:: EXERCISE ::

▶ **Questions for 2 marks:**

(1) What is android? Android is a Package of Application.

(2) What is Linux kernel? Linux kernel is a free and open source operating system kernel.

(3) What is android Runtime? Android Runtime is an application runtime environment used by And operating system.

(4) What is Application Framework?

(5) What is DVM? Dalvik virtual machine (DVM is an and virtual Machine.

▶ **Question for 3 marks :**

(1) What is Linux Kernel and list out its Classes?

(2) Provide any 5 features of Android 1.0.

(3) Android Platform

(4) Difference between Android 1.1 and android 1.0.

(5) What is the name of android 1.5, and its features?

▶ **Question for 5 marks :**

(1) Android Architecture.

(2) Open Handset Alliance.

(3) Android 4.4 kitkat

(4) Steps to download and install Android SDK

(5) Steps to build "hello world" application.

:: M.C.Q. ::

1. Android is Operating System?
   a. True b. False

2. Android Belongs to.....
   a. Microsoft b. Oracle c. SAP d. Google

3. Android architecture is combination of.....
   a. Linux Kernal b. Libraries
   c. Application Framework d. All of Above

4. Android API 1.0 Introduced in 23/11/2008
   a. 23/9/2008 b. 23/10/2008 d. 23/12/2008

5. Android API 1.1 Introduced in .....
   a. 9/2/2009 b. 9/3/2009 c. 9/4/2009 d. 9/10/2009

6. Android 1.5 API level 3 .....
   a. Cupcake b. Kitkat c. Honeycomb d. Jellybean

7. Android 1.6 API level 4 .....
   a. Cupcake b. Kitkat c. Honeycomb d. Donut

8. Android API 5/6/7 is Known as .....
   a. Jellybean b. Honeycomb c. Eclair d. Cupcake

9. Tethering is first introduced in .....
   a. Froyo b. Honeycomb c. Éclair d. Cupcake

10. OHA stands for .....
    a. Over hand applications b. Open Handset Alliance
    c. Open Handset Applications d. None of Above

11. Android 2.2/2.2.1/2.2.2/2.2.3 is a versions of .....
    a. Froyo b. Honeycomb c. Éclair d. Cupcake

12. NFC stands .....
    a. Near Field Commerce b. Near Friend company
    c. Near Field communication d. Near File communication

13. NFC was first introduced in .....
    a. Froyo b. Gingerbread c. Éclair d. Cupcake

14. Ice Cream Sandwich contains API level .....
    a. 14 b. 10 c. 11 d. 9

15. Jellybean was introduced in .....
    a. 9/7/2012 b. 9/8/2012 c. 9/9/2012 d. 9/10/2012

16. AVD stands for .....
    a. Audio video device b. Android Virtual Device
    c. Audio visual device d. None of above

17. DVM stands for.....
    a. Dalvik Virtual Machine b. Devid Virtual Machine
    c. Dalvik Visual Machine d. none of above

18. SDK stands for .....
    a. Software Desing kit b. Software Define Kit
    c. Software Development kit d. none of above

19. What is emulator
    a. Virtual device to see output b. Supporting Application.
    c. both of above d.. None of above

20. ADT Stands for.....
    a. Android Development tool b. Android Developer tool
    c. Android Device Technology d. All of above

:: EXERCISE ::

▶ **Questions for 2 marks :**

(1) What is Gen? Gen is a directory in an Android file auto generated. Inside R.java file
(2) What is Res? Res file is like Img file and xml configuration file
(3) What is assets? assets is a folders for hirechy directory and files
(4) What is adb?
(5) What is canvas? Canvas is a draw 2D object.
(6) what is DDMS? → Dalvik Debug Monitor service.
(7) What is android.xml or manifest file?
(8) What is service?

▶ **Questions for 3 marks :**

(1) What is res/drawable? drawable is a holding another
(2) What is the requirement of bin folder?
(3) Explain AndroidManifest.xml This is root file in subfolder
(4) What is .apk file? Android APP. Package file. This file in
(5) What is .dex file? dex file is a compiler code. deliver executed
(6) What is Application? my file.
(7) What is Dialog? Dialog is a Activity.
(8) What is Drawable?

▶ **Question for 5 marks :**

(1) Explain Application object.
(2) Explain Activity life cycle with example.
(3) What is android service?
(4) Explain Different Intent types. Implicit, Explicit.
(5) Explain Pending Intent.

:: MCQ ::

1. Which of the following(s) is part of android project structure?
   a. src b. res c. libs d. all of above
2. APK file found in .....
   a. src b. res c. libs d. bin
3. source codes (.java) file stores under src folder.
   a. true b. false

4. Which file file contains layout and design related xml files.
   a. res b. src c. libs d. all of above
5. ldpi stans for
   a. lower density b. larger density
   c. long description d. last details
6. Android Application package file is for .....
   a. .apk b. AAP b. both a and b d. none
7. Dalvik Executable files are known as .....
   a. .apk b. Kitkat b. .def d. .dex
8. adb is .....
   a. android detailed bug b. Advanced Droid bug
   c. Android Debug Bridge d. none
9. DDMS is known as Dalvik Debug Monitor Services.....
   a. true b. false
10. widget is known as .....
    a. tools b. utility c. folders d. controls
11. Android Manifest File defines .....
    a. package name b. Version
    c. components d. All
12. states of android activity.....
    a. Running b. Paused c. both a and b d. none
13. Called once the activity is no longer visible.
    a. onStop() b. onCreate() c. onResume() d. onPause()
14. A _____ is a component that runs in the background to perform lon running operations without needing to interact with the user.
    a. service b. activity c. Intent d. context
15. A State(s) of services .....
    a. started b. bound c. both a and b d. none
16. A type(s) of intent.
    a. Explicit b. Implicit c. both a and b d. none
17. MCC stands for .....
    a. Mobile country code b. Mobile company code
    c. Mobile customer code d. none of above
18. MNC stands for .....
    a. mobile network code b. mobile netwere code
    c. mobile new code d. none of above
19. ldpi screen density is
    a. 120dpi b. 160dpi c. 240dpi d. None of above
20. Layout directions .....
    a. ldrtl b. ldltr c. boty a and b d. none

:: PRACTICAL EXERCISE ::

(1) Create application to display current date and time.

(2) Create application to get username and display it in with greetings for ex. "Wel come : Bagdais".

(3) Create application with two buttons one to hide a control and with one control will be visible again.

(4) Create application with 2 textboxes and by clicking on first text box display datepicker dialog and get event date from user by clicking on 2nd text box display timepicker dialog and display event time.

:: EXERCISE ::

▶ Questions for 2 marks :

(1) What is View? View is a object that draw scree

(2) list of basic UI? Button, Text, TextView, RadioButton etc

(3) What is AutoCompleteTextView? This is a similar TextEdit.

(4) What is Checkbox? checkbox is Button that one more checkbox

(5) What is Progressbar? Visual feedback.

(6) list of layonts and define any one?

(7) What is canvas?

(8) View animation.

▶ Questions for 3 marks :

(1) What is ID in UI?

(2) Size attribute in brief?

(3) What is padding?

(4) What is margin?

(5) Different input types for EditText?

(6) What are different behaviors of EditText?

▶ Question for 5 marks :

(1) Explain Check box in details with example.

(2) Explain Radio Button in details with example.

(3) Explain Spinners in details with example.

(4) Explain Linear Layout in details.

(5) Explain Simple Cursor Adapter.

:: M.C.Q. ::

1. All user interface element in android app are built using ..... and ..... objects.
   a. intent, intent filter b. activity, services
   c. pending intent, intent d. **view,viewgroup**

2. ..... element creates TextView widget in your UI.
   a. button b. <button> c. textview d. <TextView>

3. ..... can be pressed, or clicked, by the user to perform an action.
   a. **Button** b. <button> c. textview d. <TextView>

4. UI to display text.
   a. **TextView** b. EditText c. Button d. none

5. ..... view enables users to select a date of the day.
   a. **Datepicker** b. calender c. Both a and b d. None

6. ..... view enables users to select a time of the day
   a. Timepicker b. Datepicker c. both a and b d. none

7. With ..... We can see preview of layout XML.
   a. res b. bin
   c. java d. **Graphical Layout Tab.**

8. Any View object may have an integer ..... Associated with it.
   a. value b. name c. **Id** d. none

9. Margins are the spaces outside the border.
   a. true b. false

10. Paddings are the spaces inside the border.
    a. true b. false

11. To define the click event handler for a button, add the .....attribute to the <Button> element in your XML layout .
    a. name b. id c. **android:onClick** d. none

12. A ..... allows the user to type text into your app.
    a. button b. list c. **Text Fields** d. xml

13. You can add a text field to your layout with the ..... object.
    a. **EditText** b. TextView c. Button d. none

14. If you want to provide suggestions to users as they type, you can use a subclass of EditText called.....
    a. **AutoCompleteTextView** b. button
    c. edittext d. textview

15. Checkboxes allow the user to select one or more options from a set.
    a. **true** b. false

16. When the user selects a checkbox, the CheckBox object receives an ..... event.
    a. focus b. curor c. **on-click** d. none

17. ..... is A layout that organizes its children into a single horizontal or vertical row.

a. Linear Layout
b. Relative Layout
c. Table Layout
d. none of above

18. ..... Enables you to specify the location of child objects relative to each other.

a. Relative Layout
b. Table Layout
c. Linear Layout
d. none of above

19. ..... Displays web pages.

a. Relative Layout
b. Table Layout
c. Linear Layout
d. webview

20. The ..... is a special subclass of View that offers a dedicated drawing surface within the View hierarchy.

a. Relative Layout
b. SurfaceView
c. Table Layout
d. webview

:: PRACTICAL EXERCISE ::

(1) Create an application to get different setting from user and use it as their choice of application work.

Fields like : User Name, Current City, and drop down to get choice for type of new they like to receive in application.

(2) Create an application to store data in .csv file provided by user and read it in application.

Fields like : username, mobileno, and email address.

(3) Create an application to get user input and store it in application using database.

Fields like : Empid, empname, empmobno, empemail, empsalary.

(4) Create an application to send image as attachment using gmail.

(5) Create an application to send image using whats app.

:: EXERCISE ::

▶ Questions for 2 marks :

(1) Methods of data storage?

(2) What is sharedpreferences?

(3) Explain Internal Storage?

(4) Explain External Storage?

(5) What is Contract Class?

▶ Questions for 3 marks :

(1) How to obtain permission to write data to external storage?

(2) How to retrieve available free space in your mobile?

(3) What is schema?

▶ Question for 5 marks :

(1) Save a File on Internal Storage with example.

(2) Save a File on External Storage with example.

(3) Create a database using SQL Helper with example.

(4) Read Information from a database.

(5) Update information in database.

:: M.C.Q. ::

1. small collection of key-values that you'd like to save, you should use the. ..... APIs.
   a. SharedPreferences b. Intent
   c. IntentFilter d. PendingIntent

2. getSharedPreferences() can call from any ..... In you app.
   a. Intent **b. Context** c. intentFilter d. pendingIntent

3. To write to a shared preferences file, create a SharedPreferences.Editor by calling ..... on your SharedPreferences.
   a. add() b. insert() **c. edit()** d. create()

4. ..... Is used to put integer values in SharedPreferences.
   a. TextView b. EditText c. Button d. putint()

5. ..... is used to get String values from SharedPreferences.
   a. getString() b. readString() c. putString() d. None

6. ..... is best when you want to be sure that neither the user nor other apps can access your files.
   a. External Storage **b. Internal storage**
   c. both a and b d. none

7. ..... Storage is not always available.
   a. External Storage **b. Internal storage**
   c. both a and b d. none

8. To write to the external storage, you must request the ..... permission in your manifest file:
   a. WRITE_EXTERNAL_STORAGE b. READ_SMS
   c. PHONE_STATE d. none

9. .....Returns a File representing an internal directory for your app.
   a. getFilesDirectory() b. getFilesDir()
   c. getFolderDir() d. none

10. To create temporary file in internal cache directory ..... is useful.
    a. getFilesDirectory() b. getFilesDir()
    c. createTempFile() d. none

11. You can query the external storage state by calling.....
    a. getFilesDirectory() b. getFilesDir()
    c. createTempFile() d. getExternalStorageState()

12. ..... Method is used to query blank space.
    a. getBlankSpace() b. getFreeSpace()
    c. FreeSpace() d. none

13. If you want to save public files on the external storage, use the ..... method.
    a. **getExternalStoragePublicDirectory()**
    b. getFilesDir()
    c. createTempFile()
    d. getExternalStorageState()

14. ..... Method is used to delete file.
    a. .remove() b. erase() c. .delete() d. none

15. Insert data into the database by passing a ..... object to the insert() method.
    a. contaxtvalue b. contextValues c. contentvalue **d. ContentValues**

16. To read from a database, use the ..... method.
    a. query() b. curor c. on-click d. none

17. ..... places the "read position" on the first entry in the results.
    a. move() **b. moveToFirst()** c. beginning() d. move(0)

18. ..... Enables you to get index of column.
    a. getid() b. getLong()
    c. getColumnIndex() d. none of above

19. ..... modify a subset of your database values.
    a. insert() b. delete() **c. update()** d. None of above

20. Binary data is shared using the .....
    a. ACTION_SEND b. ACTION c. EXTRA_STREAM d. STREAM

Many more with location based services :

Here we have seen number of different tools provided on android, using them we can find location. You can also find different tools on net.

The LocationManager supports Proximity Alerts, which are alerts that trigger a PendingIntent when the handset comes within some distance of a location. This can be useful for warning the user of an upcoming turn in directions, for scavenger hunts, or help in geocaching.

:: PRACTICAL EXERCISE ::

(1) Create an application to see simple map in android device.

(2) Create an application to fetch current latitude and longitude of device.

(3) Create an application to check if GPS is enable or not.

(4) Create an application to display your current location with longitude and latitude.

:: EXERCISE ::

▶ **Questions for 2 marks :**

(1) What is GPS? Global Positioning System. Phone

(2) What is LBS? Location Base services.

(3) What is GeoCoding Location?

(4) What is MapView?

▶ **Questions for 3 marks :**

(1) What is longitude, latitude?

▶ **Questions for 5 marks :**

(1) Explain GeoCoding Location.

(2) How GPS Works?

(3) Address Lookup Task.

:: M.C.Q. ::

1. GPS stands for .....
   a. Global Positioning System
   b. Global Place System
   c. Global Present System
   d. Global Positioning Surface

2. The location is displayed with _____ and _____.
   a. Xaxis, Yaxis
   b. longitude, latitude.
   c. Yaxis, Xaxis
   d. latitude, lontitude.

3. The Android SDK provides resources for accessing location via a built-in_____, if available in hardware
   a. Camera
   b. Speaker
   c. GPS hardware
   d. None

4. LBS stands for _____
   a. Local Base Station
   b. Last Base Station
   c. Location Bottom Service
   d. Location Base Services

5. ..... is a process of assigning locations to addresses to that they can be placed as points on a map.
   a. Geocoding
   b. Last Base Station
   c. Location Bottom Service
   d. Location Base Services

6. The central component of the location framework is the ..... system service.
   a. Geocoding
   b. LocationManager
   c. Location Bottom Service
   d. Location Base Services

7. The key class in the Google Maps Android API is .....
   a. Geocoding
   b. LocationManager
   c. MapView
   d. Location Base Services

(3) Convert InputStream to String.

(4) Manage Network Usage.

(5) Explain Android Telephony API.

▶ **Questions for 5 marks :**

(1) How to getPhoneStatus with example.

(2) How to get Service Information with example.

(3) Steps to get SIM status and its details.

(4) How to check the Network Connection.

:: M.C.Q. ::

1. Most network-connected Android apps use ..... to send and receive data.
   a. HTTP b. FTP c. HTTPS d. SMTP

2. ..... to perform a GET and download your data.
   a. HTTP **b. HttpURLConnection**
   c. HTTPS d. SMTP

3. ..... returns the connection's status code.
   a. HTTP b. HttpURLConnection
   c. getResponseCode() d. SMTP

4. ..... answers queries about the state of network connectivity.
   a. HTTP b. HttpURLConnection
   c. getResponseCode() **d. ConnectivityManager**

5. ..... Describes the status of a network interface of a given type.
   a. NetworkInfo b. HttpURLConnection
   c. getResponseCode() d. ConnectivityManager

6. To deliver a web application (or just a web page) as a part of a client application, you can do it using.....
   a. NetworkInfo **b. WebView**
   c. getResponseCode() d. ConnectivityManager

7. The WebView class is an extension of Android's ..... class that allows you to display web pages as a part of your activity layout.
   a. NetworkInfo b. WebView
   c. View d. ConnectivityManager

8. You can enable it through the ..... attached to your WebView.
   a. NetworkInfo b. WebView c. View d. WebSettings

9. Using ..... we can retrieve the current status of mobile phone and also get some basic information about device.
   a. **TelephonyManager**
   b. WebView
   c. View
   d. WebSettings

10. With the help of ..... we can check the current status of voice call in handset.
   a. TelephonyManager
   b. **getCallState()**
   c. View
   d. WebSettings

1.2 WHAT IS ANDROID?

Android is an operating system based on the Linux kernel, and designed primarily for touch screen mobile devices such as smart phones and tablet computers. Initially developed by Android, Inc., which Google backed financially and later bought in 2005.

Android is a Linux-based operating system for smart phones and tablets. It includes a touch screen user interface, widgets, camera, network data monitoring and all the other features that enable a cell phone to be called a Smartphone. Android is a platform that supports various applications, available through the Android Play Store. The Android platform also allows end users to develop, install and use their own applications on top of the Android framework. The Android framework is licensed under the Apache License, with Android application developers holding the right to distribute their applications under their customized license.

IDE => Intergrated Development environment

1.4 THE ANDROID PLATFORM

The Android Platform was launched in 2007 by the Open Handset Alliance, an alliance of famous companies that includes Google, HTC, Motorola, Texas Instruments and others. Although most of the applications that run on the Android Platform are written in Java, there is no Java Virtual Machine. Instead, the Java classes are first compiled into what are known as Dalvik Executables and run on the Dalvik Virtual Machine.

Android is an open development platform. However, it is not open in the sense that everyone can contribute while a version is under development. This is all done behind closed-doors at Google. Rather, the openness of Android starts when its source code is released to the public after it is finalized. This means once it is released anyone interested can take the code and alter it as they see fit.

To create an application for the platform, a developer requires the Android SDK, which includes tools and APIs. To shorten development time, Android developers typically integrate the SDK into graphical user IDEs (Integrated Development Environments). Beginners can also make use of the App Inventor, an application for creating Android apps that can be accessed online.

Software Development kit.

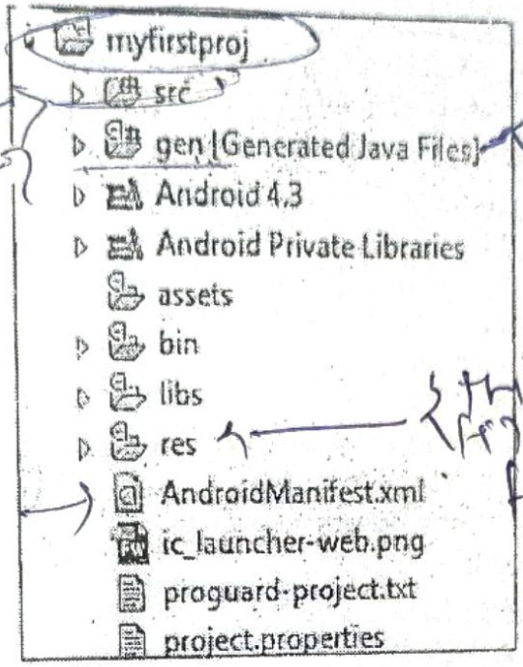
1.5 ANDROID SDK

The Android software development kit (SDK) includes a comprehensive set of development tools. These include a debugger, libraries, a handset emulator based on QEMU (short for "Quick EMUlator"), documentation, sample code, and tutorials. Currently supported development platforms include computers running Linux (any modern desktop Linux distribution), Mac OS X 10.5.8 or later, Windows XP or later; for the moment one can develop Android software on Android itself by using [AIDE - Android IDE - Java, C++] app and [Android java editor] app.

The officially supported integrated development environment (IDE) is Eclipse using the Android Development Tools (ADT) Plugin, though IntelliJ IDEA IDE (all editions) fully supports Android development out of the box, and NetBeans IDE also supports Android development via a plugin.

Additionally, developers may use any text editor to edit Java and XML files, then use command line tools (Java Development Kit and Apache Ant are required) to create, build and debug Android applications as well as control attached Android devices.

2.1 ANATOMY (STRUCTURE) OF AN ANDROID APPLICATION



- **Src :** Java source code files will be available here.
- **Gen :** The gen directory in an Android project contains auto generated files. You can see R.java inside this folder which is a generated class which contains references to certain resources of the project. R.java is automatically created by the Eclipse IDE and any manual changes are not necessary
- **Res :** Android supports resources like images and certain XML configuration files; these can be keeping separate from the source code. All these resources should be placed inside the res folder. This res folder will be having sub-folders to keep the resources based on its type.
- **/res/values :** Used to define strings, colors, dimensions, styles and static arrays of strings or integers. By convention each type is stored in a separate file, e.g. strings are defined in the res/values/strings.xml file.
- **/res/values-v11** is the values of the API version 11, and **/res/values-v14** is the values of the API version 14
  - → **/res/ values** - layout for normal screen size or default
  - → **/res/ values -small** - layout for small screen size
  - → **/res/ values -large** - layout for large screen size
  - → **/res/ values -xlarge** -layout for extra-large screen size
  - → **/res/ values -xlarge-land** - layout for extra-large in landscape orientation
  - → **/res/ values -sw600dp** - layout for tablets or layout for 7" tablets (600dp wide and bigger)
  - → **/res/ values -sw720dp** - layout for 10" tablets (720dp wide and bigger)
  - → **/res/ values -w600dp** - layout for Multi-pane (any screen with 600dp available width or more)

- /res/layout : This folder contains the layouts to be used in the application. A layout resource defines the architecture for the UI in an Activity or a component of a UI. These are resource directories in an application that provides different layout designs for different screen sizes.
- /res/menu : This folder contains menu resources to be used in the application (Options Menu, Context Menu, or submenu)
- /res/drawable : Drawable folders are resource directories in an application that provides different bitmap drawables for medium, high, and extra high density screens.
  - → /res/drawable-ldpi - bitmap for lower density
  - → /res/drawable-mdpi - bitmap for medium density
  - → /res/drawable-hdpi - bitmap for high density
  - → /res/drawable-xhdpi - bitmap for extra high density
  - → /res/drawable-xxhdpi - bitmap for X extra high density
- libs : External library files will be placed in this folder. If you want to any external library in your project place the library jar inside this folder and it will be added to the classpath automatically.
- assets : This folder contains raw hierarchy of files and directories, with no other capabilities. It is just an unstructured hierarchy of files, allowing you to put anything you want there and later retrieve as raw byte streams.
- bin : Bin folder is the area used by the compiler to prepare the files to be finally packaged to the application's APK file. This includes
  - → Compiling your Java code into class files
  - → Putting your resources (including images) into a structure to be zipped into the APK

This is the output directory of the build. This is where you can find the final .apk file and other compiled resources.

- **AndroidManifest.xml : All the android applications will have an AndroidManifest.xml file in the root directory. This file will contain essential information about the application to the Android system, information the system must have before it can run any of the application's code. This control file describes the nature of the application and each of its components**
- **ic_launcher-web.png : This is an icon to be used in Google play. Applications on Google Play require a high fidelity version of the application icon. It is not used in your actual app or the launcher, so it is not packaged in the APK. The specifications for the high-resolution icon are:**
  - → 32-bit PNG with an alpha channel
  - → 512 x 512 pixels
  - → Maximum size of 1024KB

- **proguard-project.txt** : Everything in the proguard-project.txt file will be in commented out state, because in general most people don't have any project specific needs, just to run ProGuard tool with standard settings.

  The ProGuard tool shrinks, optimizes, and obfuscates your code by removing unused code and renaming classes, fields, and methods with semantically obscure names. The result is a smaller sized .apk file that is more difficult to reverse engineer.

- **project.properties** : project.properties is the main project's properties file containing information such as the build platform target and the library dependencies has been renamed from default.properties in older SDK versions. This file is integral to the project.

2.2 ANDROID TERMINOLOGIES

The list below defines some of the basic terminology of the Android platform.

- **.apk file** : Android application package file. Each Android application is compiled and packaged in a single file that includes all of the application's code (.dex files), resources, assets, and manifest file. The application package file can have any name but must use the .apk extension. For example: myfirstproj.apk. For convenience, an application package file is often referred to as an ".apk".

- **.dex file** : Compiled Android application code file.

  Android programs are compiled into .dex (Dalvik Executable) files, which are in turn zipped into a single .apk file on the device. .dex files can be created by automatically translating compiled applications written in the Java programming language.

- **Action** : An action is a string value assigned to Intent. Action strings can be defined by Android or by a third-party developer. For example, android.intent.action.VIEW for a Web URL, or com.example.rumbler.SHAKE_PHONE for a custom application to vibrate the phone.

- **Activity** : A single screen in an application, with supporting Java code derived from the Activity class. Most commonly, an activity is visibly represented by a full screen window that can receive and handle UI events and perform complex tasks, because of the Window it uses to render its window. Though an Activity is typically full screen, it can also be floating or transparent.

- **adb** : Android Debug Bridge, a command-line debugging application included with the SDK. It provides tools to browse the device, copy tools on the device, and forward ports for debugging. If you are developing in Eclipse using the ADT Plugin, adb is integrated into your development environment.

- **Application** : From a component viewpoint, an Android application consists of one or more activities, services, listeners, and intent receivers. From a source file viewpoint, an Android application consists of code, resources, assets, and a single manifest. During compilation, these files are packaged in a single file called an application package file (.apk).

- **Canvas :** Canvas is the simplest, easiest way to draw 2D objects on the scre However, it does not support hardware acceleration, as OpenGL ES does. The b class is Canvas.
- **Content Provider :** A content provider is built on the ContentProvider cla which handles content query strings of a specific format to return data in a spec format.
- **Dalvik :** The Android platform's virtual machine. The Dalvik VM is an interpret only virtual machine that executes files in the Dalvik Executable (.dex) format format that is optimized for efficient storage and memory-mappable execution. T virtual machine is register-based, and it can run classes compiled by a Ja language compiler that have been transformed into its native format using t included "dx" tool. The VM runs on top of Posix-compliant operating systems, whic it relies on for underlying functionality (such as threading and low level memo management). The Dalvik core class library is intended to provide a famil development base for those used to programming with Java Standard Edition, but is geared specifically to the needs of a small mobile device.
- **DDMS :** Dalvik Debug Monitor Service, a GUI debugging application included wi the SDK. It provides screen capture, log dump, and process examinati capabilities. If you are developing in Eclipse using the ADT Plugin, DDMS integrated into your development environment.
- **Dialog :** A floating window that that acts as a lightweight form. A dialog can ha button controls only and is intended to perform a simple action (such as but choice) and perhaps return a value. A dialog is not intended to persist in the hist stack, contain complex layout, or perform complex actions. Android provides default simple dialog for you with optional buttons, though you can define your c dialog layout. The base class for dialogs is Dialog.
- **Drawable :** A drawable is typically loaded into another UI element, for example a background image. A drawable is not able to receive events, but does ass various other properties such as "state" and scheduling, to enable subclasses su as animation objects or image libraries. Many drawable objects are loaded fr drawable resource files — xml or bitmap files that describe the image. Drawab resources are compiled into subclasses of android.graphics.drawable.
- **Intent :** An Intent object is an instance of Intent. It includes several criteria fie that you can supply, to determine what application/activity receives the Intent an what the receiver does when handling the Intent. Available criteria include th desired action, a category, a data string, the MIME type of the data, a handlin class, and others. An application sends Intent to the Android system, rather tha sending it directly to another application/activity. The application can send th Intent to a single target application or it can send it as a broadcast, which can i turn be handled by multiple applications sequentially. The Android system responsible for resolving the best-available receiver for each Intent, based on th criteria supplied in the Intent and the Intent Filters defined by other applications

- **Intent Filter :** A filter object that an application declares in its manifest file, to tell the system what types of Intents each of its components is willing to accept and with what criteria. Through an intent filter, an application can express interest in specific data types, Intent actions, URI formats, and so on. When resolving Intent, the system evaluates all of the available intent filters in all applications and passes the Intent to the application/activity that best matches the Intent and criteria.
- **Broadcast Receiver :** An application class that listens for Intents that are broadcast, rather than being sent to a single target application/activity. The system delivers a broadcast Intent to all interested broadcast receivers, which handle the Intent sequentially.
- **Layout Resource :** An XML file that describes the layout of an Activity screen.
- **Manifest File :** An XML file that each application must define, to describe the application's package name, version, components (activities, intent filters, services), imported libraries, and describes the various activities, and so on.
- **Nine-patch / 9-patch / Ninepatch image :** A resizeable bitmap resource that can be used for backgrounds or other images on the device.
- **OpenGL ES :** Android provides OpenGL ES libraries that you can use for fast, complex 3D images. It is harder to use than a Canvas object, but better for 3D objects. The android.opengl and javax.microedition.khronos.opengles packages expose OpenGL ES functionality.
- **Resources :** Non programmatic application components that are external to the compiled application code, but which can be loaded from application code using a well-known reference format. Android supports a variety of resource types, but a typical application's resources would consist of UI strings, UI layout components, graphics or other media files, and so on. An application uses resources to efficiently support localization and varied device profiles and states. For example, an application would include a separate set of resources for each supported local or device type, and it could include layout resources that are specific to the current screen orientation (landscape or portrait).
- **Service :** An object of class Service that runs in the background (without any UI presence) to perform various persistent actions, such as playing music or monitoring network activity.
- **Surface :** An object of type Surface representing a block of memory that gets composited to the screen. A Surface holds a Canvas object for drawing, and provides various helper methods to draw layers and resize the surface. You should not use this class directly; use SurfaceView instead.
- **SurfaceView :** A View object that wraps a Surface for drawing, and exposes methods to specify its size and format dynamically. A SurfaceView provides a way to draw independently of the UI thread for resource-intensive operations (such as games or camera previews), but it uses extra memory as a result. SurfaceView supports both Canvas and OpenGL ES graphics. The base class is SurfaceView.

- Theme : A set of properties (text size, background color, and so on) bundled together to define various default display settings. Android provides a few standard themes, listed in R.style (starting with "Theme_").
- URIs in Android : Android uses URI strings as the basis for requesting data in a content provider (such as to retrieve a list of contacts) and for requesting actions in an Intent (such as opening a Web page in a browser). The URI scheme and format is specialized according to the type of use, and an application can handle specific URI schemes and strings in any way it wants. Some URI schemes are reserved by system components. For example, requests for data from a content provider must use the content://. In an Intent, a URI using an http:// scheme will be handled by the browser.
- View : An object that draws to a rectangular area on the screen and handles click, keystroke, and other interaction events. A View is a base class for most layout components of an Activity or Dialog screen (text boxes, windows, and so on). It receives calls from its parent object to draw itself, and informs its parent object about where and how big it would like to be (which may or may not be respected by the parent).
- Viewgroup : A container object that groups a set of child Views. The viewgroup is responsible for deciding where child views are positioned and how large they can be, as well as for calling each to draw itself when appropriate. Some viewgroups are invisible and are for layout only, while others have an intrinsic UI (for instance, a scrolling list box).
- Widget : One of a set of fully implemented View subclasses that render form elements and other UI components, such as a text box or popup menu. Because a widget is fully implemented, it handles measuring and drawing itself and responding to screen events. Widgets are all in the android.widget package.
- Window : In an Android application, an object derived from the abstract class Window that specifies the elements of a generic window, such as the look and feel (title bar text, location and content of menus, and so on). Dialog and Activity use an implementation of this class to render a window. You do not need to implement this class or use windows in your application.

2.3 APPLICATION CONTEXT, ACTIVITIES, SERVICES, INTENTS

- Context : is probably the most used element in android applications.
- Application : The application object is created whenever one of your Android components is started. It is started in a new process with a unique ID under a single user. Even if you do not specify one in your AndroidManifest.xml file, the Android system creates a default object for you. This object provides the following lifecycle methods:
  - → **onCreate()** : called before the first components of the application starts
  - → **onLowMemory()** : called when the Android system requests that the application cleans up memory

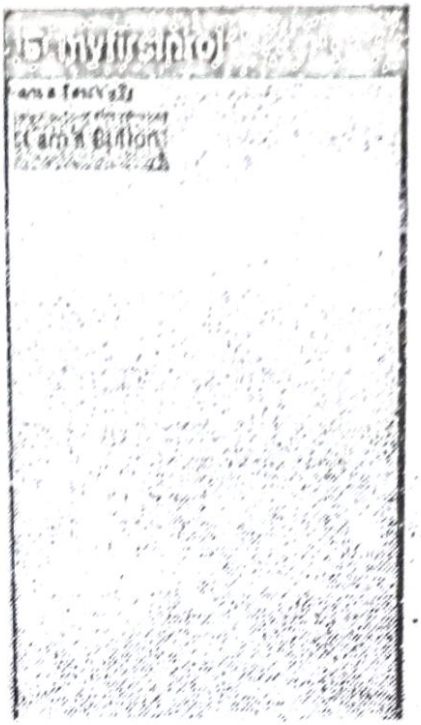
Services :

A service is a component that runs in the background to perform long-running operations without needing to interact with the user. For example, a service might play music in the background while the user is in a different application, or it might fetch data over the network without blocking user interaction with an activity. A service can essentially take two states:

State	Description
Started	A service is started when an application component, such as an activity, starts it by calling startService(). Once started, a service can run in the background indefinitely, even if the component that started it is destroyed.
Bound	A service is bound when an application component binds to it by calling bindService(). A bound service offers a client-server interface that allows components to interact with the service, send requests, get results, and even do so across processes with interprocess communication (IPC).

A service has lifecycle callback methods that you can implement to monitor changes in the service's state and you can perform work at the appropriate stage. The following diagram on the left shows the lifecycle when the service is created with startService() and the diagram on the right shows the lifecycle when the service is created with bindService().

For example, a simple vertical layout with a text view and a button looks like graphical file:



When you load a layout resource in your app, Android initializes each node o layout into a runtime object you can use to define additional behaviors, query object state, or modify the layout.

- **User Interface Components :**
  You don't have to build all of your UI using View and ViewGroup objects. And provides several app components that offer a standard UI layout for which simply need to define the content. These UI components each have a unique s APIs that are described in their respective documents, such as Action Bar, Dial and Status Notifications.

- **List of Basic UIs :**
  There are number of UI controls provided by Android that allow you to build graphical user interface for your app.

S.N.	UI Control & Description
1	TextView This control is used to display text to the user.
2	EditText EditText is a predefined subclass of TextView that includes rich edi capabilities.

3	AutoCompleteTextView The AutoCompleteTextView is a view that is similar to EditText, except that it shows a list of completion suggestions automatically while the user is typing.
4	Button A push-button that can be pressed, or clicked, by the user to perform an action.
5	ImageButton AbsoluteLayout enables you to specify the exact location of its children.
6	CheckBox An on/off switch that can be toggled by the user. You should use checkboxes when presenting users with a group of selectable options that are not mutually exclusive.
7	ToggleButton An on/off button with a light indicator.
8	RadioButton The RadioButton has two states: either checked or unchecked.
9	RadioGroup A RadioGroup is used to group together one or more RadioButtons.
10	ProgressBar The ProgressBar view provides visual feedback about some ongoing tasks, such as when you are performing a task in the background.
11	Spinner A drop-down list that allows users to select one value from a set.
12	TimePicker The TimePicker view enables users to select a time of the day, in either 24-hour mode or AM/PM mode.
13	DatePicker The DatePicker view enables users to select a date of the day.

3.2 DESIGNING LAYOUTS

A layout defines the visual structure for a user interface, such as the UI for an activity or app widget. You can declare a layout in two ways:

- **Declare UI elements in XML.** Android provides a straightforward XML vocabulary that corresponds to the View classes and subclasses, such as those for widgets and layouts.
- **Instantiate layout elements at runtime.** Your application can create View and ViewGroup objects (and manipulate their properties) programmatically.

The Android framework gives you the flexibility to use either or both of these methods for declaring and managing your application's UI. For example, you could declare your application's default layouts in XML, including the screen elements that will appear in them and their properties. You could then add code in your application that would modify the state of the screen objects, including those declared in XML, at run time.

The attributes define here are also available in properties window.

→ ID :

Any View object may have an integer ID associated with it, to uniquely identif the View within the tree. When the application is compiled, this ID is reference as an integer, but the ID is typically assigned in the layout XML file as a string in the id attribute:

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

The at-symbol (@) at the beginning of the string indicates that the XML parse should parse and expand the rest of the ID string and identify it as an ID resource. The plus-symbol (+) means that this is a new resource name that mus be created and added to our resources (in the R.java file).

In order to create views and reference them from the application, a commo pattern is to:

Working With Animation :

Android provides a variety of powerful APIs for applying animation to UI elements.

→ Animation :

The Android framework provides two animation systems: property animation (introduced in Android 3.0) and view animation. Both animation systems are viable options, but the property animation system, in general, is the preferred method to use, because it is more flexible and offers more features. In addition to these two systems, you can utilize Drawable animation, which allows you to load drawable resources and display them one frame after another.

→ Property Animation :

Introduced in Android 3.0 (API level 11), the property animation system lets you animate properties of any object, including ones that are not rendered to the screen. The system is extensible and lets you animate properties of custom types as well.

→ View Animation :

View Animation is the older system and can only be used for Views. It is relatively easy to setup and offers enough capabilities to meet many application's needs.

→ Drawable Animation :

Drawable animation involves displaying Drawable resources one after another, like a roll of film. This method of animation is useful if you want to animate things that are easier to represent with Drawable resources, such as a progression of bitmaps.

Property Animation :

The property animation system is a robust framework that allows you to animate almost anything. You can define an animation to change any object property over time, regardless of whether it draws to the screen or not. A

GPS :- Global Positioning Services

5.1 INTRODUCTION

A GPS receiver calculates its position by precisely timing the signals sent by GPS satellites high above the Earth. Each satellite continually transmits messages that include:

- the time the message was transmitted and,
- Satellite position at time of message transmission.

▶ How GPS Works?

The receiver uses the messages it receives to determine the transit time of each message and computes the distance to each satellite using the speed of light. Each of these distances and satellites' locations defines a area. The receiver is on the surface of each of these areas when the distances and the satellites' locations are correct. These distances and satellites' locations are used to compute the location of the receiver using the navigation equations. This location is then displayed, perhaps with a moving map display or latitude and longitude.

Basic GPS measurements surrender only a position, and neither speed nor direction. However, most GPS units can automatically derive speed and direction of movement from two or more position measurements. The disadvantage of this principle is that changes in speed or direction can only be computed with a delay, and that derived direction becomes inaccurate when the distance travelled between two position measurements drops below or near the random error of position measurement. More advanced navigation systems use additional sensors like a compass or an inertial navigation system to complement GPS.

In typical GPS operation, four or more satellites must be visible to obtain an accurate result. The solution of the navigation equations gives the position of the receiver along with the difference between the time kept by the receiver's on-board

```
// Remove following section
```

- Building Web Apps in WebView :
  How to embed web pages into your Android application using WebView and bind JavaScript to Android APIs.
- Debugging Web Apps :
  How to debug web apps using JavaScript Console APIs.
- Best Practices for Web Apps :
  A list of practices you should follow, in order to provide an effective web application on Android-powered devices.

```
// Remove section ends
```

8.1 SERVICES

A Service is an application component that can perform long-running operations in the background and does not provide a user interface. Another application component can start a service and it will continue to run in the background even if the user switches to another application. Additionally, a component can bind to a service to interact with it and even perform interprocess communication (IPC). For example, a service might handle network transactions, play music, perform file I/O, or interact with a content provider, all from the background.

A service can essentially take two forms:

- **Started :**
  A service is "started" when an application component (such as an activity) starts it by calling startService(). Once started, a service can run in the background indefinitely, even if the component that started it is destroyed. Usually, a started service performs a single operation and does not return a result to the caller. For example, it might download or upload a file over the network. When the operation is done, the service should stop itself.

- **Bound :**
  A service is "bound" when an application component binds to it by calling bindService(). A bound service offers a client-server interface that allows components to interact with the service, send requests, get results, and even do so across processes with interprocess communication (IPC). A bound service runs only as long as another application component is bound to it. Multiple components can bind to the service at once, but when all of them unbind, the service is destroyed.

  Here we discusses two types of services separately, your service can work both ways—it can be started (to run indefinitely) and also allow binding. It's simply a matter of whether you implement a couple callback methods: onStartCommand() to allow components to start it and onBind() to allow binding.

  Regardless of whether your application is started, bound, or both, any application component can use the service (even from a separate application), in the same way that any component can use an activity—by starting it with an Intent. However, you can declare the service as private, in the manifest file, and block access from other applications.

Note : A service runs in the main thread of its hosting process—the service does not create its own thread and does not run in a separate process (unless you specify otherwise). This means that, if your service is going to do any CPU intensive work or blocking operations (such as MP3 playback or networking), you should create a new thread within the service to do those work. By using a separate thread, you will reduce the risk of Application Not Responding (ANR) errors and the application's main thread can remain dedicated to user interaction with your activities.