller	Pearson Education,

CONTENTS

Unit No.	Unit Title	Page No.
I	Introduction to Mobile Computing , WiFi , Bluetooth	7-20
II	Introduction to GSM , SMS ,GPRS , Mobile OS	21-37
III	Introduction to ANDROID	38-103
IV	VIEWs	104-168
V	Location Based Service and SQLite	169-225

UNIT - I : Introduction to Mobile Computing , WiFi , Bluetooth

Objectives

At the end of the unit, students can

- Define what is cell, base station, hand-off, cell sector, cell split
- Explain the concept of How Mobile Computing
- Define the mobile computing functions and devices
- Define the various types of networks used in mobile computing
- Explain the general architecture of mobile computing
- Explain the complete concepts behind telephony system
- Define the usage of WLAN
- Define Wi-fi and configure wi-fi.
- Define Bluetooth technology and differentiate it with wi-fi

1.1 INTRODUCTION TO MOBILE COMPUTING

1.1.1 Evolution of Mobile Computing

With the handheld device, conversation between two users at different places is called mobile communication. Firstly mobile communication deals with voice based conversation. Now, both audio and video data can be transmitted is possible. People more likable one of the mobile communication device is Cellular phone. Because, recent cellular technology now comes with more features such as E-mail, Internet, SMS, MMs, GPS, Games etc.

History of Cellular Mobile Communication

During the travelling time, the communication between the people is more difficult. At the beginning, the Light source is used to make communication with using of mirrors. When light is ON referred as Modulated and OFF is referred as Demodulated.

Before Radio Transmission, Lights and Flags are used as the important Wireless Communication in Navy. In 1897, Guglielmo Marconi invented the wireless telegraph communication by encoding alphanumeric characters in analog.

In 1946, the first Public Mobile Telephone System was introduced with analog frequency modulation to cover 50 miles radius. It is not fully suited because it used only 120k spectrum for voice communication.

In 1960, with the Full Duplex services improved Mobile Telephone System was developed. Late 1960, Bell Lab was introduced the Cellular with *12 channels which served 534 subscriber*. In 1983, the Advanced Mobile Phone Systems (AMPS) functioned with 666 channels. This is the First U.S Cellular System and more flexible. Because, it works with the Frequency Modulation and Frequency Division Multiple Access.

Later in the year 1991, the Digital Cellular services were introduced with more applications.

1.1.2 IMPORTANT TERMINOLOGIES

Cell

- It is the smallest Geographical Area for cellular mobile communication radio coverage.
- The shape of the cell is Hexagon.
- Each cell has a Base Station.

Base Station

- Base Station provides function between mobile unit and message switching center. Base station consists of antenna, control unit and a transceiver. It is located in each cell.

Mobile Unit

- It is a hand held device that communicates with the base station
- A mobile unit is uniquely identified by International Mobile Equipment Identity (IMEI)
- An individual subscriber can be identified by International Mobile Subscriber Identity (IMSI)

Cell Splitting

- In high cellular traffic regions, the larger cells are sub-divided into smaller cells for complete radio coverage.

Hand-off

- When a mobile unit moves from one cell to another cell, the call in progress will be handed over from one base station to another without disturbing the call in progress.

Control Channel

- It is used for necessary exchange of information which is related to settings regarding establishment of cell Base Station and the Mobile Unit.

Cell Sectoring

- A single cell can be divided into multiple sectors, where the directional antenna should focus on each sector.

Traffic Channel

- It is used for carrying data or voice connections between different users.

Fading

- When one signal path *loses its strength*, it is called Fading.

Forward Channel

- The transmission of information *from base station to mobile unit* is called Forward Channel.

Reverse Channel

- The transmission of information from mobile unit to base station is called Reverse Channel.

Transceiver

- It is a device used for both transmitted and receiving radio signals.

Subscriber

- Those who pays some amount as a subscription charges for using Mobile Communication system

Mobile Station

- It is present in the hand held personal unit.
- It is mainly intended its service while in movement at any location.

Full Duplex System

- The communication system which allows two way communications simultaneously. Transmissions and Reception is done on two different channels.
- Example : Telephone or Mobile Phone

Half Duplex System

- The communication system which allows two way communications at the same channel. Either Transmission or Reception can be done at a time.
- Wireless Chord

Simplex System

- The communication system which allows only one way communication
- Example : Walkie Talkie

Roamer

- If mobile station operated its own services from an area other than the subscribed area.

1.1.3 MOBILE COMPUTING FUNCTIONS

The following are the few characteristics supported by a mobile computing system. They are

User Mobility

The user should be able to use the same service when he moves from one place to another. The service may be remote or home network.

Bearer Mobility

The user should be capable to use to some service even if he moves from one bearer to another. These bearer are refers to some Service Station.

Host Mobility

The user device can be either a Server or a Client. In such cases, the IP should be given care if it is on move. The service can be accessible from different locations over different networks with the same device.

Service Mobility

Though the user changes from one service to another the service should remain enabled.

1.1.4 MOBILE COMPUTING DEVICES

The following are some of the Mobile Computing Devices available in the market.

Personal Digital Assistant (PDA)

- It is a hand held device
- It allows us to carry much of the digital information.

Tablet PCs

- Tablet PCs are larger and more powerful than PDA's. It includes specialized handwriting recognition software.

Notebook Computer

- It is expensive and powerful tool for carrying electronic information.

Smart Phones

- It is the combination of various portable devices such as PMPs, PDAs and Cellular Phones
- It tries to give the single solution for the portable computing needs.

Portable Media Players

- It allows us to carry music files and other digital files.

Pagers

- It is an inexpensive
- It provides with limited communication capabilities.

Cell Phones

- It provides services like the traditional phone services.
- Some of the facilities of cell phones are Calendars, Contact, Database, e-mail, etc.,

1.1.5 Networks

The following are the **network structure** available in the Mobile Computing. They are

Wireline Network

- A **LAN technology** with the special types of **cable** is used to transfer data from one place to another.
- This network transferred the signals between the system in the form of analog and digital.
- Cables carry radio between signals of different frequencies to transfer
- Due to high installation charges, this type network is very costly.
- Also, Configuration of this network is more difficult
- This is called Wireline Network

Wireless Network

- Computer is connected to one network technology without wires.
- It is used for communication and data transmission with the help of different types of waves
- It reduces the installation time and cable costs.

Ad Hoc Network

- With the help of network interfaces, two computers link directly.
- We can add more number of computers by link up through a central network Switch or Hub

1.1.6 COMPARISON OF WIRED AND WIRELESS MECHANISM

	Wired	Wireless
Installation	Difficult	Easier
Cost	High	Less
Reliability	High	Reasonably High
Performance	Very Good	Good
Security	Reasonably Good	Reasonably Good
Mobility	Limited	Unlimited
Bandwidth	High	Low
Machines	High Power	Low Power
Delay	Low	High
Operation	Connected	Disconnected

1.1.7 VARIOUS TYPES OF WIRELESS COMMUNICATION TECHNOLOGIES USED IN MOBILES, ANTENNAS

Wireless Communication Technologies are used for the following

- Perform the operation Faster and Efficient
- It raises the standard of human lives.

The different types of Wireless Communication Technologies are;

Bluetooth

- No Wires involved
- Enable the links between Mobile Computers, Mobile Phones, Portable Handheld devices and Internet Connectivity.
- It is mostly used for devices such as Cellphones, Wireless Headsets, Printers, Mouse, Keyboards, etc.,

IrDA

- The full form is Infrared data Association
- Its operation speed is 115200 bps
- It uses a series of infrared data pulses to transfer data from one device to another.
- Example : Television Remote Control

Home RF

- It is cheaper than other technology
- This technology lacks an access point
- This is used to establish wireless technology only at home
- Some of the devices at home are : Home appliances, Phones, Gaming

Wi-Fi

- The full form is Wireless Fidelity.
- Accessible Range is upto 50 meters
- Data transfer rate is 7 MBPS
- 10 times higher than Bluetooth technology
- User no need to by other access point equipments, when use Wi-Fi
- It is used in Colleges, Offices, Conference Hall, etc.,

1.2 Architecture of Mobile Computing

The diagrammatic representation of mobile computing architecture (Fig 1.1) is as follows :

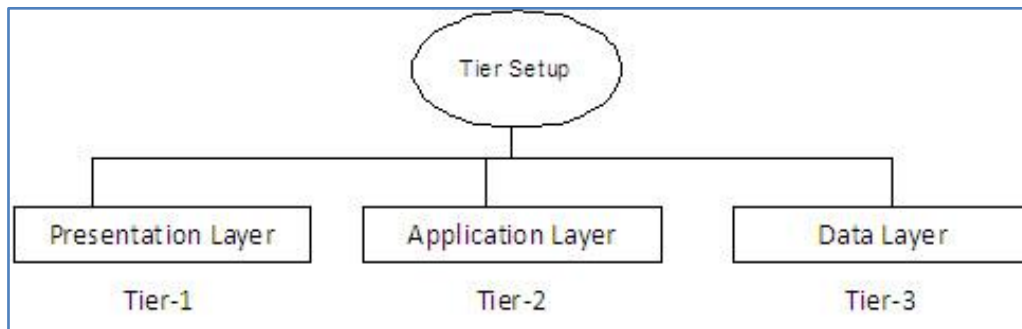


Fig. 1.1

This architecture is made up of Three-Tier. They are

Presentation (Tier-1)

A thin client based on hand held device or PC where the user enters action takes place. Web browsers and customized client programs are included in this layer.

Application (Tier-2)

- This layer plays an important role in wireless LAN applications.
- It acts as interface between the Presentation Layer and the Data Layer.
- It processes the user input, obtains the information and makes the decisions.
- It uses the technologies like .NET, JAVA, ZEND.

Data (Tier-3)

- Data layer includes the database systems in which the processed data can be stored and retrieved.

1.3 Evolution through telephony

Mobile telephony is the provision of telephone services to phones which may move around freely rather than stay fixed in one location. Mobile phones connect to a terrestrial cellular network of base stations (cell sites), whereas satellite phones connect to orbiting satellites. Both networks are interconnected to the public switched telephone network (PSTN) to allow any phone in the world to be dialed.

Public mobile phone systems were first introduced in the years after the Second World War and made use of technology developed before and during the conflict. The first system opened in St Louis, Missouri, USA in 1946 whilst other countries followed in the succeeding decades. The UK introduced its 'System 1' manual radiotelephone service as the South Lancashire Radiophone Service in 1958.

In January 1985 of two cellular systems - the British Telecom/Securicor 'Cellnet' service and the Racal/Millicom/Barclays 'Vodafone' (from voice + data + phone) service. These cellular systems were based on US Advanced Mobile Phone Service (AMPS) technology, the modified technology being named Total Access Communication System (TACS).

In 1947 Bell Labs was the first to propose a cellular radio telephone network. The primary innovation was the development of a network of small overlapping cell sites supported by a call switching infrastructure that tracks users as they move through a network and passes their calls from one site to another without dropping the connection.

In 1956 the MTA system was launched in Sweden. The early efforts to develop mobile telephony faced two significant challenges: allowing a great number of callers to use the comparatively few available frequencies simultaneously and allowing users to seamlessly move from one area to another without having their calls dropped. Both problems were solved by Bell Labs employee Amos Joel who, in 1970 applied for a patent for a mobile communications system.

Bell Labs went on to install the first trial cellular network in Chicago in 1978. This trial system was licensed by the FCC to ATT for commercial use in 1982 and, as part of the divestiture arrangements for the breakup of ATT, the AMPS technology was distributed to local telcos. The first commercial system opened in Chicago in October 1983. A system designed by Motorola also operated in the Washington D.C./Baltimore area from summer 1982 and became a full public service later the following year. Japan's first commercial radiotelephony service was launched by NTT in 1978.

The first fully automatic first generation cellular system was the Nordic Mobile Telephone (NMT) system, simultaneously launched in 1981 in Denmark, Finland, Norway and Sweden.] NMT was the first mobile phone network featuring international roaming.

1.4 Wireless LAN

1.4.1 Introduction

A network that is formed without the use of Cables or Wires is called Wireless LAN. It is also referred as Wi-Fi Network or WLAN A Wireless LAN is a flexible data communication system implemented as an extension to, or as an alternative for a Wired LAN within a building or campus. Using Electromagnetic Waves, WLANs transmit and receive data over the air. It minimizes the need of Wires connections. Thus, WLAN combines data connectivity with user mobility and through, simplified configuration, enable movable LANs.

1.4.2 Applications of Wireless LAN

Wireless LANs have a great deal of applications. Modern implementations of WLANs range from small in-home networks to large, campus-sized ones to completely mobile networks on airplanes and trains.

Users can access the Internet from WLAN hotspots in restaurants, hotels, and now with portable devices that connect to 3G or 4G networks. Oftentimes these types of public access points require no registration or password to join the network. Others can be accessed once registration has occurred and/or a fee is paid.

Existing Wireless LAN infrastructures can also be used to work as indoor positioning systems with no modification to the existing hardware.

1.4.3 Infra Red Vs Radio Transmission

The following two different transmission technologies can be used to set up WLANs

Infra Red Light

- This technology uses diffuse, light reflected at Walls (Or) Directed Light if a Line of Sight (LoS) exists between the sender and receiver.
- Senders can be simple Light Emitting Diodes (LEDs) or Laser Diodes
- Photo Diodes act as receivers

Advantages

- Simple and Cheap
- Available in many mobile Devices (Example : Laptops, Mobile Phones, etc.,)
- No License Needed
- Shielding is simple

Disadvantages

- Low Bandwidth
- In this, **Direct Connection** is gives only the good transaction quality and high data rates.

Radio Transmission

Radio Transmission are used for data transmission with the specifications of GSM at 900, 1800 and 1900 MHz and using the license free ISM band at 2.5GHz.

Advantages

- Long term experiences made with radio transmission for Wide Area Networks. Microwaves links and Mobile Cellular Phones are the examples of Radio Transmission
- Coverage of larger areas and penetrate walls, furniture, etc.,

Disadvantages

- Shielding is not simple
- Very limited ranges of license free bands are available.

1.4.4 Features of WI-FI and WI-MAX

Introduction

Wi-Fi or WiFi is a technology for wireless local area networking with devices based on the IEEE 802.11 standards. Wi-Fi most commonly uses the 2.4 gigahertz (12 cm) UHF and 5 gigahertz (6 cm) SHF ISM radio bands.

Wi-Fi Features

Convenient and everywhere

WiFi is a convenient technology and where the range station exists you are online during travel you can equip with a Wi-Fi network and set up shop anywhere. We will automatically connect to the internet if you are near hotspot. These days WiFi exist everywhere with all its wonders.

Faster

With WiFi, we can get a high speed of internet because it is very fast than DSL and Cable connection you can establish a Wifi network in small space now you don't need any professional installation just connect to a power outlet with an Ethernet cord, and start browsing.

Secure

WiFi security system for Threats makes it more renewable and its tool protect your VPN and secure web page. We can easily configure the device to take better performance. The standard devices, embedded systems, and network security make it more powerful.

No Limitation

We can use a "WiFi" network with no limitation because it can connect you worldwide. Also, we can easily reach to your requirements with WiFi networking applications because the power consumption is very high as compared to another bandwidth.

High data transmission

Its range is 50 meters with 7 mbps speed for transfer of information. Its speed is 10 times faster than other technologies.

WI-MAX Features

Introduction

Wi-MAX (Worldwide Interoperability for Microwave Access) is a family of wireless communication standards based on the IEEE 802.16 set of standards, which provide multiple physical layer (PHY) and Media Access Control (MAC) options. Wi-MAX was initially designed to provide 30 to 40 megabit-per-second data rates, with the 2011 update providing up to 1 Gbit/s for fixed stations.

The following are the some of its significant features

Long Range

Connectivity from long range is much highlighted feature of WiMax. Theoretically it is believed that WiMax technology provides range up 30 miles (50 Km). However in practical experience it is observed that it provides range up to 10 km which is still huge break through in long range connectivity. Users can connect to WiMax base station using SU1 from their homes.

Mobility

Another significant feature of WiMax technology is the connectivity on move. This technology amaze the world hence increasing the use of WiMax in some parts of the world dramatically. Especially to the urban places where cables and other network equipments are not feasible to install. Using its long range feature it provides connectivity on such area where other means of high speed Internet are not available.

The other major mobility feature of WiMax technology is that connecting devices (Laptops, PDA etc) can move in certain area and still be connected to the network.

Interfacing

WiMax radio signals are broadcasted to its subscribers from base stations. Base stations are area where all necessary hardware is installed to provide WiMax with in its range. Since one Base station covers limited range so connecting multiple base stations to each other increase that much range for connectivity. Connecting multiple base stations to each other is not big task and can be perform in few hours.

Accessibility

Accessing WiMax base station is not a difficult task, its long range capability provides connectivity at home from base stations within the range. To get high speed internet it's only required to be because subscriber to WiMax service providers. They will provide you appropriate hardware to become of the subscriber.

Ease of Installation

Installing the hardware is very easy; it is just plug and play. Hardware mostly connects through USB ports or Ethernet and connection can be made by clicking just a button.

1.4.5 Bluetooth

Introduction

Bluetooth is a wireless technology standard for exchanging data over short distances from fixed and mobile devices and building personal area networks (PANs). Bluetooth is managed by the Bluetooth Special Interest Group (SIG), which has more than 30,000 member companies in the areas of telecommunication, computing, networking, and consumer electronics.

The Bluetooth SIG includes promoter member companies Microsoft, Ericsson, IBM, Intel, Agere, Motorola, Nokia and Toshiba, Plus thousands of Associate and Adopter member companies.

Bluetooth Applications

Bluetooth is designed to operate in an environment of many users. Up to eight devices can communicate in a small network called a *piconet*. Ten of these piconets can coexist in the same coverage range of the Bluetooth radio. To provide security, each link is encoded and protected against eavesdropping and interference.

Bluetooth provides support for three general application areas using short-range wireless connectivity:

Data and voice access points

Bluetooth facilitates real-time voice and data transmissions by providing effortless wireless connection of portable and stationary communications devices.

Cable replacement

Bluetooth eliminates the need for numerous, often proprietary cable attachments for connection of practically any kind of communications device. Connections are instant and are maintained even when devices are not within line of sight. The range of each radio is approximately 10 m, but can be extended to 100 m with an optional amplifier.

Ad hoc networking

A device equipped with a Bluetooth radio can establish instant connection to another Bluetooth radio as soon as it comes into range.

Summary

- ✓ Firstly mobile communication deals with voice based conversation.
- ✓ At the beginning, the Light source is used to make communication with using of mirrors.
- ✓ In 1960, with the Full Duplex services improved Mobile Telephone System was developed.
- ✓ Late 1960, Bell Lab was introduced the Cellular with *12 channels which served 534 subscriber.*
- ✓ In 1983, the Advanced Mobile Phone Systems (AMPS) functioned with 666 channels.
- ✓ Some of the mobile computing functions are User Mobility, Bearer Mobility, Host Mobility and Service Mobility
- ✓ Some of the mobile computing devices are PDA, Tablet PC, Smart Phone, Cell Phones, etc.,
- ✓ In mobile computing, the important network structures are Wireline, Wireless and Adhoc
- ✓ Bluetooth, IrDA, Home RF and Wi-Fi are the important wireless technologies used in mobiles and antennas
- ✓ Mobile computing architecture is made up of 3-tier. They are Presentation tier, Application tier and Data tier
- ✓ There are two different transmission technologies can be used to set up WLANs. They are Infrared Transmission and Radio Transmission
- ✓ WiFi is a convenient technology and where the range station exists that are online during travel we can equip with a Wi-Fi network and set up anyplace.
- ✓ Connectivity from long range is much highlighted feature of WiMax. Practically, it supported up to 10 km.
- ✓ Bluetooth is designed to operate in an environment of many users. Up to eight devices can communicate in a small network.

Review Questions

Part-A

1. What is Mobile Communication?
2. Define : Cell
3. Define : Cell Splitting
4. Define : Cell Sectoring
5. Define : Handsoff
6. What is a Fading?
7. List out the mobile computing functions.
8. What are the mobile computing devices?
9. What are Wireless devices?
10. Define : Bluetooth

Part-B

1. What is a Wired Network?
2. What is a Wireless Network?
3. What is Adhoc Network?
4. Give the applications of WLAN.
5. Give the features of Wi-Fi
6. Give the features of Wi-Max
7. Give the applications of Bluetooth

Part-C

1. Give the introduction and history about Mobile Communication
2. List and explain the important terminologies of Mobile Communication.
3. What are Mobile Computing Functions? Explain
4. Discuss about the various Mobile Computing Devices.
5. Describe the various networks structure in Mobile Computing.
6. Give the comparison between Wired and Wireless Mechanisms
7. List out the various types of Wireless Communication Technologies used in Mobiles, Antennas
8. With a neat diagram, explain the Architecture of Mobile Computing
9. Discuss about the Mobile Computing through Telephony
10. Give the applications of WLAN.

11. Give the comparisons of Infrared and Radio transmission
12. List out the features of Wi-Fi
13. List out the features of WI-MAX
14. Give the applications of Bluetooth.

UNIT - II : Introduction to GSM , SMS ,GPRS , Mobile OS

OBJECTIVES

At the end of the unit, students can

- Explain the internal operations of GSM with their usage
- Explain the CDMA benefits and differentiate with TDMA
- Explain the working principles of SMS, strength and its usages
- Define some of the Value Added Services through SMS
- Explain the data transmission through GPRS
- Define the applications of GPRS with its limitations
- Explain the features of 3G and 4G data services
- Describe the evolution of Mobile Operating Systems
- List the Mobile Operating Systems presently available in the market with their use
- Differentiate the Linux Kernel Based Mobile Operating System and its facilities

2 GLOBAL SYSTEMS FOR MOBILE COMMUNICATION (GSM)

2.1.1 Introduction

GSM stands for Global System for Mobile Communication. GSM is the most widely accepted standard in telecommunications and it is implemented globally. It is a digital cellular technology used for transmitting mobile voice and data services.

GSM is the name of a standardization group established in 1982 to create a common European mobile telephone standard. GSM operates in the bands 850 MHz and 1900 MHz. GSM makes use of narrowband Time Division Multiple Access (TDMA) technique for transmitting signals. GSM was developed using digital technology. It has an ability to carry 64 kbps to 120 Mbps of data rates.

2.1.2 GSM Architecture

The GSM network architecture consists of three major subsystems. They are;

- Mobile Station
- Base Station Subsystem
- Network Subsystem

The overall architecture of GSM (Fig. 2.1) is

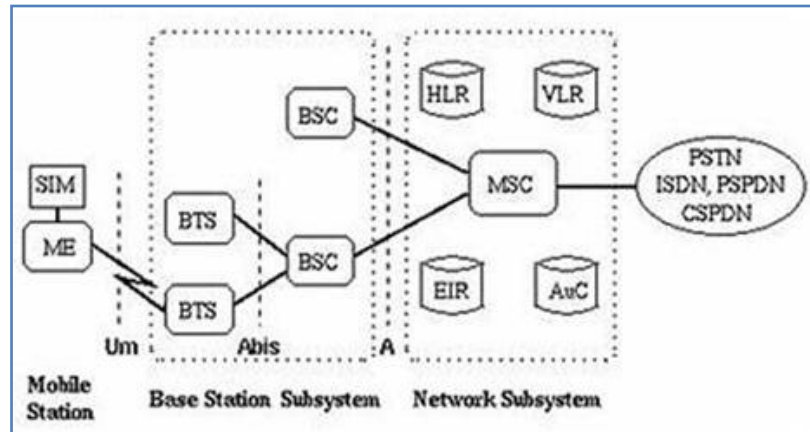


Fig. 2.1

1. Mobile Station (MS):

A mobile station communicates across the air interface with a base station transceiver in the same cell in which the mobile subscriber unit is located. The MS has two elements. The Mobile Equipment (ME) refers to the physical device, which comprises of transceiver, digital signal processors, and the antenna. The second element of the MS is the GSM is the Subscriber Identity Module (SIM). The SIM card is unique to the GSM system. It has a memory of 32 KB.

2. Base Station Subsystem (BSS):

A base station subsystem consists of a base station controller and one or more base transceiver station. Each Base Transceiver Station defines a single cell. A cell can have a radius of between 100m to 35km, depending on the environment. A Base Station Controller may be connected with a BTS. It may control multiple BTS units and hence multiple cells. There are two main architectural elements in the BSS – the Base Transceiver Subsystem (BTS) and the Base Station Controller (BSC). The interface that connects a BTS to a BSC is called the A-bis interface. The interface between the BSC and the MSC is called the A interface, which is standardized within GSM.

3. Network and switching subsystem (NSS)

The NSS is responsible for the network operation. The NSS has one hardware, Mobile switching center and four software database elements. It provides the link between the cellular network and the Public switched telecommunications Networks (PSTN or ISDN or Data Networks). The NSS controls handoffs between cells in different BSSs, authenticates user and validates their accounts, and includes functions for enabling worldwide roaming of mobile subscribers. In particular the switching subsystem consists of:

- **Mobile switch center (MSC)**

It performs the switching function of the system by controlling calls to and from other telephone and data systems. It includes functions such as network interfacing and common channel signalling.

- **Home location register (HLR)**

The HLR is database software that handles the management of the mobile subscriber account. The HLR is the reference database that permanently stores data related to subscribers, including subscriber's service profile, location information, and activity status.

- **Visitor location Register (VLR)**

The VLR is temporary database software similar to the HLR identifying the mobile subscribers visiting inside the coverage area of an MSC. The VLR assigns a Temporary mobile subscriber Identity (TMSI) that is used to avoid using IMSI on the air. The visitor location register maintains information about mobile subscriber that is currently physically in the range covered by the switching center.

- **Authentications center (Auc)**

The AuC database holds different algorithms that are used for authentication and encryptions of the mobile subscribers that verify the mobile user's identity and ensure the confidentiality of each call. The AuC holds the authentication and encryption keys for all the subscribers in both the home and visitor location register.

- **Equipment Identity Register (EIR)**

The EIR is another database that keeps the information about the identity of mobile equipment such the International mobile Equipment Identity (IMEI) that reveals the details about the manufacturer, country of production, and device type. This information is used to prevent calls from being misused, to prevent unauthorised or defective MSs, to report stolen mobile phones

- **Interworking Functions (IWF)**

It is a system in the PLMN that allows for non speech communication between the GSM and the other networks. The tasks of an IWF are particularly to adapt transmission parameters and protocol conversions.

2.1.3 Introduction to CDMA

Code Division Multiple Access (CDMA) is a digital cellular technology used for mobile communication. CDMA is the base on which access methods such as cdmaOne, CDMA-2000, and WCDMA are built. The CDMA is a digital modulation and radio access system that acts as a signature codes to arrange simultaneous and continuous access to a radio network by multiple users. CDMA cellular systems are deemed superior to FDMA and TDMA, which is why CDMA plays a critical role in building efficient, robust, and secure radio communication systems.

CDMA is altering the face of cellular and PCS communication by:

- Dramatically improving the telephone traffic capacity
- Dramatically improving the voice quality and eliminating the audible effects of multipath fading
- Reducing the incidence of dropped calls due to handoff failures

- Providing reliable transport mechanism for data communications, such as facsimile and internet traffic
- Reducing the number of sites needed to support any given amount of traffic
- Simplifying site selection
- Reducing deployment and operating costs because fewer cell sites are needed
- Reducing average transmitted power
- Reducing interference to other electronic devices
- Reducing potential health risks

2.2 Short Message Service (SMS)

2.2.1 Mobile computing over SMS

GSM supports data access over Circuit Switched Data. GSM is digitized but not packetized. In case of CSD, a circuit established and the user is charged base on the time the circuit is active and not on the number of packets transacted. GPRS also known as 2.5G, which is the next phase within the evolution of GSM, supports data over packets. WAP is a data service supported by GPRS and GSM to access Internet and remote data services. A unique data service of GSM is the Short Message Service (SMS). SMS enables sending and receiving text messages to and from, GSM mobile phones.

2.2.2 Short Message Service

Today, SMS is the most popular data bearer / service with in GSM with an average of one billion SMS messages transacted every day around the world. Short Message Service is a mechanism of delivery of short messages over the mobile networks. It is a store and forward way of transmitting messages to and from mobiles. The message (text only) from the sending mobile is stored in a Central Short Message Centre, which then forwards it to the destination mobile. This means that in the case that the recipient is not available the short message is stored and can be sent later.

Each short message is up to 160 characters. These characters can be text or binary Non-Text Short messages. SMS used signaling channels as opposed to dedicated channels, so that these messages can be sent / received simultaneously with the voice / data / fax service over a GSM network.

SMS supports national and international roaming. This means that you can send short messages to any other GSM mobile user around the world.

2.2.3 Strength of SMS

The following are the Strength of SMS

1. Always Connected

As SMS uses the SS7 signaling channel for its data traffic, the bearer media is always on. Users cannot switch OFF, BAR or DIVERT any SMS message. SMS message is delivered to the Mobile Station without any interruption to the ongoing call.

2. Self Configurable and last mile problem resistant

SMS is self configurable and subscriber is always connected to the SMS bearer irrespective of the home and visiting network configuration.

3. Asynchronous

SMS is completely an Asynchronous. In case of SMS, even if the recipient is out of service the transmission will not be abandoned and the particular SMS will be placed in the message queue.

4. Stateless

SMS is session-less and stateless as every SMS message is unidirectional and independent of any context. This makes SMS the best bearer for notifications, alerts and paging.

5. Omnibus nature of SMS

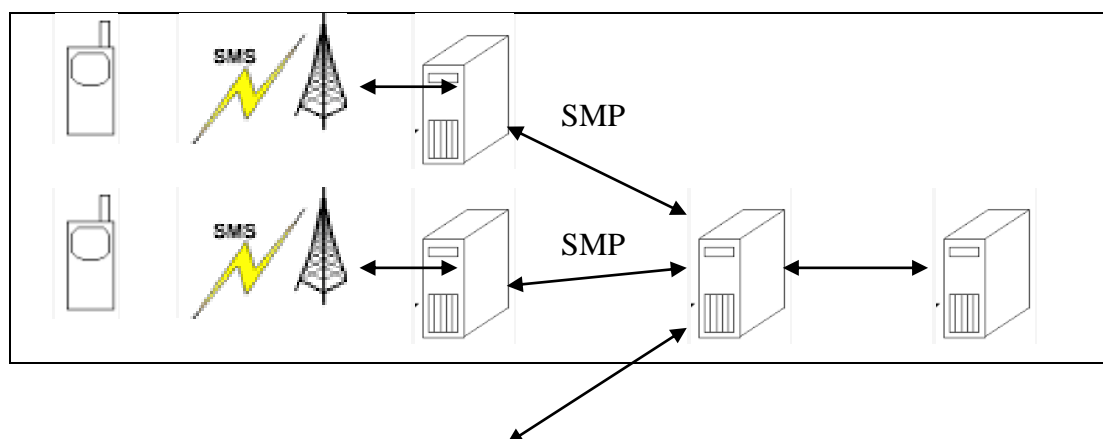
SMS uses SS7 signaling channel which is available throughout the world.

6. Non-repudiable

SMS message carries the Service Center and the source MSISDN as a part of the message header through which any SMS can prove beyond doubt its origin.

2.2.4 SMS Architecture

At the beginning, if a user sends a SMS, the SMS first deliver from the MS which is know as Mobile Station A to **SM-SC (Short Message Service Center)** Via the Base Station System (BSS), and then it catch up to the Mobile Station center(MSC) and finally combine with Interworking MSC (IW-MSC). The following Figure (Fig. 2.2) explain the architecture of SMS.



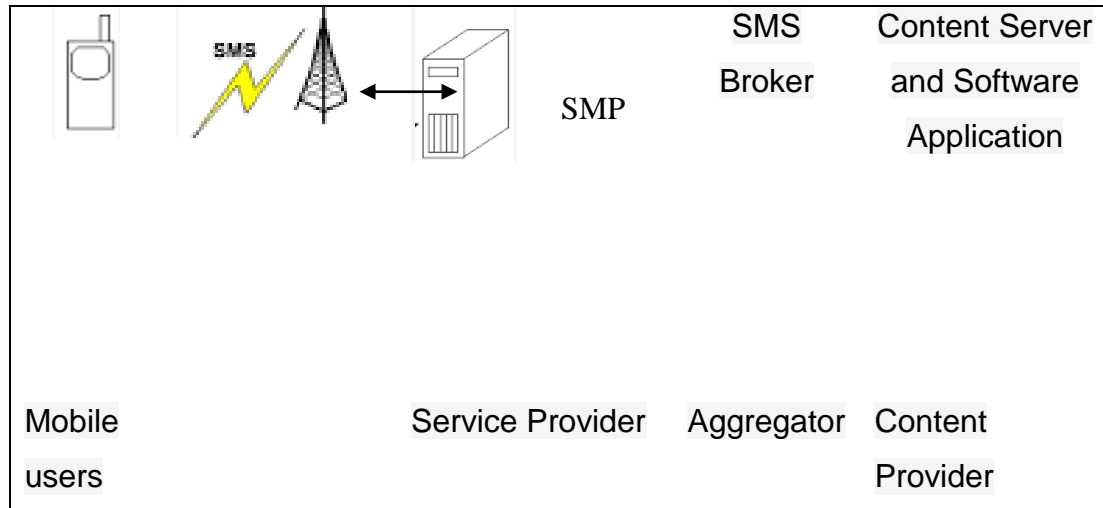


Fig. 2.2

Content aggregator uses the SMPP (Short Message Peer-to-peer Protocol) to maintain connections with carrier networks.

Content Providers

A mobile content provider is an entity that provides value-added content and applications for mobile devices.

When a mobile phone user sends an interactive text message to retrieve information, the content provider returns the information through the aggregator. The aggregator is responsible for transmitting the message to the end user.

SMS Centers

When the user sends a text message to another user, the phone actually sends the message to the SMSC, which stores the message and then delivers it, when the recipient is on the network. This is the store and forward operation.

The use of Short Message Service Center (SM-SC) to carry ahead the SMS message to the GSM network through a definite GSM-MSC called the Short Message Service gateway MSC (SMS-GMSC). The SM-SC is allowed to link with several GSM networks and to several SM-GMSCs in a GSM network. The SMS-GMSC come across the contemporary MSC of the message acceptor and then step ahead the SMS message to that Mobile Station center, pursue the Global System for Mobile Communication (GSM) roaming protocol. The MSC then Publish the SMS through the Base Station System (BSS) to the destination MSB.

2.2.5 Value added services through SMS

Value Added Services (VAS) can be defined as services, which share one or more of the following characteristics.

- Supplementary service but adds value to total service offering
- Stimulates incremental demand for core services offering
- Stands alone in terms of profitability and revenue generation potential
- Can sometimes stand-alone operationally
- Does not cannibalize basic service unless clearly favorable
- Can be an add-on basic service, and as such, may be sold at a premium price
- May provide operational and / or administrative synergy between or among other services and not merely for diversification.

Some examples

- News/Stock Quotes Service
- Session-based Chat Application
- Email through SMS
- Health Care Services
- Micro-payment Services

2.3 GENERAL PACKET RADIO SERVICE (GPRS)

2.3.1 Introduction

General Packet Radio Services (GPRS) is a packet-based wireless communication service that promises data rates from 56 up to 114 Kbps and continuous connection to the Internet for mobile phone and computer users. The higher data rates allow users to take part in video conferences and interact with multimedia Web sites and similar applications using mobile handheld devices as well as notebook computers. GPRS is based on Global System for Mobile (GSM) communication and complements existing services such circuit-switched cellular phone connections and the Short Message Service (SMS).

GPRS packet-based services cost users less than circuit-switched services since communication channels are being used on a shared-use, as-packets-are-needed basis rather than dedicated to only one user at a time. It is also easier to make applications available to mobile users because the faster data rate means that middleware currently needed to adapt applications to the slower speed of wireless systems are no longer be needed. As GPRS has become more widely available, along with other 2.5G and 3G services, mobile users of Virtual Private Networks (VPNs) have been able to access

the private network continuously over wireless rather than through a rooted dial-up connection.

2.3.2 GPRS Packet data Network

PDP stands for Packet Data Protocol. The PDP addresses are network layer addresses (Open Standards Interconnect [OSI] model Layer 3). GPRS systems support both X.25 and IP network layer protocols. Therefore, PDP addresses can be X.25, IP, or both.

Each PDP address is anchored at a Gateway GPRS Support Node (GGSN), as shown in figure (Fig. 2.3) below. All packet data traffic sent from the public packet data network for the PDP address goes through the gateway (GGSN).

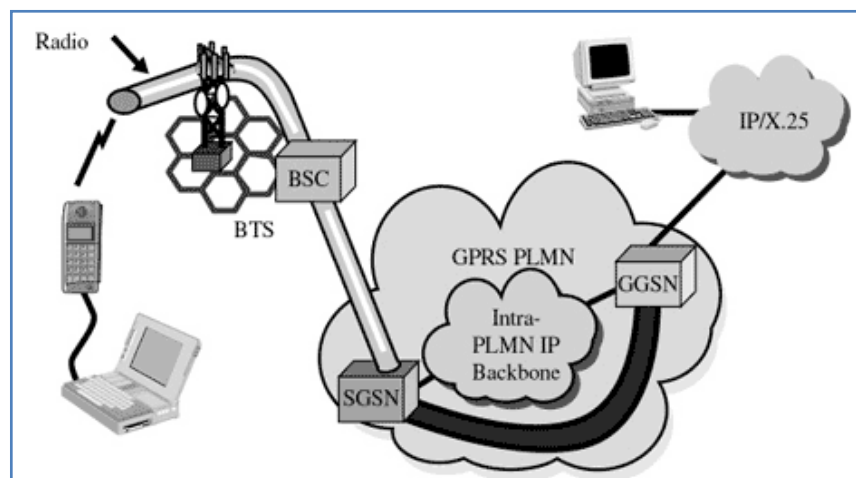


Fig. 2.3

The public packet data network is only concerned that the address belongs to a specific GGSN. The GGSN hides the mobility of the station from the rest of the packet data network and from computers connected to the public packet data network.

Statically assigned PDP addresses are usually anchored at a GGSN in the subscriber's home network. Conversely, dynamically assigned PDP addresses can be anchored either in the subscriber's home network or the network that the user is visiting.

When a MS is already attached to a SGSN and it is about to transfer data, it must activate a PDP address. Activating a PDP address establishes an association between the current SGSN of mobile device and the GGSN that anchors the PDP address.

The record kept by the SGSN and the GGSN regarding this association is called the PDP context.

It is important to understand the difference between a MS attaching to a SGSN and a MS activating a PDP address. A single MS attaches to only one SGSN, however, it may have multiple PDP addresses that are all active at the same time.

Each of the addresses may be anchored to a different GGSN. If packets arrive from the public packet data network at a GGSN for a specific PDP address and the GGSN does not have an active PDP context corresponding to that address, it may simply discard the

packets. Conversely, the GGSN may attempt to activate a PDP context with a MS if the address is statically assigned to a particular mobile device.

2.3.3 Applications for GPRS

Some of the important characteristics of GPRS are Mobility, Immediacy and Localization. With these characteristics, mobile subscribers can develop varied applications.

In general, it can be divided into two high-level categories:

- Corporation
- Consumer

These two levels further include:

- **Communications** - E-mail, fax, unified messaging and intranet/internet access, etc.
- **Value-added services** - Information services and games, etc.
- **E-commerce** - Retail, ticket purchasing, banking and financial trading, etc.
- **Location-based applications** - Navigation, traffic conditions, airline/rail schedules and location finder, etc.
- **Vertical applications** - Freight delivery, fleet management and sales-force automation.
- **Advertising** - Advertising may be location sensitive. For example, a user entering a mall can receive advertisements specific to the stores in that mall.

2.3.3.1 Generic Applications

- It is like information services, Internet access, email, Web Browsing, which are very useful while mobile
- Due to high bandwidth mobile Internet browsing will be better suited to GPRS
- Access to corporate Intranet can add new dimension to mobile workers.
- Banking over wireless is another generic applications people may like to use while mobile

2.3.3.2 GPRS Specific Applications

1. Chat

It is used as means to communicate and discuss matters of common interest. GPRS will offer ubiquitous chat by integrating Internet Chat and Wireless Chat using SMS and WAP.

2. Multimedia Services

- Multimedia objects like photographs, pictures postcards, greeting cards and presentations, static web pages can be sent and received over the mobile network.
- It can be used for monitoring parking lots or building sites for intruders and thieves.

- This can also be used by law enforcement agents, journalists and insurance agents for sending images of accident site.

3. Virtual Private Network

- GPRS is used to provide VPN services.
- As the bandwidth is higher so many banks in India are migrating to GPRS-based networks.
- This is expected to reduce the transaction time by about 25%

4. Vehicle Positioning

- This application integrates GPS that tell people where they are.
- Anyone with a GPS receiver can receive their satellite position and thereby find out where they are.
- Vehicle positioning applications can be used to deliver several services including remote vehicle diagnostics.

2.3.4 Limitations of GPRS

Although GPRS will provide better data rates than is currently available, there are some limitations.

- Speeds of 177.2kbps would require a user to use all eight timeslots without any error protection - which simply won't happen. Initial terminals are likely to use only 1-3 timeslots anyway, limiting the available bandwidth to a GPRS user.
- When GPRS packets are sent to a destination they are sent in all different directions - allowing for the potential for one or some of those packets to be corrupted or even lost altogether during the data transmission over the radio link. This is inherent in wireless packet technologies so data integrity and retransmission strategies are incorporated - which in turn result in potential transit delays.
- Although available radio resource can be concurrently shared between several users, an increase in the numbers of users will slow data services down for each user.

2.3.5 FEATURES OF 3G AND 4G DATA SERVICE

2.3.5.1 Features of 3G

The ITU (International Telecommunication Union) has proposed 3G telecommunications standards to provide cost efficient high quality, wireless multimedia applications and enhanced wireless communications.

The features of 3G can be divided into two categories. One is data rates and the other is security.

- The main feature of 3G technology is that it supports greater voice and data capacity and high data transmission at low-cost. 3G mobiles can operate on 2G and 3G technologies.

- The second major feature is the security: 3G offers greater security features than 2G like Network Access Security, Network Domain Security, User Domain Security and Application Security.
- This technology provides localized services for accessing traffic and weather updates. Video calls and video conference is another major feature in 3G mobile technology. These features reduce the communication barriers between people that were not sacked even with mobile phones.

Date transfer rates are high and can support even live TV channels over phone. Online media is another exciting feature in 3G mobiles. 3G mobiles highly attract the music lovers as they can listen to music and watch videos online and can download huge files with in less time.

2.3.5.2 Features of 4G

4G is referred to the Fourth Generation of Mobile Communications Technology, which combines 3G and WLAN into one and is able to transmit high-quality video images, and the quality of image transmission comparable to high-definition television technology products. Some of the important features are;

- This system based on orthogonal frequency division multiplexing (OFDM) as a core technology. The characteristics of the OFDM technology network architecture is highly scalable, and has a good anti-noise performance and anti-channel interference, can provide a higher quality of wireless data technology (high rate, delay), and better cost performance, able to provide better solutions for 4G wireless network.
- 4G communications gives the most impressive features than it has a faster wireless communication speed, even the highest you can reach up to 100Mbps speed transmission of wireless data.
- It is estimated that each 4G channel will occupy the 100MHz spectrum, equivalent to 20 times the W-CDMA 3G network.
- The 4G mobile communications have high intelligence, not only on the performance of 4G mobile terminals in design and operation, but also on the menu and scroll operation which are greatly reduced.
- The 4G mobile communication system in the future should have global roaming, open interfaces, and be able to talk to a variety of network interconnection, terminal diversification as well as a smooth transition from the second generation of the characteristics.
- It has a new integration technology such as the OFDM / CDMA, digital audio broadcast.
- It has many powerful breakthrough technology developed process, such as fiber optic communications, cover different types of communication interface, which means that the 4G use routing technology (Routing) based network architecture. Due to the use of several different techniques, it is much more effective than the 2G and 3G system.

2.4 MOBILE OPERATING SYSTEMS

2.4.1 Evolution of Mobile Operating System

A mobile operating system, also called a **mobile OS**, is an Operating System that is specifically designed to run on mobile devices such as mobile phones, Smart phones, PDAs, tablet computers and other handheld devices. Naturally, the evolution process is driven by the technology advancements in Hardware, Software and the Internet. The following table explains the evolution of Mobile Operating System

Year	Month	Product Name
1996	---	Palm OS 1.0
1997	OCT	Palm OS 2.0
1998	SEP	Palm OS 3.0
2000	---	Pocket PC 2000
2001	---	Pocket PC 2002
	JUN	Palm OS 4.0
2002	JUN	Palm OS 5.0
2003	---	Windows Mobile 2003
2004	JUN	Windows Mobile 2003 SE
		Palm OS Cobalt
2005	MAY	Windows Mobile 5
	OCT	Blackberry OS 4.1
		Google Acquired Android
2007	JAN	iPhone
	FEB	Windows Mobile 6
	APR	Blackberry OS 4.2
2008	JUN	iPhone OS 2.0
	SEP	Blackberry OS 4.5
	---	Windows Mobile 6.1
	---	Android 1.0
	---	Symbian OS
2009	FEB	Android 1.1
	OCT	Blackberry OS 5
	MAY	Windows Mobile 6.5
	NOV	Samsung Bada 1.0
	JUN	iPhone OS 3.0
	---	HP WB OS
	APR	Android 1.5 (Cupcake)
	Sep	Android 1.6 (Donut)
	OCT	Android 2.0 / 2.1 (Éclair)

2010	JUN	iPhone
	AUG	Blackberry OS 6
	OCT	Windows Phone 7
	MAY	Android 2.2 (Froyo)
	DEC	Android 2.3 (GingerBread)
		Symbian^2 OS
	FEB	Symbian^3 OS
2011	---	Android 3.0 (HoneyComb)

2.4.2 Handset Manufacturers and their Mobile OS

There are phones of every shape and size from a single manufacturer and there are a lot of manufacturers. Since it's the age of Smart Phones, Mobile Phones are now almost powerful. But then again, hardware specifications aside, the heart and soul of mobile phones are the operating systems. Some operating systems are open source and available to all manufactures, and some are closed source and used only by one manufacturer. Each differs in the look, feel, utilization of hardware, number of applications, etc., But most of the operating system are usually cross-platform. Here, we list some of the handset manufacturers and their mobile OS.

Sno.	Handset Manufacturers	Mobile OS	Special Features
1.	Samsung Electronics	Bada OS	Multipoint-touch, 3D graphics and of course, application downloads and installation.
2.	Blackberry	Blackberry OS	Synchronization with Microsoft Exchange, Lotus Domino, Novell GroupWise email and other business software
3.	Apple	iPhone OS (IOS)	IOS is a closed source, proprietary mobile operating system that is used on a wide range of Apple devices that include mobile phones, portable music players and a tablet PC.
4.	Nokia and Intel	MeeGo OS	Using Intel Atom and ARMv7 architecture
5.	Nokia	Symbian OS	High-level of integration with communication and personal information management (<u>PIM</u>) functionality
6.	HP	webOS	improving security features and management
7.	Microsoft	Windows Phone	Integration with Microsoft Services like Xbox Live, Windows Live and Zune along with third party applications like Facebook and Google accounts.

2.4.3 Mobile OS and their features

Mobile OS	Features
Palm OS 1.0	RIM Applications Address, Date Book, Memo Pad and To Do List
Palm OS 2.0	Mail and Expense
Palm OS 3.0	Hotsync Support, Web Clipping support, native 8 bit color support
Pocket PC 2002	MSN Messenger, Media Player 8 Enhanced UI
Palm OS 4.0	External File Systems, 16-bit Color Screens
Palm OS 5.0	PACE Emulator, Bluetooth Emulator
Windows Mobile 2003	Bluetooth Integration, Pocket Internet Explorer, Windows Media Player 9.0
Windows Mobile 2003 SE	SE Portrait and Landscape Switching for Pocket PCs and WPA
Palm OS Cobalt	Telecommunication, Wi-Fi and Bluetooth Connectivity
Windows Mobile 5	Windows Media Player 10, GPS Interface
Windows Mobile 6	.NET Compact Framework V2 SP2, Microsoft SQL Server 2005 Compact Edition Windows Live
Blackberry OS 4.2	Voice Notes Option, Email and SMS in separate folders
iPhone OS 2.0	Third-Party application Support, Sync Google Contacts
Blackberry OS 4.5	HTML e-mails, Faster Performance and improved Multi-tasking, Microsoft Office Documents ToGo
Windows Mobile 6.1	Threaded SMS, Full Page Zooming in IE and Domain Enroll
Symbian OS	Desktop interactive Widgets, Facebook IM Chat
Android 1.1	Support for saving attachments from MMS, Marquee in Layouts, API Changes
Blackberry OS 5	Wireless Sync, Blackberry Enterprise Server 4, almost
Windows Mobile 6.5	Internet Explorer Mobile 6 and Multi touch Support
Samsung Bada 1.0	Multipoint-touch, 3D graphics and of course, application downloads and installation.
iPhone OS 3.0	Push notifications, cut, copy and paste, Turn-by-Turn Navigation, Voice Memos
HP WB OS	Synergy App, Multi-touch gestures and Multi-tasking
Android 1.5 (Cupcake)	Bluetooth A2DP and AVRCP support, Uploading videos to YouTube and Pictures to Picasa
Android 1.6 (Donut)	WVGA Screen resolution support, Google free

	turn-by-turn navigation
Android 2.0 / 2.1 (Éclair)	HTMLS support, Microsoft Exchange Server, Bluetooth 2.1
iPhone	Multi-tasking, folders
Blackberry OS 6	New Media Interface, Stronger Social Media integration, Multiple Contact Lists, Trackpad support for Swipe gestures
Windows Phone 7	Tiled UL, loud-based service support, Multitasking
Android 2.2 (Froyo)	USB tethering and Wi-Fi hotspot functionality, Adobe Fash 10.1 support
Android 2.3 (GingerBread)	Multi-touch software keyboard, Support for extra-large screen sizes and resolutions
Symbian^3 OS	Native Webkit based browser, 2D and 3D graphics architecture, UI improvements and support for external displays through HDMI.
Android 3.0 (HoneyComb)	Optimized tablet support with a new user interface. Three dimensional desktop, video chat with Gtalk Support.

2.4.4 Linux Kernel based Mobile OS

Linux Kernel based mobile operating system is Android OS. It is developed by Google Corporation. Android OS give the open choice to the user to modify and add any new applications without even bring on the notice of Google.

Android is the Linux based technology tat uses Unix as an Operating System. Any one can upload a new application on the Android Platform as App store to entire free or payable. These applications uploads by the users can be easily download by the users and enjoy the additional games, interactive media and business parts.

Thus the flexibility of Android technology makes it more convenient to the operating system to have this as a base for smart phones.

Summary

- ✓ GSM is the most widely accepted standard in telecommunications and it is implemented globally.
- ✓ It is a digital cellular technology used for transmitting mobile voice and data services.
- ✓ The GSM network architecture consists of three major subsystems. They are Mobile Station, Base Station Subsystem and Network Subsystem
- ✓ Code Division Multiple Access (CDMA) is a digital cellular technology used for mobile communication.
- ✓ CDMA cellular systems are deemed superior to FDMA and TDMA, which is why CDMA plays a critical role in building efficient, robust, and secure radio communication systems.

- ✓ Short Message Service is a mechanism of delivery of short messages over the mobile networks. It is a store and forward way of transmitting messages to and from mobiles.
- ✓ Each short message is up to 160 characters.
- ✓ General Packet Radio Services (GPRS) is a packet-based wireless communication service that promises data rates from 56 up to 114 Kbps and continuous connection to the Internet for mobile phone and computer users.
- ✓ PDP stands for Packet Data Protocol. The PDP addresses are network layer addresses
- ✓ In general, GPRS can be divided into two high-level categories. They are Corporation and Consumer. Further it divided into Communications, Value-added services, E-commerce, Location-based applications, Vertical applications , Advertising
- ✓ Some of the applications of GPRS are Chat, Vehicle Positioning, Multimedia Services, etc.,
- ✓ The ITU (International Telecommunication Union) has proposed 3G telecommunications standards to provide cost efficient high quality, wireless multimedia applications and enhanced wireless communications.
- ✓ 4Gis referred to the Fourth Generation of Mobile Communications Technology, which combines 3G and WLAN into one and is able to transmit high-quality video images, and the quality of image transmission comparable to high-definition television technology products.
- ✓ Naturally, the evolution process is driven by the technology advancements in Hardware, Software and the Internet.
- ✓ Some of the latest mobile operating systems are Palm OS, Windows OS, Blackberry OS, Android OS, Symbian OS
- ✓ Android is the Linux based mobile operating system developed by Google Corporation that gives the open choice to the user to modify and add any new applications without even bring on the notice of Google.

Review Questions

PART-A

1. Define : GSM
2. What is CDMA?
3. Define : SMS
4. What is GPRS?
5. List some handset manufacturers and their mobile OS
6. Expand : VAS
7. Give any two features in Android OS

PART-B

1. Discuss any two basic entities in GSM
2. Discuss any three strength of SMS
3. Discuss any two Generic Applications of GPRS
4. List out the limitations of GPRS

PART-C

1. With a neat diagram, explain the architecture of GSM
2. Explain the SMS architecture with a neat diagram.
3. List some of the Value Added Services through SMS with an example
4. List and explain the GPRS Specific applications
5. Give the features of 3G data service
6. Give the features of 4G data service
7. Explain the evolution of Mobile Operating System.
8. List out the different mobile operating systems with their features
9. Explain the Linux Kernel based Mobile OS in detail

-- -

UNIT-III

INTRODUCTION TO ANDROID

OBJECTIVES

At the end of the unit, students can

- **Explain Android versions and features**
- **Explain architecture of Android**
- **Explain Android SDK & SDT**
- **Explain activities and Intents**
- **Explain Views and viewgroups**
- **Explain Different types of Layouts.**
- **Explain UI Notifications**

3.1. ANDROID

- Android is a mobile operating system that is based on a modified version of Linux.
- It was originally developed by a startup of the same name, Android, Inc. In 2005, as part of its strategy to enter the mobile space, Google purchased Android and took over its development work (as well as its development team).
- Google wanted Android to be open and free; hence, most of the Android code was released under the open-source Apache License, which means that anyone who wants to use Android can do so by downloading the full Android source code.
- Moreover, vendors (typically hardware manufacturers) can add their own proprietary extensions to Android and customize Android to differentiate their products from others.
- This simple development model makes Android very attractive and has thus piqued the interest of many vendors.
- This has been especially true for companies affected by the phenomenon of Apple's iPhone, a hugely successful product that revolutionized the smart phone industry.
- The main advantage of adopting Android is that it offers a unified approach to application development.
- Developers need only develop for Android, and their applications should be able to run on numerous different devices, as long as the devices are powered using Android.

3.1.1. Android versions

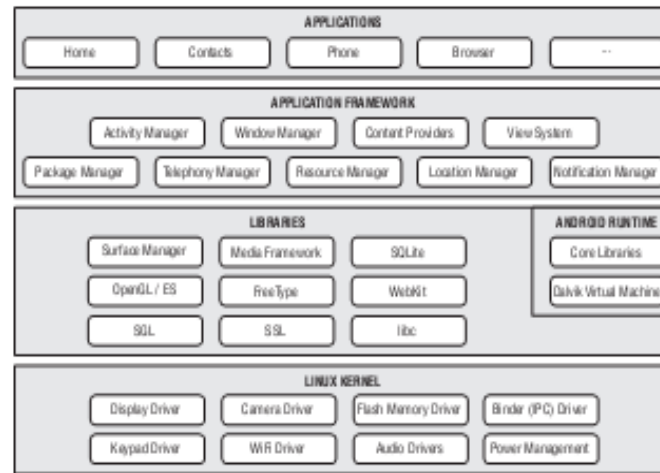
ANDROID VERSION	RELEASE DATE	CODENAME
1.1	9 February 2009	
1.5	30 April 2009	Cupcake
1.6	15 September 2009	Donut
2.0/2.1	26 October 2009	Edlair
2.2	20 May 2010	Froyo
2.3	6 December 2010	Gingerbread
3.0	Unconfirmed at the time of writing	Honeycomb

3.1.2. Features of Android

As Android is open source and freely available to manufacturers for customization, there are no fixed hardware and software configurations. However, Android itself supports the following features:

- **Storage** :Uses SQLite, a lightweight relational database, for data storage.
- **Connectivity** :Supports GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth (includes A2DP and AVRCP), WiFi, LTE, and WiMAX.
- **Messaging** :Supports both SMS and MMS.
- **Web browser**: Based on the open-source WebKit, together with Chrome's V8 JavaScript engine
- **Media support** :Includes support for the following media: H.263, H.264 (in 3GP or MP4 container), MPEG-4 SP, AMR, AMR-WB (in 3GP container), AAC, HE-AAC (in MP4 or 3GP container), MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, and BMP
- **Hardware** : support Accelerometer Sensor, Camera, Digital Compass, Proximity Sensor, and GPS.
- **Multi-touch** : Supports multi-touch screens
- **Multi-tasking** : Supportsmulti-tasking applications.
- **Flash support** : Android 2.3 supports Flash 10.1.
- **Tethering** :Supports sharing of Internet connections as a wired/wireless hotspot

3.1.3. Architecture of Android



The Android OS is roughly divided into five sections in four main layers:

Linux kernel — This is the kernel on which Android is based. This layer contains all the low-level device drivers for the various hardware components of an Android device.

Libraries — These contain all the code that provides the main features of an Android OS. For example, the SQLite library provides database support so that an application can use it for data storage. The WebKit library provides functionalities for web browsing.

Android runtime — At the same layer as the libraries, the Android runtime provides a set of core libraries that enable developers to write Android apps using the Java programming language. The Android runtime also includes the Dalvik virtual machine, which enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine (Android applications are compiled into the Dalvik executables). Dalvik is a specialized virtual machine designed specifically for Android and optimized for battery-powered mobile devices with limited memory and CPU.

Application framework — Exposes the various capabilities of the Android OS to application developers so that they can make use of them in their applications.

Applications — At this top layer, you will find applications that ship with the Android device (such as Phone, Contacts, Browser, etc.), as well as applications that you download and install from the Android Market.

3.1.4.The Android market

- One of the main factors determining the success of a smartphone platform is the applications that support it.
- It is clear from the success of the iPhone that applications play a very vital role in determining whether a new platform swims or sinks. In addition, making these applications accessible to the general user is extremely important.
- As such, in August 2008, Google announced the Android Market, an online application store for Android devices, and made it available to users in October 2008.
- Using the Market application that is preinstalled on their Android device, users can simply download third-party applications directly onto their devices.
- Both paid and free applications are supported on the Android Market, though paid applications are available only to users in certain countries due to legal issues.
- Similarly, in some countries, users can buy paid applications from the Android Market, but developers cannot sell in that country.
- As an example, at the time of writing, users in India can buy apps from the Android Market, but developers in India cannot sell apps on the Android Market.
- The reverse may also be true; for example, users in South Korea cannot buy apps, but developers in South Korea can sell apps on the Android Market.

3.1.5. Android runtime (Dalvik Virtual Machine)

- The Android runtime provides a set of core libraries that enable developers to write Android apps using the Java programming language.
- The Android runtime also includes the Dalvik virtual machine, which enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine (Android applications are compiled into the Dalvik executables).
- Dalvik is a specialized virtual machine designed specifically for Android and optimized for battery-powered mobile devices with limited memory and CPU.

3.2.ANDROID SDK AND ADT

3.2.1.ANDROID SDK

- The Android SDK contains a debugger, libraries, an emulator, documentation, sample code, and tutorials.
- You can download the Android SDK from <http://developer.android.com/sdk/index.html>.

- Once the SDK is downloaded, unzip its content (the android-sdk-windows folder) into the C:\Android\ folder, or whatever name you have given to the folder you just created.

3.2.2.ANDROID DEVELOPMENT TOOLS (ADT) [INSTALLING AND CONFIGURING ANDROID]

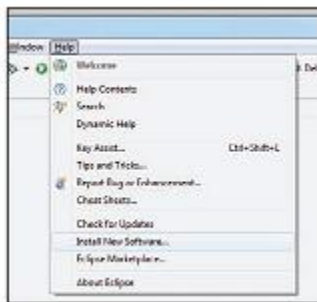
The Android Development Tools (ADT) plug-in for Eclipse is an extension to the Eclipse IDE that supports the creation and debugging of Android applications. Using the ADT, you will be able to do the following in Eclipse:

- Create new Android application projects.
- Access the tools for accessing your Android emulators and devices.
- Compile and debug Android applications.
- Export Android applications into Android Packages (APK).
- Create digital certificates for code-signing your APK.

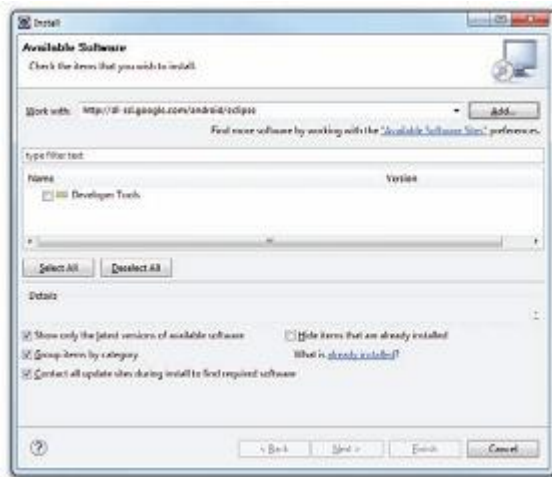
1.To install the ADT, first launch Eclipse by double-clicking on the eclipse.exe file located in the eclipse folder.

2.When Eclipse is first started, you will be prompted for a folder to use as your workspace. In Eclipse, a workspace is a folder where you store all your projects. Take the default suggested and click OK.

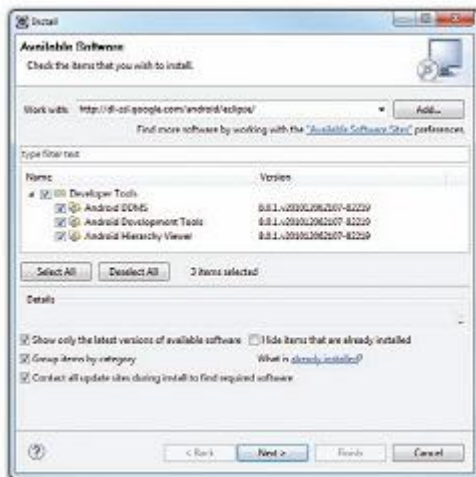
3.Once Eclipse is up and running, select the Help ➡ Install New Software... menu item



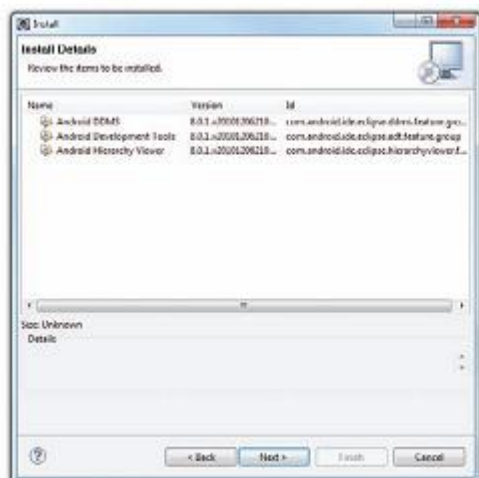
4.In the Install window that appears, type <http://dl-ssl.google.com/android/eclipse> in the text box



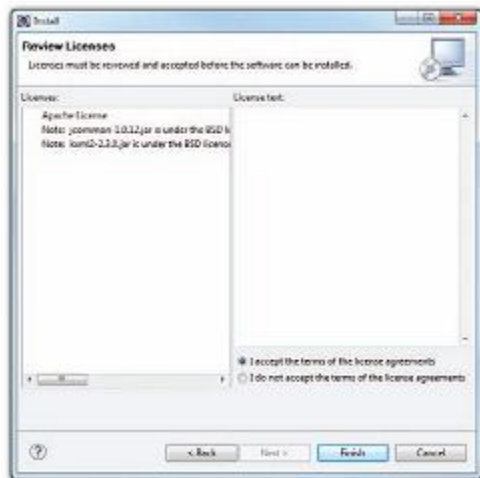
5. After a while, you will see the Developer Tools item appear in the middle of the window



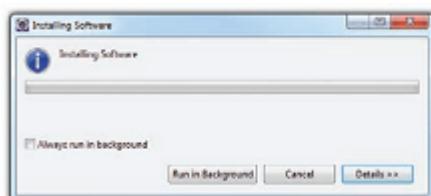
6. Expand it, and it will reveal its content: Android DDMS, Android Development Tools, and Android Hierarchy Viewer. Check all of them and click Next. When you see the installation details, click Next.



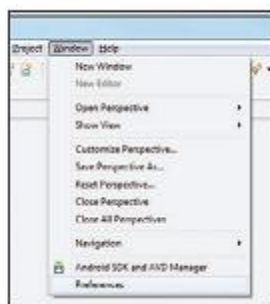
7.You will be asked to review the licenses for the tools. Check the option to accept the license agreements ,Click Finish to continue.



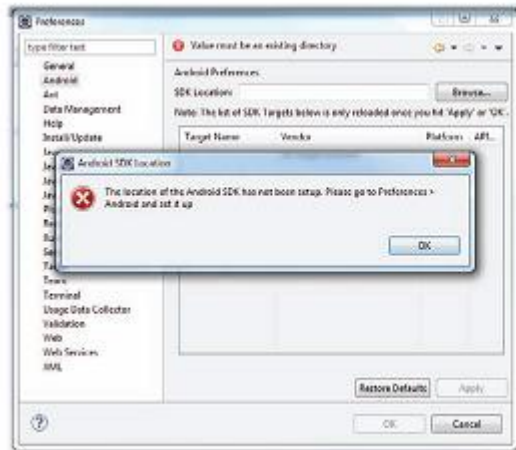
8.Eclipse will now proceed to download the tools from the Internet and install them .



9.Once the ADT is installed, you will be prompted to restart Eclipse. After doing so, go to Window ⇄ Preferences



10.In the Preferences window that appears, select Android. You will see an error message saying that the SDK has not been set up .Click OK to dismiss it.



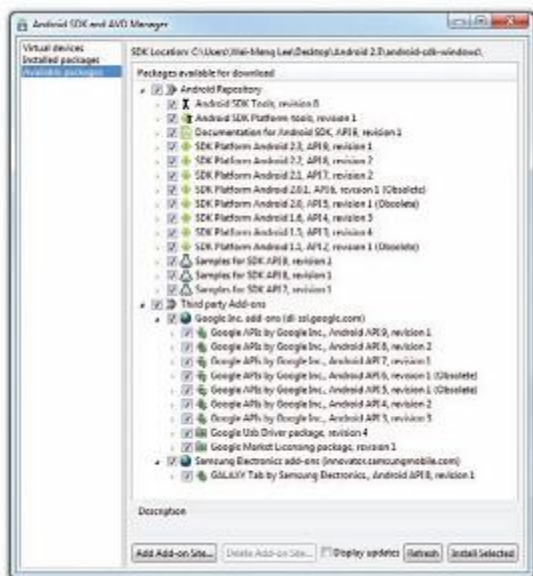
11. Enter the location of the Android SDK folder. In this example, it would be C:\Android\android-sdk-windows. Click OK.

3.2.3.ANDROID VIRTUAL DEVICES (AVDS)

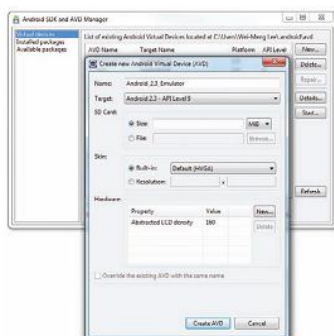
- AVD stands for Android Virtual Devices. An AVD is an emulator instance that enables you to model an actual device.
- Each AVD consists of a hardware profile, a mapping to a system image, as well as emulated storage, such as a secure digital (SD) card.
- You can create as many AVDs as you want in order to test your applications with several different configurations. This testing is important to confirm the behavior of your application when it is run on different devices with varying capabilities.

Procedure:

1. To create an AVD, go to Windows ⇄ Android SDK and AVD Manager.
2. Select the Available packages option in the left pane and expand the package name shown in the right pane.



3. Check the relevant tools, documentation, and platforms you need for your project. Once you have selected the items you want, click the Install Selected button to download them.
4. Each version of the Android OS is identified by an API level number. For example, Android 2.3 is level 9 (API 9), while Android 2.2 is level 8 (API 8), and so on. For each level, two platforms are available. For example, level 9 offers the following: SDK Platform Android 2.3 Google APIs by Google Inc.
5. The key difference between the two is that the Google APIs platform contains the Google Maps library.
6. Click the Virtual Devices item in the left pane of the window. Then click the New... button located in the right pane of the window.
7. In the Create new Android Virtual Device (AVD) window, enter the items .
8. Click the Create AVD button when you are done.



3.3.ACTIVITIES AND INTENTS

3.3.1.Understanding activities

To create an activity, you create a Java class that extends the `Activity` base class:

```
package net.learn2develop.Activities;
import android.app.Activity;
import android.os.Bundle;
public class MainActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Your activity class would then load its UI component using the XML file defined in your `res/layout` folder. In this example, you would load the UI from the `main.xml` file:

```
setContentView(R.layout.main);
```

Every activity you have in your application must be declared in your `AndroidManifest.xml` file, like this:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="net.learn2develop.Activities"
    android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon"
        android:label="@string/app_name">
        <activity android:name=".MainActivity"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
                    android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

```

</intent-filter>

</activity>

</application>

<uses-sdk android:minSdkVersion="9" />

</manifest>

```

The Activity base class defines a series of events that governs the life cycle of an activity.

The Activity class defines the following events:

onCreate() — Called when the activity is first created

onStart() — Called when the activity becomes visible to the user

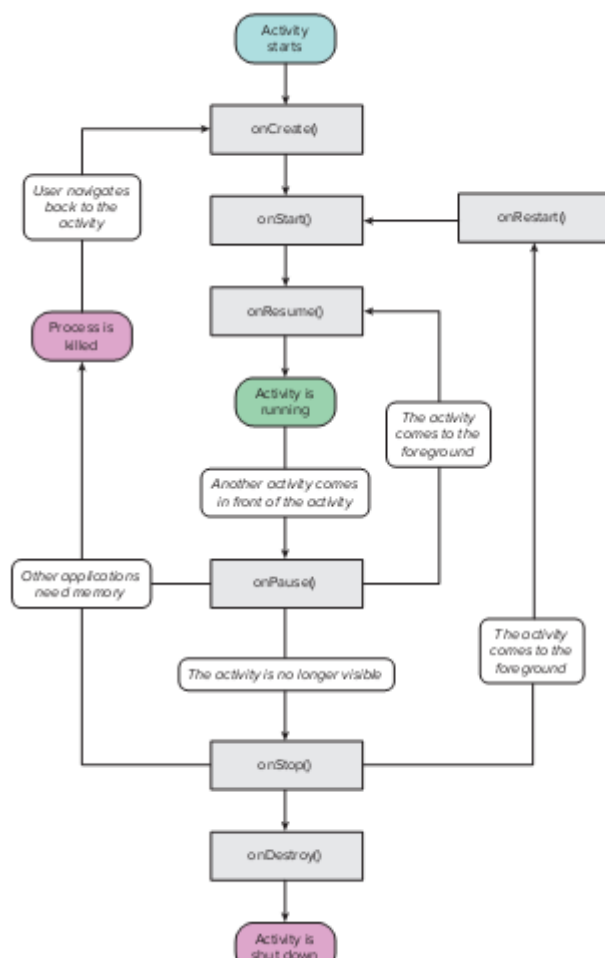
onResume() — Called when the activity starts interacting with the user

onPause() — Called when the current activity is being paused and the previous activity is being resumed

onStop() — Called when the activity is no longer visible to the user

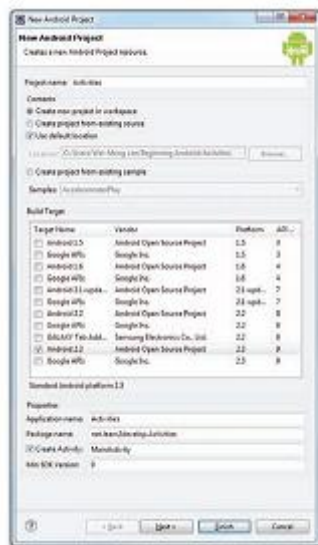
onDestroy() — Called before the activity is destroyed by the system (either manually or by the system to conserve memory)

onRestart() — Called when the activity has been stopped and is restarting again



Example:

- 1 . Using Eclipse, create a new Android project and name it



- 2 . In the MainActivity.java file, add the following statements

```
packagenet.learn2develop.Activities;  
import android.app.Activity;  
import android.os.Bundle;  
import android.util.Log;  
public class MainActivity extends Activity {  
String tag = "Events";  
/** Called when the activity is first created. */  
@Override  
public void onCreate(Bundle savedInstanceState) {  
super.onCreate(savedInstanceState);  
setContentView(R.layout.main);  
Log.d(tag, "In the onCreate() event");  
}  
public void onStart()  
{  
super.onStart();  
Log.d(tag, "In the onStart() event");  
}  
public void onRestart()
```

```

{
    super.onRestart();
    Log.d(tag , "In the onRestart() event");
}

public void onResume()
{
    super.onResume();
    Log.d(tag , "In the onResume() event");
}

public void onPause()
{
    super.onPause();
    Log.d(tag , "In the onPause() event");
}

public void onStop()
{
    super.onStop();
    Log.d(tag , "In the onStop() event" );
}

public void onDestroy()
{
    super.onDestroy();
    Log.d(tag , "In the onDestroy() event");
}
}

```

- 3 . Press F11 to debug the application on the Android Emulator.
- 4 . When the activity is first loaded, you should see the following in the LogCat window

12-2813:45:28.115: DEBUG/Events(334): Inthe onCreate()event

12-2813:45:28.115: DEBUG/Events(334): Inthe onStart()event

12-2813:45:28.115: DEBUG/Events(334): Inthe onResume()event

- 5 . When you now press the back button on the Android Emulator, observe that the following is printed:

12-2813:59:46.266:DEBUG/Events(334):IntheonPause()event

12-2813:59:46.806:DEBUG/Events(334):IntheonStop()event

12-2813:59:46.806:DEBUG/Events(334):IntheonDestroy()event

6 . Click the Home button and hold it there. Click the Activities icon and observe the following:

12-2814:00:54.115:DEBUG/Events(334):IntheonCreate()event

12-2814:00:54.156:DEBUG/Events(334):IntheonStart()event

12-2814:00:54.156:DEBUG/Events(334):IntheonResume()event

7 . Press the Phone button on the Android Emulator so that the activity is pushed to the background. Observe the output in the LogCat window:

12-2814:01:16.515:DEBUG/Events(334):IntheonPause()event

12-2814:01:17.135:DEBUG/Events(334):IntheonStop()event

8 . Notice that the onDestroy() event is not called, indicating that the activity is still in memory. Exit the phone dialer by pressing the Back button. The activity is now visible again. Observe the output in the LogCat window:

12-2814:02:17.255:DEBUG/Events(334):IntheonRestart()event

12-2814:02:17.255:DEBUG/Events(334):IntheonStart()event

12-2814:02:17.255:DEBUG/Events(334):IntheonResume()event

The onRestart() event is now fired, followed by the onStart() and onResume() events.

Working Principle:

- As you can see from this simple experiment, an activity is destroyed when you press the Back button.
- At this point, note that the onPause() event is called in both scenarios — when an activity is sent to the background, as well as when it is killed when the user presses the Back button.
- When an activity is started, the onStart() and onResume() events are always called, regardless of whether the activity is restored from the background or newly created.

3.3.2.Linking activities using intents

- An Android application can contain zero or more activities. When your application has more than one activity, you may need to navigate from one activity to another.
- In Android, you navigate between activities through what is known as an intent .

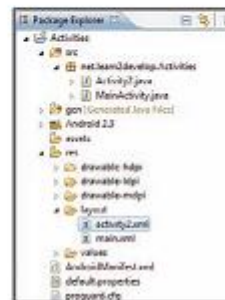
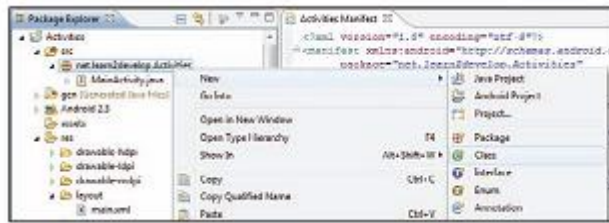
Example:

- 1 . Using the Activities project created earlier, add the following statements

AndroidManifest.xml file:

```
<?xmlversion="1.0"encoding="utf-8"?>
<manifestxmlns:android="http://schemas.android.com/apk/res/android"
package="net.learn2develop.Activities"
android:versionCode="1"
android:versionName="1.0">
<applicationandroid:icon="@drawable/icon"
android:label="@string/app_name">
<activityandroid:name=".MainActivity"
android:label="@string/app_name"
android:theme="@android:style/Theme.Dialog">
<intent-filter>
<actionandroid:name="android.intent.action.MAIN" />
<categoryandroid:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
<activity android:name=".Activity2"
android:label="Activity 2">
<intent-filter>
<action android:name="net.learn2develop.ACTIVITY2" />
<category android:name="android.intent.category.DEFAULT" />
</intent-filter>
</activity>
</application>
<uses-sdkandroid:minSdkVersion="9" />
```

- 2 . Right click on the package name under the src folder and select New ➡ Class.
Name the new class file Activity2 and click Finish.
3. Make a copy of the main.xml file by right-clicking on it and selecting Copy. Then,
right-click on the res/layout folder and select Paste. Name the file activity2.xml .
4. The res/layout folder will now contain the activity2.xml file



5 . Modify the activity2.xml file as follows:

```
<?xmlversion= "1.0"encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text= "This is Activity 2!"
    />
</LinearLayout>
```

6 . In the Activity2.java file, add the following statements

```
packagenet.learn2develop.Activities;
import android.app.Activity;
import android.os.Bundle;
public class Activity2 extends Activity {
    @Override
```

```

public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity2);
}
}

```

- 7 . Modify the MainActivity.java file as shown

```

package net.learn2develop.Activities;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.Window;
import android.view.KeyEvent;
import android.content.Intent;
public class MainActivity extends Activity {
    String tag = "Events" ;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        //---hide the titlebar---
        //requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(R.layout.main);
        Log.d(tag, "In the onCreate() event");
    }
    public boolean onKeyDown(int keyCode, KeyEvent event)
    {
        if (keyCode == KeyEvent.KEYCODE_DPAD_CENTER)
        {
            startActivity(new Intent( "net.learn2develop.ACTIVITY2"));
        }
        return false;
    }
    public void onStart(){//...}
}

```

```

public void onRestart() { //... }
public void onResume() { //... }
public void onPause() { //... }
public void onStop() { //... }
public void onDestroy() { //... }
}

```

8 . Press F11 to debug the application on the Android Emulator. When the first activity is loaded, click the center of the directional pad .The second activity will now be loaded.



Working Principle:

- An activity is made up of a UI component (for example, `main.xml`) and a class component (for example, `MainActivity.java`). Hence, if you want to add another activity to a project, you need to create these two components.
- In the `AndroidManifest.xml` file, specifically you have added the following:

```

<activity android:name= ".Activity2"
    android:label="Activity2">
    <intent-filter>
        <action android:name= "net.learn2develop.ACTIVITY2"/>
        <category android:name= "android.intent.category.DEFAULT" />
    </intent-filter>
</activity>

```

Here, you have added a new activity to the application. Note the following:

- The name of the new activity added is “Activity2”.
- The label for the activity is named “Activity 2”.

- The intent filter name for the activity is “>> net.learn2develop.ACTIVITY2”. Other activities that wish to call this activity will invoke it via this name. Ideally, you should use the reverse domain name of your company as the intent filter name in order to reduce the chances of another application having the same intent filter. The next section discusses what happens when two or more activities have the same intent filter.
- The category for the intent filter is “android.intent.category.DEFAULT”. You need to add this to the intent filter so that this activity can be started by another activity using the `startActivity()`
- In the `MainActivity.java` file, you implemented the `onKeyDown` event handler. This event is fired whenever the user presses one of the keys on the device. When the user presses the center key on the directional pad (as represented by the `KeyEvent.KEYCODE_DPAD_CENTER` constant), you use the `startActivity()` method to display Activity2 by creating an instance of the `Intent` class and passing it the intent filter name of Activity2 (which is `net.learn2develop.ACTIVITY2`):

```
public boolean onKeyDown(int keyCode, KeyEvent event)
{
    if(keyCode ==KeyEvent.KEYCODE_DPAD_CENTER)
    {
        startActivity(new Intent( “net.learn2develop.ACTIVITY2”));
    }
    return false;
}
```

- Activities in Android can be invoked by any application running on the device.
- If the activity that you want to invoke is defined within the same project, you can rewrite the preceding statement like this:
startActivity(new Intent(this,Activity2.class));
- However, this approach is applicable only when the activity you want to display is within the same project as the current activity.

3.3.4. Calling Built-in applications using intents

- One of the key aspects of Android programming is using the intent to call activities from other applications.
- In particular, your application can call the many built-in applications that are included with an Android device.
- For example, if your application needs to enable a user to call a particular person saved in the Contacts application, you can simply use an Intent object to bring up the Contacts application, from which the user can select the person to call.
- This enables your application to present a consistent user experience, and enables you to avoid building another application to retrieve all the contacts in the Contacts application.
- The following Try It Out demonstrates how to call some of the built-in applications commonly found on an Android device.

Example:

1. Using Eclipse, create a new Android project and name it Intents .
2. Add the following statements to the main.xml file:

```
<?xmlversion="1.0"encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent" >
<Button
android:id="@+id/btn_webbrowser"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Web Browser" />
<Button
android:id="@+id/btn_makecalls"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Make Calls" />
<Button
```

```

android:id="@+id/btn_showMap"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text= "Show Map" />

<Button
android:id="@+id/btn_chooseContact"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text= "Choose Contact" />

</LinearLayout>

```

3.Add the following statements to the MainActivity.java file:

```

packagenet.learn2develop.Intents;
importandroid.app.Activity;
importandroid.os.Bundle;
import android.content.Intent;
import android.net.Uri;
import android.provider.ContactsContract;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;
publicclassMainActivityextendsActivity {
Button b1, b2, b3, b4;
int request_Code = 1;
/** Calledwhenthe activity isfirstcreated. */
@Override
publicvoidonCreate(Bundle savedInstanceState){
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
//---Web browser button---
b1 = (Button) findViewById(R.id.btn_webbrowser);
b1.setOnClickListener(new OnClickListener()
{

```

```

public void onClick(View arg0){
    Intent i = new
    Intent(android.content.Intent.ACTION_VIEW,
        Uri.parse("http://www.amazon.com" ));
    startActivity(i);
}

});

//---Make calls button---

b2 = (Button) findViewById(R.id.btn_makecalls );
b2.setOnClickListener(new OnClickListener()
{
    public void onClick(View arg0){
        Intent i = new
        Intent(android.content.Intent.ACTION_DIAL,
            Uri.parse("tel:+651234567"));
        startActivity(i);
    }
});

//---Show Map button---

b3 = (Button) findViewById(R.id.btn_showMap);
b3.setOnClickListener(new OnClickListener()
{
    public void onClick(View arg0){
        Intent i = new
        Intent(android.content.Intent.ACTION_VIEW,
            Uri.parse("geo:37.827500,-122.481670"));
        startActivity(i);
    }
});

//---Choose Contact button---

b4 = (Button) findViewById(R.id.btn_chooseContact);
b4.setOnClickListener(new OnClickListener()
{

```

```

public void onClick(View arg0){
    Intent i = new
    Intent(android.content.Intent.ACTION_PICK);
    i.setType(ContactsContract.Contacts.CONTENT_TYPE);
    startActivityForResult(i,request_Code);
}
});
}

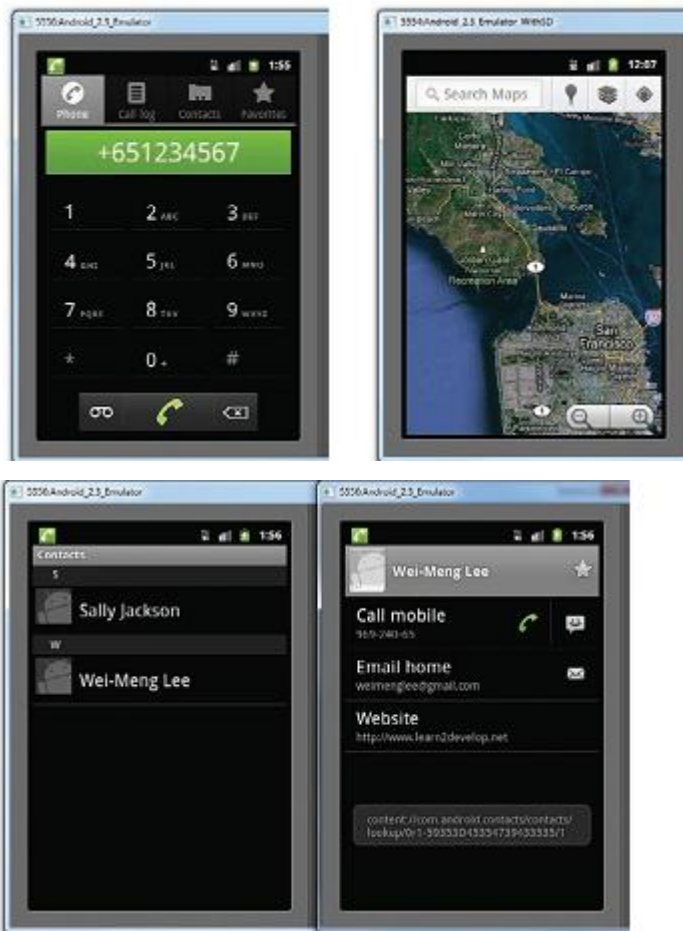
public void onActivityResult( int requestCode, int resultCode, Intent data)
{
    if (requestCode == request_Code)
    {
        if (resultCode == RESULT_OK)
        {
            Toast.makeText (this,data.getData().toString(),
            Toast.LENGTH_SHORT).show();
            Intent i = new Intent(
            android.content.Intent.ACTION_VIEW,
            Uri.parse(data.getData().toString()));
            startActivity(i);
        }
    }
}
}

```

- 4 . Press F11 to debug the application on the Android Emulator.
- 5 . Click the Web Browser button to load the Browser application on the emulator .



- 6 . Click the Make Calls button and the Phone application will load
 - 7 . Similarly, to load the Maps application, click the Show Map button.
 - 8 . Click the Choose Contact application to show a list of contacts that you can select
- Selecting a contact will show details about that contact.



3.3.4.Fragments:

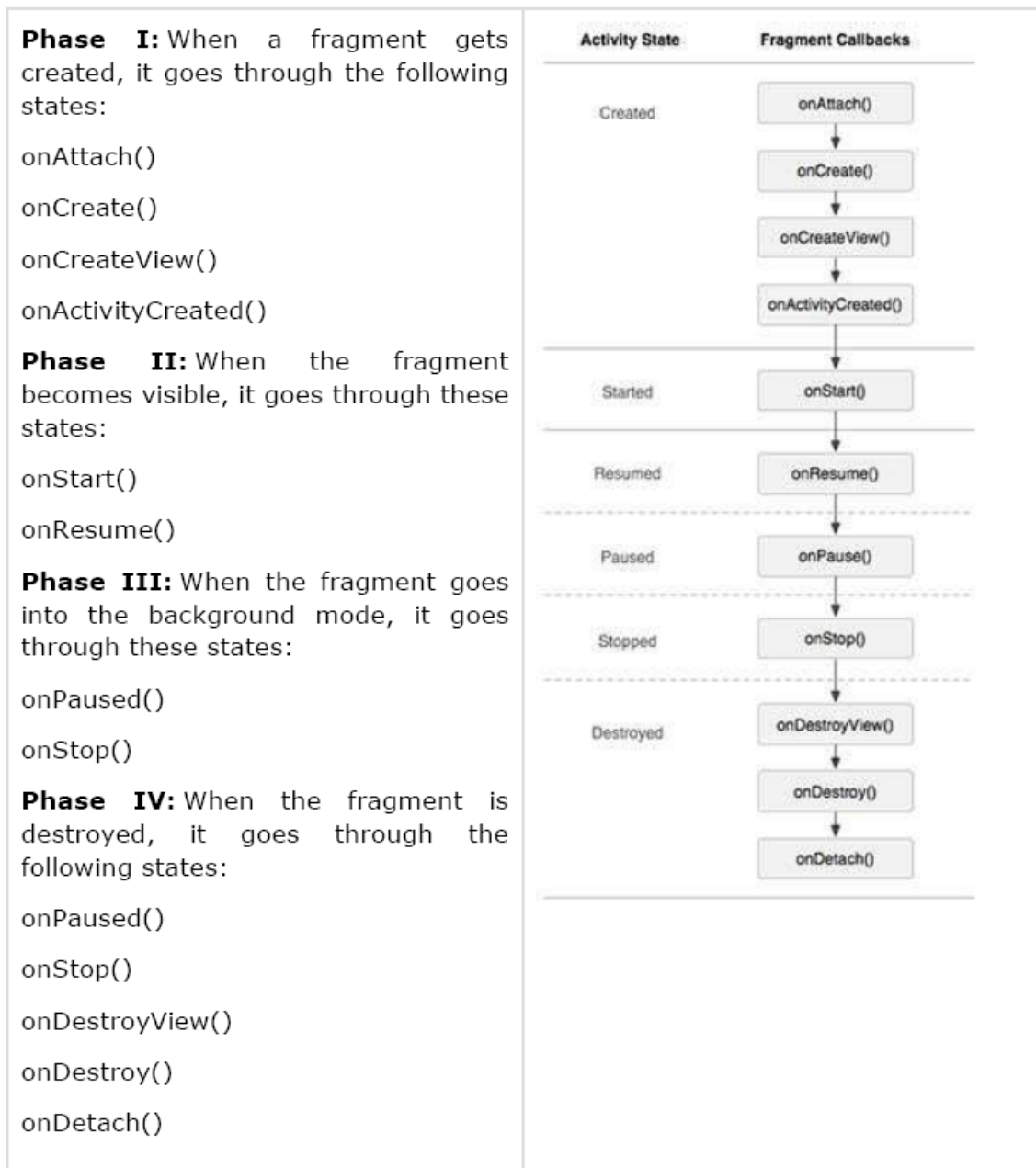
A Fragment is a piece of an application's user interface or behavior that can be placed in an Activity which enable more modular activity design. It will not be wrong if we say, a fragment is a kind of **sub-activity**. Following are the important points about fragment:

- A fragment has its own layout and its own behavior with its own lifecycle callbacks.
- You can add or remove fragments in an activity while the activity is running.
- You can combine multiple fragments in a single activity to build a multipane UI.
- A fragment can be used in multiple activities.
- Fragment life cycle is closely related to the lifecycle of its host activity which means when the activity is paused, all the fragments available in the activity will also be stopped.
- A fragment can implement a behavior that has no user interface component.

Fragment Life Cycle

Android fragments have their own life cycle very similar to an android activity.

This section briefs different stages of its life cycle.



Example:

Following is the content of the modified main activity file

```

package com.example.myfragments;

import android.os.Bundle;
import android.app.Activity;
import android.app.FragmentManager;
import android.app.FragmentTransaction;
import android.content.res.Configuration;
import android.view.WindowManager;

```

```

public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Configuration config = getResources().getConfiguration();
        FragmentManager fragmentManager = getFragmentManager();
        FragmentTransaction fragmentTransaction =
            fragmentManager.beginTransaction();
        /**
         * Check the device orientation and act accordingly
         */
        if (config.orientation == Configuration.ORIENTATION_LANDSCAPE) {
            /**
             * Landscape mode of the device
             */
            LM_Fragment ls_fragment = new LM_Fragment();
            fragmentTransaction.replace(android.R.id.content, ls_fragment);
        } else {
            /**
             * Portrait mode of the device
             */
            PM_Fragment pm_fragment = new PM_Fragment();
            fragmentTransaction.replace(android.R.id.content, pm_fragment);
        }
        fragmentTransaction.commit();
    }
}

```

Create two fragment files **LM_Fragment.java** and **PM_Fragment.java** under *com.example.mycontentprovider* package.

Following is the content of **LM_Fragment.java** file:

```

package com.example.myfragments;
import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;

```

```

import android.view.View;
import android.view.ViewGroup;
public class LM_Fragment extends Fragment{
    @Override
    public View onCreateView(LayoutInflater inflater,
        ViewGroup container, Bundle savedInstanceState) {
        /**
        * Inflate the layout for this fragment
        */
        return inflater.inflate(
            R.layout.lm_fragment, container, false);
    }
}

```

Following is the content of **PM_Fragment.java** file:

```

package com.example.myfragments;
import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
public class PM_Fragment extends Fragment{
    @Override
    public View onCreateView(LayoutInflater inflater,
        ViewGroup container, Bundle savedInstanceState) {
        /**
        * Inflate the layout for this fragment
        */
        return inflater.inflate(
            R.layout.pm_fragment, container, false);
    }
}

```

Create two layout

files: **lm_fragment.xml** and **pm_fragment.xml** under *res/layout* directory.

Following is the content of **lm_fragment.xml** file:

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:background="#7bae16">
<TextView
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/landscape_message"
android:textColor="#000000"
android:textSize="20px" />
<!-- More GUI components go here -->
</LinearLayout>

```

Following is the content of **pm_fragment.xml** file:

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="horizontal"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:background="#666666">
<TextView
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/portrait_message"
android:textColor="#000000"
android:textSize="20px" />
<!-- More GUI components go here -->
</LinearLayout>

```

Following will be the content of **res/layout/activity_main.xml** file which includes your fragments:

```

<?xml version="1.0" encoding="utf-8"?>

```

```

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="horizontal">
<fragment
android:name="com.example.fragments"
android:id="@+id/lm_fragment"
android:layout_weight="1"
android:layout_width="0dp"
android:layout_height="match_parent" />
<fragment
android:name="com.example.fragments"
android:id="@+id/pm_fragment"
android:layout_weight="2"
android:layout_width="0dp"
android:layout_height="match_parent" />
</LinearLayout>

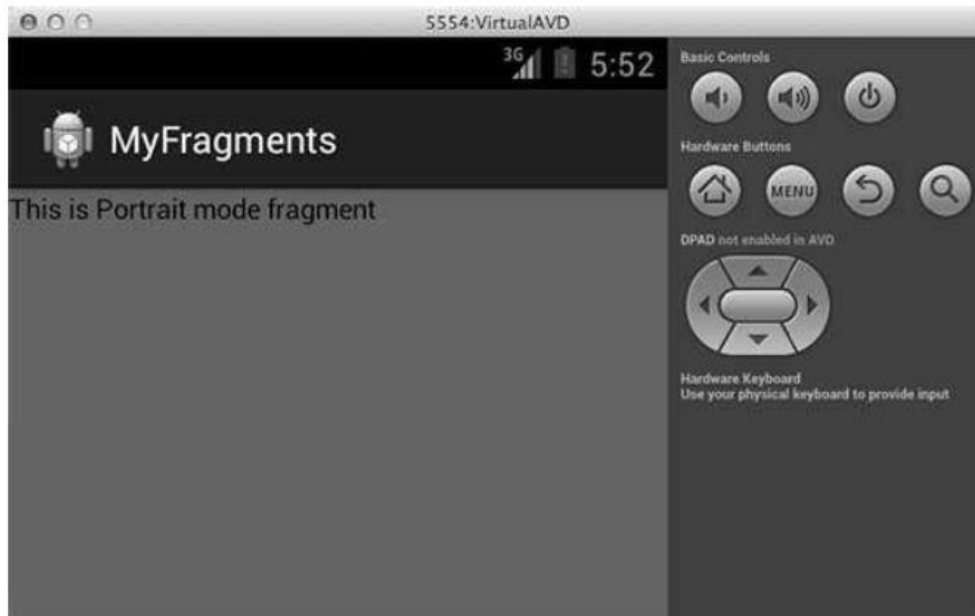
```

Make sure you have following content of **res/values/strings.xml** file:

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="app_name">MyFragments</string>
<string name="action_settings">Settings</string>
<string name="hello_world">Hello world!</string>
<string name="landscape_message">This is Landscape mode fragment
</string>
<string name="portrait_message">This is Portrait mode fragment
</string>
</resources>

```



3.3.5. Displaying notifications

- Toast class is a handy way to show users alerts, it is not persistent. It flashes on the screen for a few seconds and then disappears.
- If it contains important information, users may easily miss it if they are not looking at the screen.
- For messages that are important, you should use a more persistent method. In this case, you should use the NotificationManager to display a persistent message at the top of the device, commonly known as the status bar (sometimes also referred to as the notification bar).

Example:

- 1 . Using Eclipse, create a new Android project and name it Notifications .
- 2 . Add a new class file named NotificationView.java to the src folder of the project
In addition, add a new notification.xml file to the res/layout folder as well.
- 3 . Populate the notification.xml file as follows:

```
<?xmlversion= "1.0"encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent" >
<TextView
```

```

android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text= "Here are the details for the notification..." />
</LinearLayout>

```

- 4 . Populate the NotificationView.java file as follows:

```

packagenet.learn2develop.Notifications;
import android.app.Activity;
import android.app.NotificationManager;
import android.os.Bundle;
public class NotificationView extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.notification);

        //--look up the notification manager service---
        NotificationManager nm = (NotificationManager)
        getSystemService(NOTIFICATION_SERVICE);
        //--cancel the notification that we started
        nm.cancel(getIntent().getExtras().getInt("notificationID"));
    }
}

```

- 5 . Add the following statements to the AndroidManifest.xml file:

```

<?xmlversion= "1.0"encoding="utf-8"?>
<manifestxmlns:android="http://schemas.android.com/apk/res/android"
package= "net.learn2develop.Notifications"
android:versionCode="1"
android:versionName="1.0">
<applicationandroid:icon=                                "@drawable/icon"
android:label="@string/app_name" >
<activityandroid:name= ".MainActivity"

```

```

        android:label="@string/app_name" >
        <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER"/>
        </intent-filter>
    </activity>
    <activity android:name=".NotificationView"
        android:label="Details of notification">
        <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.DEFAULT" />
        </intent-filter>
    </activity>
</application>
<uses-sdk android:minSdkVersion="9" />
<uses-permission android:name="android.permission.VIBRATE" />
</manifest>

```

- 6 . Add the following statements to the main.xml file:

```

<?xmlversion="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <Button
        android:id="@+id/btn_displaynotif"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Display Notification" />
    </LinearLayout>

```

- 7 . Finally, add the following statements to the MainActivity.java file:

```

packagenet.learn2develop.Notifications;
import android.app.Activity;

```

```

import android.os.Bundle;
import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Intent;
import android.view.View;
import android.widget.Button;
public class MainActivity extends Activity {
    int notificationID = 1;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Button button = (Button) findViewById(R.id. btn_displaynotif);
        button.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                displayNotification();
            }
        });

        protected void displayNotification()
        {
            //---PendingIntent to launch activity if the user selects
            // this notification---
            Intent i = new Intent( this, NotificationView.class);
            i.putExtra("notificationID", notificationID);
            PendingIntent pendingIntent =
            PendingIntent.getActivity(this, 0, i, 0);
            NotificationManager nm = (NotificationManager)
            getSystemService(NOTIFICATION_SERVICE);
            Notification notif = new Notification(

```

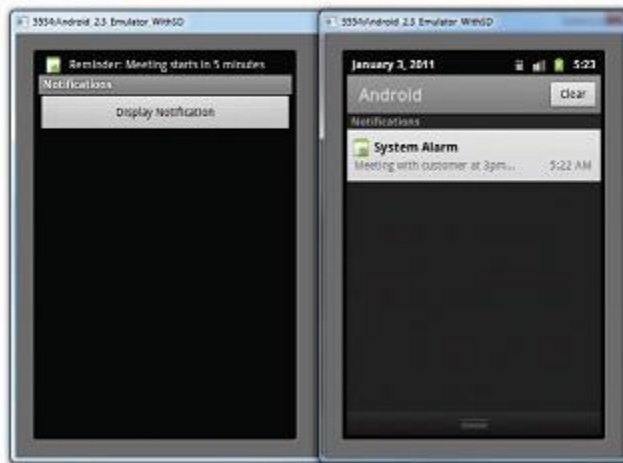
```

R.drawable.icon,
"Reminder: Meeting starts in 5 minutes",
System.currentTimeMillis());
CharSequence from = "System Alarm";
CharSequence message = "Meeting with customer at 3pm..." ;

notif.setLatestEventInfo(this, from, message, pendingIntent);
//---100ms delay, vibrate for 250ms, pause for 100 ms and
// then vibrate for 500ms---
notif.vibrate = new long[] { 100, 250, 100, 500};
nm.notify(notificationID, notif);
}
}

```

- 8 . Press F11 to debug the application on the Android Emulator.
- 9 . Click the Display Notification button (and a notification will appear on the status bar).
- 10 . Clicking and dragging the status bar down will reveal the notification
- 11 . Clicking on the notification will reveal the Notification View activity. This also causes the notification to be dismissed from the status bar.



Working Principle:

- To display a notification, you first created an Intent object to point to the NotificationView class:

```
//---PendingIntent to launch activity if the user selects  
//this notification---  
Intent i = new Intent(this, NotificationView.class);  
i.putExtra("notificationID", notificationID);
```

- This intent will be used to launch another activity when the user selects a notification from the list of notifications. In this example, you added a key/value pair to the Intent object so that you can tag the notification ID, identifying the notification to the target activity.
- The getActivity() method retrieves a PendingIntent object and you set it using the following arguments:
 1. context — Application context
 2. request code — Request code for the intent
 3. intent — The intent for launching the target activity
 4. flags — The flags in which the activity is to be launched
- You then obtain an instance of the NotificationManager class and create an instance of the Notification class:

```
NotificationManager nm = (NotificationManager)  
getSystemService(NOTIFICATION_SERVICE);  
Notification notif = new Notification(  
R.drawable.icon,  
"Reminder: Meeting starts in 5 minutes",  
System.currentTimeMillis());
```

- The Notification class enables you to specify the notification's main information when the notification first appears on the status bar. The second argument to the Notification constructor sets the "ticker text" on the status bar
- Next, you set the details of the notification using the setLatestEventInfo() method:

```
CharSequence from = "System Alarm";  
CharSequence message = "Meeting with customer at 3pm..." ;  
notif.setLatestEventInfo(this, from, message, pendingIntent);  
//---100ms delay, vibrate for 250ms, pause for 100 ms and
```

```
// then vibrate for 500ms---
```

```
notif.vibrate = new long[] { 100, 250, 100, 500};
```

- The preceding also sets the notification to vibrate the phone. Finally, to display the notification you use the `notify()` method:

```
nm.notify(notificationID, notif);
```

- When the user clicks on the notification, the NotificationView activity is launched. Here, you dismiss the notification by using the `cancel()` method of the NotificationManager object and passing it the ID of the notification (passed in via the Intent object):

```
//---look up the notification manager service---
```

```
NotificationManager nm = (NotificationManager)
```

```
getSystemService(NOTIFICATION_SERVICE);
```

```
//---cancel the notification that we started
```

```
nm.cancel(getIntent().getExtras().getInt("notificationID"));
```

3.4.USER INTERFACE

3.4.1.Views and viewgroups

- A view is a widget that has an appearance on screen.
- Examples of views are buttons, labels, and text boxes. A view derives from the base class `android.view.View`.
- One or more views can be grouped together into a ViewGroup. A ViewGroup (which is itself a special type of view) provides the layout in which you can order the appearance and sequence of views.
- Examples of ViewGroups include `LinearLayout` and `FrameLayout`. A ViewGroup derives from the base class `android.view.ViewGroup`.
- Android supports the following ViewGroups:
 1. `LinearLayout`
 2. `AbsoluteLayout`
 3. `TableLayout`
 4. `RelativeLayout`
 5. `FrameLayout`
 6. `ScrollView`

3.4.2.Layouts:

Linear layout

- The LinearLayout arranges views in a single column or a single row. Child views can be arranged either vertically or horizontally.
- To see how LinearLayout works, consider the following elements typically contained in the main.xml file:

```
<?xmlversion="1.0"encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>
<TextView
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/hello"
/>
</LinearLayout>
```

- In the main.xml file, observe that the root element is <LinearLayout> and it has a <TextView> element contained within it. The <LinearLayout> element controls the order in which the views contained within it appear.
- Each View and ViewGroup has a set of common attributes

Attribute	description
layout_width :	Specifies the width of the View or ViewGroup
layout_height :	Specifies the height of the View or ViewGroup
layout_margin :	Top Specifies extra space on the top side of the View or ViewGroup
layout_margin :	Bottom Specifies extra space on the bottom side of the View or ViewGroup
layout_margin :	Left Specifies extra space on the left side of the View or ViewGroup

layout_margin : Right Specifies extra space on the right side of the View or ViewGroup

- This will set the width of the view to be equal to the width of the text contained within it. Consider the following layout:

```
<?xmlversion="1.0"encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>
<TextView
android:layout_width="105dp"
android:layout_height="wrap_content"
android:text="@string/hello"
/>
<Button
android:layout_width="160dp"
android:layout_height="wrap_content"
android:text="Button"
/>
</LinearLayout>
```



- The preceding example also specifies that the orientation of the layout is vertical:

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>
```

- The default orientation layout is horizontal, so if you omit the android:orientation attribute, the views will appear as shown in



- In LinearLayout, you can apply the layout_weight and layout_gravity attributes to views contained within it, as the following modifications to main.xml show:

```
<?xmlversion="1.0"encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>
<TextView
android:layout_width="105dp"
android:layout_height="wrap_content"
android:text="@string/hello"
/>
```

```

<Button
    android:layout_width="160dp"
    android:layout_height="wrap_content"
    android:text="Button"
        android:layout_gravity="right"
        android:layout_weight="0.2"
    />

<EditText
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textSize="18sp"
    android:layout_weight="0.8"
    />

</LinearLayout>

```



Absolute layout

The `AbsoluteLayout` enables you to specify the exact location of its children. Consider the following UI defined in `main.xml` :

```

<?xmlversion="1.0"encoding="utf-8"?>

<AbsoluteLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android=http://schemas.android.com/apk/res/android
>

```

```

<Button
    android:layout_width="188dp"
    android:layout_height="wrap_content"
    android:text="Button"
    android:layout_x="126px"
    android:layout_y="361px"
/>

<Button
    android:layout_width="113dp"
    android:layout_height="wrap_content"
    android:text="Button"
    android:layout_x="12px"
    android:layout_y="361px"
/>

</AbsoluteLayout>

```



Table layout

- The `TableLayout` groups views into rows and columns. You use the `<TableRow>` element to designate a row in the table. Each row can contain one or more views.
- Each view you place within a row forms a cell. The width of each column is determined by the largest width of each cell in that column. Consider the content of `main.xml` shown here:

```

<?xmlversion="1.0" encoding="utf-8"?>

<TableLayout
    xmlns:android="http://schemas.android.com/apk/res/android"

```

```

    android:layout_height="fill_parent"
    android:layout_width="fill_parent"
    >
    <TableRow>
        <TextView
            android:text= "UserName:"
            android:width="120px"
        />
        <EditText
            android:id="@+id/txtUserName"
            android:width="200px"/>
    </TableRow>
    <TableRow>
        <TextView
            android:text= "Password:"
        />
        <EditText
            android:id="@+id/txtPassword"
            android:password="true"
        />
    </TableRow>
    <TableRow>
        <TextView/>
        <CheckBoxandroid:id="@+id/chkRememberPassword"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text= "RememberPassword"
        />
    </TableRow>
    <TableRow>
        <Button
            android:id="@+id/buttonSignIn"
            android:text= "LogIn" />

```

</TableRow>

</TableLayout>



Relative layout

The RelativeLayout enables you to specify how child views are positioned relative to each other. Consider the following main.xml file:

```
<?xmlversion= "1.0" encoding="utf-8"?>
<RelativeLayout
    android:id="@+id/RLLayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
>
<TextView
    android:id="@+id/lblComments"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text= "Comments"
        android:layout_alignParentTop= "true"
        android:layout_alignParentLeft="true"
/>
<EditText
    android:id="@+id/txtComments"
    android:layout_width="fill_parent"
    android:layout_height="170px"
    android:textSize="18sp"
```

```

        android:layout_alignLeft= "@+id/lblComments"
        android:layout_below="@+id/lblComments"
        android:layout_centerHorizontal="true"
    />
    <Button
        android:id="@+id/btnSave"
        android:layout_width="125px"
        android:layout_height="wrap_content"
        android:text= "Save"
            android:layout_below="@+id/txtComments"
            android:layout_alignRight="@+id/txtComments"
    />
    <Button
        android:id="@+id/btnCancel"
        android:layout_width="124px"
        android:layout_height="wrap_content"
        android:text= "Cancel"
            android:layout_below="@+id/txtComments"
            android:layout_alignLeft= "@+id/txtComments"
    />
</RelativeLayout>

```

Notice that each view embedded within the RelativeLayout has attributes that enable it to align with another view. These attributes are as follows:

```

layout_align:    ParentTop
layout_align:    ParentLeft
layout_align:    Left
layout_align:    Right
layout_center:   Horizontal

```

The value for each of these attributes is the ID for the view that you are referencing. The preceding XML UI creates the screen shown below



Frame layout

The `FrameLayout` is a placeholder on screen that you can use to display a single view. Views that you add to a `FrameLayout` are always anchored to the top left of the layout. Consider the following content in `main.xml` :

```
<?xmlversion="1.0"encoding="utf-8"?>
<RelativeLayout
    android:id="@+id/RLLayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
>
    <TextView
        android:id="@+id/lblComments"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Thisismylovelydog,Ookii"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
    />
    <FrameLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/lblComments"
        android:layout_below="@+id/lblComments"
        android:layout_centerHorizontal="true"
```

```

>
<ImageView
android:src="@drawable/ookii"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
/>
</FrameLayout>
</RelativeLayout>

```

Here, you have a FrameLayout within a RelativeLayout. Within the FrameLayout, you embed an ImageView.

```

<?xmlversion="1.0"encoding="utf-8"?>
<RelativeLayout
android:id="@+id/RLayout"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
xmlns:android="http://schemas.android.com/apk/res/android"
>
<TextView
android:id="@+id/lblComments"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Thisismylovelydog,Ookii"
android:layout_alignParentTop="true"
android:layout_alignParentLeft="true"
/>
<FrameLayout
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignLeft="@+id/lblComments"
android:layout_below="@+id/lblComments"
android:layout_centerHorizontal="true"
>

```

```

<ImageView
    android:src="@drawable/ookii"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
/>

<Button
    android:layout_width="124dp"
    android:layout_height="wrap_content"
    android:text="Print Picture"
/>

</FrameLayout>
</RelativeLayout>

```



Scroll view

A `ScrollView` is a special type of `FrameLayout` in that it enables users to scroll through a list of views that occupy more space than the physical display. The `ScrollView` can contain only one child view or `ViewGroup`, which normally is a `LinearLayout`.

The following `main.xml` content shows a `ScrollView` containing a `LinearLayout`, which in turn contains some `Button` and `EditText` views:

```

<?xmlversion="1.0"encoding="utf-8"?>

<ScrollView
    android:layout_width="fill_parent"

```

```

    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
>
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
>
    <Button
        android:id="@+id/button1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text= "Button1"
    />
    <Button
        android:id="@+id/button2"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text= "Button2"
    />
    <Button
        android:id="@+id/button3"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text= "Button3"
    />
    <EditText
        android:id="@+id/txt"
        android:layout_width="fill_parent"
        android:layout_height="300px"
    />
    <Button
        android:id="@+id/button4"

```

```

    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text= "Button4"
/>

<Button
    android:id="@+id/button5"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text= "Button5"
/>

</LinearLayout>

</ScrollView>

```



3.4.3.Display orientation

- One of the key features of modern smartphones is their ability to switch screen orientation, and Android is no exception. Android supports two screen orientations: portrait and landscape .
- By default, when you change the display orientation of your Android device, the current activity that is displayed will automatically redraw its content in the new orientation. This is because the onCreate() event of the activity is fired whenever there is a change in display orientation.



In general, you can employ two techniques to handle changes in screen orientation:

Anchoring — The easiest way is to “anchor” your views to the four edges of the screen.

When the screen orientation changes, the views can anchor neatly to the edges.

Resizing and repositioning — Whereas anchoring and centralizing are simple techniques to ensure that views can handle changes in screen orientation, the ultimate technique is resizing each and every view according to the current screen orientation.

Anchoring v i e w s

Anchoring could be easily achieved by using `RelativeLayout`. Consider the following `main.xml` containing five `Button` views embedded within the `<RelativeLayout>` element:

```
<?xmlversion="1.0"encoding="utf-8"?>
<RelativeLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
>
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TopLeftButton"
            android:layout_alignParentLeft="true"
            android:layout_alignParentTop="true"
    />
    <Button
        android:id="@+id/button2"
```

```

    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text= "TopRightButton"
        android:layout_alignParentTop= "true"
        android:layout_alignParentRight="true"
    />
<Button
    android:id="@+id/button3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text= "BottomLeftButton"
        android:layout_alignParentLeft="true"
        android:layout_alignParentBottom="true"
    />
<Button
    android:id="@+id/button4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text= "BottomRightButton"
        android:layout_alignParentRight="true"
        android:layout_alignParentBottom="true"
    />
<Button
    android:id="@+id/button5"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text= "MiddleButton"
        android:layout_centerVertical= "true"
        android:layout_centerHorizontal="true"
    />
</RelativeLayout>

```

Observe the following attributes found in the various Button views:

- **layout_alignParentLeft** — Aligns the view to the left of the parent view
- **layout_alignParentRight** — Aligns the view to the right of the parent view
- **layout_alignParentTop** — Aligns the view to the top of the parent view
- **layout_alignParentBottom** — Aligns the view to the bottom of the parent view
- **layout_centerVertical** — Centers the view vertically within its parent view
- **layout_centerHorizontal** — Centers the view horizontally within its parent view



Resizing and repositioning

- Apart from anchoring your views to the four edges of the screen, an easier way to customize the UI based on screen orientation is to create a separate `res/layout` folder containing the XML files for the UI of each orientation.
- To support landscape mode, you can create a new folder in the `res` folder and name it as `layout-land` (representing landscape).

The following shows the content of `main.xml` under the `layout` folder:

```
<?xmlversion= "1.0" encoding="utf-8"?>

<RelativeLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
>
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text= "TopLeftButton"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
    />
    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text= "TopRightButton"
        android:layout_alignParentTop="true"
        android:layout_alignParentRight="true"
    />
    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```

        android:text= "BottomLeftButton"
        android:layout_alignParentLeft="true"
        android:layout_alignParentBottom= "true"
    />
    <Button
        android:id="@+id/button4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text= "BottomRightButton"
        android:layout_alignParentRight="true"
        android:layout_alignParentBottom= "true"
    />
    <Button
        android:id="@+id/button5"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text= "MiddleButton"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"
    />
</RelativeLayout>

```

The following shows the content of main.xml under the layout-land folder are the additional views to display in landscape mode):

```

<?xmlversion= "1.0" encoding="utf-8"?>
<RelativeLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
>
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

```

```

    android:text= "TopLeftButton"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
/>
<Button
    android:id="@+id/button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text= "TopRightButton"
    android:layout_alignParentTop="true"
    android:layout_alignParentRight="true"
/>
<Button
    android:id="@+id/button3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text= "BottomLeftButton"
    android:layout_alignParentLeft="true"
    android:layout_alignParentBottom="true"
/>
<Button
    android:id="@+id/button4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text= "BottomRightButton"
    android:layout_alignParentRight="true"
    android:layout_alignParentBottom="true"
/>
<Button
    android:id="@+id/button5"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text= "MiddleButton"

```

```

    android:layout_centerVertical="true"
    android:layout_centerHorizontal="true"
/>

<Button
    android:id="@+id/button6"
    android:layout_width="180px"
    android:layout_height="wrap_content"
    android:text="Top Middle Button"
    android:layout_centerVertical="true"
    android:layout_centerHorizontal="true"
    android:layout_alignParentTop="true"
/>

<Button
    android:id="@+id/button7"
    android:layout_width="180px"
    android:layout_height="wrap_content"
    android:text="Bottom Middle Button"
    android:layout_centerVertical="true"
    android:layout_centerHorizontal="true"
    android:layout_alignParentBottom="true"
/>

</RelativeLayout>

```



3.4.4` Action Bar

The action bar is a dedicated bar at the top of each screen that is generally persistent throughout the app. It provides you several key function which are as following:

- Makes important actions prominent and accessible
- Supports consistent navigation and view switching within apps
- Reduces clutter by providing an action overflow for rarely used actions
- Provides a dedicated space for giving your app an identity

Action Bar Components

Action Bar has four major components which can be seen in the following image.



Sr.No	Action Bar Components
1	App Icon The app icon establishes your app's identity. It can be replaced with a different logo or branding if you wish.
2	View control If your app displays data in different views, this segment of the action bar allows users to switch views.
3	Action buttons Shows the most important actions of your app in the actions section.
4	Action overflow Moves less often used actions to the action overflow.

3.4.5.Listening for UI notifications

Users interact with your UI at two levels: the activity level and the views level. At the activity level, the Activity class exposes methods that you can override. Some common methods that you can override in your activities include the following:

- onKeyDown — Called when a key was pressed and not handled by any of the views contained within the activity
- onKeyUp — Called when a key was released and not handled by any of the views contained within the activity

- `onMenuItemSelected` — Called when a panel's menu item has been selected by the user
- `onMenuOpened` — Called when a panel's menu is opened by the user overriding methods defined in an Activity

Example:

- 1 . Using Eclipse, create a new Android project and name it UIActivity.
- 2 . Add the following statements :

```

<?xmlversion= "1.0" encoding="utf-8"?>
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    xmlns:android="http://schemas.android.com/apk/res/android"
>
    <TextView
        android:layout_width="214dp"
        android:layout_height="wrap_content"
        android:text="Your Name"
    />
    <EditText
        android:id="@+id/txt1"
        android:layout_width="214dp"
        android:layout_height="wrap_content"
    />
    <Button
        android:id="@+id/btn1"
        android:layout_width="106dp"
        android:layout_height="wrap_content"
        android:text="OK"
    />
    <Button
        android:id="@+id/btn2"
        android:layout_width="106dp"

```

```

        android:layout_height="wrap_content"
        android:text="Cancel"

    />
</LinearLayout>

```

- 3 . Add the following statements to the MainActivity.java file:

```

package net.learn2develop.UIActivity;
import android.app.Activity;
import android.os.Bundle;
import android.view.KeyEvent;
import android.widget.Toast;
public class MainActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event)
    {
        switch (keyCode)
        {
            case KeyEvent.KEYCODE_DPAD_CENTER:
                Toast.makeText (getBaseContext(),
                    "Center was clicked",
                    Toast.LENGTH_LONG).show();
                break;
            case KeyEvent.KEYCODE_DPAD_LEFT:
                Toast.makeText (getBaseContext(),
                    "Left arrow was clicked",
                    Toast.LENGTH_LONG).show();
                break;
            case KeyEvent.KEYCODE_DPAD_RIGHT :

```

```

        Toast.makeText (getBaseContext(),
            "Right arrow was clicked",
            Toast.LENGTH_LONG).show();
        break;
    case KeyEvent.KEYCODE_DPAD_UP:
        Toast.makeText (getBaseContext(),
            "Up arrow was clicked",
            Toast.LENGTH_LONG).show();
        break;
    case KeyEvent.KEYCODE_DPAD_DOWN:
        Toast.makeText (getBaseContext(),
            "Down arrow was clicked",
            Toast.LENGTH_LONG).show();
        break;
    }
    return false;
}
}

```

- 4 . Press F11 to debug the application on the Android Emulator.
- 5 . When the activity is loaded, type some text into it. Next, click the down arrow key on the directional pad. Observe the message shown on the screen



Registering events for views

- Views can fire events when users interact with them. For example, when a user touches a Button view, you need to service the event so that the appropriate action can be performed.
- Using the same example discussed in the previous section, recall that the activity has two Button views; therefore, you can register the button click events using an anonymous class as shown here:

```
packagenet.learn2develop.UIActivity;
importandroid.app.Activity;
importandroid.os.Bundle;
importandroid.view.KeyEvent;
importandroid.view.View;
importandroid.widget.Toast;
import android.view.View.OnClickListener;
import android.widget.Button;
publicclassMainActivityextendsActivity {
/** Calledwhenthe activity isfirstcreated. */
@Override
publicvoidonCreate(Bundle savedInstanceState){
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
    //---the two buttons are wired to the same event handler---
    Button btn1 = (Button)findViewById(R.id. btn1);
    btn1.setOnClickListener(btnListener);
    Button btn2 = (Button)findViewById(R.id. btn2);
    btn2.setOnClickListener(btnListener);
}
//---create an anonymous class to act as a button click listene r---
private OnClickListener btnListener = new OnClickListener()
{
    public void onClick(View v)
    {
        Toast. makeText (getBaseContext(),
```

```

        ((Button) v).getText() + " was clicked",
        Toast.LENGTH_LONG).show();
    }
};

@Override
public boolean onKeyDown(int keyCode, KeyEvent event)
{
    switch(keyCode)
    {
        //...
        //...
    }
    return false;
}
}

```

If you now press either the OK button or the Cancel button, the appropriate message will be displayed .proving that the event is wired up properly.



Besides defining an anonymous class for the event handler, you can also define an anonymous inner class to handle an event. The following example shows how you can handle the `onFocusChange()`

event for the `EditText` view:

```

import android.widget.EditText;

public class MainActivity extends Activity {
    /** Called when the activity is first created. */

```

@Override

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.main);  
    //---the two buttons are wired to the same event handler---  
    Button btn1 = (Button) findViewById(R.id.btn1);  
    btn1.setOnClickListener(btnListener);  
    Button btn2 = (Button) findViewById(R.id.btn2);  
    btn2.setOnClickListener(btnListener);
```

```
    EditText txt1 = (EditText) findViewById(R.id.txt1);  
    //---create an anonymous inner class to act as an onFocus listener---  
    txt1.setOnFocusChangeListener( new View.OnFocusChangeListener()   
    {  
        @Override  
        public void onFocusChange(View v, boolean hasFocus) {  
            Toast.makeText (getBaseContext(),  
                ((EditText) v).getId() + " has focus - " + hasFocus,  
                Toast.LENGTH_LONG).show();  
        }  
    });  
}
```



Review Questions

PART-A(2 Mark Questions)

1. What is an Android?
2. Mention any 4 android versions.
3. List any 4 features of Android.
4. What is Dalvik Virtual Machine?
5. What is ADT?
6. What is AVD?
7. What is an activity?
8. What is an Intent?
9. What do you mean by linking activities?
10. What do you mean by fragment?
11. Define view.
12. Define Viewgroup.
13. What is a layout?
14. What are the orientations provided by the activity?
15. Define notification.
16. What do you mean by an ActionBar?

PART-B(3 Mark Questions)

1. Explain any 3 features of Android.
2. Briefly explain Android SDK.
3. How will you create AVD?
4. What are the types of intent? Briefly explain them.
5. Draw flowchart for Android fragment lifestyle.
6. What are the types of fragments? Explain.
7. Explain view.
8. Explain view group.
9. List any three layout attributes and explain them.
10. Briefly explain Action bar.

PART-C(5&10 Mark Questions))

1. Explain Architecture of Android?(10)
2. Explain Dalvik Virtual Machine(5)
3. Explain ADT(5)

4. With activity lifecycle flowchart ,explain activities in detail.(10)
5. Explain intents in detail.(10)
6. Explain various types of Layouts(5)
7. Explain display Orientation in detail(10)
8. Explain display notifications(5)
9. Explain listening for UI notifications(5)

UNIT-IV

VIEWS

OBJECTIVES

At the end of the unit, students can

- **Explain Basic views in Android**
- **Explain Advanced views in Android**
- **Explain Displaying Pictures and menus with views**
- **Explain how to send SMS**
- **Explain how to make Phonecall**

4.1 Basic views

Basic Views in Android applications:

- TextView
- EditText
- Button
- ImageButton
- CheckBox
- ToggleButton
- RadioButton
- RadioGroup

These basic views enable you to display text information, as well as perform some basic selection.

4.1.1 Text view

When you create a new Android project, Eclipse always creates the main.xml file (located in the res/layout folder), which contains a <TextView> element:

```
<?xmlversion="1.0"encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:layout_width="fill_parent"
```

```

        android:layout_height="wrap_content"
        android:text= "@string/hello"
    />
</LinearLayout>

```

The `TextView` view is used to display text to the user.

4.1.2 Button, ImageButton, editText, checkBox, toggleButton, radioButton, and radiogroup views

Some Basic UI Controls are Button, ImageButton, EditText , CheckBox , ToggleButton, RadioButton, and RadioGroup

- Button — Represents a push-button widget
- ImageButton — Similar to the Button view, except that it also displays an image
- EditText — A subclass of the TextView view, except that it allows users to edit its text content
- CheckBox — A special type of button that has two states: checked or unchecked
- RadioGroup and RadioButton — The RadioButton has two states: either checked or unchecked.
- Once a RadioButton is checked, it cannot be unchecked. A RadioGroup is used to group together one or more RadioButton views, thereby allowing only one RadioButton to be checked within the RadioGroup.
- ToggleButton — Displays checked/unchecked states using a light indicator

Example:

- 1 . Using Eclipse, create an Android project and name it
- 2 . Modify the `main.xml` file located in the `res/layout` folder by adding the following elements

```

<?xmlversion="1.0"encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <Button android:id="@+id/btnSave"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text= "Save" />

```

```

<Button android:id="@+id/btnOpen"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Open" />
<ImageButton android:id="@+id/btnImg1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:src="@drawable/icon" />
<EditText android:id="@+id/txtName"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
<CheckBox android:id="@+id/chkAutosave"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Autosave" />
<CheckBox android:id="@+id/star"
    style="?android:attr/starStyle"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
<RadioGroup android:id="@+id/rdbGp1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical" >
    <RadioButton android:id="@+id/rdb1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Option 1" />
    <RadioButton android:id="@+id/rdb2"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Option 2" />
</RadioGroup>
<ToggleButton android:id="@+id/toggle1"
    android:layout_width="wrap_content"

```

```
android:layout_height="wrap_content" />
```

```
</LinearLayout>
```

3 . To see the views in action, debug the project in Eclipse by selecting the project name and pressing F11



FIGURE 4-1



FIGURE 4-2

4 . Click on the various views and note how they vary in their look and feel.

- The first CheckBox view (Autosave) is checked.
- The second CheckBox View (star) is checked.
- The second RadioButton (Option 2) is selected.
- The ToggleButton is turned on.



FIGURE 4-3

Working Principle:

1. For the first Button, the `layout_width` attribute is set to `fill_parent` so that its width occupies the entire width of the screen:

```
<Buttonandroid:id="@+id/btnSave"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Save"/>
```

2. For the second Button, the `layout_width` attribute is set to `wrap_content` so that its width will be the width of its content — specifically, the text that it is displaying (i.e., “Open”):

```
<Buttonandroid:id="@+id/btnOpen"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Open"/>
```

3. The `ImageButton` displays a button with an image. The image is set through the `src` attribute. In this case, you simply use the image used for the application icon:

```
<ImageButtonandroid:id="@+id/btnImg1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:src="@drawable/icon"/>
```

4. The `EditText` view displays a rectangular region where the user can enter some text. You set the `layout_height` to `wrap_content` so that if the user enters a long string of text, its height will automatically be adjusted to fit the content

```

<EditText android:id="@+id/txtName"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"/>

```

5.The `CheckBox` displays a checkbox that users can tap to check or uncheck it:

```

<CheckBox android:id="@+id/chkAutosave"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Autosave"/>

```

6.If you do not like the default look of the `CheckBox` , you can apply a style attribute to it to display it as some other image, such as a star:

```

<CheckBox android:id="@+id/star"
    style="?android:attr/starStyle"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>

```

7.The `RadioGroup` encloses two `RadioButtons`. This is important because radio buttons are usually used to present multiple options to the user for selection. When a `RadioButton` in a `RadioGroup` is selected, all other `RadioButtons` are automatically unselected:

```

<RadioGroup android:id="@+id/rdbGp1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <RadioButton android:id="@+id/rdb1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Option 1"/>
    <RadioButton android:id="@+id/rdb2"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Option 2"/>
</RadioGroup>

```

8.The `ToggleButton` displays a rectangular button that users can toggle on and off by clicking it:

```

<ToggleButton android:id="@+id/toggle1"
    android:layout_width="wrap_content"

```

android:layout_height="wrap_content"/>

4.1.3. Progressbar view

- The ProgressBar view provides visual feedback of some ongoing tasks, such as when you are performing a task in the background.
- For example, you might be downloading some data from the Web and need to update the user about the status of the download.
- In this case, the ProgressBar view is a good choice for this task.

Example

- 1 . Using Eclipse, create an Android project and name it as BasicViews2.
- 2 . Modify the main.xml file located in the res/layout folder by adding the following code

```
<?xmlversion= "1.0"encoding="utf-8"?>
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
android:orientation="vertical"
```

```
android:layout_width="fill_parent"
```

```
android:layout_height="fill_parent" >
```

```
<ProgressBar android:id="@+id/progressbar"
```

```
android:layout_width="wrap_content"
```

```
android:layout_height="wrap_content" />
```

```
</LinearLayout>
```

- 3 . In the MainActivity.java file, add the following statements:

```
packagenet.learn2develop.BasicViews2;
```

```
import android.app.Activity;
```

```
import android.os.Bundle;
```

```
import android.os.Handler;
```

```
import android.widget.ProgressBar;
```

```
public class MainActivity extends Activity {
```

```
private static int progress ;
```

```
private ProgressBar progressBar;
```

```
private int progressStatus = 0;
```

```
private Handler handler = new Handler();
```

```
/** Called when the activity is first created. */
```

```
@Override
```

```
public void onCreate(Bundle savedInstanceState){
```

```

super.onCreate(savedInstanceState);
setContentView(R.layout.main);
progress = 0;
progressBar = (ProgressBar) findViewById(R.id.progressbar);
//---do some work in background thread---
new Thread( new Runnable()
{
public void run()
{
//---do some work here---
while (progressStatus < 10)
{
progressStatus = doSomeWork();
}
//---hides the progress bar---
handler.post(new Runnable()
{
public void run()
{
//---0 - VISIBLE; 4 - INVISIBLE; 8 - GONE---
progressBar.setVisibility(8);
}
});
}
//---do some long lasting work here---
private int doSomeWork()
{
try {
//---simulate doing some work---
Thread.sleep(500);
} catch (InterruptedException e)
{
e.printStackTrace();
}
}

```

```

return ++ progress ;
}
}).start();
}
}

```

4 . Press F11 to debug the project on the Android Emulator. after about five seconds, it will disappear.



FIGURE 4-6

4.1.4 Autocomplete textview

The `AutoCompleteTextView` is a view that is similar to `EditText` (in fact it is a subclass of `EditText`), except that it shows a list of completion suggestions automatically while the user is typing.

Example:

- 1 . Using Eclipse, create an Android project and name it BasicViews3.
- 2 . Modify the `main.xml` file located in the `res/layout` folder

```

<?xmlversion= "1.0"encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text= "Name of President"/>
    <AutoCompleteTextView android:id="@+id/txtCountries"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />
</LinearLayout>

```

3 . Add the following statements to the MainActivity.java file:

```
packagenet.learn2develop.BasicViews3;  
importandroid.app.Activity;  
importandroid.os.Bundle;  
import android.widget.AdapterView;  
import android.widget.AutoCompleteTextView;  
publicclassMainActivityextendsActivity {  
String[] presidents = {  
    “Dwight D. Eisenhower”,  
    “John F. Kennedy”,  
    “Lyndon B. Johnson”,  
    “Richard Nixon”,  
    “Gerald Ford” ,  
    “Jimmy Carter”,  
    “Ronald Reagan”,  
    “George H. W. Bush”,  
    “Bill Clinton”,  
    “George W. Bush”,  
    “Barack Obama”  
};  
  
/** Called when the activity is firstcreated. */  
@Override  
publicvoidonCreate(Bundle savedInstanceState){  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.main);  
    ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,  
    android.R.layout.simple_dropdown_item_1line, presidents);  
    AutoCompleteTextView textView = (AutoCompleteTextView)  
    findViewById(R.id. txtCountries);  
    textView.setThreshold(3);  
    textView.setAdapter(adapter);  
}  
}
```

4 . press F11 to debug the application on the Android Emulator.



FIGURE 4-8

Working Principle:

In the MainActivity class, you first create a String array containing a list of presidents' names:

```
String[] presidents={  
    "DwightD.Eisenhower",  
    "JohnF.Kennedy" ,  
    "LyndonB.Johnson" ,  
    "Richard Nixon",  
    "GeraldFord",  
    "JimmyCarter",  
    "RonaldReagan",  
    "GeorgeH.W.Bush",  
    "BillClinton" ,  
    "GeorgeW.Bush",  
    "BarackObama"  
};
```

- The ArrayAdapter object manages the array of strings that will be displayed by the autoCompleteTextView.
- In the preceding example, you set the autoCompleteTextView to display in the simple_dropdown_item_1line mode:
- The setThreshold() method sets the minimum number of characters the user must type before the suggestions appear as a drop-down menu:

```
textView.setThreshold(3);
```

- The list of suggestions to display for the `AutoCompleteTextView` is obtained from the `ArrayAdapter` object:

```
textView.setAdapter(adapter);
```

4.2. Advanced Views

4.2.1. Time picker view

The Time Picker view enables users to select a time of the day, in either 24-hour mode or AM/PM mode.

Example:

- 1 . Using Eclipse, create an Android project and name it BasicViews4.
- 2 . Modify the `main.xml` file located in the `res/layout` folder

```
<?xmlversion= "1.0"encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
android:orientation="vertical"  
android:layout_width="fill_parent"  
android:layout_height="fill_parent" >  
<TimePicker android:id="@+id/timePicker"  
android:layout_width="wrap_content"  
android:layout_height="wrap_content" />  
<Button android:id="@+id/btnSet"  
android:layout_width="wrap_content"  
android:layout_height="wrap_content"  
android:text= "I am all set!" />  
</LinearLayout>
```

- 3 . Press F11 to debug the application on the Android Emulator.

Besides clicking on the plus (+) and minus (-) buttons, you can use the numeric keypad on the device to change the hour and minute, and click the AM button to toggle between AM and PM.



FIGURE 4-9

4 . Back in Eclipse, add the following statements to the MainActivity.java file:

```
packagenet.learn2develop.BasicViews4;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TimePicker;
import android.widget.Toast;
public class MainActivity extends Activity {
    TimePicker timePicker;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        timePicker = (TimePicker) findViewById(R.id.timePicker);
        timePicker.setIs24HourView(true);
        //---Button view---
        Button btnOpen = (Button) findViewById(R.id.btnSet);
        btnOpen.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                Toast.makeText (getBaseContext(),
```

```

“Time selected:” +
timePicker.getCurrentHour() +
“:” + timePicker.getCurrentMinute(),
Toast.LENGTH_SHORT).show();
}
});
}
}

```

5 . Press F11 to debug the application on the Android Emulator. This time, the TimePicker will be displayed in the 24-hour format.



FIGURE 4-10

Working Principle:

- The TimePicker displays a standard UI to enable users to set a time. By default, it displays the time in the AM/PM format. If you wish to display the time in the 24-hour format, you can use the `setIs24HourView()` method.
- To programmatically get the time set by the user, use the `getCurrentHour()` and `getCurrentMinute()` methods:

```

Toast.makeText (getBaseContext(),
“Timeselcted:”+
timePicker.getCurrentHour() +
“:” +timePicker.getCurrentMinute(),
Toast.LENGTH_SHORT).show();

```

4.2.2.Datepicker view

Another view that is similar to the TimePicker is the DatePicker. Using the DatePicker, we can enable users to select a particular date on the activity.

Example:

1 . Using the same project created in the previous Try It Out, modify the main.xml file as shown here:

```
<?xmlversion= "1.0"encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <DatePicker android:id="@+id/datePicker"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <TimePicker android:id="@+id/timePicker"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
    <Button android:id="@+id/btnSet"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text= "I am all set!"/>
</LinearLayout>
```

2 . Press F11 to debug the application on the Android Emulator.



FIGURE 4-12

3 . Back in Eclipse, add in the following statements to the MainActivity.java file:

```
packagenet.learn2develop.BasicViews4;
import android.app.Activity;
```

```

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
import android.app.Dialog;
import android.app.TimePickerDialog;
import android.widget.TimePicker;
import android.widget.DatePicker;
public class MainActivity extends Activity {
    TimePicker timePicker;
    DatePicker datePicker;
    int hour, minute;
    static final int TIME_DIALOG_ID = 0;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //showDialog(TIME_DIALOG_ID);
        timePicker = (TimePicker) findViewById(R.id.timePicker);
        timePicker.setIs24HourView(true);

        datePicker = (DatePicker) findViewById(R.id.datePicker);
        //---Button view---
        Button btnOpen = (Button) findViewById(R.id.btnSet);
        btnOpen.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                Toast.makeText(getApplicationContext(),
                    "Date selected:" + datePicker.getMonth() + 1 +
                    "/" + datePicker.getDayOfMonth() +
                    "/" + datePicker.getYear() + "\n" +
                    "Time selected:" + timePicker.getCurrentHour() +
                    ":" + timePicker.getCurrentMinute(),
                    Toast.LENGTH_SHORT).show();
            }
        });
    }
}

```

```

    }
    });
}
@Override
protected Dialog onCreateDialog(int id)
{
    switch(id){
    case TIME_DIALOG_ID:
        return new TimePickerDialog(
            this, mTimeSetListener, hour, minute, false);
    }
    return null;
}

private TimePickerDialog.OnTimeSetListener mTimeSetListener =
    new TimePickerDialog.OnTimeSetListener()
    {
        public void onTimeSet(
            TimePicker view, int hourOfDay, int minuteOfHour)
        {
            hour = hourOfDay;
            minute = minuteOfHour;
            Toast.makeText (getBaseContext(),
                "You have selected : "+hour+":" +minute,
                Toast.LENGTH_SHORT).show();
        }
    };
}

```

4 . Press F11 to debug the application on the Android Emulator. Once the date is set, clicking the Button will display the date set .



FIGURE 4-13

Working Principle:

Like the TimePicker, you call the `getMonth()`, `getDayOfMonth()`, and `getYear()` methods to get the month, day, and year, respectively:

**“Dateselectd:”+datePicker.getMonth()+1+“/” +datePicker.getDayOfMonth()+“/” +
datePicker.getYear()+“\n”+**

4.2.3.List View

- List views are views that enable you to display a long list of items.
- In Android, there are two types of list views: `ListView` and `SpinnerView` . Both are useful for displaying long lists of items

Example:

1. Using Eclipse, create an Android project and name it BasicView5.
- 2 . Modify the MainActivity.java file by inserting the statements

```
packagenet.learn2develop.BasicViews5;  
import android.app.Activity;  
import android.os.Bundle;  
import android.app.ListActivity;  
import android.view.View;  
import android.widget.AdapterView;  
import android.widget.ListView;
```

```

import android.widget.Toast;
public class MainActivity extends ListActivity{
String[] presidents = {
    "Dwight D. Eisenhower",
    "John F. Kennedy",
    "Lyndon B. Johnson",
    "Richard Nixon",
    "Gerald Ford" ,
    "Jimmy Carter",
    "Ronald Reagan",
    "George H. W. Bush",
    "Bill Clinton",
    "George W. Bush",
    "Barack Obama"
};
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState){
    super.onCreate(savedInstanceState);
    //setContentView(R.layout.main);
    setListAdapter(new ArrayAdapter<String>(this,
    android.R.layout.simple_list_item_1 , presidents));
}
public void onItemClick(
    ListView parent, View v, int position, long id)
{
    Toast.makeText (this,
    "You have selected " + presidents[position],
    Toast.LENGTH_SHORT).show();
}
}

```

3 . Press F11 to debug the application on the Android Emulator.



FIGURE 4-15

4 . Click on an item. A message containing the item selected will be displayed.

Working Principle:

- The first thing to notice in this example is that the `MainActivity` class extends the `ListActivity` class. The `ListActivity` class extends the `Activity` class and it displays a list of items by binding to a data source. Also, note that there is no need to modify the `main.xml` file to include the `ListView` ; the `ListActivity` class itself contains a `ListView` . Hence, in the `onCreate()` method, there is no need to call the `setContentView()` method to load the UI from the `main.xml` file:

```
//---noneedtocallthis---
```

```
//setContentView(R.layout.main);
```

- In the `onCreate()` method, you use the `setListAdapter()` method to programmatically fill the entire screen of the activity with a `ListView` .
- The `ArrayAdapter` object manages the array of strings that will be displayed by the `ListView` . In the preceding example, you set the `ListView` to display in the `simple_list_item_1` mode:

```
setListAdapter(new ArrayAdapter<String>(this,  
    android.R.layout.simple_list_item_1 ,presidents));
```

The `onListItemClick()` method is fired whenever an item in the `ListView` has been clicked:

```
publicvoidonListItemClick(  
    ListView parent,Viewv,int position,longid)  
{  
    Toast.makeText (this,
```

```

        "You have selected "+presidents[position],
        Toast.LENGTH_SHORT).show();
    }

```

4.2.4. ImageView

In Android, you can use "[android.widget.ImageView](#)" class to display an image file. Image file is easy to use but hard to master, because of the various screen and dpi in Android devices.

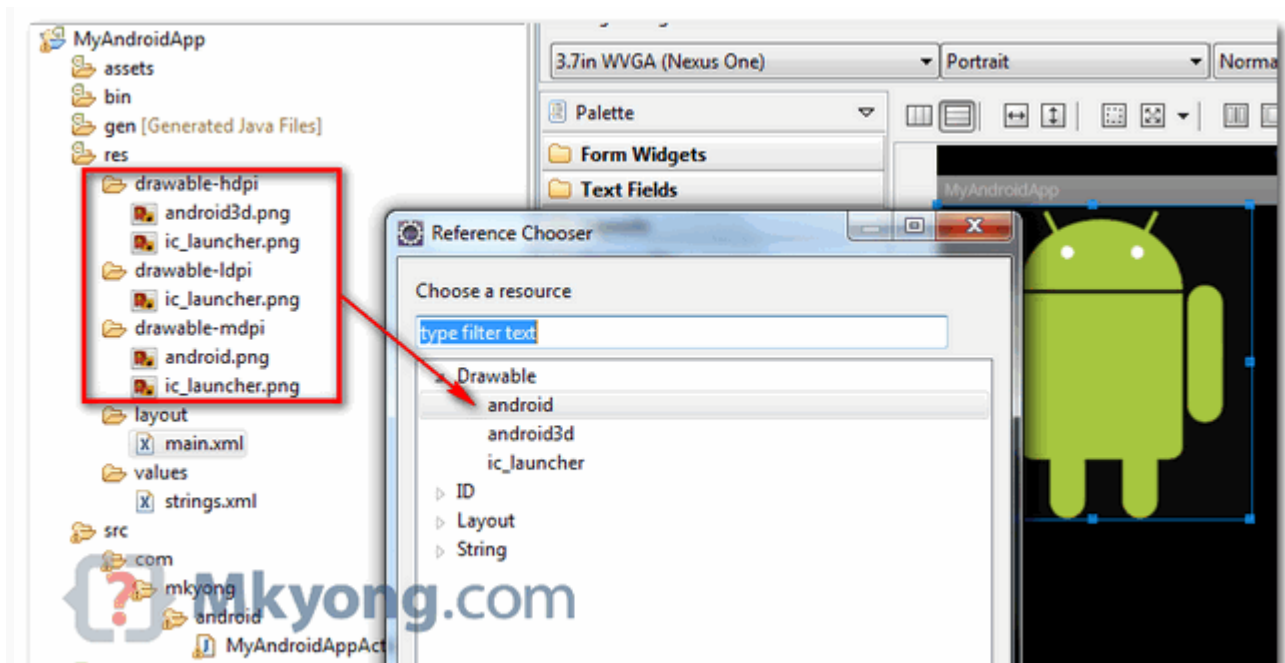
Note

Please refer to this official Android's "[Drawable Resource](#)" and "[Screen Support](#)" article for better understand of how image works in Android.

1. Add Image to Resources

Put your images into folder "res/drawable-ldpi", "res/drawable-mdpi" or "res/drawable-hdpi".

See figure below, no matter which folder you put, Android will find your image automatically. In this case, both "android.png" and "android3d.png" images are used for demonstration.



Note

Again, read official Android's "[Drawable Resource](#)" and "[Screen Support](#)" article to understand what is dpi and resources in Android.

2. Add ImageView

Open "**res/layout/main.xml**" file, just add an `ImageView` and `Button` for demonstration. By default, `imageView1` will display "android.png".

File : res/layout/main.xml

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/android" />

    <Button
        android:id="@+id/btnChangeImage"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Change Image" />

</LinearLayout>

```

3. Code

Simple, when button is clicked, change it to “android3d.png”.

File : MyAndroidAppActivity.java

```

package com.mkyong.android;

import android.app.Activity;
import android.os.Bundle;
import android.widget.Button;
import android.widget.ImageView;
import android.view.View;
import android.view.View.OnClickListener;

public class MyAndroidAppActivity extends Activity {

    Button button;
    ImageView image;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        addListenerOnButton();
    }

    public void addListenerOnButton() {

```

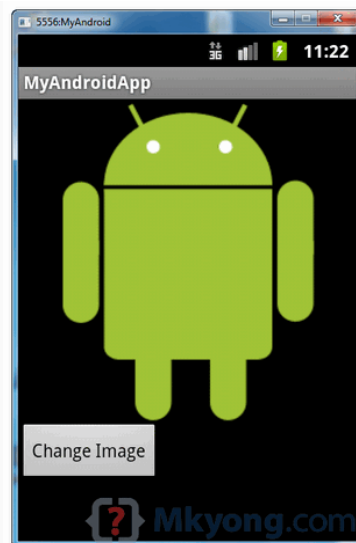
```
image = (ImageView) findViewById(R.id.imageView1);
```

```
button = (Button) findViewById(R.id.btnChangeImage);  
button.setOnClickListener(new OnClickListener() {
```

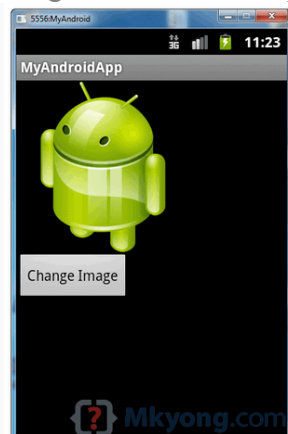
```
    @Override  
    public void onClick(View arg0) {  
        image.setImageResource(R.drawable.android3d);  
    }  
});
```

```
    }  
}
```

1. Result, “android.png” is displayed.



2. Click on the button, image will changed to “android3d.png”.



4.2.5.Menus

Menus are a common user interface component in many types of applications. To provide a familiar and consistent user experience, you should use the Menu APIs to present user actions and other options in your activities.

Options menu and app bar

The options menu is the primary collection of menu items for an activity. It's where you should place actions that have a global impact on the app, such as "Search," "Compose email," and "Settings."

Context menu and contextual action mode

A context menu is a floating menu that appears when the user performs a long-click on an element. It provides actions that affect the selected content or context frame.

The contextual action mode displays action items that affect the selected content in a bar at the top of the screen and allows the user to select multiple items.

Popup menu

A popup menu displays a list of items in a vertical list that's anchored to the view that invoked the menu. It's good for providing an overflow of actions that relate to specific content or to provide options for a second part of a command. Actions in a popup menu should **not** directly affect the corresponding content—that's what contextual actions are for. Rather, the popup menu is for extended actions that relate to regions of content in your activity.

Defining a Menu in XML

For all menu types, Android provides a standard XML format to define menu items. Instead of building a menu in your activity's code, you should define a menu and all its items in an XML menu resource. You can then inflate the menu resource (load it as a Menu object) in your activity or fragment.

Using a menu resource is a good practice for a few reasons:

- It's easier to visualize the menu structure in XML.
- It separates the content for the menu from your application's behavioral code.
- It allows you to create alternative menu configurations for different platform versions, screen sizes, and other configurations by leveraging the app resources framework.

To define the menu, create an XML file inside your project's `res/menu/` directory and build the menu with the following elements:

```
<menu>
```

Defines a Menu, which is a container for menu items. A `<menu>` element must be the root node for the file and can hold one or more `<item>` and `<group>` elements.

`<item>`

Creates a MenuItem, which represents a single item in a menu. This element may contain a nested `<menu>` element in order to create a submenu.

`<group>`

An optional, invisible container for `<item>` elements. It allows you to categorize menu items so they share properties such as active state and visibility. For more information, see the section about Creating Menu Groups.

Here's an example menu named `game_menu.xml`:

```
<?xmlversion="1.0"encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
  <item android:id="@+id/new_game"
        android:icon="@drawable/ic_new_game"
        android:title="@string/new_game"
        android:showAsAction="ifRoom"/>
  <item android:id="@+id/help"
        android:icon="@drawable/ic_help"
        android:title="@string/help" />
</menu>
```

The `<item>` element supports several attributes you can use to define an item's appearance and behavior. The items in the above menu include the following attributes:

`android:id`

A resource ID that's unique to the item, which allows the application to recognize the item when the user selects it.

`android:icon`

A reference to a drawable to use as the item's icon.

`android:title`

A reference to a string to use as the item's title.

`android:showAsAction`

Specifies when and how this item should appear as an action item in the app bar.

These are the most important attributes you should use, but there are many more available. For information about all the supported attributes, see the Menu Resource document.

You can add a submenu to an item in any menu (except a submenu) by adding a `<menu>` element as the child of an `<item>`. Submenus are useful when your application has a lot of functions that can be organized into topics, like items in a PC application's menu bar (File, Edit, View, etc.). For example:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
  <item android:id="@+id/file"
        android:title="@string/file" >
    <!-- "file" submenu -->
    <menu>
      <item android:id="@+id/create_new"
            android:title="@string/create_new" />
      <item android:id="@+id/open"
            android:title="@string/open" />
    </menu>
  </item>
</menu>
```

To use the menu in your activity, you need to inflate the menu resource (convert the XML resource into a programmable object) using `MenuInflater.inflate()`. In the following sections, you'll see how to inflate a menu for each menu type

4.2.6 Analogclock and digitalclock views

- The `AnalogClock` view displays an analog clock with two hands — one for minutes and one for
- hours. Its counterpart, the `DigitalClock` view, displays the time digitally.
- Both display the system time, and do not allow you to display a particular time. Hence, if you want to display the time for a particular region, you have to build your own custom views.

1. Using the `AnalogClock` and `DigitalClock` views are straightforward; simply declare them in your XML file (such as `main.xml`), like this:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <AnalogClock
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
```

<DigitalClock

android:layout_width="wrap_content"

android:layout_height="wrap_content" />

</LinearLayout>

4.2.7 .Dialog Boxes:



Main.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
android:orientation="vertical"
```

```
android:layout_width="fill_parent"
```

```
android:layout_height="fill_parent"
```

```
>
```

```
<TextView
```

```
android:layout_width="fill_parent"
```

```
android:layout_height="wrap_content"
```

```
android:text="@string/hello"
```

```
/>
```

```
<Button android:text="Ok" android:id="@+id/button1"
```

```
android:layout_width="wrap_content"
```

```
android:layout_height="wrap_content"></Button>
```

```
<Button android:text="YesNo" android:id="@+id/button2"
```

```
android:layout_width="wrap_content"
```

```
android:layout_height="wrap_content"></Button>
```

```
<Button android:text="YesNoCancel" android:id="@+id/button3"
```

```
android:layout_width="wrap_content"
```

```
android:layout_height="wrap_content"></Button>
```

```
</LinearLayout>
```

Main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
    />
    <Button android:text="Ok" android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"></Button>
    <Button android:text="YesNo" android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"></Button>
    <Button android:text="YesNoCancel" android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"></Button>
</LinearLayout>
```

Act.java

```
package com.AlertDialogDemo;

public class act extends Activity {

    /** Called when the activity is first created. */

    Button b1,b2,b3;

    @Override

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        b1=(Button)findViewById(R.id.button1);
        b2=(Button)findViewById(R.id.button2);
        b3=(Button)findViewById(R.id.button3);
```

```

b1.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        // TODO Auto-generated method stub
        AlertDialog alt=new AlertDialog.Builder(act.this).create();
        alt.setTitle("New Version");
        alt.setMessage("Welcome");
        alt.setIcon(R.drawable.icon);
        alt.setButton("Ok", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
                // TODO Auto-generated method stub
                Toast.makeText(getBaseContext(), "Ok", Toast.LENGTH_LONG).show();
            }
        });
        alt.show();
    }
});

b2.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        // TODO Auto-generated method stub
        AlertDialog.Builder alt=new AlertDialog.Builder(act.this);
        alt.setTitle("Confirm Save");
        alt.setMessage("Save it");
        alt.setIcon(R.drawable.icon);
        alt.setPositiveButton("Yes", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
                // TODO Auto-generated method stub
                Toast.makeText(getBaseContext(), "Yes", Toast.LENGTH_LONG).show();
            }
        });
        alt.setNegativeButton("No", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
                // TODO Auto-generated method stub
                Toast.makeText(getBaseContext(), "No", Toast.LENGTH_LONG).show();
            }
        });
    }
});

```

```

});
alt.show();
}
});
b3.setOnClickListener(new OnClickListener() {
public void onClick(View v) {
// TODO Auto-generated method stub
AlertDialog.Builder alt=new AlertDialog.Builder(act.this);
alt.setTitle("Confirm Delete");
alt.setMessage("Delete it");
alt.setIcon(R.drawable.icon);
alt.setPositiveButton("Yes", new DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog, int which) {
// TODO Auto-generated method stub
Toast.makeText(getBaseContext(), "Yes", Toast.LENGTH_LONG).show();
}
});
alt.setNegativeButton("No", new DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog, int which) {
// TODO Auto-generated method stub
Toast.makeText(getBaseContext(), "No", Toast.LENGTH_LONG).show();
}
});
alt.setNeutralButton("Cancel", new DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog, int which) {
// TODO Auto-generated method stub
Toast.makeText(getBaseContext(), "Cancel", Toast.LENGTH_LONG).show();
}
});
alt.show();
}
});
}
}

```

4.3.Displaying Pictures & Menus with views

To displaying images, we use the ImageView, Gallery, ImageSwitcher , and GridView views.

4.3.1.Image View:

Gallery View:

The Gallery is a view that shows items (such as images) in a center-locked, horizontal scrolling list.



FIGURE 5-1

1. Using Eclipse, create a new Android project



FIGURE 5-2

2 . Modify the main.xml fi le as shown

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
```

```

android:text="Images of San Francisco" />
<Gallery
android:id="@+id/gallery1"
android:layout_width="fill_parent"
android:layout_height="wrap_content" />
<ImageView
android:id="@+id/image1"
android:layout_width="320px"
android:layout_height="250px"
android:scaleType="fitXY" />
</LinearLayout>

```

3 . Right-click on the res/values folder and select New ⇨ File. Name the file attrs.xml.

4 . Populate the attrs.xml file as follows:

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
<declare-styleable name="Gallery1">
  <attr name="android:galleryItemBackground" />
</declare-styleable>
</resources>

```

5 . Prepare a series of images and name them pic1.png , pic2.png , and so on for each subsequent image

6 . Drag and drop all the images into the res/drawable-mdpi folder .When a dialog is displayed, check the copy option and click OK.

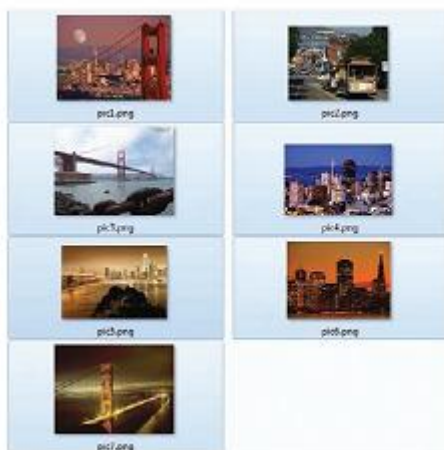


FIGURE 5-3

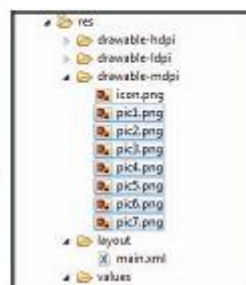


FIGURE 5-4

7 . Add the following statements to the MainActivity.java file:

```

packagenet.learn2develop.Gallery;
import android.app.Activity;
import android.os.Bundle;
import android.content.Context;
import android.content.res.TypedArray;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.BaseAdapter;
import android.widget.Gallery;
import android.widget.ImageView;
import android.widget.Toast;
public class MainActivity extends Activity {
    //---the images to display---
    Integer[] imageIDs = {
        R.drawable.pic1,
        R.drawable.pic2,
        R.drawable.pic3,
        R.drawable.pic4,
        R.drawable.pic5,
        R.drawable.pic6,
        R.drawable.pic7
    };
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Gallery gallery = (Gallery) findViewById(R.id.gallery1 );
        gallery.setAdapter(new ImageAdapter(this));
        gallery.setOnItemClickListener(new OnItemClickListener()
        {
            public void onItemClick(AdapterView<?> parent, View v,

```

```

int position, long id)
{
    Toast.makeText (getBaseContext(),
        "pic" + (position + 1) + " selected",
        Toast.LENGTH_SHORT).show();
}
});
}

public class ImageAdapter extends BaseAdapter
{
    private Context context;
    private int itemBackground;
    public ImageAdapter(Context c)
    {
        context = c;
        //---setting the style---
        TypedArray a = obtainStyledAttributes(R.styleable.Gallery1 );
        itemBackground = a.getResourceId(
            R.styleable.Gallery1_android_galleryItemBackground , 0);
        a.recycle();
    }
    //---returns the number of images---
    public int getCount() {
        return imageIDs .length;
    }
    //---returns the ID of an item---
    public Object getItem(int position) {
        return position;
    }
    //---returns the ID of an item---
    public long getItemId(int position) {
        return position;
    }
    //---returns an ImageView view---

```

```

public View getView(int position, View convertView, ViewGroup parent) {
    ImageView imageView = new ImageView(context);
    imageView.setImageResource(imageIDs [position]);
    imageView.setScaleType(ImageView.ScaleType. FIT_XY);
    imageView.setLayoutParams(new Gallery.LayoutParams(150, 120));
    imageView.setBackgroundResource(itemBackground);
    return imageView;
}
}
}

```

- 8 . Press F11 to debug the application on the Android Emulator.
- 9 . You can swipe the images to view the entire series of images. Observe that as you click on an image, the Toast class will display its name



FIGURE 5-5



FIGURE 5-6

- 10 . To display the selected image in the ImageView, add the following statements MainActivity.java file:

```

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState){
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    Gallery gallery=(Gallery)findViewById(R.id. gallery1 );

    gallery.setAdapter(new ImageAdapter( this));
    gallery.setOnItemClickListener(new OnItemClickListener()
    {
    public void onItemClick(AdapterView<?>parent,Viewv,
    int position,long id)
    {
    Toast.makeText (getBaseContext(),
    “pic”+(position+1)+“selected”,
    Toast.LENGTH_SHORT).show();
    //---display the images selected---
    ImageView imageView = (ImageView) findViewById(R.id.image1);
    imageView.setImageResource(imageIDs [position]);
    }
    });
    }

```

11 . Press F11 to debug the application again. This time, you will see the image selected in the ImageView



FIGURE 5-7

Image switcher

- Gallery view together with an ImageView to display a series of thumbnail images so that when one is selected, the selected image is displayed in the ImageView.
- However, sometimes we don't want an image to appear abruptly when the user selects it in the Gallery view.
- In this case, you need to use the ImageSwitcher together with the Gallery view.

Example:

- 1 . Using Eclipse, create a new Android project and name it as ImageSwitcher .
- 2 . Modify the main.xml file by adding the following statements

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#ff000000">
    <Gallery
        android:id="@+id/gallery1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />
    <ImageSwitcher
        android:id="@+id/switcher1"
        android:layout_width="fill_parent"
```

```

android:layout_height="fill_parent"
android:layout_alignParentLeft="true"
android:layout_alignParentRight="true"
android:layout_alignParentBottom="true" />
</RelativeLayout>

```

3 . Right-click on the res/values folder and select New ⇨ File. Name the file attrs.xml.

4 . Populate the attrs.xml file as follows:

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
<declare-styleable name="Gallery1">
<attr name="android:galleryItemBackground" />
</declare-styleable>
</resources>

```

5 . Drag and drop a series of images into the res/drawable-mdpi folder When a dialog is displayed, check the copy option and click OK.

6 . Add the following statements to the MainActivity.java file:

```

packagenet.learn2develop.ImageSwitcher;
import android.app.Activity;
import android.os.Bundle;
import android.content.Context;
import android.content.res.TypedArray;
import android.view.View;
import android.view.ViewGroup;
import android.view.ViewGroup.LayoutParams;
import android.view.animation.AnimationUtils;
import android.widget.BaseAdapter;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.Gallery;
import android.widget.ViewSwitcher.ViewFactory;
import android.widget.ImageSwitcher;
import android.widget.ImageView;
public class MainActivity extends Activity implements ViewFactory {

```

```

//---the images to display---
Integer[] imageIDs = {
    R.drawable.pic1,
    R.drawable.pic2,
    R.drawable.pic3,
    R.drawable.pic4,
    R.drawable.pic5,
    R.drawable.pic6,
    R.drawable.pic7
};

private ImageSwitcher imageSwitcher ;
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState){
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    imageSwitcher = (ImageSwitcher) findViewById(R.id.switcher1);
    imageSwitcher .setFactory(this);
    imageSwitcher .setInAnimation(AnimationUtils.loadAnimation (this,
    android.R.anim.fade_in));
    imageSwitcher .setOutAnimation(AnimationUtils.loadAnimation (this,
    android.R.anim.fade_out ));
    Gallery gallery = (Gallery) findViewById(R.id.gallery1 );
    gallery.setAdapter(new ImageAdapter(this));
    gallery.setOnItemClickListener(new OnItemClickListener()
    {
        public void onItemClick(AdapterView<?> parent,
        View v, int position, long id)
        {
            imageSwitcher .setImageResource( imageIDs [position]);
        }
    });
}

public View makeView()

```

```

{
    ImageView imageView = new ImageView(this);
    imageView.setBackgroundColor(0xFF000000);
    imageView.setScaleType(ImageView.ScaleType. FIT_CENTER);
    imageView.setLayoutParams(new
    ImageSwitcher.LayoutParams(
    LayoutParams. FILL_PARENT,
    LayoutParams. FILL_PARENT));
    return imageView;
}

public class ImageAdapter extends BaseAdapter
{
    private Context context;
    private int itemBackground;
    public ImageAdapter(Context c)
    {
        context = c;
        //---setting the style---
        TypedArray a = obtainStyledAttributes(R.styleable.Gallery1 );
        itemBackground = a.getResourceId(
        R.styleable.Gallery1_android_galleryItemBackground , 0);
        a.recycle();
    }
    //---returns the number of images---
    public int getCount()
    {
        return imageIDs .length;
    }
    //---returns the ID of an item---
    public Object getItem(int position)
    {
        return position;
    }
    public long getItemId(int position)

```

```

{
return position;
}
//---returns an ImageView view---
public View getView(int position, View convertView, ViewGroup parent)
{
ImageView imageView = new ImageView(context);

imageView.setImageResource(imageIDs [position]);
imageView.setScaleType(ImageView.ScaleType. FIT_XY);
imageView.setLayoutParams(new Gallery.LayoutParams(150, 120));
imageView.setBackgroundResource(itemBackground);

return imageView;
}
}
}

```

7 . Press F11 to debug the application on the Android Emulator.



FIGURE 5-8

4.3.2. Grid view

The `GridView` shows items in a two-dimensional scrolling grid.

We use the `GridView` together with an `ImageView` to display a series of images.

Example:

- 1 . Using Eclipse, create a new Android project and name it Grid.
- 2 . Drag and drop a series of images into the `res/drawable-mdpi` folder. When a dialog is displayed, check the copy option and click OK.
- 3 . Populate the `main.xml` file with the following content:

```
<?xml version="1.0" encoding="utf-8"?>
<GridView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/gridview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:numColumns="auto_fit"
    android:verticalSpacing="10dp"
    android:horizontalSpacing="10dp"
    android:columnWidth="90dp"
    android:stretchMode="columnWidth"
    android:gravity="center"
/>
```

- 4 . Add the following statements to the `MainActivity.java` file:

```
packagenet.learn2develop.Grid;
import android.app.Activity;
import android.os.Bundle;
import android.content.Context;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.BaseAdapter;
import android.widget.GridView;
import android.widget.ImageView;
import android.widget.Toast;
```

```

public class MainActivity extends Activity {
    //---the images to display---
    Integer[] imageIDs = {
        R.drawable.pic1,
        R.drawable.pic2,
        R.drawable.pic3,
        R.drawable.pic4,
        R.drawable.pic5,
        R.drawable.pic6,
        R.drawable.pic7
    };
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        GridView gridView = (GridView) findViewById(R.id.gridview );
        gridView.setAdapter(new ImageAdapter(this));
        gridView.setOnItemClickListener(new OnItemClickListener()
        {
            public void onItemClick(AdapterView<?> parent,
                View v, int position, long id)
            {
                Toast.makeText(getApplicationContext(),
                    "pic" + (position + 1) + " selected",
                    Toast.LENGTH_SHORT).show();
            }
        });
    }
    public class ImageAdapter extends BaseAdapter
    {
        private Context context;
        public ImageAdapter(Context c)
        {

```

```

context = c;
}
//---returns the number of images---
public int getCount() {
return imageIDs .length;
}
//---returns the ID of an item---
public Object getItem(int position) {
return position;
}
//---returns the ID of an item---
public long getItemId(int position) {
return position;
}
//---returns an ImageView view---
public View getView(int position, View convertView,
ViewGroup parent)
{
ImageView imageView;
if (convertView == null) {
imageView = new ImageView(context);
imageView.setLayoutParams(new
GridView.LayoutParams(85, 85));
imageView.setScaleType(
ImageView.ScaleType.CENTER_CROP);
imageView.setPadding(5, 5, 5, 5);
} else {
imageView = (ImageView) convertView;
}
imageView.setImageResource(imageIDs [position]);
return imageView;
}
}
}

```

5 . Press F11 to debug the application on the Android Emulator.



FIGURE 5-10

4.3.3. Creating the Menu Helper Methods

Menus with views:

Menus are useful for displaying additional options that are not directly visible on the main UI of an application. There are two main types of menus in Android:

- Options menu — Displays information related to the current activity. In Android, you activate the options menu by pressing the MENU key.
- Context menu — Displays information related to a particular view on an activity. In Android, to activate a context menu you tap and hold on to it.

To activate the context menu, the user selects an item on the screen and either taps and holds it or simply presses the center button on the directional keypad.

Creating the helper methods

Before you go ahead and create your options and context menus, you need to create two helper methods. One creates a list of items to show inside a menu, while the other handles the event that is fired when the user selects an item inside the menu.

Example:

- 1 . Using Eclipse, create a new Android project and name it as Menus.
- 2 . In the MainActivity.java file, add the following statements:

```
packagenet.learn2develop.Menus;
```

```
import android.app.Activity;
```

```
import android.os.Bundle;
```

```
import android.view.Menu;
```

```

import android.view.MenuItem;
import android.widget.Button;
import android.widget.Toast;
public class MainActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    private void CreateMenu(Menu menu)
    {
        MenuItem mnu1 = menu.add(0, 0, 0, "Item 1");
        {
            mnu1.setAlphabeticShortcut('a');
            mnu1.setIcon(R.drawable.icon);
        }
        MenuItem mnu2 = menu.add(0, 1, 1, "Item 2");
        {
            mnu2.setAlphabeticShortcut('b');
            mnu2.setIcon(R.drawable.icon);
        }
        MenuItem mnu3 = menu.add(0, 2, 2, "Item 3");
        {
            mnu3.setAlphabeticShortcut('c');
            mnu3.setIcon(R.drawable.icon);
        }
        MenuItem mnu4 = menu.add(0, 3, 3, "Item 4");
        {
            mnu4.setAlphabeticShortcut('d');
        }
        menu.add(0, 3, 3, "Item 5");
        menu.add(0, 3, 3, "Item 6");
        menu.add(0, 3, 3, "Item 7");
    }
}

```

```

}
private boolean MenuChoice(MenuItem item)
{
    switch (item.getItemId()) {
        case 0:
            Toast.makeText (this, "You clicked on Item 1" ,
            Toast.LENGTH_LONG).show();
            return true;
        case 1:
            Toast.makeText (this, "You clicked on Item 2" ,
            Toast.LENGTH_LONG).show();
            return true;
        case 2:
            Toast.makeText (this, "You clicked on Item 3" ,
            Toast.LENGTH_LONG).show();
            return true;
        case 3:
            Toast.makeText (this, "You clicked on Item 4" ,
            Toast.LENGTH_LONG).show();
            return true;
        case 4:
            Toast.makeText (this, "You clicked on Item 5" ,
            Toast.LENGTH_LONG).show();
            return true;
        case 5:
            Toast.makeText (this, "You clicked on Item 6" ,
            Toast.LENGTH_LONG).show();
            return true;
        case 6:
            Toast.makeText (this, "You clicked on Item 7" ,
            Toast.LENGTH_LONG).show();
            return true;
    }
    return false;
}

```

```
}  
}
```

Working Principle:

- The preceding example creates two methods: `CreateMenu()` and `MenuChoice()`.
- The `CreateMenu()` method takes a `Menu` argument and adds a series of menu items to it.
- To add a menu item to the menu, you create an instance of the `MenuItem` class and use the `add()` method of the `Menu` object.

```
MenuItem mnu1=menu.add(0,0,0,"Item1");  
{  
mnu1.setAlphabeticShortcut('a');  
mnu1.setIcon(R.drawable.icon);  
}
```

The four arguments of the `add()` method are as follows:

- **groupId** — The group identifier that the menu item should be part of. Use 0 if an item is not in a group.
- **itemId** — Unique item ID
- **order** — The order in which the item should be displayed
- **title** — The text to display for the menu item
- use the `setAlphabeticShortcut()` method to assign a shortcut key to the menu item so that
- users can select an item by pressing a key on the keyboard. The `setIcon()` method sets an image to be displayed on the menu item.
- The `MenuChoice()` method takes a `MenuItem` argument and checks its ID to determine the menu item that is clicked. It then displays a `Toast` message to let the user know which menu item was clicked.

4.2.4.Options menu

Example:

1 . Using the same project created in the previous section, add the following statements to the

MainActivity.java file:

```
packagenet.learn2develop.Menus;
importandroid.app.Activity;
importandroid.os.Bundle;
importandroid.view.Menu;
importandroid.view.MenuItem;
importandroid.widget.Button;
importandroid.widget.Toast;
publicclassMainActivityextendsActivity {
/** Calledwhenthe activity isfirstcreated. */
@Override
publicvoidonCreate(Bundle savedInstanceState){
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);
    CreateMenu(menu);
    return true;
}
@Override
public boolean onOptionsItemSelected(MenuItem item)
{
    return MenuChoice(item);
}
privatevoidCreateMenu(Menumenu)
{
    //...
```

```

}
private boolean MenuChoice(MenuItem item)
{
    //...
}
}

```

2 . Press F11 to debug the application on the Android Emulator.

Working Principle:

- To display the options menu for your activity, you need to override two methods in your activity:
- `onCreateOptionsMenu()` and `onOptionsItemSelected()`. The `onCreateOptionsMenu()` method is called when the MENU button is pressed.
- In this event, you call the `CreateMenu()` helper method to display the options menu. When a menu item is selected, the `onOptionsItemSelected()` method is called.
- In this case, you call the `MenuChoice()` method to display the menu item selected (and do whatever you want to do).
- Observe the icons displayed for menu items 1, 2, and 3. Also, if the options menu has more than six items, a “More” menu item will be displayed to indicate the additional options.



FIGURE 5-13



FIGURE 5-14

4.3.7.Context menu

- A context menu is usually associated with a view on an activity, and it is displayed when the user long clicks an item.
- For example, if the user taps on a Button view and hold it for a few seconds, a context menu can be displayed.
- If you want to associate a context menu with a view on an activity, you need to call the `setOnCreateContextMenuListener()` method of that particular view.

Example:

1 . Using the same project created in the previous section, add the following statements to the MainActivity.java file:

```
packagenet.learn2develop.Menus;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
```

```

import android.widget.Button;
import android.widget.Toast;
import android.view.View;
import android.view.ContextMenu;
import android.view.ContextMenu.ContextMenuInfo;
public class MainActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button btn = (Button) findViewById(R.id.btn1);
        btn.setOnCreateContextMenuListener(this);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        super.onCreateOptionsMenu(menu);
        CreateMenu(menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item)
    {
        return MenuChoice(item);
    }
    @Override
    public void onCreateContextMenu(ContextMenu menu, View view,
        ContextMenuInfo menuInfo)
    {
        super.onCreateContextMenu(menu, view, menuInfo);
        CreateMenu(menu);
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item)

```

```

{
return MenuChoice(item);
}
private void CreateMenu(Menu menu)
{
//...
}
private boolean MenuChoice(MenuItem item)
{
//...
}
}

```

2 . Press F11 to debug the application on the Android Emulator.



FIGURE 5-15

4.4.SMS messaging

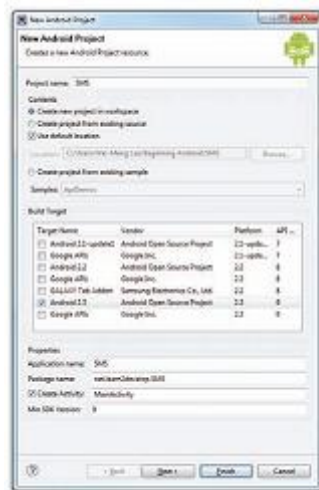
- SMS messaging is one of the main killer applications on a mobile phone today
- Any mobile phone you buy today should have at least
- SMS messaging capabilities, and nearly all users of any age know how to send and receive such messages.
- Android comes with a built-in SMS application that enables you to send and receive SMS messages.
- However, in some cases you might want to integrate SMS capabilities into your own

Android application. For example, you might want to write an application that automatically sends a SMS message at regular time intervals.

4.4.1 Sending SmS

Using this approach, your application can automatically send an SMS message to a recipient without user intervention.

1. Using Eclipse, create a new Android project



- 2 . Add the following statements to the main.xml file:

```
<?xmlversion= "1.0"encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <Button
        android:id="@+id/btnSendSMS"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text= "Send SMS" />
</LinearLayout>
```

- 3 . In the AndroidManifest.xml file, add the following statements

```
<?xmlversion= "1.0"encoding="utf-8"?>
<manifestxmlns:android="http://schemas.android.com/apk/res/android"
    package= "net.learn2develop.SMS"
    android:versionCode="1"
```

```

android:versionName="1.0">
<applicationandroid:icon= "@drawable/icon"android:label="@string/app_name"
>
<activityandroid:name= ".MainActivity"
android:label="@string/app_name" >
<intent-filter>
<actionandroid:name= "android.intent.action.MAIN" />
<categoryandroid:name= "android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
</application>
<uses-sdkandroid:minSdkVersion="8" />
<uses-permission android:name="android.permission.SEND_SMS"></uses-
permission>
</manifest>

```

4 . Add the following statements to the MainActivity.java file:

```

packagenet.learn2develop.SMS;
importandroid.app.Activity;
importandroid.os.Bundle;
import android.app.PendingIntent;
import android.content.Intent;
import android.telephony.SmsManager;
import android.view.View;
import android.widget.Button;
publicclassMainActivityextendsActivity {
Button btnSendSMS;
/** Calledwhenthe activity isfirstcreated. */
@Override
publicvoidonCreate(Bundle savedInstanceState){
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
btnSendSMS = (Button) findViewById(R.id.btnSendSMS);
btnSendSMS.setOnClickListener(new View.OnClickListener()
{

```

```

public void onClick(View v)
{
    sendSMS( "5556", "Hello my friends!");
}

});
}

//---sends an SMS message to another device---
private void sendSMS(String phoneNumber, String message)
{
    SmsManager sms = SmsManager. getDefault();
    sms.sendTextMessage(phoneNumber, null, message, null, null);
}
}

```

5 . Press F11 to debug the application on the Android Emulator. Using the Android SDK and AVD Manager, launch another AVD.

6 . On the first Android Emulator, click the Send SMS button to send an SMS message to the second emulator.



Working Principle:

To send an SMS message programmatically, you use the `SmsManager` class. Unlike other classes, you do not directly instantiate this class; instead, you call the `getDefault()` static method to obtain a `SmsManager` object. You then send the SMS message using the `sendTextMessage()` method:

```

private void sendSMS(String phoneNumber, String message)
{

```

```

SmsManagersms =SmsManager.getDefault();
sms.sendTextMessage(phoneNumber,null,message, null,null);
}

```

Following are the five arguments to the sendTextMessage() method:

- destinationAddress — Phone number of the recipient
- scAddress — Service center address; use null for default SMSC
- text — Content of the SMS message
- sentIntent — Pending intent to invoke when the message is sent
- deliveryIntent — Pending intent to invoke when the message has been delivered

4.4.2.Receiving sms messages

Besides sending SMS messages from your Android applications, you can also receive incoming SMS messages from within your application by using a BroadcastReceiver object. This is useful when you want your application to perform an action when a certain SMS message is received.

Example:

- 1 . Using the same project created in the previous section, add the following statements in to the AndroidManifest.xml file:

```

<?xmlversion="1.0"encoding="utf-8"?>
<manifestxmlns:android=http://schemas.android.com/apk/res/android
package="net.learn2develop.SMS"
android:versionCode="1"
android:versionName="1.0">
<applicationandroid:icon= "@drawable/icon"android:label="@string/app_name"
>
<activityandroid:name= ".MainActivity"
android:label="@string/app_name" >
<intent-filter>
<actionandroid:name= "android.intent.action.MAIN" />
<categoryandroid:name= "android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
<receiver android:name=".SMSReceiver">
<intent-filter>

```

```

<action android:name=
“android.provider.Telephony.SMS_RECEIVED” />
</intent-filter>
</receiver>
</application>
<uses-sdk android:minSdkVersion=”8” />
<uses-permission android:name=          ”android.permission.SEND_SMS”></uses-
permission>
<uses-permission android:name=”android.permission.RECEIVE_SMS”>
</uses-permission>
</manifest>

```

2 . In the src folder of the project, add a new Class file to the package name and call it SMSReceiver.java

3 . Code the SMSReceiver.java file as follows:

```

package net.learn2develop.SMS;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.SmsMessage;
import android.widget.Toast;
public class SMSReceiver extends BroadcastReceiver
{
    @Override
    public void onReceive(Context context, Intent intent)
    {
        //---get the SMS message passed in---
        Bundle bundle = intent.getExtras();
        SmsMessage[] msgs = null;
        String str = “”;
        if (bundle != null)
        {
            //---retrieve the SMS message received---
            Object[] pdus = (Object[]) bundle.get( “pdus”);

```

```

msgs = new SmsMessage[pdus.length];
for (int i=0; i<msgs.length; i++){
    msgs[i] = SmsMessage.createFromPdu ((byte[])pdus[i]);
    str += "SMS from " + msgs[i].getOriginatingAddress();
    str += " :";
    str += msgs[i].getMessageBody().toString();
    str += "\n";
}
//---display the new SMS message---
Toast.makeText (context, str, Toast.LENGTH_SHORT).show();
}
}
}

```

- 4 . Press F11 to debug the application on the Android Emulator.
- 5 . Using the DDMS, send a message to the emulator. Your application should be able to receive the message and display it using the Toast class



Working Principle:

- To listen for incoming SMS messages, you create a BroadcastReceiver class. The BroadcastReceiver class enables your application to receive intents sent by other applications using the `sendBroadcast()` method. Essentially, it enables your application to handle events raised by other applications.
- When an intent is received, the `onReceive()` method is called; hence, you need to override this.

- When an incoming SMS message is received, the `onReceive()` method is fired. The SMS message is contained in the Intent object (intent; the second parameter in the `onReceive()` method) via a Bundle object. The messages are stored in an Object array in the PDU format.
- To extract each message, you use the static `createFromPdu()` method from the `SmsMessage` class.
- The SMS message is then displayed using the Toast class. The phone number of the sender is obtained via the `getOriginatingAddress()` method, so if you need to send an autoreply to the sender, this is the method to obtain the sender's phone number.

4.4.3 Making phone calls

- Besides sending SMS messages to the emulator, we use the Telnet client to make a phone call to the emulator. To do so, simply use the following commands.
- sTo Telnet to the emulator, use this command:

C:\telnetlocalhost5554

- To make a phone call to the emulator, use this command:

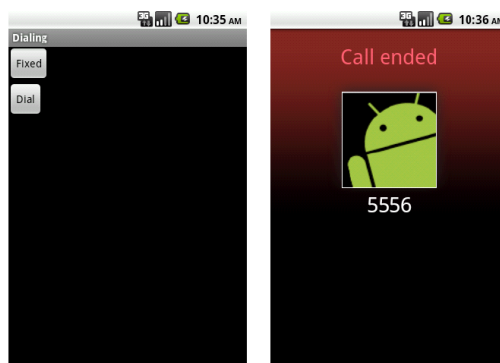
gsm call+651234567

- The syntax of the `gsm send` command is as follows:

gsm call<phone_number>

As with sending SMS, you can also make phone calls between AVDs by using their port numbers as phone numbers.

Fixed Dialing



Add the Following Permissions in `androidmanifest.xml`

android.permission.CALL_PHONE

android.permission.MODIFY_PHONE_STATE

android.permission.PROCESS_OUTGOING_CALLS

android.permission.READ_PHONE_STATE

Main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <Button android:text="Fixed" android:id="@+id/button1"
        android:layout_width="wrap_content" android:layout_height="wrap_content"></Button>
    <Button android:text="Dial" android:id="@+id/button2"
        android:layout_width="wrap_content" android:layout_height="wrap_content"></Button>
</LinearLayout>
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.dialing"
    android:versionCode="1"
    android:versionName="1.0">
    <uses-sdk android:minSdkVersion="8" />
    <uses-permission android:name="android.permission.CALL_PHONE">
    </uses-permission>
    <uses-permission android:name="android.permission.MODIFY_PHONE_STATE">
    </uses-permission>
    <uses-permission
        android:name="android.permission.PROCESS_OUTGOING_CALLS">
    </uses-permission>
    <uses-permission android:name="android.permission.READ_PHONE_STATE">
    </uses-permission>
    <application android:icon="@drawable/icon" android:label="@string/app_name">
    <activity android:name=".act"
        android:label="@string/app_name">
    <intent-filter>
```

```

<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
</application>
</manifest>

```

Act.java

```

package com.dialing;

public class act extends Activity {
    /** Called when the activity is first created. */
    Button b1;
    Button b2;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        b1=(Button)findViewById(R.id.button1);
        b2=(Button)findViewById(R.id.button2);
        b1.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                // TODO Auto-generated method stub
                Intent in=new Intent(Intent.ACTION_CALL,Uri.parse("tel:"+5556));
                startActivity(in);
            }
        });
        b2.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                // TODO Auto-generated method stub
                Intent in=new Intent(Intent.ACTION_CALL,Uri.parse("tel:"));
                startActivity(in);
            }
        });
    }
}

```


Review Questions

PART-A(2 Mark Questions)

1. What is a view?
2. What is meant by viewgroup?
3. What is the use of text view?
4. Define button.
5. What is an Image button?
6. Define Edit text.
7. What is the use of Checkbox?
8. What is Togglebutton?
9. What is the use of Radiobuttonview?
10. Define Progressbar View.
11. Define Timepicker View.
12. Define Datepicker View.
13. Mention some advanced views
14. What is the use of Imageview?
15. Mention the dialog boxes used in Android.
16. What is the use of Gridview?
17. Mention the types of menus used in Android.
18. What are the two ways to send SMS?

PART-B(3 Mark Questions)

1. Explain Textview.
2. Explain Button Control.
3. Explain Image button.
4. Explain Edit text.
5. Explain ToggleButton.
6. Explain Checkbox control.
7. Explain Timepicker view.
8. Explain Datepicker view
9. Explain List view
10. Explain Analog and Digital view.

PART-C(5&10 Mark Questions)

1. Explain basic views with example.(10)
2. Explain Radio button and Radiobutton group views with example(10)
3. Explain Progressbar view(5)
4. With example coding, explain Auto complete Text view(10)
5. Explain any 3 Advanced views in detail.(10)
6. Explain dialog boxes(5)
7. Explain displaying pictures with views.(10)
8. Briefly explain how to receive SMS using Android Studio.(5)
9. Explain how to make phone calls using Android Studio(5)

UNIT-V

LOCATION BASED SERVICES AND SQLITE

OBJECTIVES

At the end of the unit, students can

- Explain Location based Services
- Explain Sharing data
- Explain View,Modify,Add,Delete Contacts
- Explain Store and retrieve data in Internal and External Storage.
- Explain Creating and using Data bases in Android
- Explain Android Service
- Explain downloading binary data

5.1 Location Based Services

5.1.1. Obtaining the maps API key

- Beginning with the Android SDK release v1.0, you need to apply for a free Google Maps API key before you can integrate Google Maps into your Android application.
- When you apply for the key, you must also agree to Google's terms of use, so be sure to read them carefully.
- To apply for a key, follow the series of steps outlined next.
- First, if you are testing the application on the Android Emulator or an Android device directly connected to your development machine, locate the SDK debug certificate located in the default folder (C:\Users\<username>\.android for Windows 7 users).
- You can verify the existence of the debug certificate by going to Eclipse and selecting Window ⇨ Preferences. Expand the Android item and select Build .
- On the right side of the window, you will be able to see the debug certificate's location.
- In this example, my MD5 fingerprint is
EF:7A:61:EA:AF:E0:B4:2D:FD:43:5E:1D:26:04:34:BA.
- Copy the MD5 certificate fingerprint and navigate your web browser to:
<http://code.google.com/>

- android/maps-api-signup.html. Follow the instructions on the page to complete the application and obtain the Google Maps key. When you are done, you should see something similar to what is shown in Figure below.



5.1.2.Displaying the map

You are now ready to display Google Maps in your Android application. This involves two main tasks:

- Modify your AndroidManifest.xml file by adding both the <uses-library> element and the INTERNET permission.
- Add the MapView element to your UI.

Example:

1 . Using the project created in the previous section, add the following lines in bold to the main.xml file (be sure to replace the value of the apiKey attribute with the API key you obtained earlier):

```
<?xmlversion= "1.0"encoding="utf-8"?>
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
android:orientation="vertical"
```

```
android:layout_width="fill_parent"
```

```
android:layout_height="fill_parent"
```

```
>
```

```
<com.google.android.maps.MapView
```

```
android:id="@+id/mapView"
```

```
android:layout_width="fill_parent"
```

```

        android:layout_height="fill_parent"
        android:enabled="true"
        android:clickable="true"
        android:apiKey="<YOUR KEY>" />
</LinearLayout>
2 . Add the following lines in bold to the main.xml file:
<?xmlversion= "1.0"encoding="utf-8"?>
<manifestxmlns:android="http://schemas.android.com/apk/res/android"
package= "net.learn2develop.LBS"
android:versionCode="1"
android:versionName="1.0">
<applicationandroid:icon= "@drawable/icon"android:label="@string/app_name" >
    <uses-library android:name="com.google.android.maps" />
    <activityandroid:name= ".MainActivity"
        android:label="@string/app_name" >
        <intent-filter>
        <actionandroid:name= "android.intent.action.MAIN" />
        <categoryandroid:name= "android.intent.category.LAUNCHER"/>
        </intent-filter>
    </activity>
</application>
<uses-sdkandroid:minSdkVersion="8" />
<uses-permission android:name="android.permission.INTERNET"></uses-
permission>
</manifest>

```

3 . Add the following statements in bold to the MainActivity.java file. Note that MainActivity is now extending the MapActivity class.

```

package net.learn2develop.LBS;
import android.app.Activity;
import android.os.Bundle;
import com.google.android.maps.MapActivity;
public class MainActivity extends MapActivity{
/** Called when the activity is first created. */
@Override

```

```

public void onCreate(Bundle savedInstanceState){
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
}

@Override
protected boolean isRouteDisplayed() {
    // TODO Auto-generated method stub
    return false;
}
}

```

4 . Press F11 to debug the application on the Android Emulator.



5.1.3.Zoom control

- We can pan the map to any desired location and it will be updated on-the-fly. However, on the emulator there is no way to zoom in or out from a particular location (on a real Android device you can pinch the map to zoom it).
- Thus, in this section, you will learn how you can let users zoom in or out of the map using the built-in zoom controls.

Example:

1 . Using the project created in the previous activity, add in the following statements

```

package net.learn2develop.LBS;
import android.app.Activity;
import android.os.Bundle;
import com.google.android.maps.MapActivity;

```

```

import com.google.android.maps.MapView;
public class MainActivity extends MapActivity{
    MapView mapView;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mapView = (MapView) findViewById(R.id.mapView);
        mapView.setBuiltInZoomControls(true);
    }
    @Override
    protected boolean isRouteDisplayed() {
        // TODO Auto-generated method stub
        return false;
    }
}

```

2 . Press F11 to debug the application on the Android Emulator. Observe the built-in zoom controls that appear at the bottom of the map when you click and drag the map. You can click the minus (–) icon to zoom out of the map and the plus (+) icon to zoom into the map.

Working Principle

- To display the built-in zoom controls, you first get a reference to the map and then call the `setBuiltInZoomControls()` method:

```

mapView=(MapView)findViewById(R.id. mapView);
mapView.setBuiltInZoomControls(true);

```

- Besides displaying the zoom controls, you can also programmatically zoom in or out of the map using the `zoomIn()` or `zoomOut()` method of the `MapController` class.

1 . Using the project created in the previous activity, add the following statements to the `MainActivity.java` file:

```

package net.learn2develop.LBS;
import android.app.Activity;

```

```

import android.os.Bundle;
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapView;
import android.view.KeyEvent;
import com.google.android.maps.MapController;
public class MainActivity extends MapActivity{
    MapView mapView;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mapView = (MapView) findViewById(R.id. mapView);
        mapView.setBuiltInZoomControls(true);
    }

    public boolean onKeyDown(int keyCode, KeyEvent event)
    {
        MapController mc = mapView.getController();
        switch (keyCode)
        {
            case KeyEvent.KEYCODE_3:
                mc.zoomIn();
                break;
            case KeyEvent.KEYCODE_1:
                mc.zoomOut();
                break;
        }
        return super.onKeyDown(keyCode, event);
    }

    @Override
    protected boolean isRouteDisplayed() {
        // TODO Auto-generated method stub
        return false;
    }
}

```

```
}
```

2 . Press F11 to debug the application on the Android Emulator. You can now zoom into the map by pressing the numeric 3 key on the emulator. To zoom out of the map, press the numeric 1 key.

Working Principle:

To handle key presses on your activity, you handle the `onKeyDown` event:

```
publicbooleanonKeyDown(int keyCode, KeyEvent event)
```

```
{
```

```
//...
```

```
}
```

- To manage the panning and zooming of the map, you need to obtain an instance of the `MapController` class from the `MapView` object.
- The `MapController` class contains the `zoomIn()` and `zoomOut()` methods (plus some other methods to control the map) to enable users to zoom in or out of the map, respectively.

5.1.4.Navigating to a Specific location

- By default, Google Maps displays the map of the United States when it is first loaded. However, you can also set Google Maps to display a particular location.
- In this case, you can use the `animateTo()` method of the `MapController` class.

Example:

1 . Using the project created in the previous activity, add the following statements to the `MainActivity.java` file:

```
packagenet.learn2develop.LBS;  
importandroid.app.Activity;  
importandroid.os.Bundle;  
importandroid.view.KeyEvent;  
importcom.google.android.maps.MapActivity;  
importcom.google.android.maps.MapController;  
importcom.google.android.maps.MapView;  
import com.google.android.maps.GeoPoint;  
publicclassMainActivityextendsMapActivity{  
MapViewmapView;  
MapController mc;
```

```

    GeoPoint p;
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState){
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    mapView=(MapView)findViewById(R.id. mapView);
    mapView.setBuiltInZoomControls(true);
        //mapView.setSatellite(true);
    mapView.setStreetView(true);
        mc = mapView.getController();
        String coordinates[] = {"1.352566007" , "103.78921587"};
        double lat = Double.parseDouble(coordinates[0]);
        double lng = Double.parseDouble(coordinates[1]);
        p = new GeoPoint(
            ( int ) (lat * 1E6),
            ( int ) (lng * 1E6));
        mc.animateTo(p);
        mc.setZoom(13);
        mapView.invalidate();
}
public boolean onKeyDown(int keyCode, KeyEvent event)
{
    MapController mc=mapView.getController();
    switch(keyCode)
    {
case KeyEvent.KEYCODE_3:
        mc.zoomIn();
        break;
case KeyEvent.KEYCODE_1:
        mc.zoomOut();
        break;
    }
    return super.onKeyDown(keyCode,event);

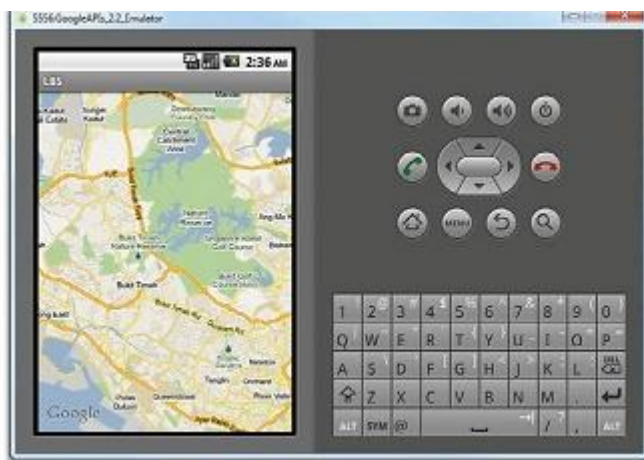
```

```

}
@Override
protected boolean isRouteDisplayed() {
    //TODO Auto-generated method stub
    return false;
}
}

```

2 . Press F11 to debug the application on the Android Emulator. When the map is loaded, observe that it now animates to a particular location in Singapore .



Working Principle:

- In the preceding code, you first obtain a map controller from the MapView instance and assign it to a MapController object (mc). You then use a GeoPoint object to represent a geographical location.
- To navigate the map to a particular location, you can use the animateTo() method of the MapController class. The setZoom() method enables you to specify the zoom level at which the map is displayed (the bigger the number, the more details you see on the map). The invalidate() method forces the MapView to be redrawn.

5.1.5 Adding markers

Adding markers to a map to indicate places of interest enables your users to easily locate the places they are looking for. The following Try It Out shows you how to add a marker to Google Maps.

Example:

1 . Create a GIF image containing a pushpin and copy it into the `res/drawable-mdpi` folder of the project. For the best effect, make the background of the image transparent so that it does not

block parts of the map when the image is added to the map.

2 . Using the project created in the previous activity, add the following statements to the `MainActivity.java` file:

```
packagenet.learn2develop.LBS;
import android.app.Activity;
import android.os.Bundle;
import android.view.KeyEvent;
import com.google.android.maps.GeoPoint;
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapController;
import com.google.android.maps.MapView;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Point;
import com.google.android.maps.Overlay;
import java.util.List;
public class MainActivity extends MapActivity{
    MapView mapView;
    MapController mc;
    GeoPoint p;
    class MapOverlay extends com.google.android.maps.Overlay
    {
        @Override
        public boolean draw(Canvas canvas, MapView mapView,
        boolean shadow, long when)
        {
            super.draw(canvas, mapView, shadow);
            //---translate the GeoPoint to screen pixels---
            Point screenPts = new Point();
            mapView.getProjection().toPixels(p, screenPts);

```

```

        //---add the marker---
        Bitmap bmp = BitmapFactory.decodeResource(
            getResources(), R.drawable.pushpin);
        canvas.drawBitmap(bmp, screenPts.x, screenPts.y-50, null);
        return true;
    }
}

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    mapView = (MapView) findViewById(R.id.mapView);
    mapView.setBuiltInZoomControls(true);
    //mapView.setSatellite(true);
    //mapView.setStreetView(true);
    mc = mapView.getController();
    String coordinates[] = {"1.352566007", "103.78921587"};
    double lat = Double.parseDouble(coordinates[0]);
    double lng = Double.parseDouble(coordinates[1]);
    p = new GeoPoint(
        (int)(lat*1E6),
        (int)(lng*1E6));
    mc.animateTo(p);
    mc.setZoom(13);
    //---Add a location marker---
    MapOverlay mapOverlay = new MapOverlay();
    List<Overlay> listOfOverlays = mapView.getOverlays();
    listOfOverlays.clear();
    listOfOverlays.add(mapOverlay);
    mapView.invalidate();
}

public boolean onKeyDown(int keyCode, KeyEvent event)
{

```

```

MapController mc=mapView.getController();
switch(keyCode)
{
caseKeyEvent.KEYCODE_3:
mc.zoomIn();
break;
caseKeyEvent.KEYCODE_1:
mc.zoomOut();
break;
}
returnsuper.onKeyDown(keyCode,event);
}

@Override
protectedbooleanisRouteDisplayed() {
//TODOAuto-generatedmethodstub
returnfalse;
}
}

```

3 . Press F11 to debug the application on the Android Emulator. Figure 9-13 shows the marker added to the map.



Working Principle:

1)To add a marker to the map, you first need to define a class that extends the Overlay class:

```
class MapOverlay extends com.google.android.maps.Overlay  
{  
    @Override  
    public boolean draw(Canvas canvas, MapView mapView,  
        boolean shadow, long when)  
    {  
    //...  
    }  
}
```

2)An overlay represents an individual item that you can draw on the map. You can add as many overlays as you want. In the MapOverlay class, override the draw() method so that you can draw the pushpin image on the map. In particular, note that you need to translate the geographical location (represented by a GeoPoint object, p) into screen coordinates:

//---translate the GeoPoint to screen pixels---

```
Point screenPts = new Point();  
mapView.getProjection().toPixels(p, screenPts);
```

3)Because you want the pointed tip of the pushpin to indicate the position of the location, you need to deduct the height of the image (which is 50 pixels) from the y coordinate of the point and draw the image at that location:

//---add the marker---

```
Bitmap bmp = BitmapFactory.decodeResource(  
getResources(), R.drawable.pushpin);  
canvas.drawBitmap(bmp, screenPts.x, screenPts.y-50, null);
```

4)To add the marker, create an instance of the MapOverlay class and add it to the list of overlays available on the MapView object:

//---Add a location marker---

```
MapOverlay mapOverlay = new MapOverlay();  
List<Overlay> listOfOverlays = mapView.getOverlays();  
listOfOverlays.clear();  
listOfOverlays.add(mapOverlay);
```

5.1.6. Geocoding and reverse geocoding

- As mentioned in the preceding section, if you know the latitude and longitude of a location, you can find out its address using a process known as reverse geocoding.
- Google Maps in Android supports this via the `Geocoder` class.
- The following code snippet shows how you can retrieve the address of a location just touched using the `getFromLocation()` method:

Example:

```
import android.location.Address;
import android.location.Geocoder;
import java.util.Locale;
import java.io.IOException;
//...
@Override
public boolean onTouchEvent(MotionEvent event, MapView mapView)
{
    //---when user lift his finger---
    if(event.getAction() == 1){
        GeoPoint p = mapView.getProjection().fromPixels(
            (int) event.getX(),
            (int) event.getY());
        /*
        Toast.makeText(getBaseContext(),
        "Location: "+
        p.getLatitudeE6()/1E6 + ", " +
        p.getLongitudeE6()/1E6,
        Toast.LENGTH_SHORT).show();
        */

        Geocoder geoCoder = new Geocoder(
            getBaseContext(), Locale.getDefault());
        try {
            List<Address> addresses = geoCoder.getFromLocation(
                p.getLatitudeE6() / 1E6,
                p.getLongitudeE6() / 1E6, 1);
            String add = "";
```

```

        if (addresses.size() > 0)
        {
            for (int i=0; i<addresses.get(0).getMaxAddressLineIndex();
                i++)
                add += addresses.get(0).getAddressLine(i) + "\n";
        }
        Toast.makeText (getBaseContext(), add,
Toast.LENGTH_SHORT).show();
    }
    catch (IOException e) {
        e.printStackTrace();
    }
    return true;
}
return false;
}
}

```

Working Principle:

- The Geocoder object converts the latitude and longitude into an address using the `getFromLocation()` method. Once the address is obtained, you display it using the Toast class.
- If you know the address of a location but want to know its latitude and longitude, you can do so via geocoding.
- Again, you can use the Geocoder class for this purpose. The following code shows how you can find the exact location of the Empire State Building by using the `getFromLocationName()` method:

//---geo-coding---

```

Geocoder geoCoder = new Geocoder(this, Locale.getDefault());
try {
    List<Address> addresses = geoCoder.getFromLocationName(
        "empire state building" , 5);
    String add = "";
    if (addresses.size() > 0) {

```

```

p = new GeoPoint(
    (int ) (addresses.get(0).getLatitude() * 1E6),
    (int ) (addresses.get(0).getLongitude() * 1E6));
mc.animateTo(p);
mapView.invalidate();
}
} catch (IOException e) {
    e.printStackTrace();
}
}

```



5.2. Content Provider

5.2.1. Sharing data in Android

- In Android, using a content provider is the recommended way to share data across packages.
- Think of a content provider as a data store.
- How it stores its data is not relevant to the application using it; what is important is how packages can access the data stored in it using a consistent programming interface.
- A content provider behaves very much like a database ,you can query it, edit its content, as well as add or delete its content. However, unlike a database, a content provider can use different ways to store its data.
- The data can be stored in a database, in files, or even over a network.
- Android ships with many useful content providers, including the following:
 - ❖ **Browser** — Stores data such as browser bookmarks, browser history, and so on
 - ❖ **CallLog** — Stores data such as missed calls, call details, and so on
 - ❖ **Contacts** — Stores contact details
 - ❖ **MediaStore** — Stores media files such as audio, video and images

❖ **Settings** — Stores the device's settings and preferences

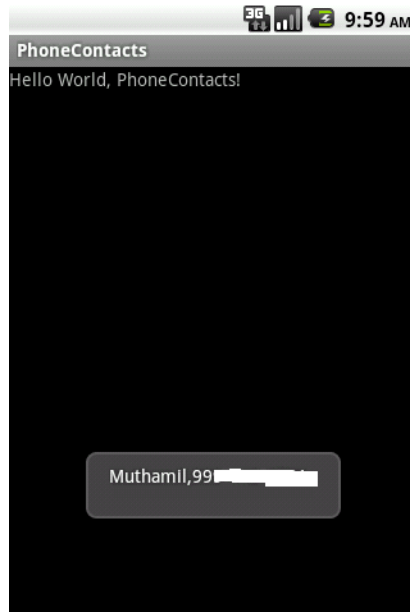
- Besides the many built-in content providers, you can also create your own content providers.
- To query a content provider, you specify the query string in the form of a URI, with an optional specifier for a particular row.
- The format of the query URI is as follows:

<standard_prefix>://<authority>/<data_path>/<id>

- The various parts of the URI are as follows:
 - The *standard prefix* for content providers is always `content://`.
 - The *authority* specifies the name of the content provider. An example would be `contacts` for the built-in Contacts content provider. For third-party content providers, this could be the fully qualified name, such as `com.wrox.provider` or `net.learn2develop.provider`.
 - The *data path* specifies the kind of data requested. For example, if you are getting all the contacts from the Contacts content provider, then the data path would be `people`, and the URI would look like this: `content://contacts/people`.
 - The *id* specifies the specific record requested. For example, if you are looking for contact number 2 in the Contacts content provider, the URI would look like this: `content://contacts/people/2`.

QUERY STRING	DESCRIPTION
<code>content://media/internal/images</code>	Returns a list of all the internal images on the device
<code>content://media/external/images</code>	Returns a list of all the images stored on the external storage (e.g., SD card) on the device
<code>content://call_log/calls</code>	Returns a list of all calls registered in the Call Log
<code>content://browser/bookmarks</code>	Returns a list of bookmarks stored in the browser

5.2.2.Phone Contacts:



Set Permission

AndroidManifest.xml

```
<uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission android:name="android.permission.WRITE_CONTACTS" />
package com.phonecontacts;
import android.app.Activity;
import android.database.Cursor;
import android.os.Bundle;
import android.provider.ContactsContract;
import android.widget.Toast;
```

```
public class PhoneContacts extends Activity {
    /** Called when the activity is first created. */
    int count=0;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        try{
            Cursor cursor = getContentResolver().query(ContactsContract.
                CommonDataKinds.Phone.CONTENT_URI, null, null,null,
            null);
            while (cursor.moveToNext())
            {
                String name =cursor.getString(cursor.getColumnIndex
                    (ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME));
                String phoneNumber = cursor.getString(cursor.getColumnIndex
                    (ContactsContract.CommonDataKinds.Phone.NUMBER));
                String data =name+", "+phoneNumber+"\n";
```

```

        Toast.makeText(getBaseContext(), data, Toast.LENGTH_LONG).show();
        count=count +1;
    }
} catch(Exception e)
{
    Toast.makeText(getBaseContext(), e.getMessage(),
Toast.LENGTH_LONG).show();
}
    Toast.makeText(getBaseContext(), "Total Contacts =" + count,
        Toast.LENGTH_LONG).show();
}
}

```

```

Cursor cursor = getContentResolver().query(ContactsContract.
        CommonDataKinds.Phone.CONTENT_URI, null, null,null,
        null);

```

public ContentResolver getContentResolver ()

Return a ContentResolver instance for your application's package.

1.1.1.1 public final Cursor query (Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder)

Query the given URI, returning a Cursor over the result set.

For best performance, the caller should follow these guidelines:

- Provide an explicit projection, to prevent reading data from storage that aren't going to be used.
- Use question mark parameter markers such as 'phone=?' instead of explicit values in the selection parameter, so that queries that differ only by those values will be recognized as the same for caching purposes.

1.1.1.1.1 Parameters

uri	The URI, using the content:// scheme, for the content to retrieve.
projection	A list of which columns to return. Passing null will return all columns, which is inefficient.
selection	A filter declaring which rows to return, formatted as an SQL WHERE clause (excluding the WHERE itself). Passing null will return all rows for the given URI.
selectionArgs	You may include ?s in selection, which will be replaced by the values from selectionArgs, in the order that they appear in the selection. The values will be bound as Strings.
sortOrder	How to order the rows, formatted as an SQL ORDER BY clause (excluding the ORDER BY itself). Passing null will use the default sort order, which may be unordered.

1.1.1.1.2 Returns

- A Cursor object, which is positioned before the first entry, or null

5.3.Data Persistence

5.3.1.Programmatically Retrieving and Modifying the Preferences valuesPreferences

- Android has a concept of shared preferences using which application preferences data can be stored persistently.
- That means the data or state won't be lost until the application is uninstalled. The preferences data can be stored as key/value pairs and are available across all the Activities of the given application or can also be restricted to a particular Activity.


SharedPreferences

- Using the SharedPreferences interface implementations we can store persistent sets of data in the filesystem. The data will be available across application restarts or even device stop/start.
- Shared Preferences can be stored at 2 levels – activity or application. To get a SharedPreferences object for your activity or application in order to start storing, retrieving and updating data there are two methods:
 - `getSharedPreferences()` – Application-wide preferences file identified by the name passed to it as the first argument.
 - `getPreferences()` – Activity-level preferences object where no name is specified as there will be only one file for an Activity.

Working Principle:

`SharedPreferences pref = getSharedPreferences("MyPrefs", Context.MODE_PRIVATE);`
You can store as many shared preferences as you want for your application and all of them will keep on getting saved in the XML file named `MyPrefs` which is usually located at `/data/data/[package name]/shared_prefs/MyPrefs.xml`.

You can browse that in the File Explorer in the DDMS view or if your device is rooted then go ahead and explore it in \$ adb shell.



▼ com.pycitup.pyc	2014-10-27 04:48 drwxr-x--x
▶ app_Parse	2014-10-26 07:59 drwxrwx--x
cache	2014-10-18 02:15 drwxrwx--x
lib	2014-10-27 04:48 lrwxrwxrwx -> /data/...
▼ shared_prefs	2014-10-26 13:52 drwxrwx--x
GestureActivity.xml	106 2014-10-26 13:52 -rw-rw----
MyPrefs.xml	106 2014-10-26 13:48 -rw-rw----
com.pycitup.pyc_preferences.xml	110 2014-10-26 07:59 -rw-rw----
▶ com.svox.pico	2014-06-14 05:46 drwxr-x--x
▶ jp.co.omronsoft.openwnn	2014-06-14 05:44 drwxr-x--x

The image shows my app specific data (including shared preferences) in DDMS. `com.pycitup.pyc` is my application's package name.

Storing Preferences

- Once you've decided whether you want application preferences or activity one, then you'll start storing data into it. Once you've a valid SharedPreferences object you call `edit()` method on it to fetch a `SharedPreferences.Editor` object whose public method will allow us to start writing data to the file.
- To store primitive data it has various methods like `putBoolean()`, `putFloat()`, `putInt()`, `putLong()` and `putString()`.

Let's try storing some values:

```
SharedPreferences pref = getSharedPreferences("MyPrefs", Context.MODE_PRIVATE);
```

```
// We need an editor object to make changes
```

```
SharedPreferences.Editor edit = pref.edit();
```

```
// Set/Store data
```

```
edit.putString("username", "Rishabh");
```

```
edit.putBoolean("logged_in", true);
```

```
// Commit the changes
```

```
edit.commit();
```

We stored 2 key-value pairs. This is how our /data/data/[package name]/shared_prefs/MyPrefs.xml should look like now:

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
```

```
<map>
```

```
  <string name="username">Rishabh</string>
```

```
  <boolean name="logged_in" value="true" />
```

```
</map>
```

Notice we had to use the commit() method to commit our changes. There's another method to do the same which is apply() but that is asynchronous and won't report failures.

Modifying Preferences

- Updating the preferences is similar to the setting them that we just learnt a bit back. Get the SharedPreferences.Editor object, set values using the put*() methods and then commit your changes.

```
SharedPreferences pref = getSharedPreferences("MyPrefs", Context.MODE_PRIVATE);
```

```
// We need an editor object to make changes
```

```
SharedPreferences.Editor edit = pref.edit();
```

```
// Set/Store data
```

```
edit.putString("username", "CodeTheory");
```

```
edit.putBoolean("logged_in", false);
```

```
// Commit the changes
```

```
edit.commit();
```

Same code but different values. The new XML contents looks like this:

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
```

```
<map>
```

```
  <boolean name="logged_in" value="false" />
```

```
  <string name="username">CodeTheory</string>
```

</map>

Retrieving Preferences

- Fetching the preferences is done directly on the SharedPreferences object. So SharedPreferences.Editor is not required.
- There are several `get` methods for this job like getBoolean(), getFloat(), getInt(), getLong() and getString(). All of them accept two arguments where the first is the name of the key while the second non-optional one is the default value to return if the preference does not exist (is undefined).

```
SharedPreferences pref = getSharedPreferences("MyPrefs", Context.MODE_PRIVATE);
String username = pref.getString("username", "");
boolean logged_in = String.valueOf(pref.getBoolean("logged_in", false));
```

```
Log.d(TAG, username);
Log.d(TAG, String.valueOf(logged_in));
```

The code logged CodeTheory and false in separate lines.

Deleting Preferences

Deleting a particular preference is super simple. Just call the remove() method and commit your changes.

```
// Remove a particular key
pref.remove("username");
```

```
// Commit changes
pref.commit();
```

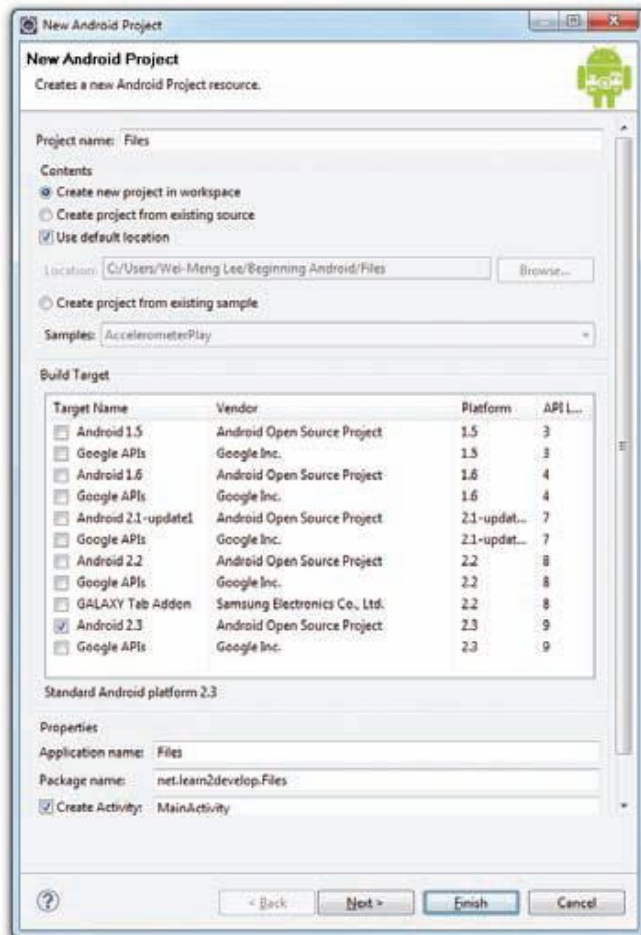
5.3.2. Saving to Internal Storage

The first way to save files in your Android application is to write to the device's internal storage.

1. Using Eclipse, create an Android project and name it as
2. In the main.xml file, add the following statements in bold:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <TextView
        android:layout_width="fill_parent"
```

```
    android:layout_height="wrap_content"
    android:text="Please enter some text"
/>
<EditText
    android:id="@+id/txtText1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
<Button
    android:id="@+id/btnSave"
    android:text="Save"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
<Button
    android:id="@+id/btnLoad"
    android:text="Load"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
</LinearLayout>
```



3. In the MainActivity.java file, add the following statements in bold:

```
package net.learn2develop.Files;

import android.app.Activity;
import android.view.View;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
public class MainActivity extends Activity {
private EditText textBox;
private static final int READ_BLOCK_SIZE = 100;
```

```

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    textBox = (EditText) findViewById(R.id.txtText1);
    Button saveBtn = (Button) findViewById(R.id.btnSave);
    Button loadBtn = (Button) findViewById(R.id.btnLoad);
    saveBtn.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
    String str = textBox.getText().toString();
    try
    {
    FileOutputStream fOut =
    openFileOutput("textfile.txt",
    MODE_WORLD_READABLE);
    OutputStreamWriter osw = new
    OutputStreamWriter(fOut);
    //---write the string to the file---
    osw.write(str);
    osw.flush();
    osw.close();
    //---display file saved message---
    Toast.makeText(getApplicationContext(),
    "File saved successfully!",
    Toast.LENGTH_SHORT).show();
    //---clears the EditText---
    textBox.setText("");
    }
    catch (IOException ioe)
    {
    ioe.printStackTrace();
    }
    }

```

```

});
loadBtn.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        try
        {
            FileInputStream fIn =
            openFileInput("textfile.txt");
            InputStreamReader isr = new
            InputStreamReader(fIn);
            char[] inputBuffer = new char[READ_BLOCK_SIZE];
            String s = "";
            int charRead;
            while ((charRead = isr.read(inputBuffer))>0)
            {
                //---convert the chars to a String---
                String readString =
                String.valueOf(inputBuffer, 0,
                charRead);
                s += readString;
                inputBuffer = new char[READ_BLOCK_SIZE];
            }
            //---set the EditText to the text that has been
            // read---
            textBox.setText(s);
            Toast.makeText(getApplicationContext(),
            "File loaded successfully!",
            Toast.LENGTH_SHORT).show();
        }
        catch (IOException ioe) {
            ioe.printStackTrace();
        }
    }
});
}

```

}

4. Press F11 to debug the application on the Android Emulator.
5. Type some text into the EditText view and then click the Save button.



6. If the file is saved successfully, you will see the Toast class displaying the “File saved successfully!” message. The text in the EditText view should disappear.
 7. Click the Load button and you should see the string appearing in the EditText view again.
- This confirms that the text is saved correctly.

Working Principle:

- To save text into a file, you use the `FileOutputStream` class. The `openFileOutput()` method opens a named file for writing, with the mode specified.
- In this example, you use the **MODE_WORLD_READABLE** constant to indicate that the file is readable by all other applications:

`FileOutputStream fOut`

`=openFileOutput(“textfile.txt”,MODE_WORLD_READABLE);`

- Apart from the **MODE_WORLD_READABLE** constant, you can select from the following:

MODE_PRIVATE (file can only be accessed by the application that created it),

MODE_APPEND (for appending to an existing file),

and MODE_WORLD_WRITEABLE (all other applications have write access to the file).

- To convert a character stream into a byte stream, you use an instance of the `OutputStreamWriter` class, by passing it an instance of the `FileOutputStream` object:

```
OutputStreamWriter osw = new OutputStreamWriter(fOut);
```

MODE_PRIVATE (file can only be accessed by the application that created it),

MODE_APPEND (for appending to an existing file),

and MODE_WORLD_WRITEABLE (all other applications have write access to the file).

- You then use its `write()` method to write the string to the file. To ensure that all the bytes are written to the file, use the `flush()` method. Finally, use the `close()` method to close the file:

```
osw.write(str);
```

```
osw.flush();
```

```
osw.close();
```

- To read the content of a file, you use the `FileInputStream` class, together with the `InputStreamReader` class:

```
FileInputStream fIn = openFileInput("textfile.txt");
```

```
InputStreamReader isr = new InputStreamReader(fIn);
```

- As you do not know the size of the file to read, the content is read in blocks of 100 characters into a buffer (character array). The characters read are then copied into a `String` object:

```
char[] inputBuffer = new char[READ_BLOCK_SIZE];
```

```
String s = "";int charRead;
```

```
while ((charRead = isr.read(inputBuffer)) > 0)
```

```
{
```

```
//---convert the chars to a String---
```

```
String readString = String.valueOf(inputBuffer, 0, charRead);
```

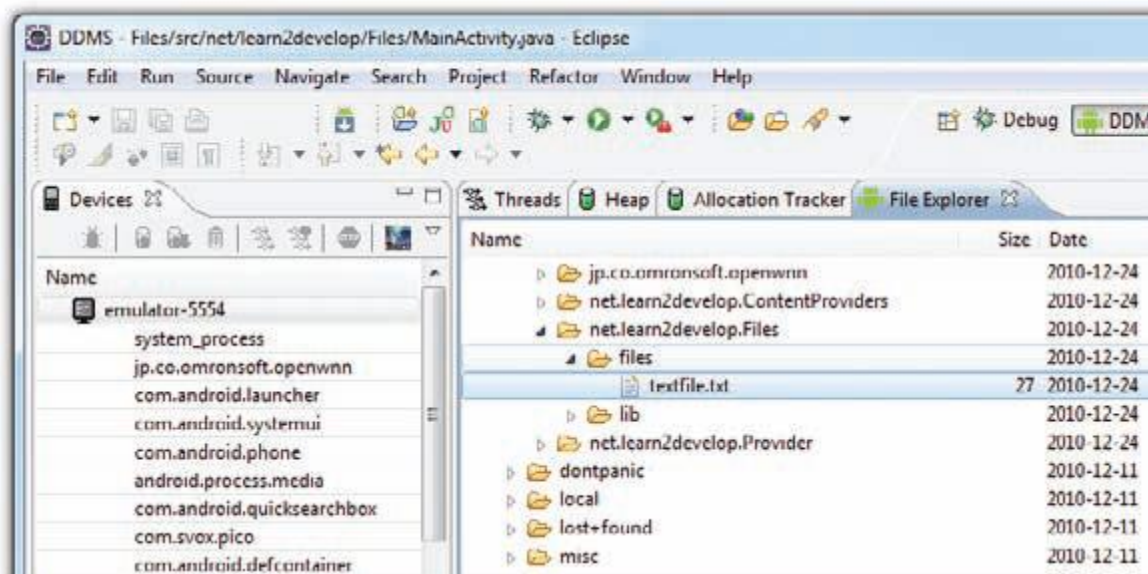
```
s += readString;
```

```
inputBuffer = new char[READ_BLOCK_SIZE];
```

```
}
```

- The `read()` method of the `InputStreamReader` object reads the number of characters read and returns -1 if the end of the file is reached.

- When testing this application on the Android Emulator, you can use the DDMS to verify that the application did indeed save the file into the application's files directory



5.3.2. Saving to External Storage (SD Card)

- Sometimes, Information would be useful to save them to external storage (such as an SD card) because of its larger capacity, as well as the capability to share the files easily with other users (by removing the SD card and passing it to somebody else).
- Using the project created in the previous section as the example, to save the text entered by the user in the SD card, modify the `onClick()` method of the Save button as shown in bold here:

```
saveBtn.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        String str = textBox.getText().toString();
        try
        {
            //---SD Card Storage---
            File sdCard = Environment.getExternalStorageDirectory();
            File directory = new File (sdCard.getAbsolutePath() +
            "/MyFiles");
            directory.mkdirs();
            File file = new File(directory, "textfile.txt");
            FileOutputStream fOut = new FileOutputStream(file);
            OutputStreamWriter osw = new
```

```

OutputStreamWriter(fOut);
//---write the string to the file---
osw.write(str);
osw.flush();
osw.close();
//---display file saved message---
Toast.makeText(getApplicationContext(),
"File saved successfully!",Toast.LENGTH_SHORT).show();
//---clears the EditText---
textBox.setText("");
}
catch (IOException ioe)
{
ioe.printStackTrace();
}
}
});

```

- The preceding code uses the `getExternalStorageDirectory()` method to return the full path to the external storage. Typically, it should return the `"/sdcard"` path for a real device, and `"/mnt/sdcard"` for an Android Emulator.
- However, you should never try to hardcode the path to the SD card, as manufacturers may choose to assign a different path name to the SD card. Hence, be sure to use the `getExternalStorageDirectory()` method to return the full path to the SD card.
- You then create a directory called `MyFiles` in the SD card. Finally, you save the file into this directory. To load the file from the external storage, modify the `onClick()` method for the Load button:

```

loadBtn.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        try
        {
            //---SD Storage---
            File sdCard = Environment.getExternalStorageDirectory();

```

```

File directory = new File (sdCard.getAbsolutePath() +
"/MyFiles");
File file = new File(directory, "textfile.txt");
FileInputStream fIn = new FileInputStream(file);
InputStreamReader isr = new InputStreamReader(fIn);
char[] inputBuffer = new char[READ_BLOCK_SIZE];
String s = "";
int charRead;
while ((charRead = isr.read(inputBuffer))>0)
{
//---convert the chars to a String---
String readString =
String.valueOf(inputBuffer, 0, charRead);
s += readString;
inputBuffer = new char[READ_BLOCK_SIZE];
}
//---set the EditText to the text that has been
// read---
textBox.setText(s);
Toast.makeText(getApplicationContext(),
"File loaded successfully!",
Toast.LENGTH_SHORT).show();
}
catch (IOException ioe) {
ioe.printStackTrace();
}
});

```

- Note that in order to write to the external storage, you need to add the WRITE_EXTERNAL_STORAGE permission in your Android Manifest.xml file:

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="net.learn2develop.Files"
android:versionCode="1"

```

```

    android:versionName="1.0">
    <application android:icon="@drawable/icon"
    android:label="@string/app_name">
    <activity android:name=".MainActivity"
    android:label="@string/app_name">
    <intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
    </activity>
    </application>
    <uses-sdk android:minSdkVersion="9" />
    <uses-permission
    android:name="android.permission.WRITE_EXTERNAL_STORAGE"
    >
    </uses-permission>

```

5.3.4. Creating and Using Databases

- All the techniques you have seen are useful for saving simple sets of data. For saving relational data, using a database is much more efficient.
- For example, if you want to store the results of all the students in a school, it is much more efficient to use a database to represent them because you can use database querying to retrieve the results of the specific students.
- Moreover, using databases enables you to enforce data integrity by specifying the relationships between different sets of data. Android uses the SQLite database system.
- The database that you create for an application is only accessible to itself; other applications will not be able to access it.
- For Android, the SQLite database that you create programmatically in an application is always stored in the /data/data/<package_name>/databases folder.

Creating the DBAdapter Helper Class

- A good practice for dealing with databases is to create a helper class to encapsulate all the complexities of accessing the data so that it is transparent to the calling code.

- In this example, you are going to create a database named MyDB containing one table named contacts. This table will have three columns: _id,name, and email

Example:

1. Using Eclipse, create an Android project and name it Databases.
2. Add a new class file to the project and name it DBAdapter.java
3. Add the following statements in bold to the DBAdapter.java file:

```
package net.learn2develop.Databases;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;

public class DBAdapter {
    public static final String KEY_ROWID = "_id";
    public static final String KEY_NAME = "name";
    public static final String KEY_EMAIL = "email";
    private static final String TAG = "DBAdapter";
    private static final String DATABASE_NAME = "MyDB";
    private static final String DATABASE_TABLE = "contacts";
    private static final int DATABASE_VERSION = 1;
    private static final String DATABASE_CREATE =
        "create table contacts (_id integer primary key autoincrement, "
        + "name text not null, email text not null);";
    private final Context context;
    private DatabaseHelper DBHelper;
    private SQLiteDatabase db;
    public DBAdapter(Context ctx)
    {
        this.context = ctx;
        DBHelper = new DatabaseHelper(context);
    }
    private static class DatabaseHelper extends SQLiteOpenHelper
```

```

{
DatabaseHelper(Context context)
{
super(context, DATABASE_NAME, null, DATABASE_VERSION);
}
@Override
public void onCreate(SQLiteDatabase db)
{
try {
db.execSQL(DATABASE_CREATE);
} catch (SQLException e) {
e.printStackTrace();
}
}
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
{
Log.w(TAG, "Upgrading database from version " + oldVersion + " to "
+ newVersion + ", which will destroy all old data");
db.execSQL("DROP TABLE IF EXISTS contacts");
onCreate(db);
}
}
//---opens the database---
public DBAdapter open() throws SQLException
{
db = DBHelper.getWritableDatabase();
return this;
}
//---closes the database---
public void close()
{
DBHelper.close();
}

```

```

//---insert a contact into the database---
public long insertContact(String name, String email)
{
    ContentValues initialValues = new ContentValues();
    initialValues.put(KEY_NAME, name);
    initialValues.put(KEY_EMAIL, email);
    return db.insert(DATABASE_TABLE, null, initialValues);
}

//---deletes a particular contact---
public boolean deleteContact(long rowId)
{
    return db.delete(DATABASE_TABLE, KEY_ROWID + "=" + rowId, null) > 0;
}

//---retrieves all the contacts---
public Cursor getAllContacts()
{
    return db.query(DATABASE_TABLE, new String[] {KEY_ROWID, KEY_NAME,
    KEY_EMAIL}, null, null, null, null, null);
}

//---retrieves a particular contact---
public Cursor getContact(long rowId) throws SQLException
{
    Cursor mCursor =
    db.query(true, DATABASE_TABLE, new String[] {KEY_ROWID,
    KEY_NAME, KEY_EMAIL}, KEY_ROWID + "=" + rowId, null,
    null, null, null, null);
    if (mCursor != null) {
        mCursor.moveToFirst();
    }
    return mCursor;
}

```

//---updates a contact---

```
public boolean updateContact(long rowId, String name, String email)  
{  
    ContentValues args = new ContentValues();  
    args.put(KEY_NAME, name);  
    args.put(KEY_EMAIL, email);  
    return db.update(DATABASE_TABLE, args, KEY_ROWID + "=" + rowId, null) > 0;  
}  
}
```

Working Principle:

- You first defined several constants to contain the various fields for the table that you are going to create in your database:

```
public static final String KEY_ROWID = "_id";  
public static final String KEY_NAME = "name";  
public static final String KEY_EMAIL = "email";  
private static final String TAG = "DBAdapter";  
private static final String DATABASE_NAME = "MyDB";  
private static final String DATABASE_TABLE = "contacts";  
private static final int DATABASE_VERSION = 1;  
private static final String DATABASE_CREATE =  
    "create table contacts (_id integer primary key autoincrement, "  
    + "name text not null, email text not null);";
```

- In particular, the DATABASE_CREATE constant contains the SQL statement for creating the contacts table within the MyDB database.
- Within the DBAdapter class, you also extend the SQLiteOpenHelper class, which is a helper class in Android to manage database creation and version management.
- In particular, you override the onCreate() and onUpgrade() methods:

```
public class DBAdapter {  
    public static final String KEY_ROWID = "_id";  
    public static final String KEY_NAME = "name";  
    public static final String KEY_EMAIL = "email";
```

```

private static final String TAG = "DBAdapter";
private static final String DATABASE_NAME = "MyDB";
private static final String DATABASE_TABLE = "contacts";
private static final int DATABASE_VERSION = 1;
private static final String DATABASE_CREATE =
"create table contacts (_id integer primary key autoincrement, "
+ "name text not null, email text not null);";
private final Context context;
private DatabaseHelper DBHelper;
private SQLiteDatabase db;
public DBAdapter(Context ctx)
{
this.context = ctx;
DBHelper = new DatabaseHelper(context);
}
private static class DatabaseHelper extends SQLiteOpenHelper
{
DatabaseHelper(Context context)
{
super(context, DATABASE_NAME, null, DATABASE_VERSION);
}
@Override
public void onCreate(SQLiteDatabase db)
{
try {
db.execSQL(DATABASE_CREATE);
} catch (SQLException e) {
e.printStackTrace();
}
}
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion)
{

```

```

Log.w(TAG, “Upgrading database from version “ + oldVersion + “ to “
+ newVersion + “, which will destroy all old data”);
db.execSQL(“DROP TABLE IF EXISTS contacts”);
onCreate(db);
}
}

```

- The onCreate() method creates a new database if the required database is not present. The onUpgrade() method is called when the database needs to be upgraded.
- This is achieved by checking the value defined in the DATABASE_VERSION constant. For this implementation of the onUpgrade() method, you simply drop the table and create it again.
- You can then define the various methods for opening and closing the database, as well as the methods for adding/editing/deleting rows in the table:

```

public class DBAdapter {
    //...
    //...
    //---opens the database---
    public DBAdapter open() throws SQLException
    {
    db = DBHelper.getWritableDatabase();
    return this;
    }
    //---closes the database---
    public void close()
    {
    DBHelper.close();
    }
    //---insert a contact into the database---
    public long insertContact(String name, String email)
    {
    ContentValues initialValues = new ContentValues();
    initialValues.put(KEY_NAME, name);

```

```

initialValues.put(KEY_EMAIL, email);
return db.insert(DATABASE_TABLE, null, initialValues);
}

//---deletes a particular contact---
public boolean deleteContact(long rowId)
{
return db.delete(DATABASE_TABLE, KEY_ROWID + "=" + rowId,
null) > 0;
}

//---retrieves all the contacts---
public Cursor getAllContacts()
{
return db.query(DATABASE_TABLE, new String[] {KEY_ROWID,
KEY_NAME,
KEY_EMAIL}, null, null, null, null, null);
}

//---retrieves a particular contact---
public Cursor getContact(long rowId) throws SQLException
{
Cursor mCursor =
db.query(true, DATABASE_TABLE, new String[] {KEY_ROWID,
KEY_NAME, KEY_EMAIL}, KEY_ROWID + "=" + rowId, null,
null, null, null, null);
if (mCursor != null) {
mCursor.moveToFirst();
}
return mCursor;
}

//---updates a contact---
public boolean updateContact(long rowId, String name, String email)
{
ContentValues args = new ContentValues();
args.put(KEY_NAME, name);
args.put(KEY_EMAIL, email);

```

```

        return db.update(DATABASE_TABLE, args, KEY_ROWID + "=" +
            rowId, null) > 0;
    }
}

```

- Notice that Android uses the Cursor class as a return value for queries. Think of the Cursor as a pointer to the result set from a database query. Using Cursor enables Android to more efficiently manage rows and columns as needed.
- You use a Content Values object to store key/value pairs. Its put() method enables you to insert keys with values of different data types.

5.4. Android Services

5.4.1. Consuming Web Services using HTTP:

- Using the HTTP protocol, you can perform a wide variety of tasks, such as downloading web pages from a web server, downloading binary data, and so on.
- The following Try It Out creates an Android project so that you can use the HTTP protocol to connect to the Web to download all sorts of data.

Example:

1. Using Eclipse, create a new Android project and name it **Networking**.

2. Add the following statement in bold to the AndroidManifest.xml file:

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="net.learn2develop.Networking"
android:versionCode="1"
android:versionName="1.0">
<application android:icon="@drawable/icon" android:label="@string/app_name">
<activity android:name=".MainActivity"
android:label="@string/app_name">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>

```

```

</activity>
</application>
<uses-sdk android:minSdkVersion="8" />
<uses-permission android:name="android.permission.INTERNET"></uses-
permission>
</manifest>

```

3. Import the following namespaces in the MainActivity.java file:

```

package net.learn2develop.Networking;
import android.app.Activity;
import android.os.Bundle;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLConnection;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.widget.ImageView;
import android.widget.Toast;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
public class MainActivity extends Activity {
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
}

```

```
}
```

4. Define the `OpenHttpConnection()` method in the `MainActivity.java` file:

```
public class MainActivity extends Activity {  
    private InputStream OpenHttpConnection(String urlString)  
    throws IOException  
    {  
        InputStream in = null;  
        int response = -1;  
        URL url = new URL(urlString);  
        URLConnection conn = url.openConnection();  
        if (!(conn instanceof HttpURLConnection))  
        throw new IOException("Not an HTTP connection");  
        try{  
            HttpURLConnection httpConn = (HttpURLConnection) conn;  
            httpConn.setAllowUserInteraction(false);  
            httpConn.setInstanceFollowRedirects(true);  
            httpConn.setRequestMethod("GET");  
            httpConn.connect();  
            response = httpConn.getResponseCode();  
            if (response == HttpURLConnection.HTTP_OK) {  
                in = httpConn.getInputStream();  
            }  
        }  
        catch (Exception ex)  
        {  
            throw new IOException("Error connecting");  
        }  
        return in;  
    }  
    /** Called when the activity is first created. */  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
    }  
}
```

```
}  
}
```

Working Principle:

- your application needs the INTERNET permission; hence, the first thing you do is add the permission in the AndroidManifest.xml file.
- You then define the OpenHttpConnection() method, which takes a URL string and returns an InputStream object. Using an InputStream object, you can download the data by reading bytes from the stream object.
- In this method, you made use of the HttpURLConnection object to open an HTTP connection with a remoteURL. You set all the various properties of the connection, such as the request method, and so on:

```
HttpURLConnection httpConn = (HttpURLConnection) conn;  
httpConn.setAllowUserInteraction(false);  
httpConn.setInstanceFollowRedirects(true);  
httpConn.setRequestMethod("GET");
```

- After you try to establish a connection with the server, you get the HTTP response code from it. If the connection is established (via the response code HTTP_OK), then you proceed to get an InputStream object from the connection:

```
httpConn.connect();  
response = httpConn.getResponseCode();  
if (response == HttpURLConnection.HTTP_OK) {  
    in = httpConn.getInputStream();  
}
```

- Using the InputStream object, you can then start to download the data from the server.

5.4.2. Downloading Binary Data

- One of the common tasks you need to perform is downloading binary data from the Web.
- For example, you may want to download an image from a server so that you can display it in your application.

Example:

1. Using the same project created earlier, add the following statements in bold to the main.xml file:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <ImageView
        android:id="@+id/img"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center" />
</LinearLayout>
```

2. Add the following statements in bold to the MainActivity.java file:

```
public class MainActivity extends Activity {
    ImageView img;
    private InputStream OpenHttpConnection(String urlString)
    throws IOException
    {
        //...
    }
    private Bitmap DownloadImage(String URL)
    {
        Bitmap bitmap = null;
        InputStream in = null;
        try {
            in = OpenHttpConnection(URL);
            bitmap = BitmapFactory.decodeStream(in);
            in.close();
        } catch (IOException e1) {
            Toast.makeText(this, e1.getLocalizedMessage(),
            Toast.LENGTH_LONG).show();
        }
    }
}
```

```

e1.printStackTrace();
}
return bitmap;
}
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    //---download an image---
    Bitmap bitmap =
    DownloadImage(
    “http://www.streetcar.org/mim/cable/images/cable-01.jpg”);
    img = (ImageView) findViewById(R.id.img);
    img.setImageBitmap(bitmap);
}
}

```

3. Press F11 to debug the application on the Android Emulator.

Working Principle:

- The DownloadImage() method takes the URL of the image to download and then opens the connection to the server using the OpenHttpConnection() method that you have defined earlier.
- Using the InputStream object returned by the connection, the decodeStream() method from the BitmapFactory class is used to download and decode the data into a Bitmap object. The DownloadImage() method returns a Bitmap object.



5.4.3. Downloading text Files

- Besides downloading binary data, you can also download plain-text files.
- For example, you might be writing an RSS Reader application and hence need to download RSS XML feeds for processing.

Example:

1. Using the same project created earlier, add the following statements in bold to the MainActivity.java file:

```
public class MainActivity extends Activity {
    ImageView img;
    private InputStream OpenHttpConnection(String urlString)
    throws IOException
    {
        //...
    }
    private Bitmap DownloadImage(String URL)
    {
        //...
    }
}
```

```

private String DownloadText(String URL)
{
    int BUFFER_SIZE = 2000;
    InputStream in = null;
    try {
        in = OpenHttpConnection(URL);
    } catch (IOException e1) {
        Toast.makeText(this, e1.getLocalizedMessage(),
            Toast.LENGTH_LONG).show();
        e1.printStackTrace();
        return "";
    }
    InputStreamReader isr = new InputStreamReader(in);
    int charRead;
    String str = "";
    char[] inputBuffer = new char[BUFFER_SIZE];
    try {
        while ((charRead = isr.read(inputBuffer))>0)
        {
            //---convert the chars to a String---
            String readString =
                String.valueOf(inputBuffer, 0, charRead);
            str += readString;
            inputBuffer = new char[BUFFER_SIZE];
        }
        in.close();
    } catch (IOException e) {
        Toast.makeText(this, e.getLocalizedMessage(),
            Toast.LENGTH_LONG).show();
        e.printStackTrace();
        return "";
    }
    return str;
}

```

```

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    //---download an image---
    Bitmap bitmap =
    DownloadImage(
    "http://www.streetcar.org/mim/cable/images/cable-01.jpg");
    img = (ImageView) findViewById(R.id.img);
    img.setImageBitmap(bitmap);
    //---download an RSS feed---
    String str = DownloadText(
    "http://www.appleinsider.com/appleinsider.rss");
    Toast.makeText(getApplicationContext(), str,
    Toast.LENGTH_SHORT).show();
}
}

```

2. Press F11 to debug the application on the Android Emulator. Figure 8-10 shows the RSS feed

downloaded and displayed using the Toast class.



Working Principle:

- The DownloadText() method takes an URL of the text file to download and then returns the string of the text file downloaded.
- It basically opens an HTTP connection to the server and then uses an InputStreamReader object to read each character from the stream and save it in a String object.

5.4.4. Accessing Web Services

- Once the Web service returns a result in XML, you will extract the relevant parts and display its content using the Toast class.
- For this example, the web method you will be using is from <http://services.aonaware.com/>
- DictService/DictService.asmx?op=Define. This web method is from a Dictionary Web service that returns the definitions of a given word.

The web method takes a request in the following format:

GET /DictService/DictService.asmx/Define?word=string HTTP/1.1

Host: services.aonaware.com

HTTP/1.1 200 OK

Content-Type: text/xml; charset=utf-8

Content-Length: length

It returns a response in the following format:

```
<?xml version="1.0" encoding="utf-8"?>
<WordDefinition xmlns="http://services.aonaware.com/webservices/">
  <Word>string</Word>
  <Definitions>
    <Definition>
      <Word>string</Word>
      <Dictionary>
        <Id>string</Id>
        <Name>string</Name>
      </Dictionary>
    <WordDefinition>string</WordDefinition>
    </Definition>
    <Definition>
      <Word>string</Word>
      <Dictionary>
        <Id>string</Id>
        <Name>string</Name>
      </Dictionary>
    <WordDefinition>string</WordDefinition>
    </Definition>
  </Definitions>
</WordDefinition>
```

Example:

1. Using the same project created earlier, add the following statements in bold to the MainActivity.java file:

```
public class MainActivity extends Activity {
  ImageView img;
  private InputStream OpenHttpConnection(String urlString)
  throws IOException
{
  //...
```

```

}
private Bitmap DownloadImage(String URL)
{
//...
}
private String DownloadText(String URL)
{
//...
}
private void WordDefinition(String word) {
    InputStream in = null;
    try {
        in = OpenHttpConnection(
            "http://services.aonaware.com/DictService/DictService.asmx/Define?word=" + word);
        Document doc = null;
        DocumentBuilderFactory dbf =
            DocumentBuilderFactory.newInstance();
        DocumentBuilder db;
        try {
            db = dbf.newDocumentBuilder();
            doc = db.parse(in);
        } catch (ParserConfigurationException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        doc.getDocumentElement().normalize();
        //---retrieve all the <Definition> nodes---
        NodeList itemNodes =
            doc.getElementsByTagName("Definition");
        String strDefinition = "";
        for (int i = 0; i < definitionElements.getLength(); i++) {

```

```

Node itemNode = definitionElements.item(i);
if (itemNode.getNodeType() == Node.ELEMENT_NODE)
{
    ///---convert the Node into an Element---
    Element definitionElement = (Element) itemNode;
    ///---get all the <WordDefinition> elements under
    // the <Definition> element---
    NodeList wordDefinitionElements =
    (definitionElement).getElementsByTagName(
    "WordDefinition");
    strDefinition = "";
    for (int j = 0; j < wordDefinitionElements.getLength(); j++) {
        ///---convert a <WordDefinition> Node into an Element---
        Element wordDefinitionElement =
        (Element) wordDefinitionElements.item(j);
        ///---get all the child nodes under the
        // <WordDefinition> element---
        NodeList textNodes =
        ((Node) wordDefinitionElement).getChildNodes();
        strDefinition +=
        ((Node) textNodes.item(0)).getNodeValue() + ". ";
    }
    ///---display the title---
    Toast.makeText(getBaseContext(),strDefinition,
    Toast.LENGTH_SHORT).show();
}
}
} catch (IOException e1) {
    Toast.makeText(this, e1.getLocalizedMessage(),
    Toast.LENGTH_LONG).show();
    e1.printStackTrace();
}
}

/** Called when the activity is first created. */

```

```

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    //---download an image---
    Bitmap bitmap =
    DownloadImage(
    "http://www.streetcar.org/mim/cable/images/cable-01.jpg");
    img = (ImageView) findViewById(R.id.img);
    img.setImageBitmap(bitmap);
    //---download an RSS feed---
    String str = DownloadText(
    Toast.makeText(getApplicationContext(), str,
    Toast.LENGTH_SHORT).show();
    //---access a Web service using GET---
    WordDefinition("Apple");
}
}

```

2. Press F11 to debug the application on the Android Emulator. Figure 8-11 shows the result of the

Web service call being parsed and then displayed using the Toast class.



Working Principle:

It then uses the DocumentBuilderFactory and DocumentBuilder objects to obtain a Document (DOM) object from an XML file (which is the XML result returned by the Web service):

```
Document doc = null;
DocumentBuilderFactory dbf =
DocumentBuilderFactory.newInstance();
DocumentBuilder db;
try {
db = dbf.newDocumentBuilder();
doc = db.parse(in);
} catch (ParserConfigurationException e) {
// TODO Auto-generated catch block
e.printStackTrace();
} catch (Exception e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
```

```
doc.getDocumentElement().normalize();
```

Once the Document object is obtained, you will find all the elements with the <Definition> tag:

```
//---retrieve all the <Definition> nodes---
```

```
NodeList itemNodes =
```

```
doc.getElementsByTagName("Definition");
```

```
db = dbf.newDocumentBuilder();
```

```
doc = db.parse(in);
```

```
} catch (ParserConfigurationException e) {
```

```
// TODO Auto-generated catch block
```

```
e.printStackTrace();
```

```
} catch (Exception e) {
```

```
// TODO Auto-generated catch block
```

```
e.printStackTrace();
```

```
}
```

```
doc.getDocumentElement().normalize();
```

Once the Document object is obtained, you will find all the elements with the <Definition> tag:

```
//---retrieve all the <Definition> nodes---
```

```
NodeList itemNodes =
```

```
doc.getElementsByTagName("Definition");
```

```
db = dbf.newDocumentBuilder();
```

```
doc = db.parse(in);
```

```
} catch (ParserConfigurationException e) {
```

```
// TODO Auto-generated catch block
```

```
e.printStackTrace();
```

```
} catch (Exception e) {
```

```
// TODO Auto-generated catch block
```

```
e.printStackTrace();
```

```
}
```

```
doc.getDocumentElement().normalize();
```

Once the Document object is obtained, you will find all the elements with the <Definition> tag:

```
//---retrieve all the <Definition> nodes---
```

```
NodeList itemNodes =  
doc.getElementsByTagName("Definition");
```

Review Questions

PART-A(2 Mark Questions)

1. Define Location based services.
2. What are the two methods used for zoom control?
3. What is meant by geocoding?
4. What is meant by reverse geocoding?
5. Define Content Provider.
6. Define Android Service.
7. Define Web Service

PART-B(3 Mark Questions)

1. Explain zoom Control.
2. Explain how to navigate to a specific location.
3. Explain geocoding and reverse geocoding.
4. Explain accessing web service.
5. Explain downloading binary data.

PART-C(5&10 Mark Questions)

1. Explain how will you obtain maps API Key.(5)
2. Explain how to display the map in Android(5)
3. Explain adding markers in android(10)
4. Explain how to share data in android(5)
5. Explain how to view,modify,delete,and add contacts using Android(10)
6. Explain Storing and retrieving datas in internal and external storage using Android(10)
7. Explain how will you create and use databases using SQLite(10)
8. Explain Consuming Web services using HTTP
9. Explain downloading binary data and downloading text content using android(10)