**“Mobile is the future of Software Development” – Google’s Eric Schmidt.**

**Question 1: what is Android?**

**Answer**: Android is a software stack for mobile devices such as Smart phone and Tablet that include an operating systems (Linux kernel based),middleware and key applications(Mobile app s/w).

Goal of Android Project (s/w stack) is to create a successful real world product that improves mobile experience for mobile end user.

It can be thought of a mobile OS but it is not limited to mobile only it is currently being used in various devices such as mobiles, tablet, television, wears etc.

NOTE- Google has developed Android OS for CAR.

**Android History & Version: -**

1. In 2005 GOOGLE acquired Android Inc. and key employees of Android Inc. At that time not much was known about android Inc. but many assume that Google was planning to enter in the mobile phone (G phone) market with this move.

At Google, Andy Rubin (Father of Android) and his Team developed mobile device platform (OS) powered by Linux Kernel.

1. In 2007, November 07, the open Handset Alliance (OHA) is a consortium of technology companies formed by Google including Google, Devices manufacturers like HTC, LG, Sony, Motorola Samsung etc. and wireless carriers such as Sprint Nestle and T. Mobile and Chipset makers such as Texas Instrument and QUAL COMM, with a goal to develop open standard platform (OS) for mobile devices.

**Android Versions History: -** Like any Software, Android is improved over the time which reflected in its own version numbers-----

* **Note-** Since these devices make our lives so sweet, each Android version is named after a dessert.
* Pre-commercial release versions (2007–2008)
  + 1-Android alpha
  + 2- Android beta

**Version** **Nickname** **API level**

Android -1.0 - (API level 1)

Android -1.1 - (API level 2)

Android -1.5 Cupcake (API level 3)

Android -1.6 Donut (API level 4)

Android - 2.0 Éclair (API level 5)

Android - 2.0.1 Éclair (API level 6)

Android - 2.1 Éclair (API level 7)

Android - 2.2–2.2.3 Froyo (API level 8)

Android -2.3–2.3.2 Gingerbread (API level 9)

Android - 2.3.3–2.3.7 Gingerbread (API level 10)

Android - 3.0 Honeycomb (API level 11)

Android -3.1 Honeycomb (API level 12)

Android - 3.2 Honeycomb (API level 13)

Android -4.0–4.0.2 Ice Cream Sandwich (API level 14)

Android - 4.0.3–4.0.4 Ice Cream Sandwich (API level 15)

Android - 4.1 Jelly Bean (API level 16)

Android -4.2 Jelly Bean (API level 17)

Android -4.3 Jelly Bean (API level 18)

Android - 4.4 Kit-Kat (API level 19)

Android-4.4W Kit-Kat with wearable extension (API level 20)

Android -5.0 .1 Lollipop (API level 21)

Android -5.1.1 Lollipop (API level 22)

Android -6.0 Marshmallow (API level 23)

Note:- Google has released Android M preview (API level 23) on Nexus and HTC devices by 28th May 15.

**Question 2- Why it named Android?**

**Answer**- The word was coined from Greek word “Avopoid” a combination of

Avop = Human

Oid = having the form of. 

An Android OS is a Robot design to look and behave like man that’s why it has a logo of Robot. ‘Android’ word patent and copyright by Google in 2007.

**Question 3- Why Android is needed?**

**Answer** – Android OS needed because of following things;-

1. **Open source platform: -** Android is an open source platform means for developers Android provide all the tool and API for developing mobile apps quickly and easily.
2. **Platform Independent: -** Android is platform independent which separate hardware(Mobile Device) from Android App software that runs on it this allows for much larger number of devices to run same application.
3. **Ease of App development: -**
4. **License friendly: -** Android is licensed under business friendly licensed (Apache/MIT).
5. **Design for mobile devices: -** Android is purpose built platform for mobile devices. When designing Android platform the team looked on those mobile device constraints which would not going to change for seeable future changes. (Battery power and Mobile sizes).

Device Constraints Like Battery power and mobile size, including memory & processor consumption.

**Features Supported by Android: -**

Android is a powerful OS which support great features. Few of them are listed below—

1. Beautiful User interface: - Android OS basic screen provides a beautiful & interactive user interface.
2. Connectivity: - GSM, CDMA, Wi-Fi, Bluetooth etc.
3. Storage: - SQLite database (255 kb.) a light weighted relational database is used for data storage purpose.
4. Messaging: - SMS & multimedia messaging like MMS.
5. Multitasking: - User can jump from one task to another task at same time.

**Question: What is future scope of Android?**

**Answer: -** The future scope of Android is limitless. How? Allow me tell you the ongoing innovations in Android:-

1. **Health Industry -** That time is not far when we will test our blood samples using our smartphone. Some Innovations like Heart Rate monitor, Tracking sensor, Diet Schedule...etc is already available on androids.
2. **Artificial Intelligence -** AI in Android is going to automate the whole world with its [DeepMind Technology.](https://en.wikipedia.org/wiki/Google_DeepMind)
3. **Etc.**

**Question 4: - Why does Google made it open source (motivation)?**

**Answer: -** In 2005 when Google Inc. acquired Android Inc., the world thought Google was about to enter the smarter phone market & there was widely spread speculation that Google is about to enter in Mobile Market with device called G-phone.

Google’s CEO Eric Schmitt made it clear that Android ambition is much larger than a single device.

Ultimately Google is a media company and its business model is based on selling advertisement. If anyone is using Android, then Google can provide additional services on top it.

Although Google does license some proprietary apps such as g-mail, g-map & Play Store etc. make some money from the Android market.

**Survey-**

Android powers billions of mobile devices in more than 190 countries around the world. It is the largest installed of any mobile platform and still growing fast. Every day more than 1 million new android devices are activated worldwide.

**Question: - How would we develop Android Mobile application?**

**Answer:** - For development of Android application we need Android Environment Setup-

Android Environment Setup:- for development of android app. We require an environment that is called Android Environment Setup. We will be glad to know that we can start our app development either of the following OS---

1. M.S. windows XP or later version.
2. MAC OS or later version with Intel chip.
3. Linux

All required tools to develop Android apps are freely available and we can download from internet. Following is the list of all required tools to develop Android Apps—

1. JDK 5 or later version.
2. Android SDK (Software Development Kit).
3. IDE.

OR

ADT-Bundle Tool + Jdk

OR

Android Studio.exe + JDK

From -http://developer.android.com/

So let us have a look how to proceed to set required Android environment--

Step-1: - Install JDK.

Note- No need to set path if we are using Eclipse IDE, because it will automatically set and get the required Java component.

Step-2: - Setup of Android SDK—

**Question: - what is Android SDK?**

**Answer:** - The Android SDK stands for Software Development Kit set that is used to develop application. The Android SDK includes following things.—

1. API (required libraries) android.jar
2. Relevant Documentations of APIs (tutorials)
3. Build Tools (like aapt, aidl etc)
4. Extra libraries (Support and Design library like appcompat-v7, support-v4,design library etc)
5. An Emulator (AVD)
6. Platform tools (like adb )
7. Etc.

NOTE- By default, the Android SDK does not include everything you need to start developing. The SDK separates tools, platforms, and other components into packages you can download as needed using the [Android SDK Manager](http://developer.android.com/tools/help/sdk-manager.html). So before you can start, there are a few packages you should add to your Android SDK.

**NOTE-** Every time Google releases a new versions of android OS, a corresponding SDK is also has been released, to be able to write a program with the latest features. A developer must download and install a latest version of SDK by SDK Manager .

**Simple way of android development by ADT Bundle-**

We can download ADT Bundle from Android site (developer.android.com). We don’t need to have Eclipse IDE, Android SDK and ADT separately, because these all are already included in ADT Bundle.

**OR**

**ANDROID STUDIO**

**Ques:-What is Android Project?**

Ans:- A project is an organizational unit that represents a complete software solution for Android App Development(APK).

An Android project contains everything that is needed for your Android app, from app source code to build configurations files and resources. The SDK build tools require that your projects follow a specific structure so it can compile and package your application (.apk) correctly. If you're using Android Studio, it takes care of all this for you.

A module is the first level of containment within a project that encapsulates specific types of source code files and resources. There are several types of modules within an Android App project:

**Android Application Modules**

An Android Application Module is the container for your application's source code, resource files, and application level settings, such as the module-level build file, resource files, and Android Manifest file. The application module contents are eventually built into the .apk file that gets installed on a device.

**Test Modules**

These modules contain code to test your application projects and are built into test applications that run on a device. By default, Android Studio creates the androidTest module for inserting JUnit tests.

**Library Modules**

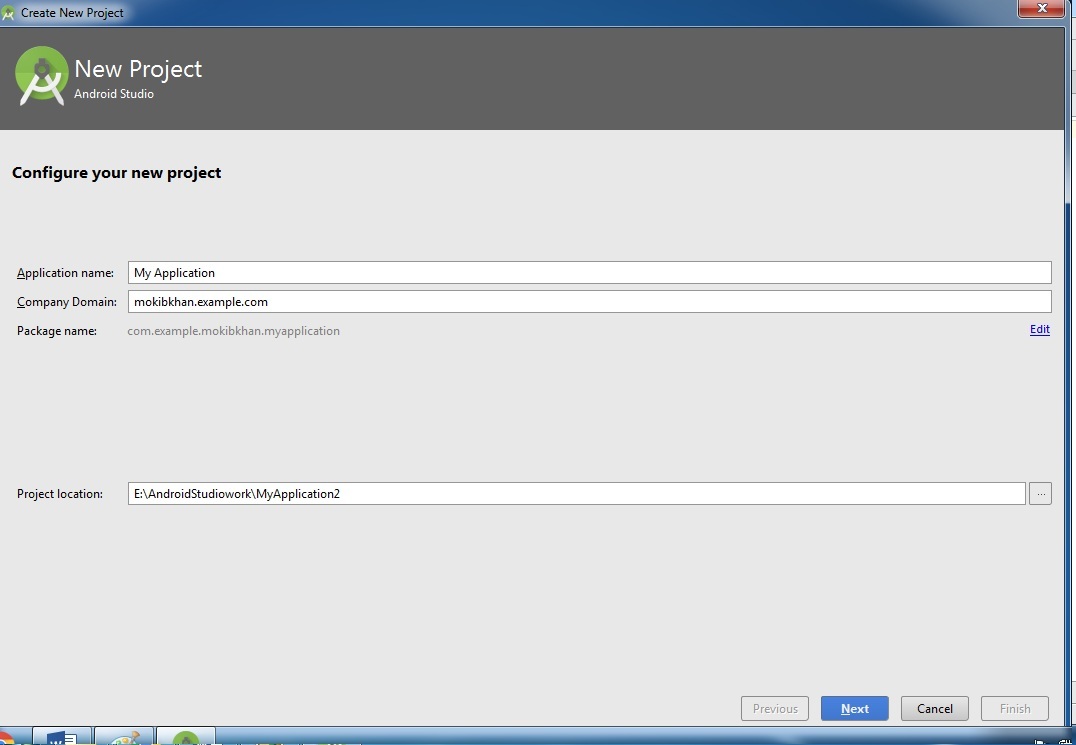
## These modules contain shareable Android source code and resources that you can reference in Android projects. This is useful when you have common code that you want to reuse. Library modules cannot be installed onto a device, however, they are pulled into the .apk file at build time.

**Etc.**

**Ques:-How to create first Android Application?**

Ans:- **Create a Project with Android Studio**

1. In Android Studio, create a new project:
   * If you don't have a project opened, in the **Welcome** screen, click **New Project**.
   * If you have a project opened, from the **File** menu, select **New Project**.



1. Under **Configure your new project**, fill in the fields as shown in figure 1 and click **Next**.

It will probably be easier to follow these lessons if you use the same values as shown.

* + **Application Name** is the app name that appears to users. For this project, use "My First App."
  + **Company domain** provides a qualifier that will be appended to the package name; Android Studio will remember this qualifier for each new project you create.
  + **Package name** is the fully qualified name for the project (following the same rules as those for naming packages in the Java programming language). Your package name must be unique across all packages installed on the Android system. You can **Edit** this value independently from the application name or the company domain.
  + **Project location** is the directory on your system that holds the project files.

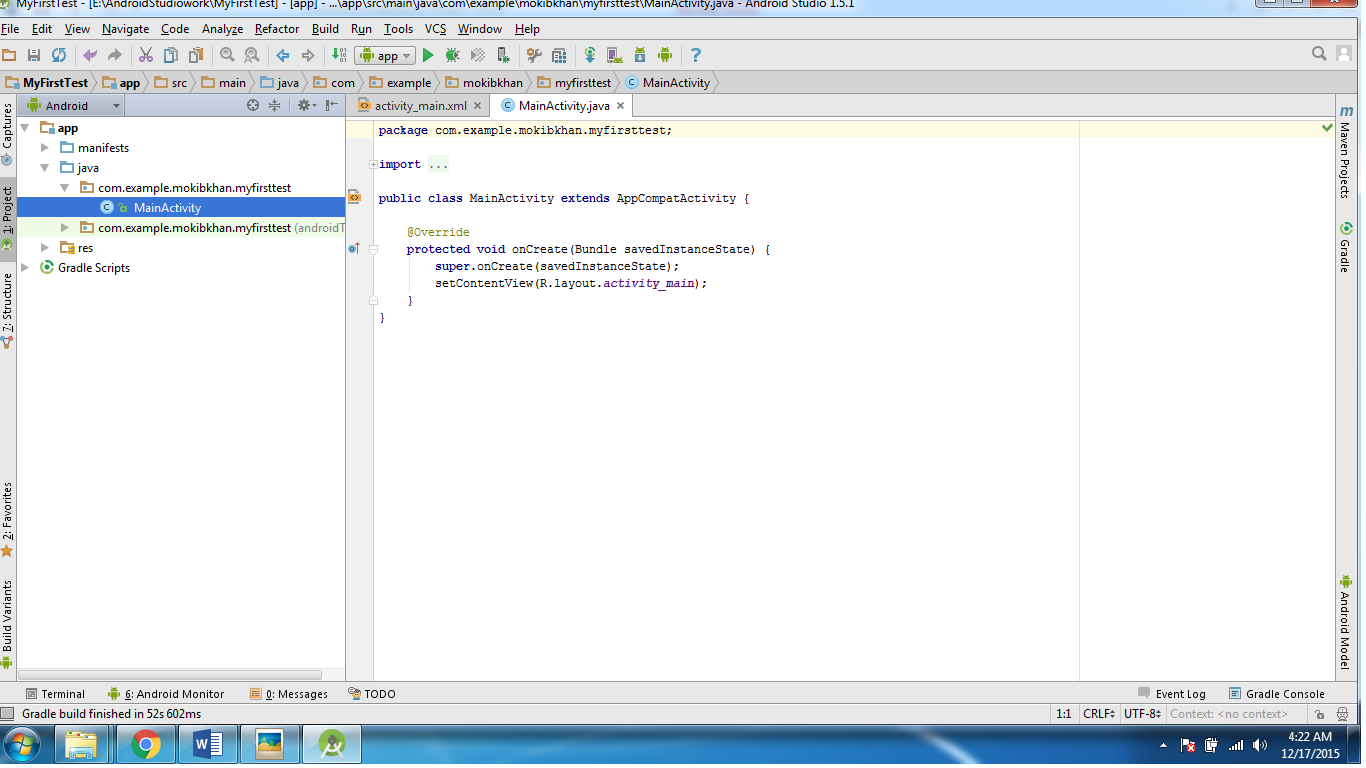
1. Under **Select the form factors your app will run on**, check the box for **Phone and Tablet**.
2. For **Minimum SDK**, select **API 15: Android 4.0.3 (IceCreamSandwich)**.

The Minimum Required SDK is the earliest version of Android that your app supports, indicated using the [API level](http://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels).

**NOTE-** To support as many devices as possible, you should set this to the lowest version available that allows your app to provide its core feature set. If any feature of your app is possible only on newer versions of Android and it's not critical to the app's core feature set, you can enable the feature only when running on the versions that support it.

1. Leave all of the other options (TV, Wear, and Glass) unchecked and click **Next.**
2. Under **Add an activity to <template>**, select **Basic Activity** or **Empty Activity** or Any other as per need of App development and click **Next**.
3. Under **Choose options for your new file**, change the **Activity Name** to MainActivity. The **Layout Name** changes to activity\_main.
4. Click the **Finish** button to create the project.

Next, follow the instructions provided and keep all other entries as default till the final step. Once your project is created successfully, you will have following project screen if you select **Empty Activity** template:



**Android Application Modules**

Android Application Modules are the modules that eventually get built into the .apk files based on your build settings. They contain things such as application source code and resource files. Most code and resource files are generated for you by default, while others should be created if required. The following directories and files comprise an Android application module:

build/

Contains build folders (generated, intermediate etc) for the specified build variants. Stored in the main application module.

libs/

Contains private libraries. Stored in the main application module.

src/

Contains your stub Activity file, which is stored at src/main/java//ActivityName>.java. All other source code files (such as .java or .aidl files) go here as well.

androidTest/

Contains the instrumentation tests. For more information, see the [Android Test documentation](http://developer.android.com/tools/testing/index.html).

main/java/com.>project<.>app<

Contains Java code source for the app activities.

main/jni/

Contains native code using the Java Native Interface (JNI). For more information, see the [Android NDK documentation](http://developer.android.com/tools/sdk/ndk/index.html).

main/gen/

Contains the Java files generated by Android Studio, such as your R.java file and interfaces created from AIDL files.

main/assets/

This is empty. You can use it to store raw asset files. Files that you save here are compiled into an .apk file as-is, and the original filename is preserved. You can navigate this directory in the same way as a typical file system using URIs and read files as a stream of bytes using the [AssetManager](http://developer.android.com/reference/android/content/res/AssetManager.html). For example, this is a good location for textures and game data.

main/res/

Contains application resources, such as drawable files, layout files, and string values in the following directories. See [Application Resources](http://developer.android.com/guide/topics/resources/index.html) for more information.

Note also the /res subdirectories that contain the [resources](http://developer.android.com/guide/topics/resources/overview.html) for your application:

anim/

For XML files that are compiled into animation objects. See the [Animation](http://developer.android.com/guide/topics/resources/animation-resource.html) resource type.

color/

For XML files that describe colors. See the [Color Values](http://developer.android.com/guide/topics/resources/color-list-resource.html) resource type.

drawable/

For bitmap files (PNG, JPEG, or GIF), 9-Patch image files, and XML files that describe Drawable shapes or Drawable objects that contain multiple states (normal, pressed, or focused). See the [Drawable](http://developer.android.com/guide/topics/resources/drawable-resource.html) resource type.

mipmap/

For app launcher icons. The Android system retains the resources in this folder (and density-specific folders such as mipmap-xxxhdpi) regardless of the screen resolution of the device where your app is installed. This behavior allows launcher apps to pick the best resolution icon for your app to display on the home screen. For more information about using the mipmap folders, see [Managing Launcher Icons as mipmap Resources](http://developer.android.com/tools/projects/index.html#mipmap).

layout/

XML files that are compiled into screen layouts (or part of a screen). See the [Layout](http://developer.android.com/guide/topics/resources/layout-resource.html) resource type.

menu/

For XML files that define application menus. See the [Menus](http://developer.android.com/guide/topics/resources/menu-resource.html) resource type.

raw/

For arbitrary raw asset files. Saving asset files here is essentially the same as saving them in the assets/directory. The only difference is how you access them. These files are processed by aapt and must be referenced from the application using a resource identifier in the R class. For example, this is a good place for media, such as MP3 or Ogg files.

values/

For XML files that define resources by XML element type. Unlike other resources in the res/ directory, resources written to XML files in this folder are not referenced by the file name. Instead, the XML element type controls how the resources defined within the XML files are placed into the R class.

AndroidManifest.xml

The control file that describes the nature of the application and each of its components. For instance, it describes: certain qualities about the activities, services, intent receivers, and content providers; what permissions are requested; what external libraries are needed; what device features are required, what API Levels are supported or required; and others. See the [AndroidManifest.xml](http://developer.android.com/guide/topics/manifest/manifest-intro.html) documentation for more information

.gitignore/

Specifies the untracked files ignored by git.

app.iml/

IntelliJ IDEA module

app/build.gradle

Android Studio uses Gradle to compile and build your app. There is a build.gradle file for each module of your project, as well as a build.gradle file for the entire project. Usually, you're only interested in the build.gradle file for the module, in this case the app or application module. This is where your app's build dependencies are set, including the defaultConfig settings:

* compiledSdkVersion is the platform version against which you will compile your app. By default, this is set to the latest version of Android available in your SDK. (It should be Android 4.1 or greater; if you don't have such a version available, you must install one using the [SDK Manager](http://developer.android.com/sdk/installing/adding-packages.html)).
* applicationId is the fully qualified package name for your application that you specified during the New Project workflow.
* minSdkVersion is the Minimum SDK version you specified during the New Project workflow. This is the earliest version of the Android SDK that your app supports.
* targetSdkVersion indicates the highest version of Android with which you have tested your application. As new versions of Android become available, you should test your app on the new version and update this value to match the latest API level and thereby take advantage of new platform features.

proguard-rules.pro

ProGuard settings file.

## Android Project Files

Android Studio project files and settings provide project-wide settings that apply across all modules in the project.

.idea

Directory for IntelliJ IDEA settings.

app

Application module directories and files.

build

This directory stores the build output for all project modules.

gradle

Contains the gradler-wrapper files. Wrapper is Batch script on windows and shell script for other OS.it starts gradle build System.

.gitignore

Specifies the untracked files that Git should ignore.

build.gradle

Customizable properties for the build system. You can edit this file to specify the default build settings used by the application modules and also set the location of your keystore and key alias so that the build tools can sign your application when building in release mode. This file is integral to the project, so maintain it in a source revision control system.

gradle.properties

Project-wide Gradle settings.

gradlew

Gradle startup script for Unix.

gradlew.bat

Gradle startup script for Windows.

local.properties

Customizable computer-specific properties for the build system, such as the path to the SDK installation. Because the content of the file is specific to the local installation of the SDK, the local.properties should not be maintained in a source revision control system.

.iml

Module file created by the IntelliJ IDEA to store module information.

settings.gradle

Specifies the sub-projects to build.

Following section will give a brief overview few of the important application files.

app/src/main/java/com.mycompany.myfirstapp/MainActivity.java

A tab for this file appears in Android Studio when the New Project workflow finishes. When you select the file you see the class definition for the activity you created. When you build and run the app, the [Activity](http://developer.android.com/reference/android/app/Activity.html) class starts the activity and loads the layout file that says "Hello World!"

**package** com.example.mokibkhan.myfirsttest;  
  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
  
**public class** MainActivity **extends** AppCompatActivity {  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 }  
}

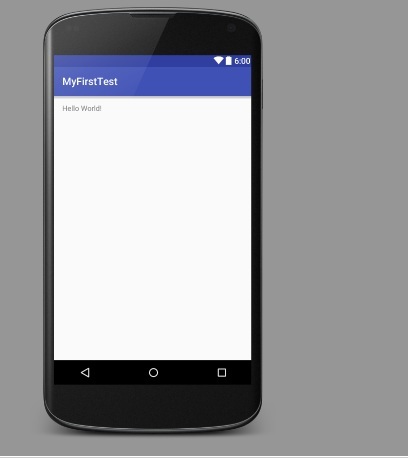
Here, *R.layout.activity\_main* refers to the *activity\_main.xml* file located in the *res/layout* folder. The *onCreate()* method is one of many methods that are called when an activity is loaded.

**The Layout File**

The **activity\_main.xml** is a layout file available in *res/layout* directory, which is referenced by your application when building its interface. You will modify this file very frequently to change the layout of your application. For your "Hello World!" application, this file will have following content related to default layout:

*<?***xml version="1.0" encoding="utf-8"***?>*<**RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:paddingBottom="@dimen/activity\_vertical\_margin"  
 android:paddingLeft="@dimen/activity\_horizontal\_margin"  
 android:paddingRight="@dimen/activity\_horizontal\_margin"  
 android:paddingTop="@dimen/activity\_vertical\_margin"  
 tools:context="com.example.mokibkhan.myfirsttest.MainActivity"**>  
  
 <**TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Hello World!"** />  
</**RelativeLayout**>

This is an example of simple *RelativeLayout* which we will study in a separate chapter. The *TextView* is an Android control used to build the GUI and it have various attribuites like *android:layout\_width*,*android:layout\_height* etc which are being used to set its width and height etc. The *@string* refers to the strings.xml file located in the res/values folder. Hence, @string/hello\_world refers to the hello string defined in the strings.xml fi le, which is "Hello World!.



**The Manifest File**

Whatever component you develop as a part of your application, you must declare all its components in a *manifest* file called **AndroidManifest.xml** which resides at the root of the application project directory. This file works as an interface between Android OS and your application, so if you do not declare your component in this file, then it will not be considered by the OS. For example, a default manifest file will look like as following file:

*<?***xml version="1.0" encoding="utf-8"***?>*<**manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="com.example.mokibkhan.aidlserver"**>  
  
 <**application  
 android:allowBackup="true"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:supportsRtl="true"  
 android:theme="@style/AppTheme"**>  
 <**activity  
 android:name=".MainActivity"  
 android:label="@string/app\_name"  
 android:theme="@style/AppTheme.NoActionBar"**>  
 <**intent-filter**>  
 <**action android:name="android.intent.action.MAIN"** />  
  
 <**category android:name="android.intent.category.LAUNCHER"** />  
 </**intent-filter**>  
 </**activity**>  
 </**application**>  
  
</**manifest**>

Following is the list of tags which you will use in your manifest file to specify different Android application components:

* <activity>elements for activities
* <service> elements for services
* <receiver> elements for broadcast receivers
* <provider> elements for content providers

**The Strings File**

The **strings.xml** file is located in the *res/values* folder and it contains all the text that your application uses. For example, the names of buttons, labels, default text, and similar types of strings go into this file. This file is responsible for their textual content. For example, a default strings file will look like as following file:

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

<resources>

<string name=*"app\_name"*>FirstApp</string>

<string name=*"action\_settings"*>Settings</string>

<string name=*"hello\_world"*>Hello world!</string>

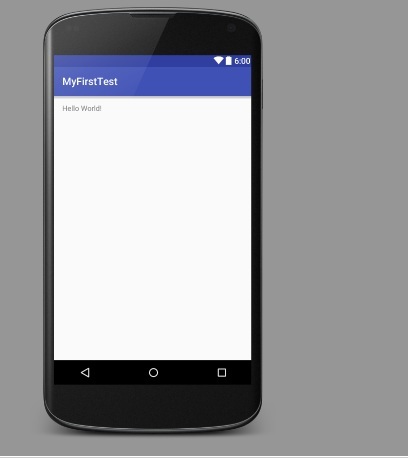
</resources>

**The R File**

The **gen/com.example.helloworld/R.java** file is the glue between the activity Java files like*MainActivity.java* and the resources like *strings.xml etc* . It is an automatically generated file and you should not modify the content of the R.java file.

**Running the Application**

Let's try to run our **Hello World!** application we just created. I assume you had created your **AVD** while doing environment setup. To run the app from Eclipse, open one of your project's activity files and click Run Eclipse Run Icon icon from the toolbar. Eclipse installs the app on your AVD and starts it and if everything is fine with your setup and application, it will display following Emulator window:



**Question: - How to run an Android Application on physical devices (Android mobile phones)?**

**Answer:** - 1- Running an app on physical devices like Samsung, Nexus, and Sony etc. during development is almost identical to running it on the emulator.

1. You need to enable USB debugging on the phone itself.

PATH= setting=>developer option => USB debugging

3-Install the Android USB device driver (eg. pdanet.exe) and then plug the phone into your computer using USB cable.

**Android Architecture: -**

Android operating system is a stack of software components which is roughly divided into five sections and four main layers as shown below in the architecture diagram

****

**NOTE :-**  Each layer provides different services to the layer just above it.

**.1)Linux kernel layer:-** This layer provides standard services like any linux kernel along with the android specific services.

**A) Provides standard services:-**

Android kernel just like any linux kernel provides generic (standard) OS services.

for example :- Its provides..

1. Security:- By providing a unique ID to each linux process with restricted access to resources .
2. Memory process management :- Each linux process will simultaneously access memory with a unique ID without interfering each other.
3. It handles the low levels details of files & networks I/O.
4. It also allows DEVICE DRIVER to be plugged in so that android can communicate with wide range of H/W components.
5. Portability so that it easily to run in various H/w architecture.

**B) Android specific service:-**

**a)** Android linux kernel includes its power management services because mobile devices run on battery power.

b) It provides memory sharing and memory management because mobile devices provide often limited memory

c) It also include Inter process communication (IPC) mechanism that allow most of the processes share data services and much more.

**2) System libraries:-**

* These libraries have typically written in C/C++ and for this, these libraries are referred as Natives libraries
* And these native libraries are to handle lots of performance sensitive cases on the devices like quickly rendering webpages, updating the display.
* These Libraries carry a set of instructions to guide the device in handling different types of data. For instance, the playback and recording of various audio and video formats is guided by the Media Framework Library.
* **System c library:-** which implements the standard OS system calls which do things like process & thread creation mathematical computation , memory allocation and much more .
* **Surface manager:-** for display management.
* **Media Framework:-** For playing the Audio and video files.
* **Webkit :-** for browser engine to display webpages.
* **Sqlite:-** for managing data in memory.
* SGL: 2D Graphics
* Open GL|ES: 3D Library
* Free Type: Font Rendering

**Note:- This layer also includes Android Runtime.**

**Android runtime:-** In android runtime, there are core libraries and AVM (Android Virtual Machine) which is responsible to run android application. AVM is like JVM but it is optimized for mobile devices. It consumes less memory and provides fast performance.

* Core java library:- these libraries are typically written in the java programming language.
* The Android runtime also provides a set of core libraries which enable Android application developers to write Android applications using standard Java programming language.

**Example:-**

* Basic java classes:- rt.jar includes basic classes for things like common data structure , concurrency mechanism &file I/O.
* android.jar packages:- one specific to life cycle of mobile Application
* org.\* packages:-Supports Internet services.
* junit.\* package:- Support unit testing of application.

**Dalvik virtual machine (DVM) :- (replaced by AVM)-** DVM is a software that actually executes the android application .

* As we know android application is written in java and then compiled into the bytecode files.
* The DX–Compiler converts java byte code files to a single DEX BYTECODE file (classes.dex).
* When user launches application then AVM executes DEX bytecode file**.**

Java source code

Java byte code

JVM

Java byte code

Java source code

Java byte code

Java compiler

Dalvik byte code

Dalvik byte code

AVM

Java compiler

DEX Compiler

**Application framework:-**

It contains s/w that many mobile application are likely to need.

For example:-

* Package manager- It sanctions the database to keep track of installed application packages on devices we are to call it application manager.
* Window Manager:- manages the windows comprising on application .
* View system – provides common user interface elements such as the icons ,entry boxes , button and more.
* Resource manager –This component manages non-compiled resources such as the strings, graphics & layout files.
* Activity manager- Manages the application life cycle and the navigation stack.
* Content provider –Provides inter – application data sharing for example- contact content provider ,media content provider etc.
* Location manager – Provides location & movement information such as generate by GPS system(Google map application).
* Notification manager – It place notification icons in the status Bar.

**Applications:-**

This layer contains all preinstalled Android application.

**Application Fundamental –**

Android apps are written in the Java programming language. The Android SDK tools compile your code—along with any data and resource files—into an APK (*Android package*), which is an archive file with an .apk suffix. One APK file contains all the contents of an Android app and is the file that Android-powered devices use to install the app.

Once installed on a device, each Android app lives in its own security sandbox:

* The Android operating system is a multi-user Linux system in which each app is a different user.
* By default, the system assigns each app a unique Linux user ID (the ID is used only by the system and is unknown to the app). The system sets permissions for all the files in an app so that only the user ID assigned to that app can access them.
* Each process has its own virtual machine (VM), so an app's code runs in isolation from other apps.
* By default, every app runs in its own Linux process. Android starts the process when any of the app's components need to be executed, then shuts down the process when it's no longer needed or when the system must recover memory for other apps.

1. **UI Components: -** Android supports numerous components & some of them are discussed below-

**NOTE:**- Every Android Widget is represented by a Java object which belongs to a Java class that are defined in android.widget package or in some other packages.

**TextView: -**

This is a class defined in android.widget.\*; an object of this class represented by an independent text (label) we can define a text view in xml view a follows:-

<Textview

android : Layout\_width= “wrap\_coontent”

android : Layout\_height= “wrap\_content”

android : text = “Enter 1st ID” />

Note: - TextView has numerous properties use by either common or specific attributes define in android namespace.

Note: - Also see CheckedTextView that is used with ListView.

**Button:** - A Button is class defined in android.widget package. An Object of this class

represents an individual click button which consist of text or both text & icon, that communicate what action occurs when the user clicks it.

For creation of every button we need to define an XML element (Tag) in Layout file as follows :-

1-Button using with text-

<Button

android: Layout\_width= “wrap\_content”

android : Layout\_height= “wrap\_content”

android : text = “@string/button\_text” />

2-Button With Icon & Text both : -

<Button :

android : Layout\_width= “wrap\_content”

android : Layout\_height= “wrap\_content”

android : text = “@string/button\_text”

android:drawableleft = “@drawable/button” />

**ImageButton –** It is a class defined in android.widget package, an object of this class provide functionality of an individual button with Icon image.

In order to use ImageButton we should define following element in layout content file.

<ImageButton

android: Layout\_width= “wrap\_content”

android : Layout\_height= “wrap\_content”

android:src = “@drawable/button\_icon” />

**EditText**: - EditText is a class defined in android.widget.\*; an object of this class is represented by an independent editable text field. An EditText can be define in xml file as follows—

<EditText android :

Layout\_width= “wrap\_coontent”

android : Layout\_height= “wrap\_content”

android : hint = “Enter 1st ID” />

Note: - we can make an EditText in different way by using different-2 attributes. Ex. Password, numbers etc.

**View :-** An **Object of** this class represents the basic building block for user interface components. A View occupies a rectangular area on the screen and is responsible for drawing and event handling. View is the base class for widgets, which are used to create interactive UI components (buttons, text fields, etc.).

**ViewGroup :-** The [ViewGroup](http://developer.android.com/reference/android/view/ViewGroup.html) is subclass of View class and the base class for layouts (Like LinearLayout, RelativeLayout etc), which are invisible containers that hold other Views (or other ViewGroups) and define their layout properties.

View

TextView View Group

Button EditText LinearLayout RelativeLayout FrameLayout TableLayout

CompoundButton

Switch RadioButton ToggleButton CheckBox

**View Hierarchy**

**Android Layout** :-

A layout defines the visual structure for a user interface, such as the UI for an [activity](http://developer.android.com/guide/components/activities.html) or [app widget](http://developer.android.com/guide/topics/appwidgets/index.html). You can declare a layout in two ways:

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

**NOTE:-** The advantage to declaring your UI in XML is that it enables you to better separate the presentation of your application from the code that controls its behavior. Your UI descriptions are external to your application code, which means that you can modify or adapt it without having to modify your source code and recompile.

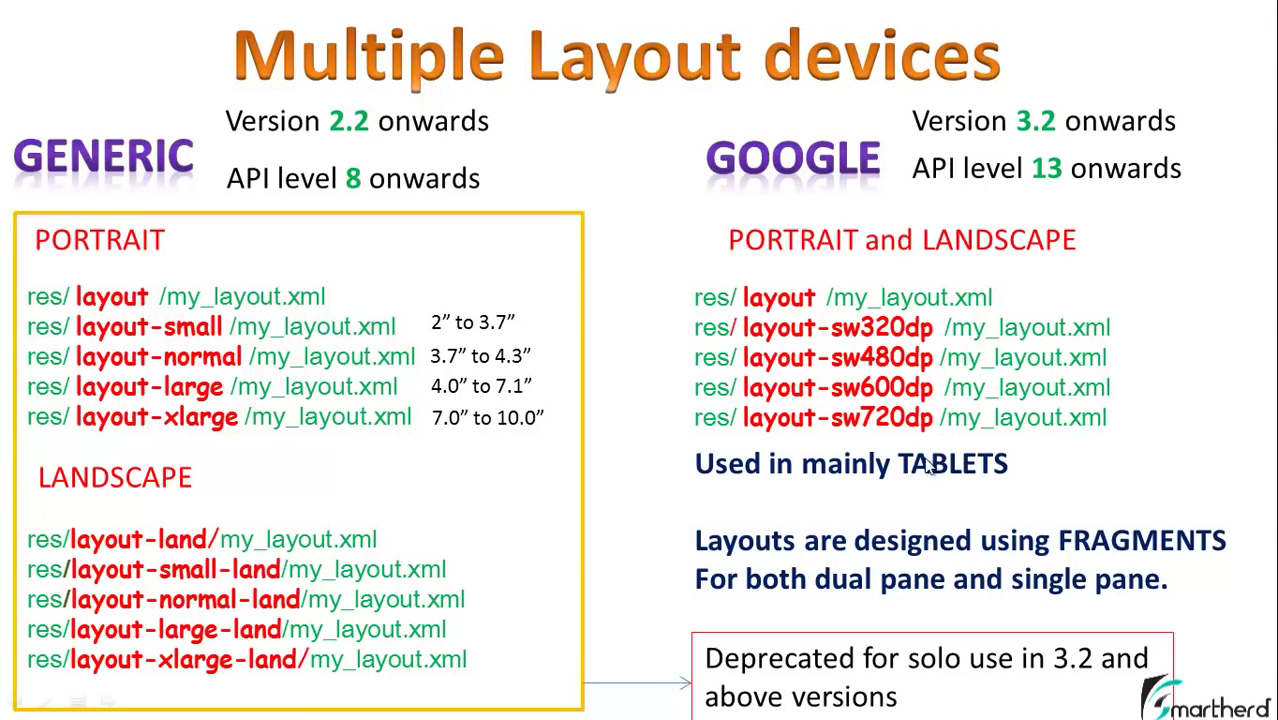
For example, you can create XML layouts for different screen orientations, different device screen sizes, and different languages. Additionally, declaring the layout in XML makes it easier to visualize the structure of your UI, so it's easier to debug problems.

**Types of Layout**s (ViewGroup):-

There are number of Layouts provided by Android which you will use in almost all the Android applications to provide different view, look and feel.

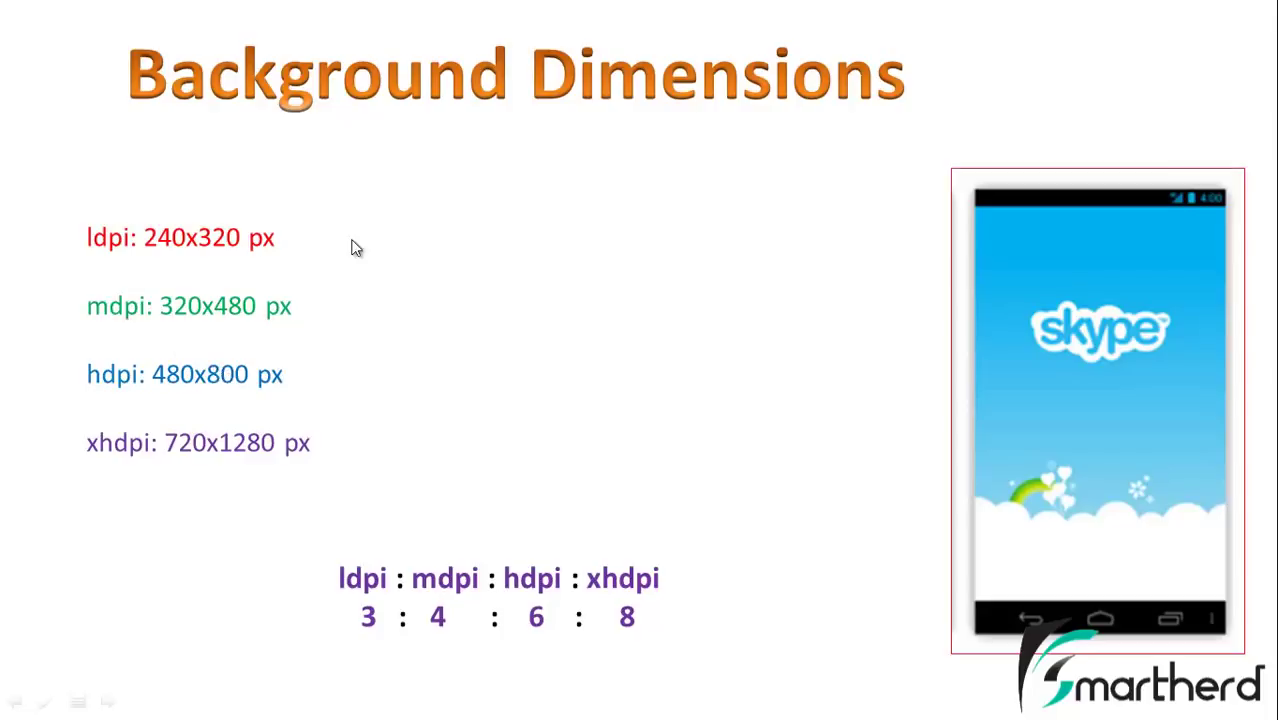
1. **RelativeLayout** :- RelativeLayout is a view group that displays child views in relative positions.
2. **LinearLayout** :- LinearLayout is a view group that aligns all children in a single direction, vertically or horizontally.
3. **TableLayout :**- TableLayout is a view that groups views into rows and columns.
4. **AbsoluteLayout** :- AbsoluteLayout enables you to specify the exact location of its children by absolute co-ordinates.
5. **FrameLayout :**- The FrameLayout is a placeholder on screen that you can use to display a single view.

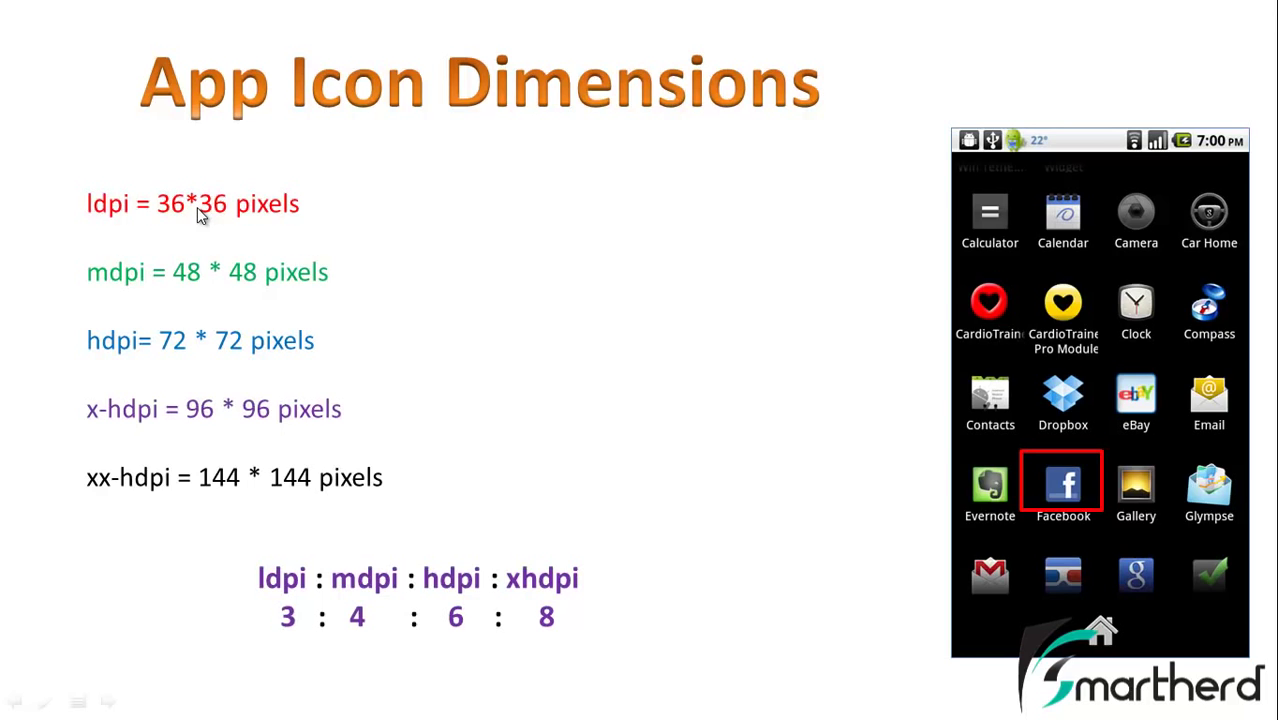
Note:- Any type of Layout must be defined in layout.xml file with respective TAG and put this file in appropriate layout subdirectory of res directory.



Where sw stands for smallest width.

**Drawable :-** contains images either for background or for icon:-





**Question: - What is an Event?**

Answer: - An Event is the change of state of any object either by user or by application itself.

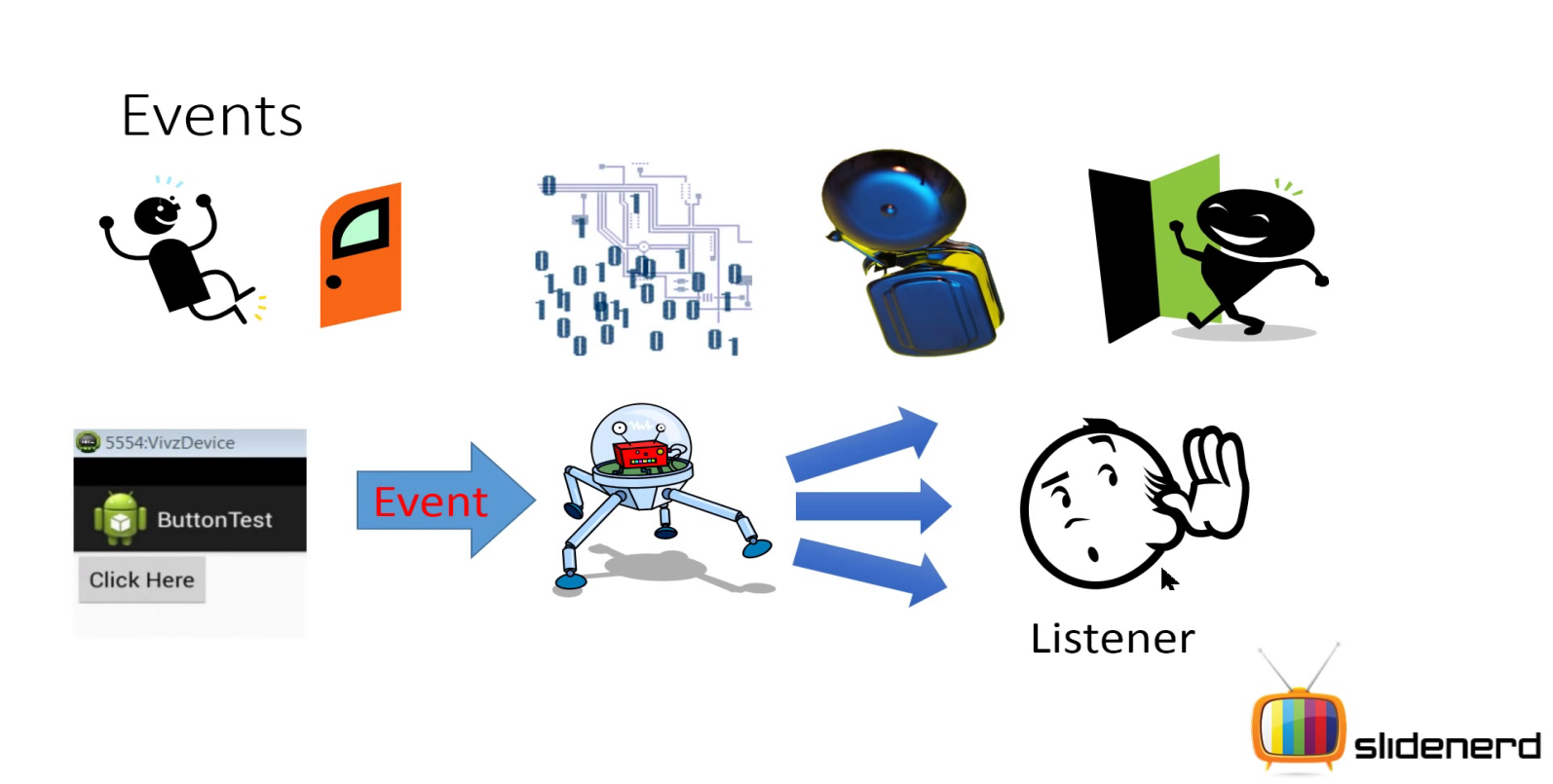
The event is broadly classified into two categories-

* 1. **Foreground Events-** Those events which require the direct interaction of User. They are generated as consequences of a person interacting with the graphical components in GUI.

**For example –** clicking a Button, checked/unchecked a RadioButton, selecting an item of list etc.

* 1. **Background Events -** Those events which don’t require the direct interaction of User.

**For example-** OS interrupts, Hardware/Software failure etc.

****

**Event Handling: -**

1. It is a mechanism in which a functionality gets executed automatically (by android OS) when any event has been generated.
2. Implementation of event handling can be done by implementation of following concept –

* Listener Interface contains declaration of event handler (call back method).
* Callback methods
* Listener class
* Listener object
* Source of event
* **Listener Interface: -** There are no. of predefined Listener Interfaces in Android API. Each listener Interface contains declaration of one or more call back method (event handler).

Note: - (1) - Call back method those methods which are defined by programmer and called by any runtime environment. Call back method can also be known as event handler.

OR

These are method which will be invoked by android OS to perform programmer specified functionality when any event has been generated.

* **Listener Class: -** listener class provides implementation of listener interface. It means we will provide definition of callback methods in listener class.
* **Listener object: -** it is an object of Listener class.
* **Source of event: -** It is any android widget (Object) which is responsible for event generating. Ex. Button, ImageButton,TextView, EditView etc.

**Hierarchy/Block Diagram---**

**General Steps for implementation of Event handling –**

1. Identify Event type.
2. Create Listener class by implementing Listener interface.
3. Override callback methods.
4. Create listener Object of listener class.
5. Identify source of event.
6. Do registration of listener object with source of event.

**How to do registration of Listener Object: -** Source of event is represented by an object of any android widget class. Each android widget class has a registration method of following signature for registration of listener object.

**setTypeListener (Listener Object);** /where type denotes type of event.

Like onClick, onLongClick, onFocusChange etc.

Note: - For each type of event there is same type of Listener Interface in Android API.

**Ex**.: - onClick-OnClickListener, onFocusChange-OnFocusChangeListener etc.

**Event Listener (listener Object) Registration process: -** it is the process by which an event listener has been registered with source of event by setTypeListener() method, so that the event handler is called when the source of event generates the event.

There are several ways to register our event listener for any event, but here we are going to know about top two ways.—

1. **Activity class implements the Listener Interface: -** By this way our Activity class treated as Listener class. Activity class implements the Listener Interface that’s why we put/define handler methods in Activity class.

For EventListener registration we have to call setTypeListener(this) on any source of event object.

1. **Registration using Layout file: -** Here we have to put our eventHandler() method in Activity class without implementing a Listener Interface & registering any Listener object.

Rather we will use Layout file to specify handler method by “android:OnClick” attribute. For Onclick event in each and every source of event.

<Button

android: Layout\_width= “wrap\_content”

android : Layout\_height= “wrap\_content”

android : text = “@string/button\_text”

android:OnClick=”addition”

/>

NOTE :- In this case programmer must define eventHandler() with following signature in Activity class.

**Signature of Programmer declared Event Handler method: -** The Handler method you declare in the android:onClick attribute must have signature as below----

1. Method should be public.
2. It has return type **void.**
3. It has the View as its only parameter.

public void methodname (View v)

{

……..………

}

Example :-

Public class MainActivity extends Activity

{

………

………

public void addition (View v)

{

……..………;

}

}

NOTE :- This way is possible in case of OnClick event only .

**Showing Pop-up Messages-**

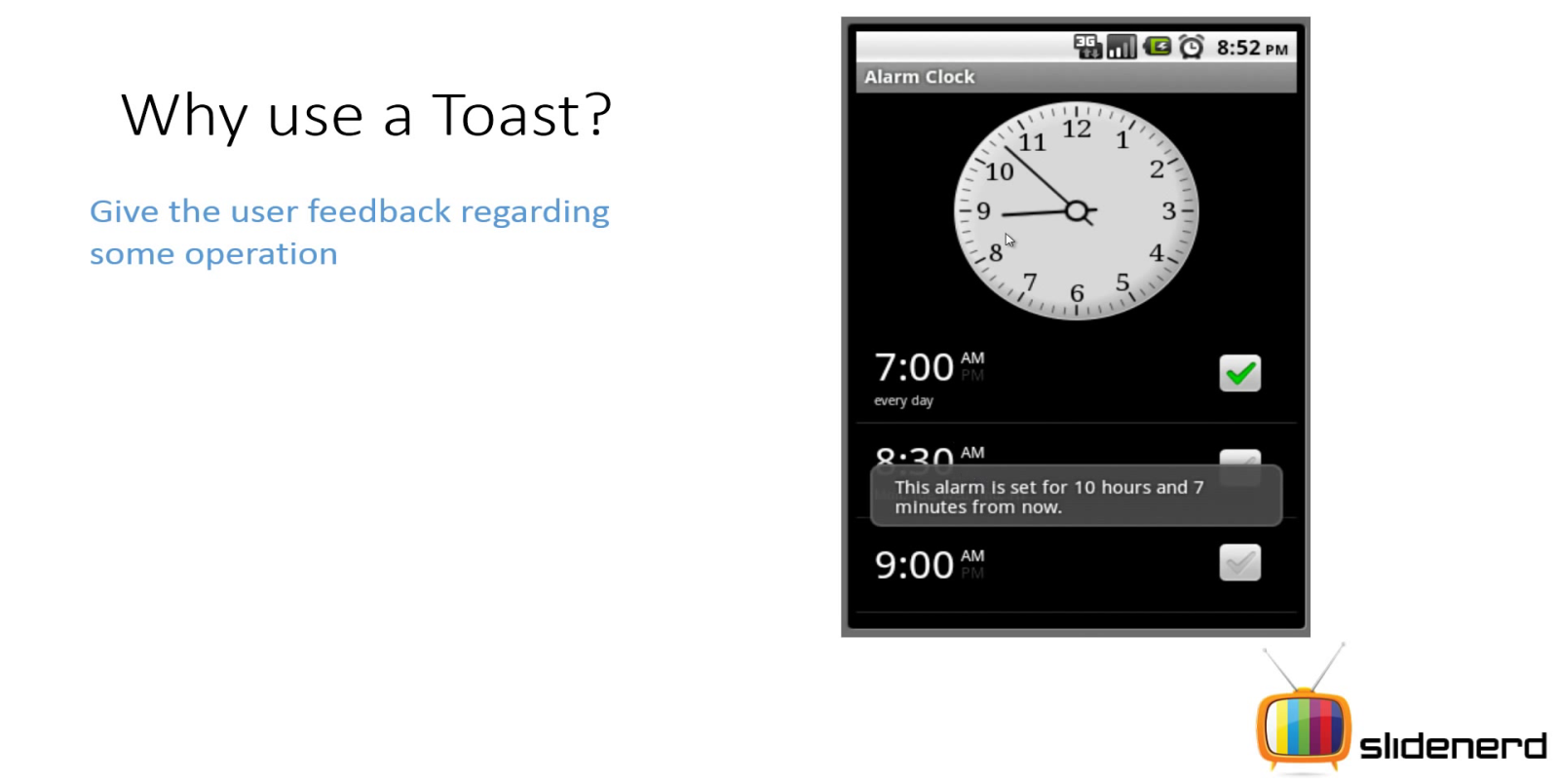
There are many situations where you might want your app to show a quick message to the user, without necessarily waiting for the user to respond. For example, when a user performs an action like sending an email or deleting a file, your app should show a quick confirmation to the user. Often the user doesn't need to respond to the message. The message needs to be prominent enough that the user can see it, but not so prominent that it prevents the user from working with your app.

**Android provides two ways to show pop-up messages -**

1. **Toast: -** It is a class defined in android.widget package. An object of this class provides facility of showing simple feedback message about an operation in a small popup.

It only fills the amount of space required for the message and current activity remains visible & interactive.

**For example: -**



**How to create a Toast: -**

By Factory method: - There is a static method makeText() in Toast class receives three arguments & returns object of Toast class.

Arguments are:-

i- Context object

ii- String message

iii- time duration

**Syntax: -** **Toast.makeText(context,“message saved in draft”,Toast.LENGTH\_SHORT);**

**What is Context: -** It is a class defined in android.content package. An object of this class is representing a buffer that has the references of all data & resources about an application.

It is basically an application environment to our android app ,by which any widget or component can access required data & resources from it.

**How to get Context object: -** we can get the context by invoking following method.

1. getApplicationContext();
2. getContext();
3. getBaseContext() ;
4. by this keyword (only in Activity in Service class)

**Note: -** All Activity & Service classes are already subclasses of Context class.

**For example-** Context is like a Hotel Room Service.

**How to show a Toast object: -** Toast class has a method called show(); (a non-static method) for representing a Toast object on screen.

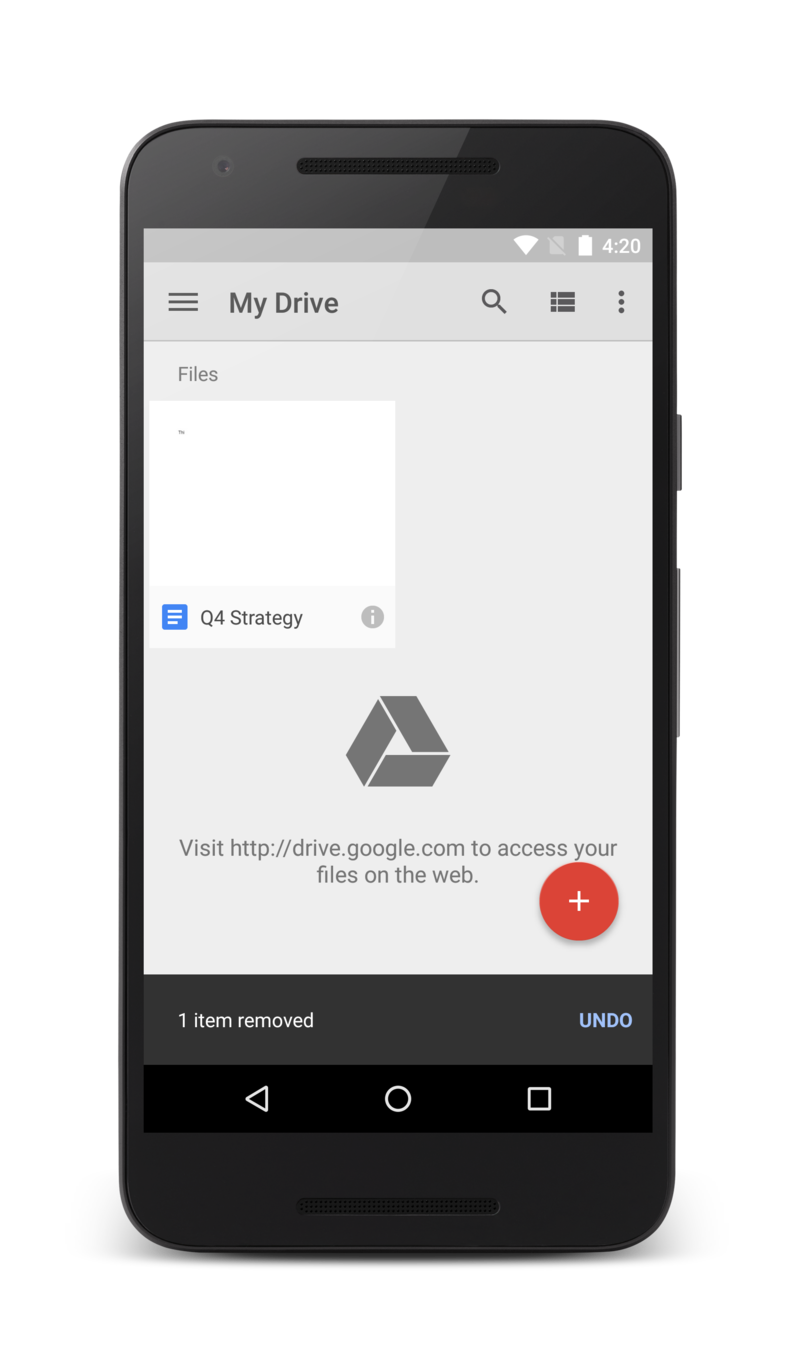
**Toast t=Toast.makeText(Context, “message saved”, Toast.LENGTH\_SHORT);**

**t.show();**

**OR**

**Toast.makeText(Context, “message saved”, Toast.LENGTH\_SHORT).show();**

1. **SnackBar** – It is a class defined in android.support.design.widget package. An object of this class provides lightweight feedback about an operation. It shows a brief message at the bottom of the screen on mobile and lower left on larger devices. Snackbars appear above all other elements on screen and only one can be displayed at a time.



A [Snackbar](http://developer.android.com/reference/android/support/design/widget/Snackbar.html) shows a message at the bottom of the activity, but the rest of the activity is still usable.

Snackbars can contain an action which is set via [setAction(CharSequence, android.view.View.OnClickListener)](http://developer.android.com/reference/android/support/design/widget/Snackbar.html#setAction(java.lang.CharSequence, android.view.View.OnClickListener)).

\*\*To be notified when a snackbar has been shown or dismissed, you can provide a [Snackbar.Callback](http://developer.android.com/reference/android/support/design/widget/Snackbar.Callback.html) via [setCallback(Callback)](http://developer.android.com/reference/android/support/design/widget/Snackbar.html#setCallback(android.support.design.widget.Snackbar.Callback))

**How to Create SnackBar** –

Snackbar  
  .make(parentLayout, “1 item removed”, Snackbar.LENGTH\_LONG)  
  .setAction(“UNDO”, myOnClickListener)  
  .show(); // Don’t forget to show!

#### public [Snackbar](http://developer.android.com/reference/android/support/design/widget/Snackbar.html) setAction ([CharSequence](http://developer.android.com/reference/java/lang/CharSequence.html) text, [View.OnClickListener](http://developer.android.com/reference/android/view/View.OnClickListener.html) listener)

Set the action to be displayed in this [Snackbar](http://developer.android.com/reference/android/support/design/widget/Snackbar.html).

##### Parameters

|  |  |
| --- | --- |
| *Text* | Text to display Action. |
| *Listener* | callback to be invoked when the action is clicked |

Whenever user clicks at Action (here UNDO) onClick event is triggered.

**Note:** The [Snackbar](http://developer.android.com/reference/android/support/design/widget/Snackbar.html) class supersedes [Toast](http://developer.android.com/reference/android/widget/Toast.html). While [Toast](http://developer.android.com/reference/android/widget/Toast.html) is currently still supported, [Snackbar](http://developer.android.com/reference/android/support/design/widget/Snackbar.html) is now the preferred way to display brief, transient messages to the user.

**Ques.- What is Material Design?**

**Ans-** It is a mechanism which guides for visual, motion, and interaction design across platforms and devices. Android now includes **support** for material design apps. To use material design in your Android apps, follow the guidelines defined in the [material design specification](http://www.google.com/design/spec) and use the new components and functionality available in Android 5.0 (API level 21) and above.

Android provides the following elements for you to build material design apps:

* A new theme
* New widgets for complex views
* New APIs (methods) for custom shadows and animations

**NOTE -** For more information about implementing material design on Android, see [Creating Apps with Material Design](http://developer.android.com/training/material/index.html).

**Ques.-What is Design Library?**

**Ans.** Android 5.0 Lollipop was one of the most significant Android releases ever, due to the introduction of material design. [Material design specification](http://www.google.com/design/spec)  is a great place to start to adopt material design, but we understand that it can be a challenge for developers, particularly ones concerned with backward compatibility. Here comes **Android Design Support Library in existence.**

With a little help from the new **Android Design Support Library,** which is brought a number of important material design components to all developers and to all Android 2.1 or higher devices. You’ll find a navigation drawer view, floating labels for editing text, a floating action button, snackbar, tabs, and a motion and scroll framework to tie them together with the help of **CoordinatorLayout**.

**Ques.-How to add Design Library in project?**

**Ans.** Open the build.gradle file inside your application module directory.

**Note:** Android Studio projects contain a top-level build.gradle file and a build.gradle file for each module. Be sure to edit the file for your application module.

Add a new build rule under dependencies. For example:

dependencies {

       compile **'com.android.support:design:23.2.1'**  
    }  
OR

file/project structure/ app /dependencies tab / + /Library modules

**FloatingActionButton –** It is a class defined in android.support.design.widget package (Design support library), an object of this class is represented by a circled icon floating above the UI.

**Note- It’s a sub class of ImageButton class.**

In order to use FloatingActionButton we should define following element in layout file.

<**android.support.design.widget.FloatingActionButton  
 android:id="@+id/fab"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="bottom|end"  
 android:layout\_margin="@dimen/fab\_margin"  
 android:src="@android:drawable/ic\_dialog\_email"** />

**Toggle Button: -** This is a class defined in android.widget . An object of this class represents functionality of an individual toggle button that allows user to change setting between two states.

We can define a Toggle button in xml Layout file with <Toggle Button> Tag.

C:\Users\Mokib Khan\Desktop\And Batches\pics\toggle.jpg

**Switch: -** Switch is a class defined in android.widget package from android 4.0 API .It introduces another kind of Toggle Button called a Switch which represents Switch button that provides a slider control to the user between to states.

To display a switch ,We have to define Switch in xml Layout file.

Eg:-

<Switch>….

</Switch>

C:\Users\Mokib Khan\Desktop\And Batches\pics\switch.jpg

**Note: -** ToggleButton & Switch are subclass of CompoundButton and both are function in same manner.

**CompoundButton class**:- A button with two states, checked and unchecked. When the button is pressed or clicked, the state changes automatically.

Note:-Button is a super class of CompoundButton.

Note :- In case of Switch ,we have to take min SDK 14 .

**Responding to Click Event: -** When an user select a toggle button or switch then the object receive an onClick event. We have to define an event handler in <Toggle Button> or Switch> by adding android:onClick

OR

### Responding to OnCheckedChangeEvent :

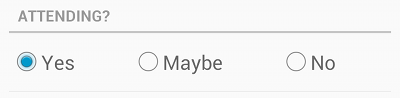
When an user slide to a switch then the object receive an onCheckedChangeEvent.

Listener- OnCheckedChangeListener

Event Handler – OnCheckedChanged(CompoundButton cb, boolean isChecked )

**Radio Button: -** This is a class defined in android.widget.An object of this class provides functionality of an individual Radio Button.

Radio Button allow user to select one option from multiple.



**Ex.=** which Technology do you prefer ?

* Java
* .net
* Php
* C

For each Radio Button option, we have to define <Radio Button> tag in your Layout file. We should use radio buttons for optional sets that are mutually exclusive then we must group them together inside a <Radio Group> to make Radio Buttons mutually exclusive. By Grouping them together the system ensure that only one Radio Button can be selected at that time.

**Responding to click event: -** Whenever the user select any Radio Button, the corresponding Radio Button receives onClick event.

**Check Box: -** This is a class defined in android.widget package. An object of this class represents functionality of Check Box which allows the user to select one or more option for selection set.

Typically we should present each Check Box option in vertical list.

To create a Check Box option we have to define <Check Box> tag in xml Layout.

**Responding to click event: -** When the user select a CheckBox, object receive onClick event.



Note:- CompoundButton is a Super class of CheckBox, RadioButton classes as well.

OR

**Responding to onCheckedChange event:** When the user checked a CheckBox, object receive onCheckedChange event.

Listener- OnCheckedChangeListener

Event Handler – OnCheckedChanged(CompoundButton cb, boolean isChecked )

**RatingBar** :- This is a class defined in android.widget package. An object of this class represents functionality of Rating Bar with Star icons, which allows the user to give rating value to anything in App by touching or dragging or clicking on it.

To create a RatingBar, we have to define <RatingBar> tag in xml Layout.

**Responding to event: -** When the user touches, drags or clicks the RatingBar, its object receive onRatingBarChanged event.

Listener Name- OnRatingBarChangeListener

Event Handler – **onRatingChanged(RatingBar r, float ratingValue, Boolean fromUser)-**

Notification that the rating has changed. Clients can use the fromUser parameter to distinguish user-initiated changes from those that occurred programmatically. This will not be called continuously while the user is dragging, only when the user finalizes a rating by lifting the touch.

##### Parameters

|  |  |
| --- | --- |
| *ratingBar* | The RatingBar whose rating has changed. |
| *rating* | The current rating. This will be in the range 0..numStars. |
| *fromUser* | True if the rating change was initiated by a user's touch gesture or arrow key/horizontal trackbell movement. |

NOTE :- By default rating Value changes by 1.0, when user is selected or deselected one star icon and 0.5 rating value for half star icon .

Eg :-

C:\Users\Mokib Khan\Pictures\rat.jpg

**ImageView** :- This is a class defined in android.widget package. An object of this class

represents functionality to hold an image file in App.

To create a Image View, we have to define <ImageView> tag in xml Layout

NOTE :- At first we have to put image file in appropriate drawable directory according to its density.

NOTE:- ImageView comes with different configuration options to support different scaleTypes. scaleType options are used for scaling the bounds of an image to the bounds of Image view. Below are the listed values for scaleType attributes supported.

* CENTER
* CENTER\_CROP
* FIT\_XY

<ImageView

android:id="@+id/imageView1"

android:layout\_width="fill\_parent"

android:layout\_height="150dp"

android:layout\_margin="5dp"

android:src="@drawable/image1" />

<ImageView

android:id="@+id/imageView2"

android:layout\_width="fill\_parent"

android:layout\_height="150dp"

android:layout\_margin="5dp"

android:background="#fff"

android:padding="3dp"

android:scaleType="fitXY"

android:src="@drawable/image2" />

<ImageView

android:id="@+id/imageView3"

android:layout\_width="fill\_parent"

android:layout\_height="200dp"

android:layout\_margin="5dp"

android:background="#fff"

android:paddingBottom="50dp"

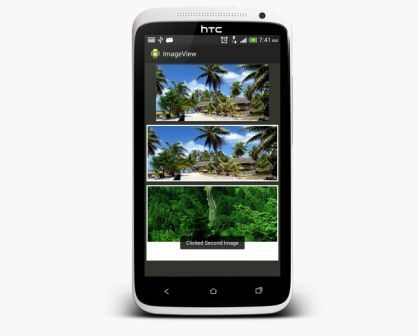
android:paddingLeft="3dp"

android:paddingRight="3dp"

android:paddingTop="3dp"

android:scaleType="fitXY"

android:src="@drawable/image2" />



**Date & Time Widgets**

**AnalogClock :-** This is a class defined in android.widget package. An object of this class

represents functionality of two handed analog watch one for hour indicator and second for minute indicator.

To create a Analog Clock, we have to define <AnalogClock> tag in xml Layout.

**Eg :-** <AnalogClock  
        android:id="@+id/analog\_clock"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content" />  
 **NOTE:**- **this class was deprecated in API level 23 (Android 6.0).**

**DigitalClock :- :-** This is a class defined in android.widget package. An object of this class represents functionality of a digital watch which displays hour, minute, seconds in digital format.

To create a Digital Clock, we have to define <DigitalClock> tag in xml Layout

**Eg :-**

<DigitalClock  
        android:id="@+id/digital\_clock"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content" />

****

**NOTE:-** Both AnalogClock and DigitalClock are unable to modify time of Device .

**NOTE:**- **this class was deprecated in API level 17 (Android 4.2).**  
It is recommended you use [TextClock](http://developer.android.com/reference/android/widget/TextClock.html) instead.

**TextClock:**- This is a class defined in android.widget package from API 17. An object of this class displays the current date and/or time as a formatted string (patterns).

To create a Text Clock, we have to define <TextClock> tag in xml Layout

NOTE:- Minimum SDK version must be 17 because backward compatibility was not provided for this widget.

<TextClock

android:format12Hour="MMM dd, yyyy | hh:mm:ss aa"

android:format24Hour="MMM dd, yyyy"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content" />

OR

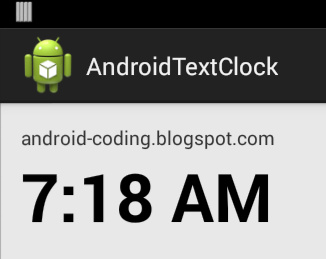
<TextClock

android:format12Hour="hh:mm:ss aa"

android:textSize="20sp”

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content" />



It is possible to determine whether the system is currently in 24-hour mode by calling [is24HourModeEnabled()](http://developer.android.com/reference/android/widget/TextClock.html#is24HourModeEnabled()).

The rules used by this widget to decide how to format the date and time are the following:

* In 24-hour mode:
  + Use the value returned by [getFormat24Hour()](http://developer.android.com/reference/android/widget/TextClock.html#getFormat24Hour()) when non-null
  + Otherwise, use the value returned by [getFormat12Hour()](http://developer.android.com/reference/android/widget/TextClock.html#getFormat12Hour()) when non-null
  + Otherwise, use a default value appropriate for the user's locale, such as h:mm a
* In 12-hour mode:
  + Use the value returned by [getFormat12Hour()](http://developer.android.com/reference/android/widget/TextClock.html#getFormat12Hour()) when non-null
  + Otherwise, use the value returned by [getFormat24Hour()](http://developer.android.com/reference/android/widget/TextClock.html#getFormat24Hour()) when non-null
  + Otherwise, use a default value appropriate for the user's locale, such as HH:mm

**Chronometer –** This is a class defined in android.widget package. An object of this class provide functionality of simple Timer.

In order to create a Chronometer, Developer has to define <Chronometer > in layout XML file as follows –

<**Chronometer  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/chronometer"  
 android:format="%s"  
 android:textSize="30sp"/>**

You can give it a start time in the [elapsedRealtime()](http://developer.android.com/reference/android/os/SystemClock.html#elapsedRealtime()) (Returns milliseconds since boot, including time spent in sleep) timebase, and it counts up from that, or if you don't give it a base time, it will use the time at which you call [start()](http://developer.android.com/reference/android/widget/Chronometer.html#start()). By default it will display the current timer value in the form "MM:SS" or "H:MM:SS", or you can use [setFormat(String)](http://developer.android.com/reference/android/widget/Chronometer.html#setFormat(java.lang.String)) to format the timer value into an arbitrary string.

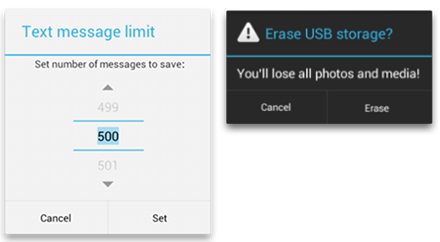
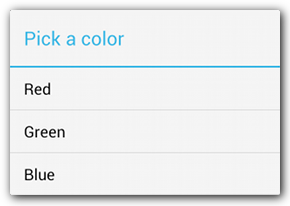
Example - chronometer.setBase(SystemClock.*elapsedRealtime*());

chronometer.start();

**On Responding to OnChronometerTick Event-** when chronometer has been incremented on its own, It receives this event.

**Listener-** OnChronometerTickListener

**Callback- onChronometerTick(Chronometer cm)-** This callback has been invoked by OS whenever chronometer is incremented .

**Dialog** :- A dialog is a small window that prompts the user to make a decision or enter additional information. A dialog does not fill the screen and is normally used for events that require users to take an action before they can proceed.  

The Dialog class is the base class for dialogs, but you should avoid instantiating Dialog directly. Instead, use one of the following subclasses:

1.AlertDialog:-A dialog that can show a title, an icon and up to three buttons, a list of selectable items.

2.DatePickerDialog:-A dialog with a pre-defined UI that allows the user to select a date

3. TimePickerDialog :- A dialog with a pre-defined UI that allows the user to select a time.

NOTE:- These classes define the style and structure for your dialog, but you should use a [DialogFragment](http://developer.android.com/reference/android/support/v4/app/DialogFragment.html) as a container for your dialog. The [DialogFragment](http://developer.android.com/reference/android/support/v4/app/DialogFragment.html) class provides all the controls you need to create your dialog and manage its appearance, instead of calling methods on any [Dialog](http://developer.android.com/reference/android/app/Dialog.html) object.

NOTE:- Although [DialogFragment](http://developer.android.com/reference/android/app/DialogFragment.html) was first added to the platform in Android 3.0 (API level 11), if your app supports versions of Android older than 3.0—even as low as Android 1.6—you can use the [DialogFragment](http://developer.android.com/reference/android/support/v4/app/DialogFragment.html) class that's available in the [support library](http://developer.android.com/tools/support-library/index.html) for backward compatibility.

If your app's [minSdkVersion](http://developer.android.com/guide/topics/manifest/uses-sdk-element.html#min) is 11 or higher, you can instead use the platform version (android.jar) of [DialogFragment](http://developer.android.com/reference/android/app/DialogFragment.html).

1. **AlertDialog** :- It is a subclass of Dialog (defined in android.app package) that can display a dialog with one, two or three buttons along with Title and Icon.

It is used, if you want to ask the user about taking a decision between yes or no or remindMelater in response of any particular action taken by the user, by remaining in the same activity and without changing the screen.

In order to make an alert dialog , we need to make an object of AlertDialog.Builder which an inner class of AlertDialog. Its syntax is given below

AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(this);

Now we have to set the positive (yes) or negative (no) or neutral button using the object of the AlertDialogBuilder class. Its syntax is

alertDialogBuilder.setPositiveButton(CharSequence text, DialogInterface.OnClickListener listener)

alertDialogBuilder.setNegativeButton(CharSequence text, DialogInterface.OnClickListener listener)

# DialogInterface.OnClickListener:-

# Interface used to allow the creator of a dialog to run some code when an item (any Button) on the dialog is clicked.

**EventHandler** :- public void [onClick](http://developer.android.com/reference/android/content/DialogInterface.OnClickListener.html#onClick(android.content.DialogInterface, int))([DialogInterface](http://developer.android.com/reference/android/content/DialogInterface.html) dialog, int which)

This method will be invoked by OS when a button in the dialog is clicked.

##### Parameters

|  |  |
| --- | --- |
| *Dialog* | The dialog that received the click. |
| *Which* | The button that was clicked (e.g. [BUTTON1](http://developer.android.com/reference/android/content/DialogInterface.html#BUTTON1)) or the position of the item clicked. public static final int BUTTON\_POSITIVE The identifier for the positive button.  Constant Value: -1 (0xffffffff) |

#### public static final int  BUTTON\_NEGATIVE

The identifier for the negative button.

Constant Value: -2 (0xfffffffe)

#### public static final int BUTTON\_NEUTRAL

The identifier for the neutral button.

Constant Value: -3 (0xfffffffd)

After creating and setting the dialog builder , we will create an alert dialog by calling the create() method of the builder class. Its syntax is

AlertDialog alertDialog = alertDialogBuilder.create();

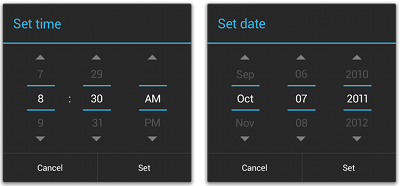
This will create the alert dialog and will show it on the screen by calling…..

alertDialog.show();

Apart from this , we can use other functions provided by the Builder class to customize the alert dialog. These are listed below

|  |  |
| --- | --- |
| **Sr.No** | **Method type & description** |
| 1 | **setIcon(Drawable icon)** This method set the icon of the alert dialog box. |
| 2 | **setCancelable(boolean cancelable)** This method sets the property that the dialog can be cancelled or not,when pressing back button or else. |
| 3 | **setMessage(CharSequence message)** This method sets the message to be displayed in the alert dialog |
| 4 | **setOnCancelListener(DialogInterface.OnCancelListener onCancelListener)** This method Sets the callback that will be called if the dialog is canceled. |
| 5 | **setTitle(CharSequence title)** This method set the title to be appear in the dialog |

1. **Pickers :-** Android provides controls for the user to pick a time or pick a date as ready-to-use dialogs. Each picker provides controls for selecting each part of the time (hour, minute, AM/PM) or date (month, day, year).



1. **TimePicker :-** This class is defined in android.widget package. An object of this class provides facility to render Time picker widget in current Activity with/without the help of TimePickerDialog, which allows the user to select the time of day, in either 24 hour or AM/PM mode.

In order to create TimePicker widget , we have to define <TimePicker> Tag in layoyt XML file as follows-

<TimePicker

android:id=*"@+id/timePicker1"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_below=*"@+id/textView1"*

android:layout\_centerHorizontal=*"true"*

android:layout\_marginTop=*"18dp"* />

**When TimePicker component rendering without TimePickerDialog** :-

**Responding to event: -** When the time has been adjusted by the user. TimePicker receives OnTimeChanged event.

Listener Name- OnTimeChangedListener

Event Handler – **public** **void** onTimeChanged(TimePicker tp, **int** selectedHour, **int** selectedMinute);

**When TimePicker component rendering with TimePickerDialog** :-

**Responding to event: -** When the time has been adjusted by the user and pressed Set/cancel Button. TimePickerDialog receives OnTimeSet event.

Listener Name- OnTimeSetListener

Event Handler – **public** **void** onTimeSet(TimePicker tp, **int** selectedHour, **int** selectedMinute);

Note:- In this case TimePickerDialog implicitly uses its own Timepicker Widget.

Note:- TimePickerDialog has been created programmatically.

1. **DatePicker :-** This class is defined in android.widget package. An object of this class provides facility to render Date picker widget in current Activity with/without the help of DatePickerDialog, which allows the user to select the day , month and year.

In order to create DatePicker widget, we have to define <DatePicker> Tag in layoyt XML file as follows-

<DatePicker

android:id=*"@+id/datePicker1"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"* />

Note:- DatePicker can be shown in either Spinner or CalendarView mode by using following attributes android:calendarViewShown=*"false"*

android:spinnersShown=*"true"*

Note:- android:datePickerMode=”spinner/calendarView” – introduced from Android Lollipop.

**When DatePicker component without DatePickerDialog** :-

**Responding to event: -** When the date has been adjusted by the user. DatePicker receives OnDateChanged event.

Listener Name- OnDateChangedListener

Event Handler – **public** **void** onDateChanged(DatePicker dp, **int** selectedYear,

**int** selectedMonth, **int** selectedDay) ;

**How to set onDateChangedListener ?**

By using following method :- init(year, month, day, **listnerObject**);

**When DatePicker component with DatePickerDialog** :-

**Responding to event: -** When the date has been adjusted by the user and pressed Set Button. DatePickerDialog receives OnDateSet event.

Listener Name- OnDateSetListener

Event Handler – **public** **void** onDateSet(DatePicker dp, **int** selectedYear,

**int** selectedMonth, **int** selectedDay) ;

Note:- In this case DatePickerDialog implicitly uses its own DatePicker Widget.

Note:- DatePickerDialog has been created programmatically.

**CalendarView :-**

This class is defined in android.widget package (API 11) and an object of this class represents a calendar widget for displaying and selecting dates. A user can select a date by taping on it and can scroll and fling the calendar to a desired date.

In order to use CalendarView, we may define <CalendarView> in layout file as follows –

<**CalendarView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/calendarView"**

/>

**Responding to event: -** When the date has been adjusted by the user. CalendarView receives OnDateChanged event.

Listener Name- CalendarView.OnDateChangedListener

Event Handler – **public** **void** onSelectedDayChange(CalendarView cv, **int** selectedYear,

**int** selectedMonth, **int** selectedDay) ;

**How to set onDateChangedListener ?**

#### By using following method :-

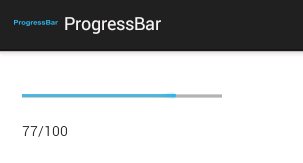
#### Public void setOnDateChangeListener ([CalendarView.OnDateChangeListener](http://developer.android.com/reference/android/widget/CalendarView.OnDateChangeListener.html) listener)

1. **ProgressBar :-**

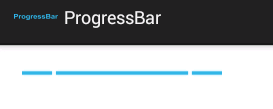
  ProgressBar is a class has been defined in android.widget package.An object of this class provides a graphical view indicator which shows some progress. It shows a bar representing the completing of the task. Normally the ProgresBar does not display the amount of completion in numbers. If we want, we can display it in the TextView.

**Important things about ProgressBar:**

            By default the progress bar maximum value is 100. It can also be changed by setting the our own maximum value using **android:max** attribute. Example;



If we know the amount of work, then we can use this progress bar showing the actual progress. But if we do not know the amount of work to be done, then we can use **progressBar.setIndeterminate(true)** in the activity class programmatically or using **android:indeterminate="true"**in the layout xml file which will enable the intermediate mode. In the indermediate mode the actual progress will not be shown. A cyclic animation will be shown to indicate that some progress is happening. This intermediate mode is shown below;



By default the progress bar will be displayed as a spinning wheel. If we want it to be displayed as a horizontal bar, we need to use  style="?android:attr/progressBarStyleHorizontal" attribute.

The default mode is shown below;



There is also a secondary progress displayable on a progress bar which is useful for displaying intermediate progress, such as the buffer level during a streaming playback progress bar.

**How to create:**

            ProgressBar can be created using **<ProgressBar>** tag in the layout xml file.

1. **ProgressDialog :-**  
   ProgressDialog is a sub class of AlertDialog which has been defined in android.app package. An object of this class provides a dialog box/dialog window which shows a progress indicator and an optional text message. The progress dialog will be used if we want the user to wait till the task complete.

**NOTE-** The dialog can be made cancelable on back key press.The progress range is 0..10000.

**NOTE:-** This is an extension of AlertDialog. ProgressDialog is almost same as ProgressBar with the exception that this is displayed as a dialog box. This will display the progress in numbers.

**Difference between ProgressBar and ProgressDialog:**

           The programmers often get this doubt. ProgressBar is a view indicating the amount of progress done. The application can change the value. This can be used in the layout to show the progress.

On the other side ProgressDialog is a dialog box (like alert). The user must wait for the progress to complete

**Important methods of ProgressDialog:**  
setMessage(); method is used to show the message to the user. Example: Loading... (or) Please wait....  
setTitle(); method is used to set a title to the dialog box.  
setProgressStyle(ProgressDialog.STYLE\_HORIZONTAL); is used to show the horizontal progress bar in the dialog box.  
setProgressStyle(ProgressDialog.STYLE\_SPINNER); is used to show the circle/spinning progress bar in the dialog box.  
setMax();method is used to set the maximum value.  
getProgress(); method is used to get the current progress value in numbers.  
[incrementProgressBy](http://developer.android.com/reference/android/app/ProgressDialog.html#incrementProgressBy(int))(int diff)-

[incrementSecondaryProgressBy](http://developer.android.com/reference/android/app/ProgressDialog.html#incrementSecondaryProgressBy(int))(int diff)-  
**How to create:**  
ProgressDialog can be created programatically using the ProgressDialog class instance as shown below;  
**ProgressDialog progressDoalog=new ProgressDialog(MainActivity.this);**  
Here MainActivity.this is the activity class.

OR

ProgressDialog pD=ProgressDialog.show(Context c, CharSequence title, CharSequence message, Boolean indeterminate);

**Layout Inflation:-** A mechanism in which precompiled XML tags get converted into Java Objects.

**Adapter**

Adapter is an interface defined in android.widget package and implementation of this interface acts as Bridge between an adapter view and data source for the creation of view.

1. The adapter provides access to the data items.
2. It is also responsible for making a child view object for each item in data set.

Adapter interface has many direct and indirect implementing classes.

But here we talk about only three that are ArrayAdapter, SimpleCursorAdapter and BaseAdapter.

**Note:-**

1. The adapter is only responsible for taking the data from data source and managing the data.
2. Adapter takes data from data source like database OR Array as source and create a view and passes to Adapter View.

On the basis of type of data Source Adapter are of 3 types (Implementing classes).

1. **ArrayAdapter:-** It takes the data from an array and create a view out of it and passes to AdapterView.
2. **SimpleCursorAdapter :**- It takes data from Cursor object and create a view out of that data and passes to Adapter View.
3. **BaseAdapter:**- It is a custom adapter that allows you to define and customize as how the data should be extracted? How to data should be processed? How the child view should be generated.

OR

Programmatically, It is an abstract sub class of common implementation for an [Adapter](http://developer.android.com/reference/android/widget/Adapter.html) that can be used in both [ListView](http://developer.android.com/reference/android/widget/ListView.html) (by implementing the specialized [ListAdapter](http://developer.android.com/reference/android/widget/ListAdapter.html) interface) and [Spinner](http://developer.android.com/reference/android/widget/Spinner.html) (by implementing the specialized [SpinnerAdapter](http://developer.android.com/reference/android/widget/SpinnerAdapter.html) interface).

Adapter

SpinnerAdapter

ListAdapter

BaseAdapter

ArrayAdapter

SimpleCursorAdapter etc.

**Note:**- This is an advance concept as for as Adapter is concerned, so we’ll talk later. It is used when app has complex Data source because By default, ArrayAdapter assumes that single row is represented by a TextView only.

**Note:-** It is predefined implementing class of ListAdapter interface and [SpinnerAdapter](http://developer.android.com/reference/android/widget/SpinnerAdapter.html) interface provides common functionalities of ListAdapter and SpinnerAdapter.

**Note:** Here ListAdapter and SpinnerAdapter (extends Adapter interface) are interfaces in android has multiple implementing classes of ListAdapter interface some of them as follows:-

* ArrayAdapter
* SimpleCursorAdapter
* BaseAdapter

**What is Cursor?**

It is an object of Cursor class in android that represents data which is taken out from database by Select Query.

**Adapter View**

This is an abstract class which has been defined in android.widget package and object its any implementing class represents a Parent view whose children are determined by adapter.

Like- ListView, GridView, Spinner, Gallery are commonly used subclass of AdapterView.

\*\* Here Adapter View is responsible for taking Child View object from adapter and controlling how to child views should display to the user.

Pintu

Sachin

Vimal

Monu

Sonu

View

Data Source

Sonu, Monu, Vimal, Sachin, Mahesh, Suresh, Chintu, Pintu

Array/Database/file etc..

Adapter View

ListView

Base Adapter

Simple Cursor Adapter

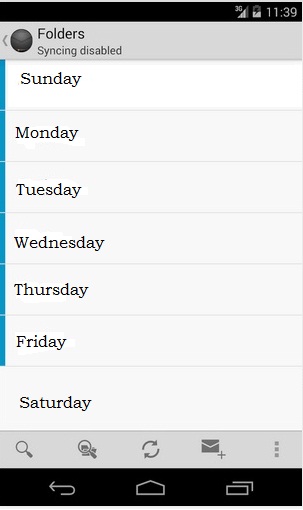
Array Adapter

GridView

Gallery

Spinner

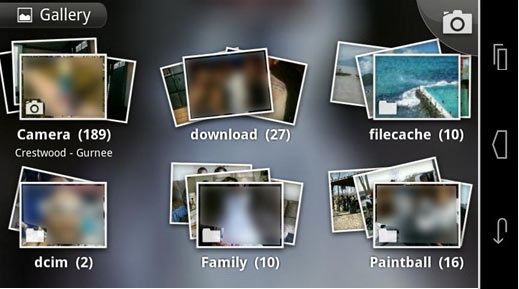
**Types of AdapterView :- (implementing classes of AdapterView)**

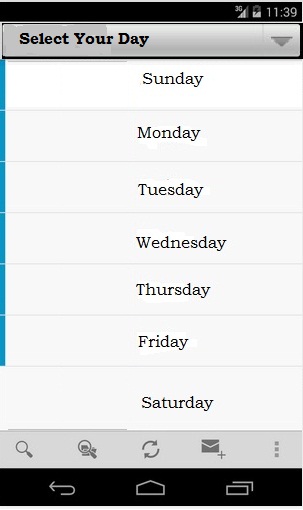
1. **ListView:-** It displays vertically scrolling list of child View objects with which user can interact.

****

1. **GridView:-** It displays each child View objects along with row and columns with which user can interact.
2. **Gallery:-** It displays a horizontal scrolling list of View object with which user can interact.

***Note:-*** From android 4.1 it has deprecated that mean it is not using any more.



1. **Spinner:**- It displays list of items similar to dropdown menu.

**ListView:-**

It is class defined in android.widget package. An object of this class created by OS from XML file (inflation) and object of this class provides facility to display multiple child views in vertical scrolling list with which user can interact.

**Que.- Why List View?**

* To display lots of information to browse easily.
* As user reaches the end of screen, more results are generated or displayed.
* When the users click on any row some action can be performed.
* Avoid the need to implement paging like—In Web pages.

**Working of List View:-**

Here, A simple architecture to how List View works.

Pintu

Sachin

Vimal

Monu

Sonu

Data Source

Sonu, Monu, Vimal, Sachin, Mahesh, Suresh, Chintu, Pintu

View

ListView

Array/Database

ImageView

TextView

**Steps to create Android App with ListView**:-

1. Define the data source like Array, database etc.

**Note**:- we can define array either programmatically or declaratively.

**Programmatically** :-

String contactName[]={“sonu”,”monu”,…….};

**Declaratively** :- In XML file.

Like For string array we have to define in string.xml as follows :

<string-array name=”contactsName”>

<item>sonu</item>

<item>monu</item>

.

.

</string-array>

1. Create an Object of the Adapter like ArrayAdapter, SimpleCursorAdapter or Custom Adapter etc.
2. Define <ListView> element in layout XML file.
3. Responding to the user interaction (Event Handling)

When any item has been clicked by the user. ListView receives OnItemClick event.

Listener Name- OnItemClickListener

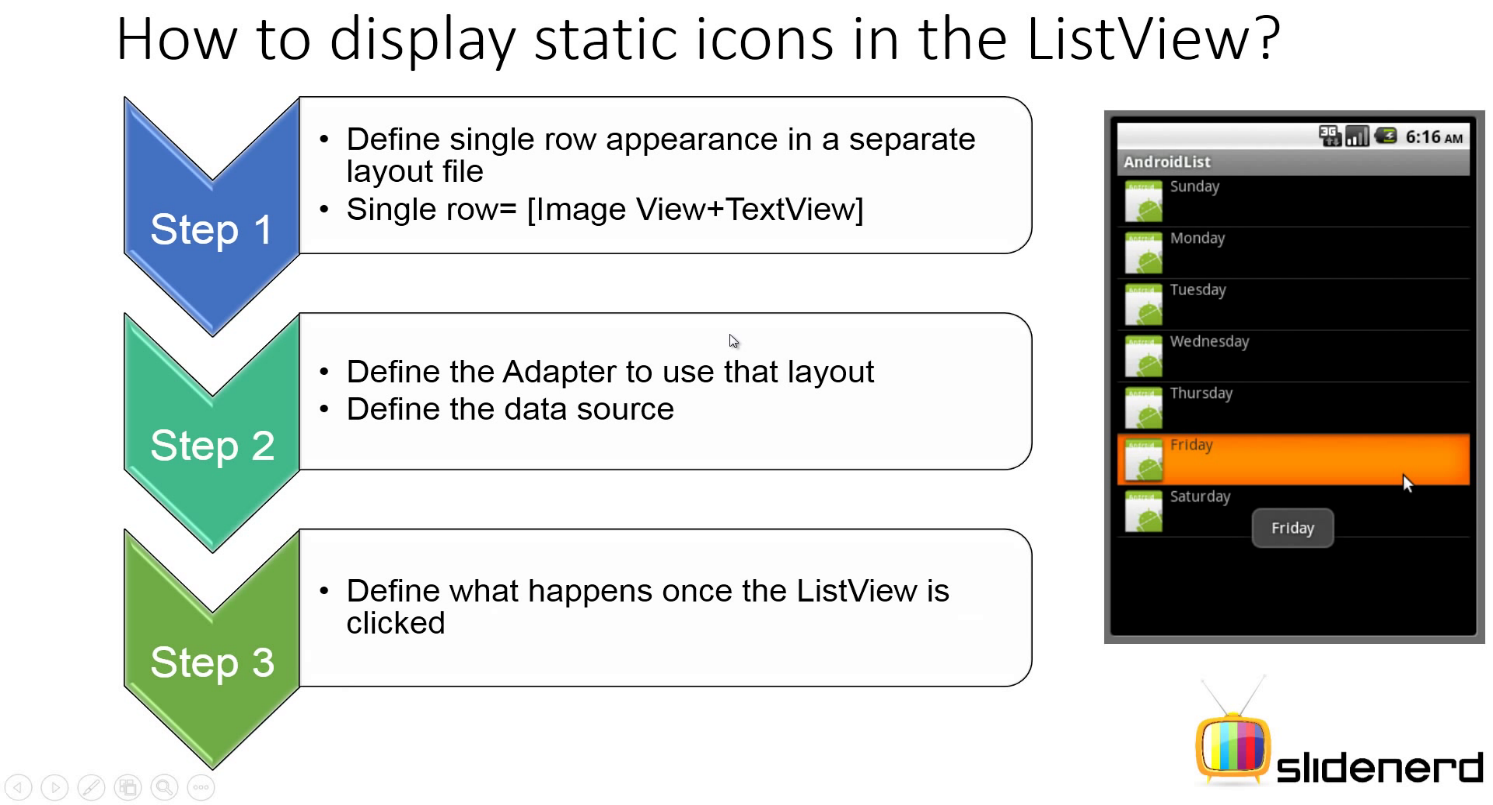
Event Handler – **public** **void** onItemClick(AdapterView adapterView, **View view, int position, long id**);

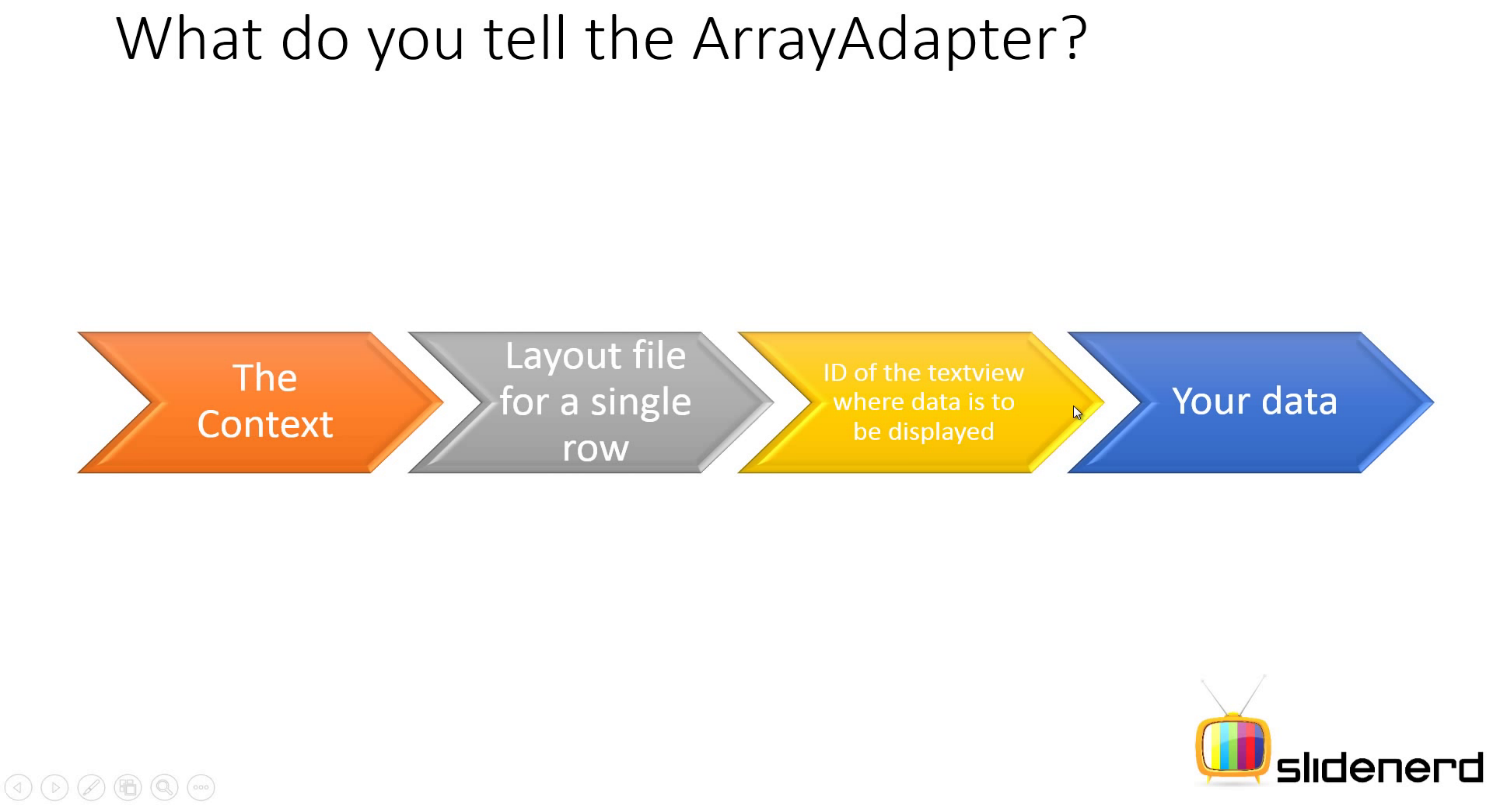
Where- adapterView represents who contains all rows. Eg:- ListView

view represents clicked row. Eg:- TextView

position represents clicked row number. Eg :- 0 (starting)

id represents id of clicked row. But remember it doesn’t matter until we are reading data from database, because it contains some useful information about clicked row.





**GridView:-**

It is class defined in android.widget package. An object of this class created by OS from XML file (inflation) and object of this class provides facility to display multiple child views vertical & horizontal scrolling list (in row-column) with which user can interact.

**Que.- Why GridView?**

* To display lots of information to browse easily.
* As user reaches the end of screen, more results are generated or displayed.
* When the users click on row some action can be performed.

**Working of GridView:-**

Here, A simple architecture to how GridView works.

Sonu

Monu

Data Source

Sonu, Monu, Vimal, Sachin, Mahesh, Suresh, Chintu, Pintu

Sachin

Vimal

Chintu

Pintu

Pappu

golu

View

GridView

Array/Database

TextView+ImageView etc. (complex View)

ImageView

TextView

**Steps to create Android App with GridView**:-

1. Define the data source like Array, database etc.
2. Define the Adapter like ArrayAdapter, SimpleCursorAdapter etc.
3. Define <GridView> element in layout XML file.
4. Responding to the user interaction (Event Handling)

When any item has been clicked by the user. GridView receives OnItemClick event.

Listener Name- OnItemClickListener

Event Handler – **public** **void** onItemClick(AdapterView adapterView, **View view, int position, long id**);

Where- adapterView represents who contains all rows. Eg:- GridView

vew represents clicked row. Eg:- TextView

position represents clicked row number. Eg :- 0 (starting)

id represents id od clicked row. But remember it doesn’t matter until we are reading data from database, because it contains some useful information about clicked row.

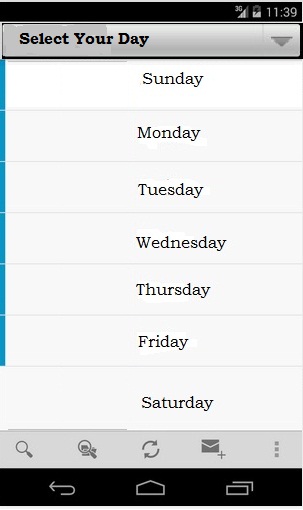
**Sppiner** :- It is class defined in android.widget package. An object of this class created by OS from XML file (inflation) and object of this class provides facility to display multiple data items in vertical scrolling list of items in drop down list after clicking by the user on this widget.

**Que.- Why Spinner?**

* It provides the quick way to select one value from set of values of drop down list.
* By default, it shows only its currently selected item.

In order to create spinner, we have to define <Spinner > element in layout XML file.

**Working of Sppiner:-**

****Here, A simple architecture to how Spinner works.

Data Source

Sonu, Monu, Vimal, Sachin, Mahesh, Suresh, Chintu, Pintu

View

Array/Database

Spinner

TextView+ImageView etc. (complex View)

ImageView

TextView

**Steps to create Android App with Spinner**:-

1. Define the <spinner> element in layout Xml file.
2. Define the data source like Array, database etc.
3. Define the Adapter like ArrayAdapter, SimpleCursorAdapter etc.
4. Responding to the user interaction (Event Handling)

When any item has been selected by the user, Spinner receives OnItemSelected event.

Listener Name- OnItemSelectedListener

Event Handler – 1-**public** **void** onItemSelected(AdapterView adapterView, **View view, int position, long id**);

**2-public** **void** onNothingSelected(AdapterView<?> arg0)

Where- adapterView represents who contains all rows. Eg:- Spinner

vew represents clicked row. Eg:- TextView

position represents clicked row number. Eg :- 0 (starting)

id represents id of clicked row. But remember it doesn’t matter until we are reading data from database, because it contains some useful information about clicked row.

onNothingSelected(AdapterView<?> arg0):- Callback method to be invoked by OS when the selection disappears from this spinner. The selection can disappear for instance when touch is activated or when the adapter becomes empty.

This method may be used so that you can set which item will be selected given that the previous item is no longer available. This is instead of letting the spinner automatically select the next item in the list.

**AutoCompleteTextView :-** Thisclass is defined in android.widget package. An object of this class represents an Editable Text field which provides facility of auto suggestions when the user is typing in it.

The list of suggestions is displayed in a drop down list, the user can choose an item to replace the content of the text field.

NOTE:- All the suggestions that come on typing are reserved strings, stored in String Array.

In order to use AutoCompleteTextView ,we have to first create < AutoCompleteTextView > in layout XML file as follows :-

**<AutoCompleteTextView**

      android:id="@+id/autoCompleteTextView1"

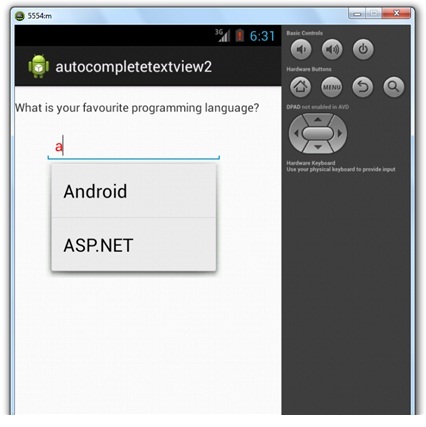
        android:layout\_width="wrap\_content"

         android:layout\_height="wrap\_content"

         android:ems="10"

    android:text=""**>**

**</AutoCompleteTextView>**

1. **Responding to the user interaction** (Event Handling)

When any item has been clicked by the user. AutoCompleteTextView receives OnItemClick event.

Listener Name- OnItemClickListener

Event Handler – **public** **void** onItemClick(AdapterView adapterView, **View view, int position, long id**);

Where- adapterView represents who contains all rows. Eg:- GridView

vew represents clicked row. Eg:- TextView

position represents clicked row number. Eg :- 0 (starting)

id represents id od clicked row. But remember it doesn’t matter until we are reading data from database, because it contains some useful information about clicked row.

**Core Components of Android Apps :-**

These are the essential building block of an android apps. These components are loosely coupled by AndroidManifest.xml file, that describe each component of an application and how do they interact to android OS.

There are four different types of app components. Each type serves a distinct purpose and has a distinct lifecycle that defines how the component is created and destroyed.

Here are the four types of app components:

1. Activity
2. Service
3. BroadCastReciever
4. ContentProvider



Note:- An Android App is built by using either all or few above core components according to the need of app development.