UNIT I

CHAPTER 1

Introduction to Android

Syllabus

What is Android? Obtaining the required tools, creating first android app, understanding the components of screen, adapting display orientation, action bar, Activities and Intents, Activity Lifecycle and Saving State, Basic Views: TextView, Button, ImageButton, EditText, CheckBox, ToggleButton, RadioButton, and RadioGroup Views, ProgressBar View, AutoCompleteTextView, TimePicker View, DatePicker View, ListView View, Spinner View

Syllabus Topic : What is Android?

1.1 Android

- Android is a mobile platform consisting of operating system, middleware and key applications,developed by Google, for smartphones and other mobile devices (such as tablets).
- The operating system in Android devices is the **Linux kernel**.
- The Linux kernel manages the following responsibilities :
  1. Memory Management
  2. Resource Management
  3. Driver Management
  4. Power Management
- Android applications can be developed using the following languages :
  1. Java(using SDK support)
  2. C/C++(using NDK support)
  3. .NET(using mono Android support)
- Initial versions of Android used to support developing apps using C/C++ and .NET. Now the latest versions allow only Java.
- It can run on different devices from different manufacturers. Android includes a software development kit for writing original code and assembling software modules to create apps for Android users. It also provides a marketplace to distribute apps.
- Altogether, Android represents an ecosystem for mobile apps.

Why develop apps for Android?

Apps are developed for a variety of reasons: addressing business requirements, building new services, creating new businesses, and providing games and other types of content for users.

Developers choose to develop for Android in order to reach the majority of mobile device users.

☞ **Most popular platform for mobile apps**

- Millions of mobile devices use android in more than 190 countries around the world. It has the largest installed base of any mobile platform.
- Every day millions of users start using Android devices for the first time and hence there is an increasing requirement of games, apps, and other digital content.

☞ **Best experience for app users**

- Android provides a touch-screen user interface (UI) for interacting with apps. Android's user interface is mainly based on direct manipulation, using touch gestures such as swiping, tapping and pinching to manipulate on-screen objects. In addition to the keyboard, there's a customizable virtual keyboard for text input.
- Android can also support game controllers and full-size physical keyboards connected by Bluetooth or USB.

1.1.1 Android Features

- Open Source
- Application framework provides infrastructure for application developer.
- DVM(Dalvik Virtual Machine) is optimized for mobile devices to work on low power, low memory and low RAM.
- OpenGLES(Open Graphics Library for Embedded Systems) library is used for displaying graphics
- Supports GPS(Global Positioning System) and different media formats
- SQLiteDb is used to maintain structured data in Android

Android Studio provides rich development environment.

Android is a product of OHA(Open Handset Alliance) which is led by Google.

- Android plays a key role in IOT(Internet of Things)

1.1.2 Android Versions

- Google provides major incremental upgrades to the Android operating system every six to nine months, using confectionery-themed names. The latest major release is Android 8.0 "Oreo"(code named **Android** during development.)

Table 1.1.1 : Android Versions

Code name	Version number	Initial release date	API level
N/A	1.0	23 September 2008	1
N/A	1.1	9 February 2009	2
Cupcake 	1.5	27 April 2009	3

Code name	Version number	Initial release date	API level
Donut 	1.6	15 September 2009	4
Éclair 	2.0 – 2.1	26 October 2009	5-7
Froyo 	2.2 – 2.2.3	20 May 2010	8
Gingerbread 	2.3 – 2.3.7	6 December 2010	9–10
Honeycomb 	3.0 – 3.2.6	22 February 2011	11–13
Ice Cream Sandwich 	4.0 – 4.0.4	18 October 2011	14–15

Code name	Version number	Initial release date	API level
Jelly Bean 	4.1 – 4.3.1	9 July 2012	16–18
KitKat 	4.4 – 4.4.4	31 October 2013	19–20
Lollipop 	5.0 – 5.1.1	12 November 2014	21–22
Marshmallow 	6.0 – 6.0.1	5 October 2015	23
Nougat 	7.0	22 August 2016	24
Oreo 	8.0	August 21, 2017	26

1.1.3 Dalvik Virtual Machine (DVM)

- The modern JVM is high performance and provides excellent memory management. However, using it for handheld devices is not feasible as it needs to be optimized for low-powered handheld devices like mobile phones, tablets etc.
- The **Dalvik Virtual Machine (DVM)** is an android virtual machine that has been optimized for mobile devices. It optimizes the virtual machine for *performance, battery life* and *memory*. However DVM has been discontinued by Google and replaced by Android Runtime (ART)
- The Dalvik VM was written by Dan Bornstein who named it after a village in Iceland with the same name.
- Android programs are commonly written in Java and compiled to bytecode for the Java virtual machine. They are then translated to Dalvik bytecode and stored in **.dex (Dalvik EXecutable)**. The Dex compiler then converts these class files into the .dex files that run on the Dalvik VM. Multiple class files are converted into one dex file.
- Fig. 1.1.1 is a diagrammatic representation of the compiling and packaging process from the source file.

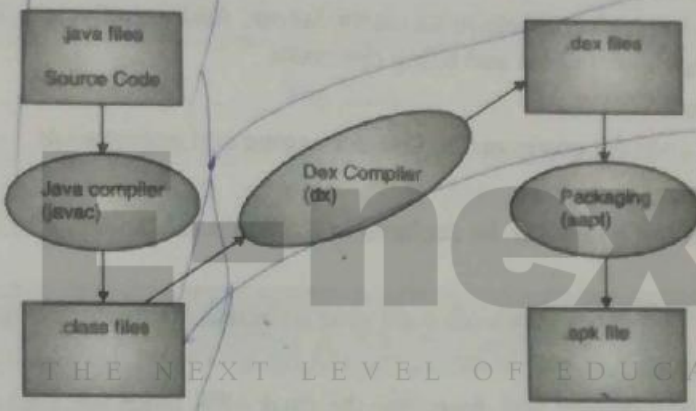


Fig. 1.1.1

1.1.4 Android Components

- There are four basic(core) Android components.

➜ **1. Activities**

A single screen in the application, with UI components, that the user can interact with is called an activity.

➜ **2. Services**

- A long running background process, without any user interaction, is called a Service.
- Eg. 1. Alarm – it continuously keeps track of the time in the background and alerts once the time is reached.
- Eg. 2. Whenever WIFI network is available our phone automatically detects it and alerts us regarding the same. This is also a service.
- Eg. 3. Media/FM player.

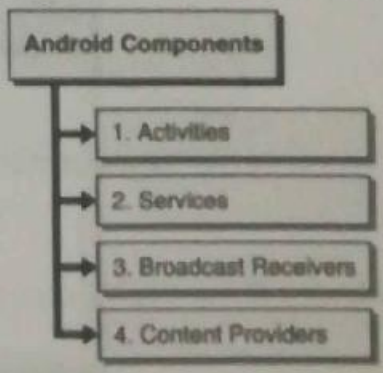


Fig. C1.1 : Android components

➜ **3. Broadcast Receivers**

Broadcast Receivers are registered for system announcements. Eg. Headset plugin, charger connected/disconnected, user is making/receiving a call etc.

➜ **4. Content Providers**

- Content Providers is used to share data between multiple applications. In Android, due to its security feature, we cannot access the data of one application in another. This is possible only if the applications make the data available to each other with content providers.
- Eg. Data being shared between WhatsApp and Contacts through content providers.

Syllabus Topic : Obtaining the Required Tools

1.2 Obtaining the Required Tools

For each activity the Android Studio can be used to do the following:

☞ **Create the layout**

Create the layout by placing UI elements on the layout. Additionally, use XML (Extensible Markup Language) to assign menu items and string resources.

☞ **Write the Java code**

Create the source code for components. Use debugging and testing tools.

☞ **Register the activity**

To register the activity it needs to be declared in the AndroidManifest.xml file.

Define the build

Use the default build configuration or create custom builds for different versions of the app.

☞ **Publishing the app**

To publish the newly created app, assemble the final APK (package file) and distribute it through channels such as the Google Play.

1.2.1 Create Android Application

- The first step is to create a simple Android Application using Android studio. On clicking on the Android studio icon, it will show screen as shown in Fig. 1.2.1.

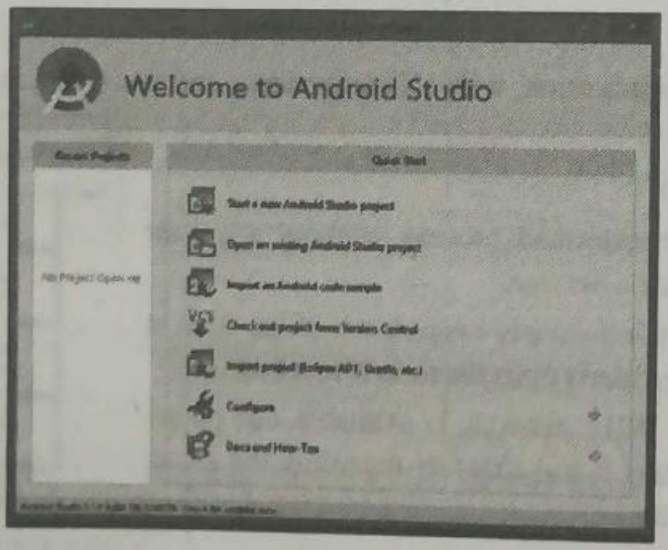


Fig. 1.2.1

- Application development can be started by calling 'Start a new Android Studio Project'.
- In a new installation frame should ask Application name, package information and location of the project.

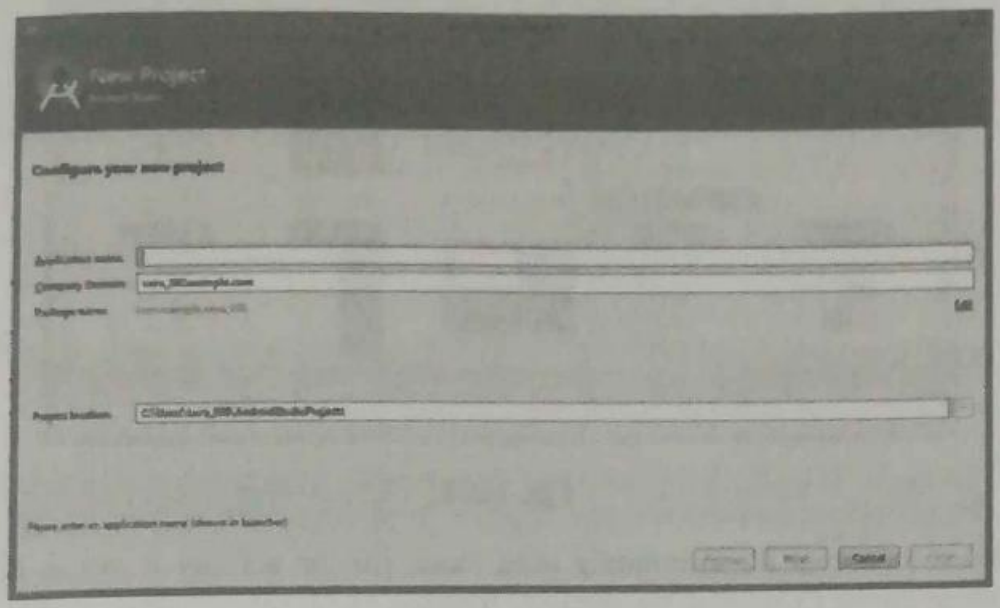


Fig. 1.2.2

- After entering application name, it asks to select the form factors that the application runs on. Here we need to specify Minimum SDK, in your tutorial, here we have declared as API23: Android 6.0(Mashmallow).

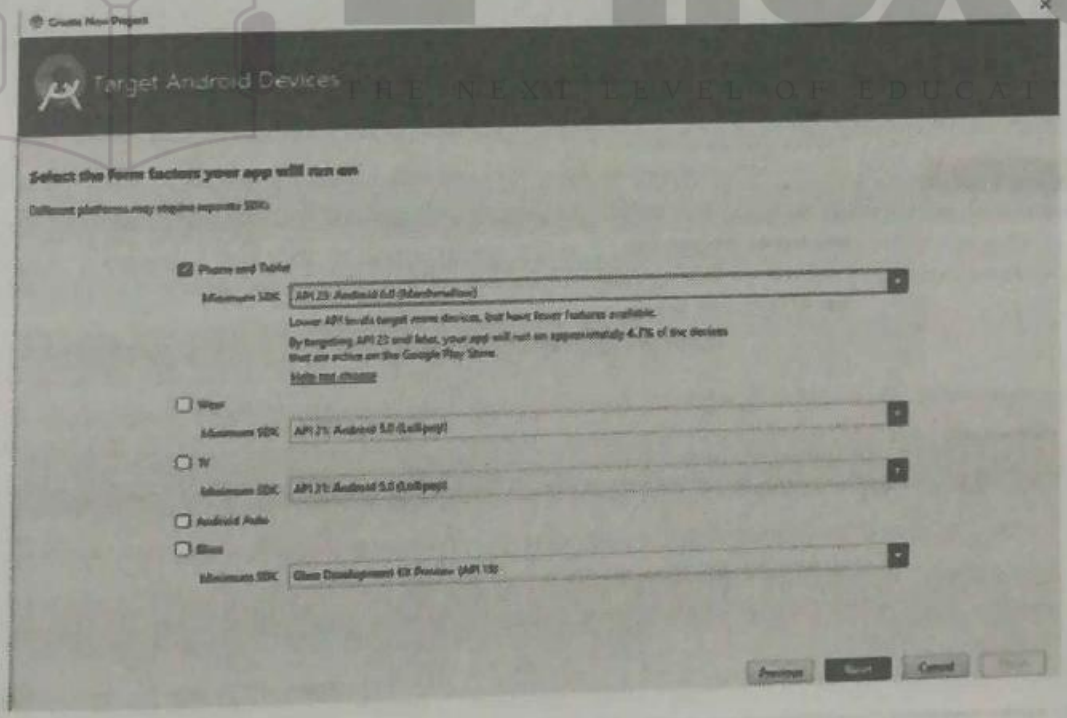


Fig. 1.2.3

- The next level of installation should contain selecting the activity to mobile, it specifies the default layout for Applications.

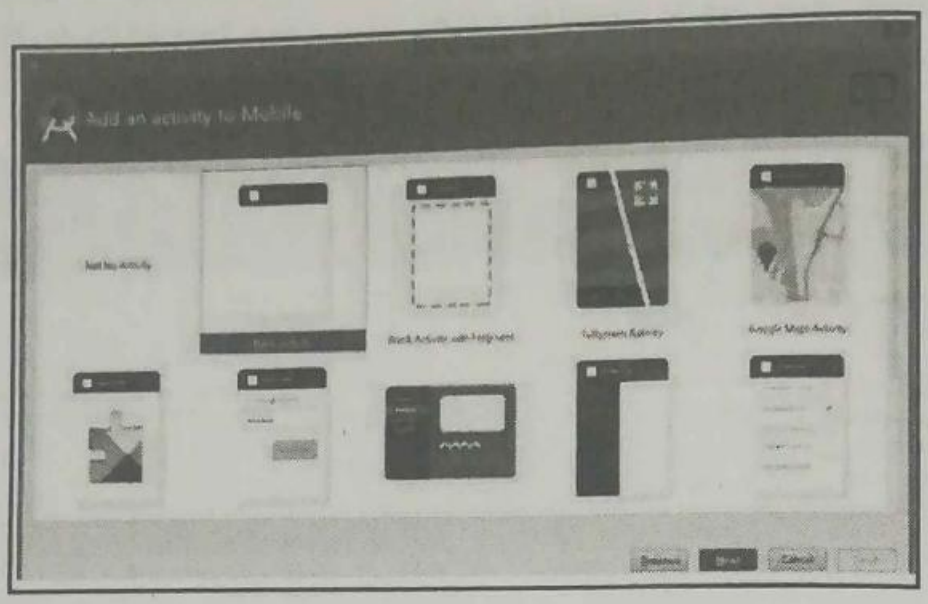


Fig. 1.2.4

We can choose to accept the commonly used name for the activity (such as MainActivity) or change the name on the Customize the Activity screen.

Also, if we use the Empty Activity template, be sure to check the following if they are not already checked.

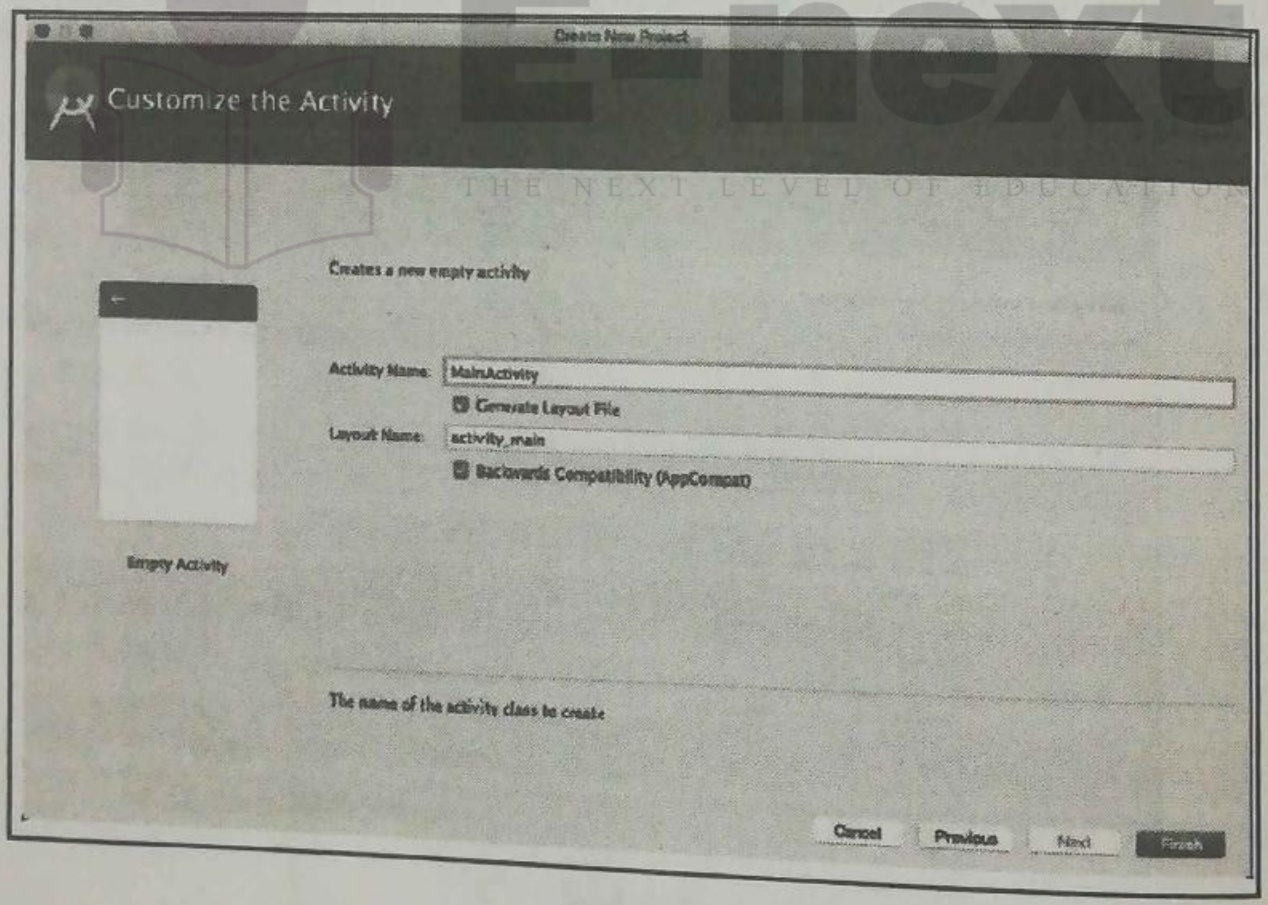


Fig. 1.2.5

At the final stage it opens development tool to write the application code.

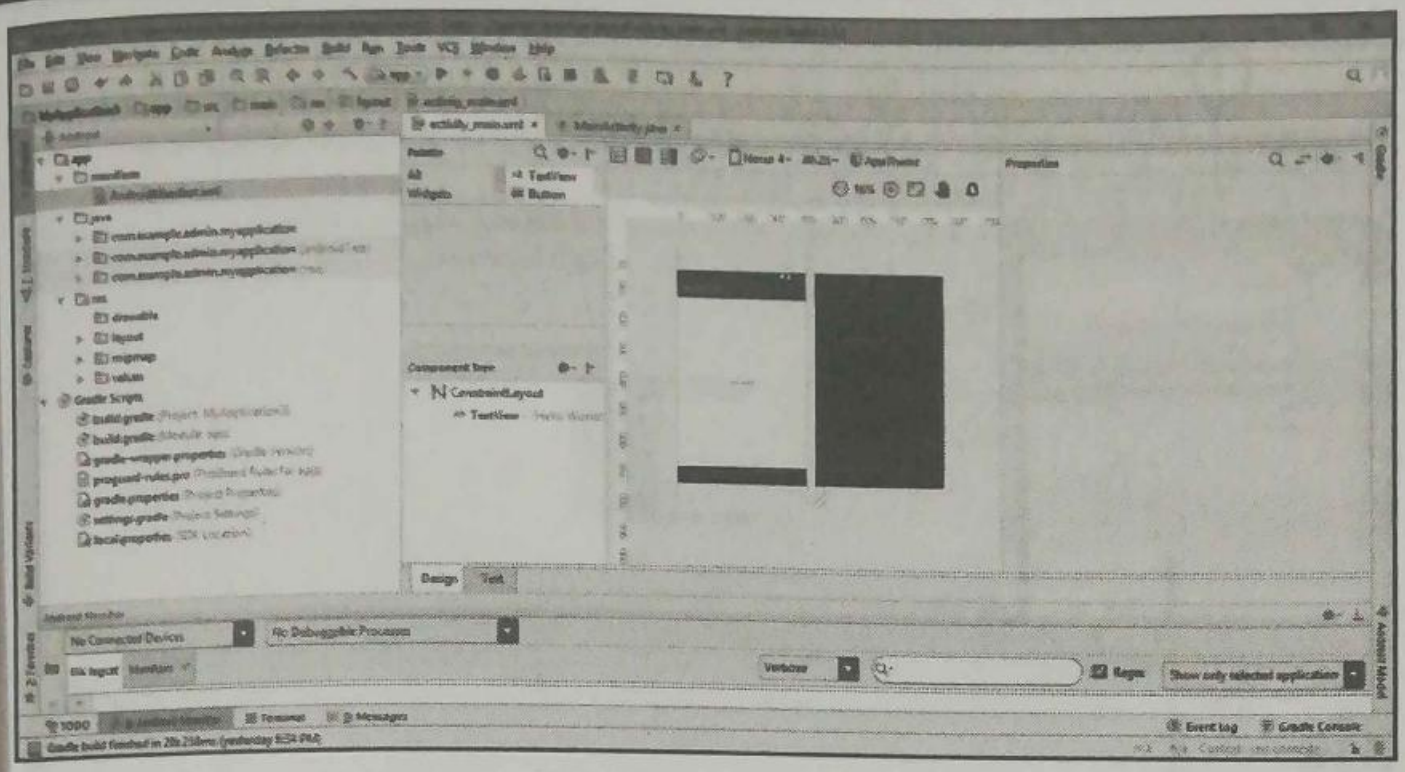


Fig. 1.2.6

- Android Studio creates a folder for the newly created project in the **AndroidStudioProjects** folder on the computer

☞ **Using Android Studio**

- Android Studio provides tools for the testing, and publishing phases of the development process, and a unified development environment for creating apps for all Android devices.
- The development environment includes code templates with sample code for common app features, extensive testing tools and frameworks, and a flexible build system.

Syllabus Topic : Understanding the Components of Screen

1.3 Understanding the Components of Screen

☞ **Android Studio window panes**

- The Android Studio main window is made up of several logical areas, or panes, as shown in the Fig. 1.3.1.

In the Fig. 1.3.1.

1. The **Toolbar:** The toolbar carries out a wide range of actions, including running the Android app and launching Android tools.
2. The **Navigation Bar:** The navigation bar allows navigation through the project and open files for editing. It provides a more compact view of the project structure.
3. The **Editor Pane** : This pane shows the contents of a selected file in the project. For example, after selecting a layout (as shown in the figure), this pane shows the layout editor with tools to edit the layout. After selecting a Java code file, this pane shows the code with tools for editing the code.
4. The **Status Bar:** The status bar displays the status of the project and Android Studio itself, as well as any warnings or messages. We can watch the build progress in the status bar.

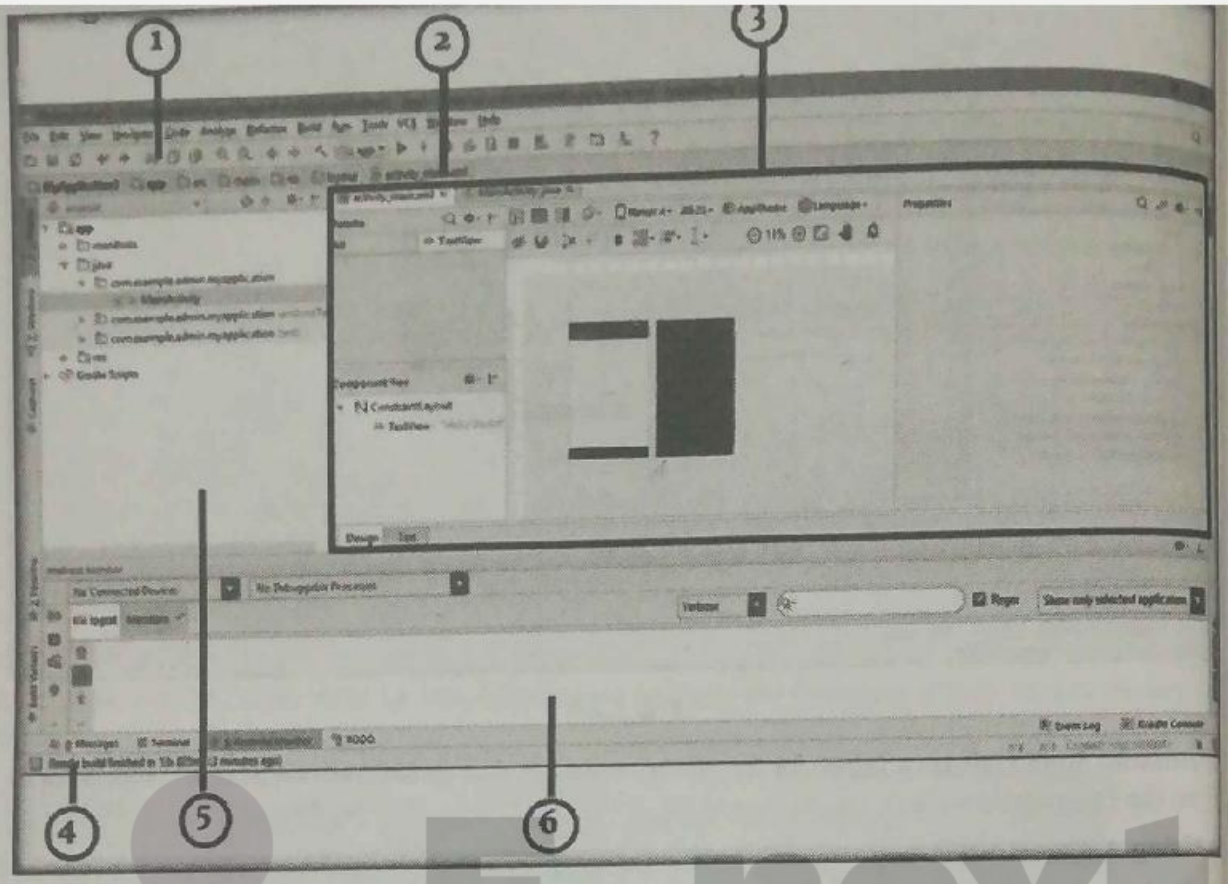


Fig. 1.3.1

5. The **Project Pane:** The project pane shows the project files and project hierarchy.
6. The **Monitor Pane:** The monitor pane offers access to the TODO list for managing tasks, th Android Monitor for monitoring app execution (shown in the figure), the logcat for viewing lo messages, and the Terminal application for performing Terminal activities

Note : We can organize the main window to give ourselves more screen space by hiding or moving panes. We can also use keyboard shortcuts to access most features.

☞ **Exploring a project**

- Each project in Android Studio contains the AndroidManifest.xml file, component source-cod files, and associatedresource files.
- By default, Android Studio organizes project files based on the file type, and displays them within theProject: Android view in the left tool pane.
- The view provides quick access to the project's key files.
- To switch back to this view from another view, click the vertical Project tab in the far left colum of the Project pane, and choose Android from the pop-up menu at the top of the Project pane, a shown in the Fig. 1.3.2.

In the Fig. 1.3.

Sr. No.	
1.	Ja
2.	re
3.	re
4.	re
5.	A
6.	B

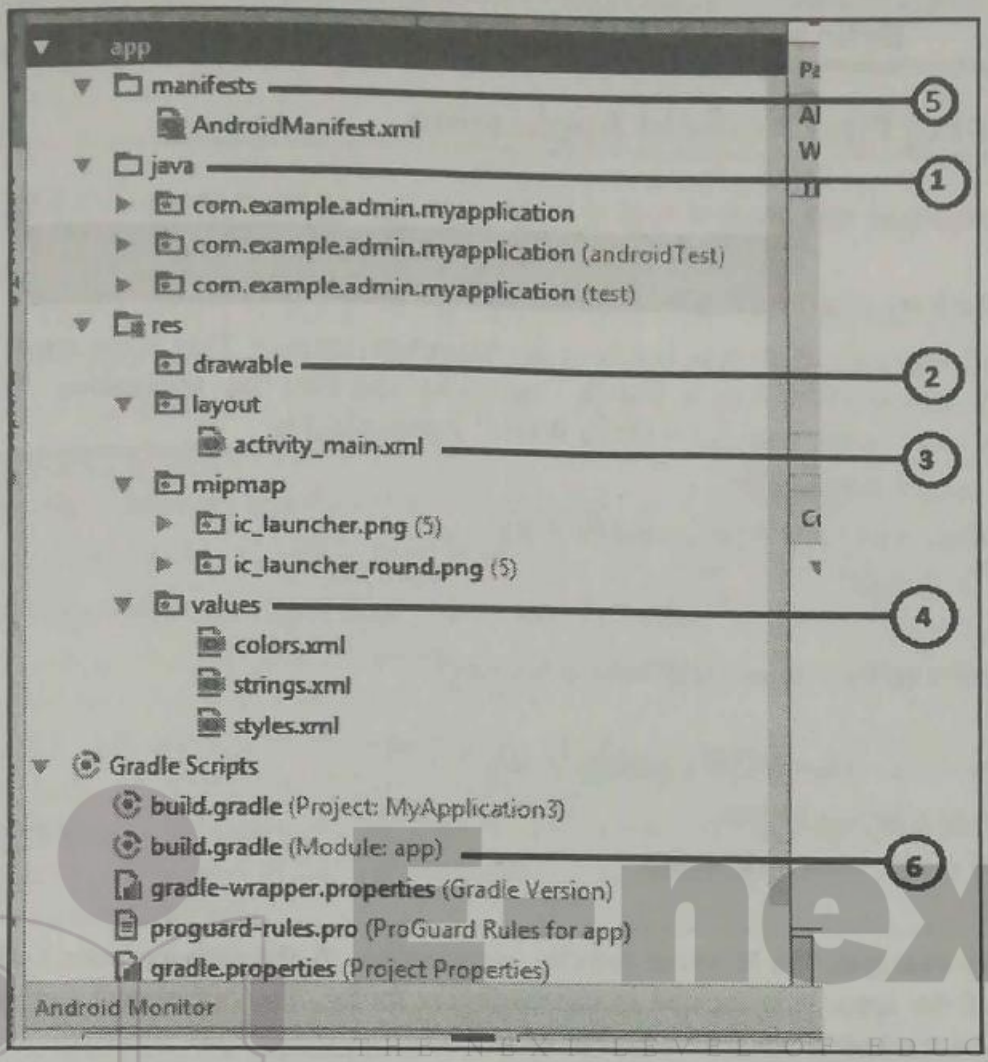


Fig. 1.3.2

In the Fig. 1.3.2.

Sr. No.	Folder, File	Description
1.	Java	This folder contains the .java source files for the project. It also includes the MainActivity.java source file which has an activity class that runs when the app is launched using the app icon.
2.	res/drawable-hdpi	The drawable objects that are designed for high-density screens are placed in this folder.
3.	res/layout	The files that define the app's user interface are placed in this folder.
4.	res/values	This directory contains a collection of XML files of resources, viz. strings and colours definitions.
5.	AndroidManifest.xml	This file describes the fundamental characteristics of the app and also defines each of its components.
6.	Build.gradle	This is an auto generated file which contains compileSdkVersion, buildToolsVersion, applicationId, minSdkVersion, targetSdkVersion, versionCode and versionName

1.4 Creating First Android Application

- In order to create an app, we first need to understand the important application files. These files ar as follows.

☞ **The Main Activity File**

The main activity code of an app, is a Java file MainActivity.java. This is the actual application fi which ultimately gets converted to a Dalvik executable and runs the application. The default cod generated, by the application wizard, for Hello World! Application is:

```
package com.example.helloworld;
import android.support.v7.app.AppCompatActivity;
import android.as.Bundle;

public class MainActivity extends AppCompatActivity{
@Override
protected void onCreate(BundlesavedlnstanceState){
super.onCreate(savedlnstaneeState);
setContentView(R.layout.activity_main);
} }
```

- In the above example, the R.layout.activity_main, refers to the activity_main.xml file(containi the layout of the app). It is located in the res/layout folder. The onCreate() method is one of t methods that are invoked when an activity is loaded.

☞ **The Manifest File**

- We must declare all the components of the app in the Androidmanifest.xml file. This file can found at the root of the application project directory.
- This file works as an interface between Android OS and the application. If we do not declare t components in this file, then it will not be considered by the OS. A default manifest file is follows.

```
<?xml version="1.0" encoding="utf-8"?>
<manifestxmlns:android="http://schemas.android.com/apk/res/android"package="com.example.tutor
alspoint7.myapplicatlon">
<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">
<intent-filter>
<action android:name="android.intent.action.MAIN"/>
<category android:name="android.intent.category.LAUNCHER"/>
```

```
</intent-filter>
</activity>
</application>
</manifest>
```

- The <application>...</application> tags enclose all the components that are related to the application.
  - Attribute android:icon points to the application icon under res/drawable-hdpi folder. Here the application uses the image named ic_launcher.png located in the drawable folders.
- The <activity> tag is used to specify an activity. Each activity of the app will have a separate <activity> tag in the manifest file.
  - android:name attribute specifies the fully qualified class name of the Activity subclass
  - android:label attribute specifies the string that is to be used as the label for the activity.
  - Multiple activities can be specified using <activity> tags.
- In order to indicate that this activity serves as the entry point for the application, the action for the intent filter is named, android.intent.action.MAIN.
- The category for the intent-filter is named android.intent.category.LAUNCHER to indicate that the application can be launched from the device's launcher icon.
- The @string refers to the string.xml file (see below). Therefore, @string/app name refers to the app name string defined in the string.xml file, (here "HelloWorld"). Similarly, other strings of the application are specified here.
- Other tags which we use in the manifest file to specify different Android application components are :
  - <activity>elements for activities
  - <service> elements for services
  - <receiver> elements for broadcast receivers
  - <provider> elements for content providers

The Strings File

- The text that any application uses is specified in the string.xml file. It is located in the res/values folder.
- For example, the names of labels, buttons, default text, and similar types of strings. This file is responsible for their textual content. For example, a default strings file will look like the following.

```
<resources>
<string name="app_name">HelloWorld</string>
<string name="hello_world">Helloworld!</strin1>
<string name="menu_settings">Settings</string>
<string name="title_activity_main">MainActivity</strin1>
</resources>
```

The Layout File

- The layout of the app is specifies in the activity_main.xml file. It is available in res/layout directory. This file is accessed by the application when building its interface.

- Modifying this file changes the layout of the applications. For the "Hello World!" application, th file will have following content related to default layout.

```
<Relativelayout
xmlns:android="http//schemas android.com/apk/res/android"xmlns:tools="http//schemas and
roid.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent">

<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerHorizontal="true"
android:layout_centerVertical="true"
android:padding="@dimen/padding_medium"
android:text="@string/hello_world"
tools:context=".MainActivity"/>
</Relativelayout>
```

☞ **Running the Application**

- In order to run the Hello World! Application, create the AVD (Android Virtual Device) while doing environment set-up. To run the app from Android studio, open one of the project's activity files and click
- Run icon from the tool bar. Android Studio installs the app on the AVD. Then it starts the app and if there are no errors, it will display the app on the Emulator window (Fig. 1.5.1).
- Congratulations!!! We have developed our first Android Application.

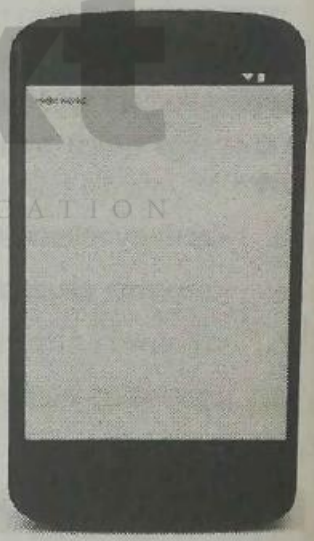


Fig. 1.4.1

☞ **The .apk file**

- .apk stands for **A**ndroid **P**ackage **K**it. It is a package file format.
- It is used for installation and distribution of mobile apps by the Android operating system.
- To install any application, we need to install the file with extension.apk
- The .apk file consists of the following :
  .dex(java bytecode generated by DVM), Manifest.xml, res, libs and assets.

1.4.1 Android R.java File

- **The Android R.java** is *an auto-generated file by aapt* (Android Asset Packaging Tool). It contains resource IDs for all the resources of res/ directory.
- Ids for the components created in activity_main.xml are automatically created in R.java file.
- This id can then be used in the activity source file to perform any action on the component.

Note : If you delete R.jar file, android creates it automatically.

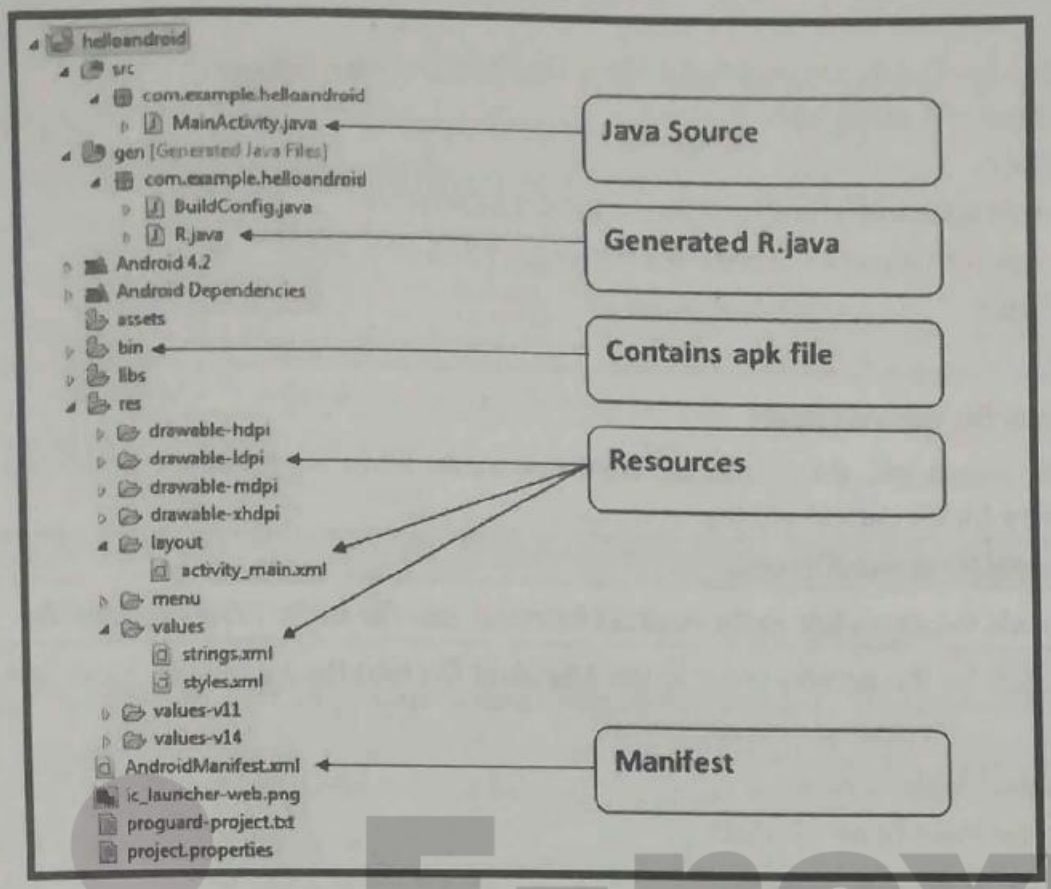


Fig. 1.4.2

Syllabus Topic : Adapting Display Orientation

1.5 Adapting Display Orientation

- Android provides two types of screen/display orientation:
  1. Landscape
  2. Portrait
- Here we will discuss regarding Android Screen Orientation Change (Screen Rotation) with help of an example.

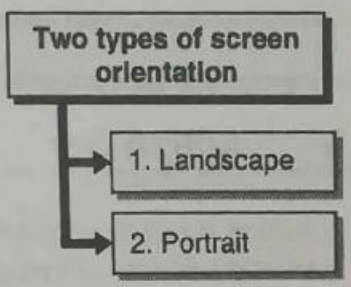


Fig. C1.2 : Types of screen orientation

☞ **Lock screen orientation change in Android**

If we want to lock the screen orientation change of any screen (activity) of the android application then add below line in the project's AndroidManifest.xml file.

➜ **1. Lock for landscape mode**

- The screen will always display in Landscape mode, when we rotate the device, no changes will apply for the current activity.

```
android:screenOrientation="landscape"
```

- So add the above line in the AndroidManifest.xml file in the following manner.

Note : Search for the activity entry in the Manifest file and then add the above line like below.

```
<activity
android:name=".MainActivity"android:screenOrientation='landscape"
android:label="@string/app_name">
<intent-filter>
<action android:name='android.intentaction.MAIN"/>
<category android:name='android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
```

➜ **2. Lock for portrait mode**

- The screen will always display in portrait mode, when we rotate the device, no changes w... apply for the current activity.

```
android:screenOrientation="portrait"
```

- So add the above line in the Android Manifest.xml file in the following manner.

Note : Search for the activity entry in the Manifest file and then add the above line like below

```
<activity
...droid:name=".MainActivity"
...roid:screenOrientation="portrait"
...droid:label="@string/app_name" >
<intent-filter>
<action android:name="android.intent.action.MAIN"/>
<category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
```


Syllabus Topic : Action Bar

1.6 Action Bar

- Action Bar is a primary toolbar within the activity that may display the activity title, applicatio... level navigation affordances, and other interactive items.
- Beginning with Android 3.0 (API level 11), the action bar appears at the top of an activity window. It is a primary toolbar within the activity that may display the activity title, applicatio... level navigation affordances, and other interactive items.
- The action bar appears at the top of an activity's window when the activity uses th... AppCompat's**AppCompat**theme (or one of its descendant themes). It can also be added to th... action bar by callingrequestFeature(FEATURE_SUPPORT_ACTION_BAR)or by declaring it in... custom theme with thewindowActionBarproperty.

1.6.1 Android Screen Orientation Example

The **screenOrientation** is the attribute of activity element. The orientation of android activity c... be portrait, landscape, sensor, unspecified etc. You need to define it in the AndroidManifest.xml file.

☞ **For example**

```
<activity
  android:name="com.example.screenOrientation.MainActivity"
android:label="@string/app_name" android:screenOrientation="landscape">
```

☞ **Attribute of screen Orientation**

The common values for screenOrientation attribute are as follows:

Value	Description
Unspecified	It is the default value. In such case, system chooses the orientation.
portrait	taller not wider
landscape	wider not taller
sensor	orientation is determined by the device orientation sensor.

1.6.2 Android Landscape Mode Screen Orientation Example

Following will be the content of res/layout/activity_main.xml file

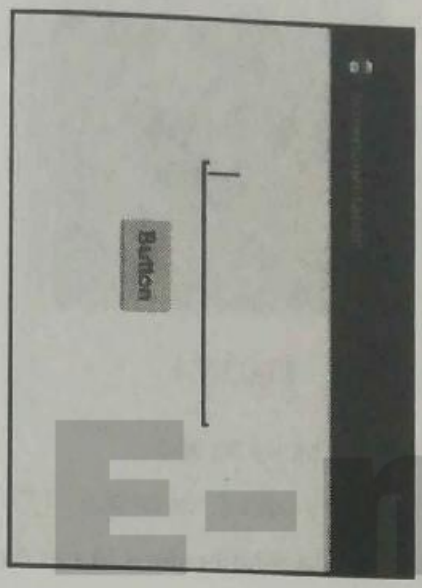
```
<RelativeLayout xmlns:android:class="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:paddingBottom="@dimen/activity_vertical_margin"
android:paddingLeft="@dimen/activity_horizontal_ma rgin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"tools:context=".MainActivity">
<Button
android:id="@+id/button1"
android :layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_margineft="66dp"
android:layout_marginTop="73dp"
android:text="Button"
android:onClick="onClick"/>
<EditText
android:id="@+id/editTextl"
android :layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerHorizontal="true11
android:ems="10"/>
</RelativeLayout>
```

- Following is the content of the modified main activity file **src/MainActivity.java**.

```
package com.example.f;
import android.os.Bundle;
```

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

Output



Syllabus Topic : Activities

1.7 Activities

- An activity broadly maps one to one to a screen in Android application. Each screen in turn is comprised of a certain number of UI elements such as buttons, text, images, maps etc. Since most applications would have multiple screens, therefore an Android application would usually have multiple activities.
  - An Activity is an application component
  - Represents one window, one hierarchy of views
  - Typically fills the screen, but can be embedded in other activity or a appear as floating window
  - Java class, typically one activity in one file
- An activity can represents any of the following:
  - such as ordering groceries, sending email, or getting directions
  - Handles user interactions, such as button clicks, text entry, or login verification
  - Can start other activities in the same or other apps
  - Has a life cycle is created, started, runs, is paused, resumed, stopped, and destroyed

☞ **Examples of activities**

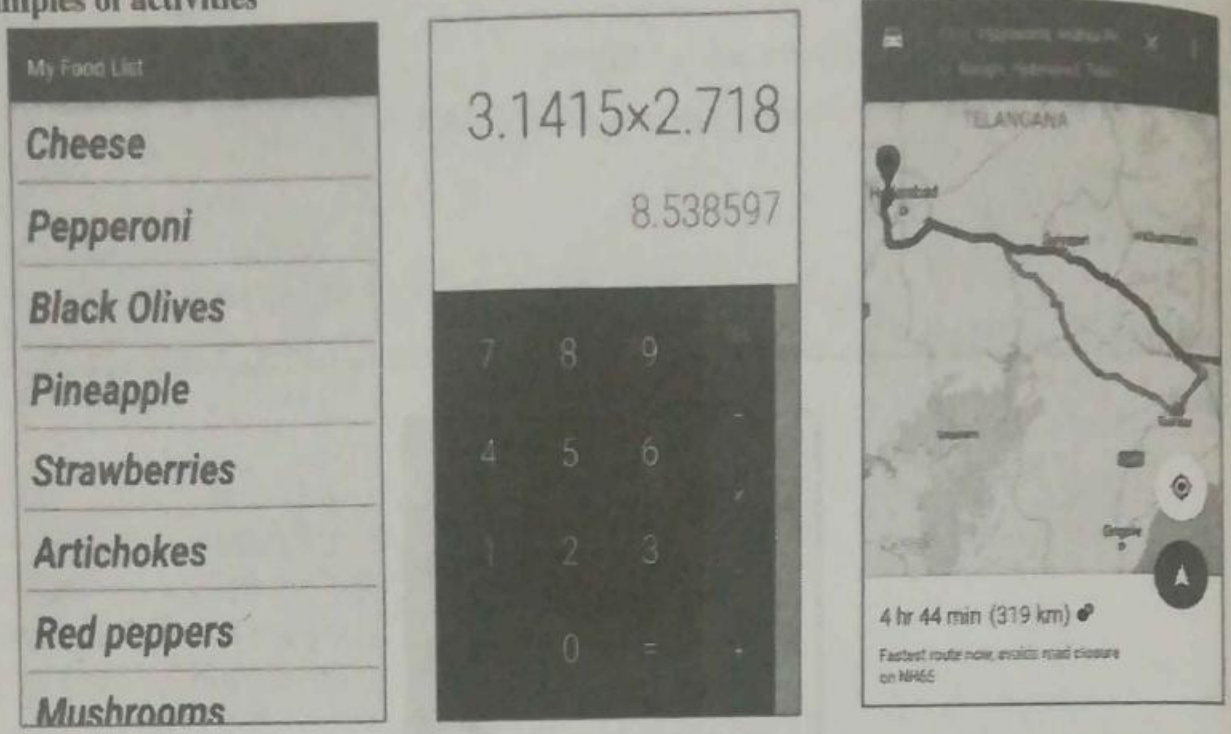


Fig. 1.7.1

- Activities are loosely tied together to make up an app
- First activity that the user sees is typically called "main activity"
- Activities can be organized in parent-child relationships in the Android manifest to aid navigation
- An activity has a UI layout
- The layout of an activity is usually defined in one or more XML files
- Activity "inflates" layout as part of being created

1.7.1 Implementing an Activity

To implement a new activity, we need to do the following:

1. **Define layout in XML**

```
<?xml version="1.0"encoding='utf-8'?>
<RelativeLayout xmlns:android=”http://schemas.android.com/apk/res/android”
android:layout_width="match_parent"
android:layout_height="match_parent">
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text=”Let's Shop for Food!"/>
</RelativeLayout>
```

2. **Define Activity Java class**

extends AppCompatActivity

```
public class MainActivity extends AppCompatActivity{
@Override
protected void onCreate(Bundle savedinstanceState){
super.onCreate(savedinstanceState);
} }
```

3. **Connect Activity with Layout**

Set content view in onCreate()

```
public class MainActivity extends AppCompatActivity{
@Override
protected void onCreate(Bundle savedinstanceState){
super.onCreate(savedinstanceState);
setContentView(R. layout. activity_main);
} }
```

Resource is Layout in this XML

4. **Declare Activity in the Android manifest**

Main Activity needs to include intent to start from launcher icon

```
<activity android:name='.MainActivity">
<intent-filter>
<action android:name="android.intent.action.MAIN"/>
<category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
```

Syllabus Topic : Activity Lifecycle

1.7.2 Activity Lifecycle

- The Activity is in a set of states during its lifetime, from when it is created until it is destroyed.
- The Fig. 1.7.2 illustrates the Android Activity Lifecycle for an application.
- An activity lifecycle shows all the states an activity can be in, and the callbacks associated with transitioning from each state to the next one.

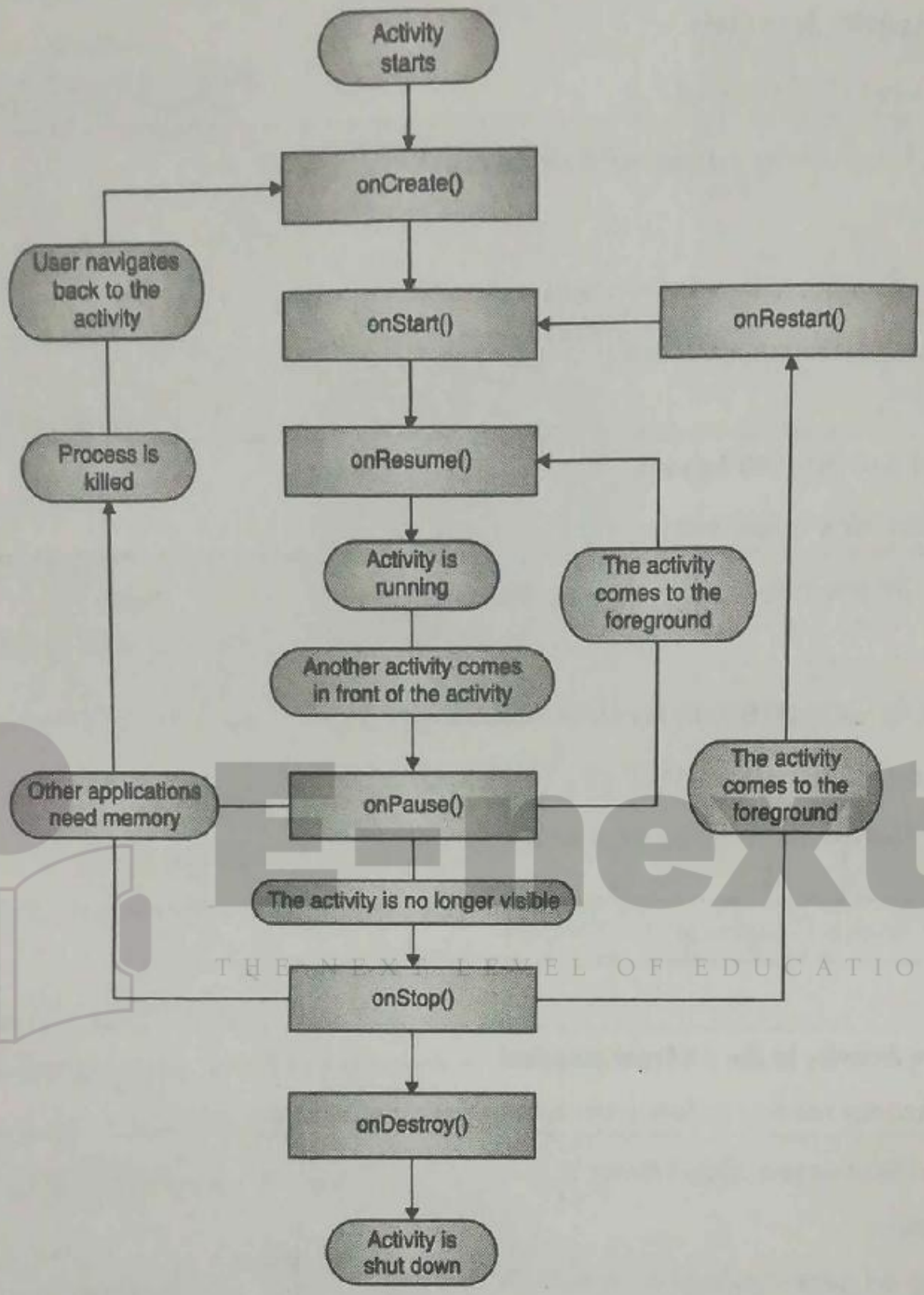


Fig. 1 7.2

1.7.3 Activity States and App Visibility

- The state changes in an activity are triggered by user action, configuration changes such as device rotation, or system action
  - Created (not visible yet)
  - Started (visible)
  - Resume (visible)
  - Paused(partially invisible)
  - Stopped (hidden)
  - Destroyed (gone from memory)

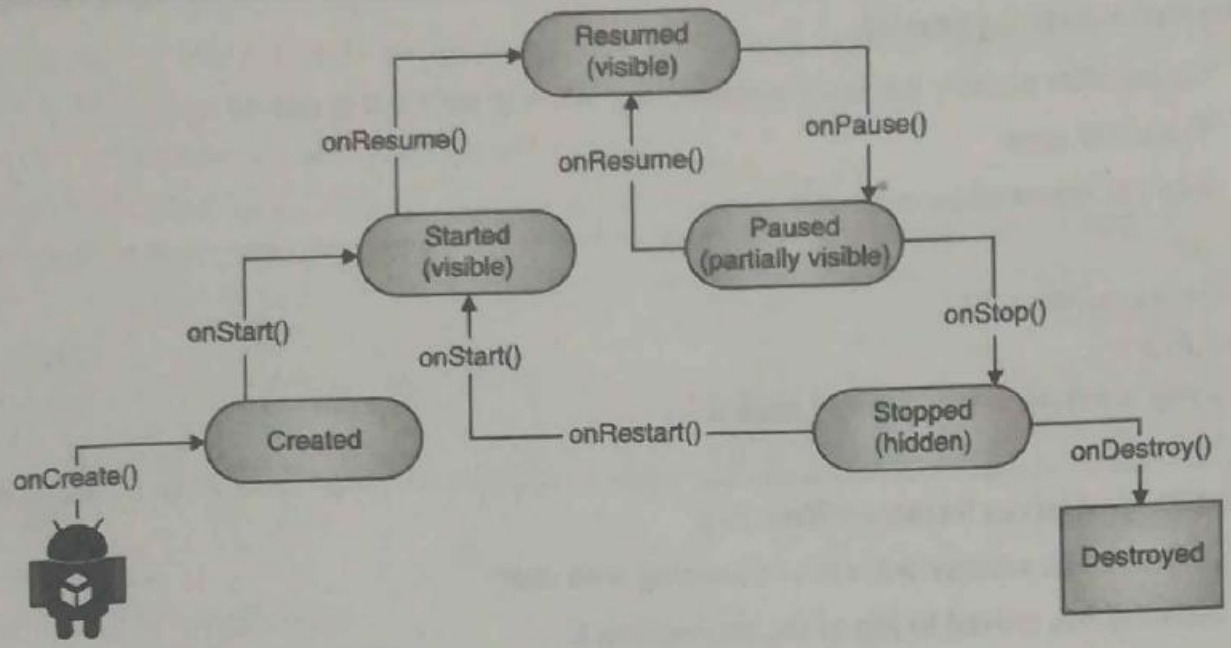


Fig. 1.7.3

1.7.4 Implementing and Overriding Callbacks

☞ **onCreate() –Activity Created**

- Called when the activity is first created, for example when user taps launcher icon
- Does all static setup: create views, bind data to lists, ...
- Only called once during an activity's lifetime
- Takes a Bundle with activity's previously frozen state, if there was one
- Created state is always followed by onStart()

```
@Override
public void onCreate(Bundle savedinstanceState){
super.onCreate(savedinstanceState);
//The activity is being created.
}
```

☞ **onStart()–Activity Started**

- Called when the activity is becoming visible to user
- Can be called more than once during lifecycle
- Followed by onResume() if the activity comes to the foreground, or onStop() if it becomes hidden

```
@Override
protected void onStart()
{super.onStart();
//The activity is about to become visible.
}
```

onRestart()–Activity Started

- Called after activity has been stopped, immediately before it is started again
- Transient state
- Always followed by onStart()

```
@Override
protected void onRestart() {
super.onRestart();
//The activity is between stopped and started.
}
```

onResume()–ActivityResumed/Running

- Called when activity will start interacting with user
- Activity has moved to top of the activity stack
- Starts accepting user input
- Running state
- Always followed by onPause()

```
@Override
protected void onResume(){
super.onResume();
//The activity has become visible
//it is now "resumed"
}
```


onPause()–ActivityPaused

- Called when system is about to resume a previous activity
- The activity is partly visible but user is leaving the activity
- Typically used to commit unsaved changes to persistent data, stop animations and anything that consumes resources
- Implementations must be fast because the next activity is not resumed until this method returns
- Followed by either onResume() if the activity returns back to the front, or onStop() if it becomes invisible to the user

```
@Override
protected void onPause(){
super.onPause();
//Another activity is taking focus
//this activity is about to be "paused"
}
```

onStop()–ActivityStopped

- Called when the activity is no longer visible to the user

- New activity is being started, an existing one is brought in front of this one, or this one is being destroyed
- Operations that were too heavy-weight for onPause
- Followed by either onRestart() if this activity is coming back to interact with the user, or onDestroy() if this activity is going away

```
@Override
protected void onStop(){
super.onStop();
//The activity is no longer visible
//it is now "stopped"
}
```

☞ **onDestroy()–ActivityDestroyed**

- Final call before activity is destroyed
- User navigates back to previous activity, or configuration changes
- Activity is finishing or system is destroying it to save space
- Call isFinishing() method to check
- System may destroy activity without calling this, so use onPause() or onStop() to save data or state

```
@Override
Protected void onDestroy(){
super.onDestroy();
//The activity is about to be destroyed.
}
```


Syllabus Topic : Intents

1.8 Intents

- An intent is a description of an operation to be performed.
- An Intent is a messaging object used to request an action from another app component via the Android system.

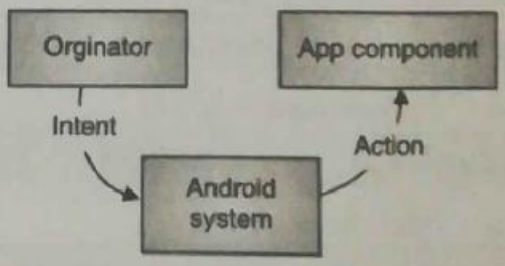


Fig. 1.8.1

- An Intent can be used to:
  1. Start an Activity
  2. Start a Service
  3. Deliver a Broadcast

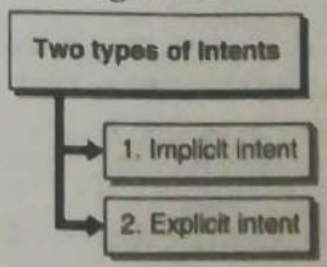


Fig. C1.3 : Types of Intents

☞ **Two types of Intents**

1. **Explicit intent :** Starts an activity of a specific class.
2. **Implicit intent :** Asks system to find an activity class with a registered handler that can handle request.

➜ 1.8.1 Implicit Intents

- An implicit intent allows you to start an activity in another app by describing an action you inte to perform, such as "share an article", "view a map", or "take a picture"
- An implicit intent specifies an action and may provide data with which to perform the action
- Implicit intents do not specify the target activity class, just the intended action
- Android runtime matches the implicit intent request with registered intent handlers
- If there are multiple matches, an App Chooser will open to let the user decide

☞ **Sending an Implicit Intent**

- Create an intent for an action

```
Intent intent = new Intent(Intent.ACTION_CALL_BUTTON);
```

- User has pressed Call button. Start an activity that allows them to make a call. No data passed in or returned.
- Start the activity

```
if (intent.resolveActivity(getPackageManager()) != null) {
startActivity(intent);
}
```

- Before starting an implicit activity, use the package manager to check that there is a packa with an activity that matches the given criteria.

```
Intent myIntent = new Intent(Intent.ACTION_CALL_BUTTON);
if (intent.resolveActivity(getPackageManager()) != null) {
startActivity(intent);
}
```

☞ **Sending an implicit intent with data**

- Create an intent for action

```
Intent intent = new Intent(Intent.ACTION_DIAL);
```

- Provide data as a URI

```
intent.setData(Uri.parse("tel:8005551234"));
```

- Start the activity

```
if (intent.resolveActivity(getPackageManager()) != null) {
startActivity(intent);
}
```

☞ **Providing the data as URI**

- Create an URI from a string using Uri.parse(String uri)

```
Uri.parse("tel:9810012345")
Uri.parse("geo:0,0?q=howrah%20bridge%2C%20howrah%2C%20cal")
Uri.parse("http://www.android.com");
```

1.8.1(A) Examples of Implicit Intents

Show a web page

```
Uri uri = Uri.parse("http://www.google.com");
Intent it = new Intent(Intent.ACTION_VIEW,uri);
startActivity(it);
```

Dial a phone number

```
Uri uri = Uri.parse("tel:9810012345");
Intent it = new Intent(Intent.ACTION_DIAL, uri);
startActivity(it);
```

Common actions for implicit intents

- Common actions for implicit intents include:
  1. ACTION_SET_ALARM
  2. ACTION_IMAGE_CAPTURE
  3. ACTION_CREATE_DOCUMENT
  4. ACTION_SENDTO
  5. And many **more**

1.8.1(B) Registering App to Receive Intents

- Declare one or more intent filters for the activity in the Android manifest
- Filter announces activity's ability to accept implicit intents
- Filter puts conditions on the intents that the activity accepts

```
<activity android:name="ShareActivity">
<intent-filter> <action android:name="android.intent.action.SEND"/>
<category android:name='android.intent.category.DEFAULT"/>
<data android:mimeType='text/plain"/>
</intent-filter> </activity>
```

☞ **An activity can have multiple filters**

```
<activity android:name="ShareActivity">
<intent-filter> <action android:name="android.intent.action.SEND"/>
</intent-filter>
<intent-filter> <action android:name="android.intent.action.SEND_MULTIPLE"/>
</intent-filter> </activity>
```

☞ **A filter can have multiple actions and data**

```
<intent-filter> <action android:name="android.intent.action.SEND"/>
<action android:name="android.intent.action.SEND_MULTIPLE"/>
<category android:name="android.intent.category.DEFAULT"/>
<data android:mimeType="image/*"/>
<data android:mimeType="video/*"/> </intent-filter>
```

1.8.2 Explicit Intent

- An explicit intent is used to launch a specific app component. Eg. a service in the app or particular activity.
- To create an explicit intent we need to, define the component name for the Intent object (all oth intent properties are optional).
- For example, to build a service in the app, named DownloadService, which downloads a file fro the web, we can start it with the following code:

```
//Executed in an Activity, so 'this' is the Context
//The fileUrl is a string URL, such as
"http://www.example.com/image.png"
Intent downloadIntent=new Intent(this, DownloadService.class);
downloadIntent.setData(Uri.Parse(fileUrl));
startService(downloadIntent);
```

- The Intent(Context, Class) constructor supplies the app Context and the component a Class obje As such, this intent explicitly starts the DownloadService class in the app.

Activity Instance State

- State information is created while the activity is running, such as a counter, user text, animati progression
- State is lost when device is rotated, language changes, back-button is pressed, or the system cle memory
- System only saves:
  - State of views with unique ID (android:id) such as text entered into EditText
  - Intent that started activity and data in its extras
- User is responsible for saving other activity and user progress data

1.9 UI Layouts

- **View** object is the basic building block for user interface. It is created from the View class. occupies a rectangular area on the screen and is responsible for event handling and drawing.
- View is also the base class for widgets, which are used to create interactive UI compone eg. buttons, text fields, etc.)

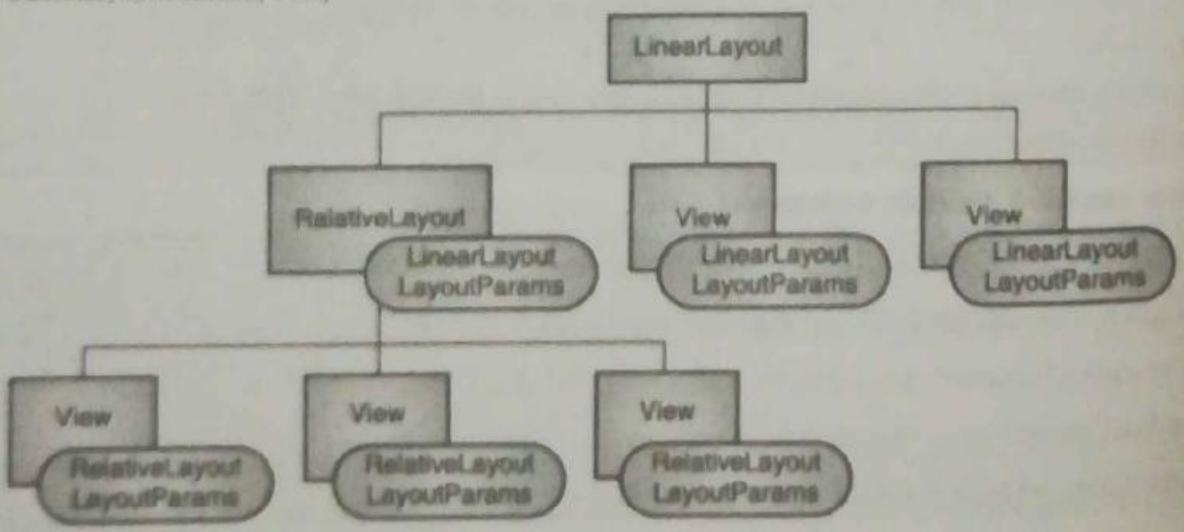


Fig. 1.9.1

- The **ViewGroup** is a subclass of **View**. It provides an invisible container that holds other Views or other ViewGroups and define their layout properties.
- The different layouts are the subclasses of the **ViewGroup** class. A typical layout defines the visual structure for an Android user interface. It can be created either at run time using **View/ViewGroup**objects or can be declared by the layout using the XML file **main_layout.xml** (located in the res/layout folder of the project).

1.9.1 Android Layout Types

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.

Linear Layout	LinearLayout is a view group It aligns all its child views in a single direction, either vertically or horizontally.
Relative Layout	RelativeLayout is a view group that displays child views in relative positions (i.e. relative to either each other or the screen)
Table Layout	TableLayout is a view that groups its child views into rows and columns.
Absolute Layout	AbsoluteLayout enables us to specify the exact location of its child views.
Frame Layout	The FrameLayout has a placeholder on the screen. It can be used to display a single view.
List View	ListView is a view group that displays a list of scrollable items.
Grid View	GridView is a ViewGroup that displays items in a two-dimensional, scrollable grid.

1.9.2 Layout Attributes

- Every layout has a set of attributes that defines the visual properties of that layout. Some common attributes among all the layouts and some attributes which are specific to that layout are:

android:id	This ID uniquely identifies the view.
android:layout_width	Specifies the width of the layout.
android:layout_height	Specifies the height of the layout
android:layout_marginRight	Specifies the extra space towards the right of the layout.
android:layout_marginLeft	Specifies the extra space towards the left of the layout.
android:layout_marginBottom	Specifies the extra space towards the bottom of the layout.
android:layout_marginTop	Specifies the extra space towards the top of the layout.
android:layout_gravity	Specifies how child Views are positioned.
android:layout_weight	This specifies how much of the extra space in the layout should be allocated to the View.
android:layout_height	Specifies the height of the layout.
android:layout_width	Specifies the width of the layout.
android:layout_y	Specifies y-coordinate of layout.
android:layout_x	Specifies x-coordinate of layout.
android:paddingLeft	Specifies the left padding for the layout.
android:paddingRight	Specifies the right padding for the layout.
android:paddingTop	Specifies the top padding for the layout.
android:paddingBottom	Specifies the bottom padding for the layout.

- Here we will be discussing two important layouts: LinearLayout and RelativeLayout

1.9.3 LinearLayout

Android LinearLayout is a view group that aligns all children in either vertically or horizontally

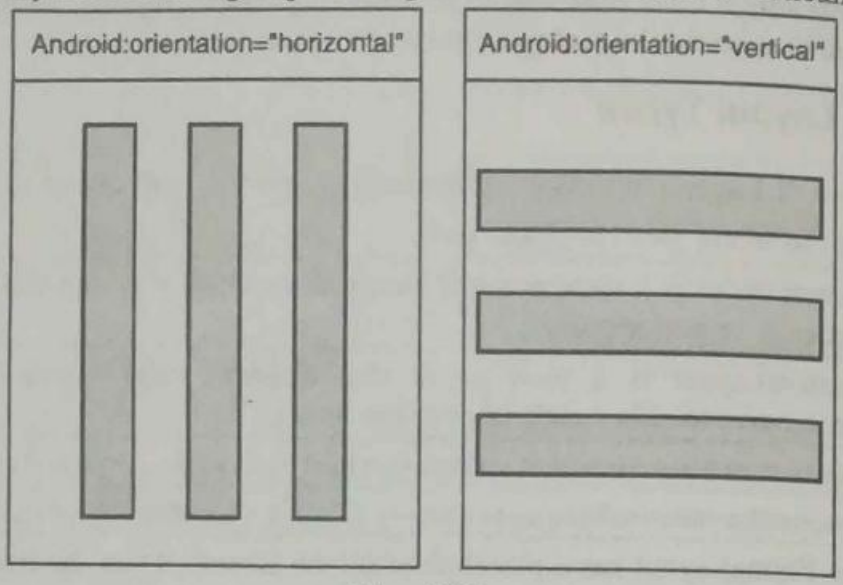


Fig. 1.9.2

Attributes of LinearLayout

Following are the important attributes specific to LinearLayout :

android:id	This is the ID which uniquely identifies the layout.
android:baselineAligned	This must be a boolean value, either "true" or "false" a prevents the layout from aligning its children's baselines.
android:baselineAlignedChildIndex	When a linear layout is part of another layout that is baseli aligned, it can specify which of its children to baseline align.
android:divider	This is drawable to use as a vertical divider between button You use a color value, in the form of "#rgb", "#argb "#rrggbb", or "#aarrggbb".
android:gravity	This specifies how an object should position its content, both the X and Y axes. Possible values are top, bottom, le right, center, center_vertical, center_horizontal etc.
android:orientation	This specifies the direction of arrangement and you will u "horizontal" for a row, "vertical" for a column. The default horizontal.
android:weightSum	Sum up of child weight

Example of LinearLayout

Following is the content of the modified main activity file **src/com.example.demo/MainActivity.ja**

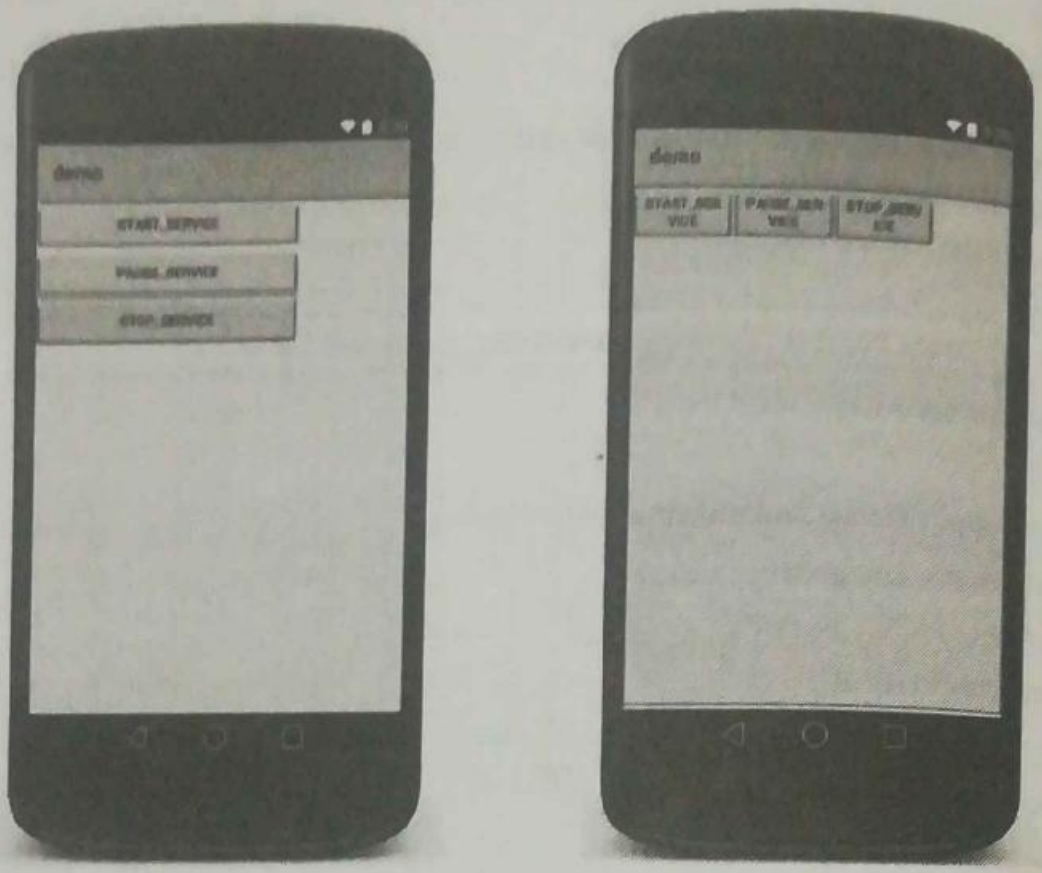
```
package com.example.demo;
import android.os.Bundle;
```

```
import android.app.Activity;
public  lass MainActivity extendsActivity{
@Override
protected void onCreate(BundlesavedInstanceState){
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
} }
```

- Following will be the content of **res/layout/activity_main.xml** file –

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

<Button android:id="@+id/btnStartService"
android:layout_width="270dp"
android:layout_height="wrap_content"
android:text="start_service"/>

<Button android:id="@+id/btnPauseService"
android:layout_width="270dp"
android:layout_height="wrap_content"
android:text="pause_service"/>

<Button android:id="@+id/btnStopService"android:layout_width="270dp"
android:layout_height="wrap_content"
android:text="stop_service"/>

</LinearLayout>
```


- Do not make any changes to the strings.xml and AndroidManifest.xml file.
- To run the app from Android studio, open one of your project's activity files and click Run icon from the toolbar. Android studio 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 (Fig. 1.9.3(a)).
- Now let's change the orientation of Layout as **android:orientation="horizontal"** and try to run the same application, it will give following screen (Fig. 1.9.3(b))



(a) (b)

Fig. 1.9.3

1.9.4 RelativeLayout

- Android RelativeLayout helps to specify the positioning of child views relative to each other.
- The position of each view can be specified relative to the parent or its sibling elements.

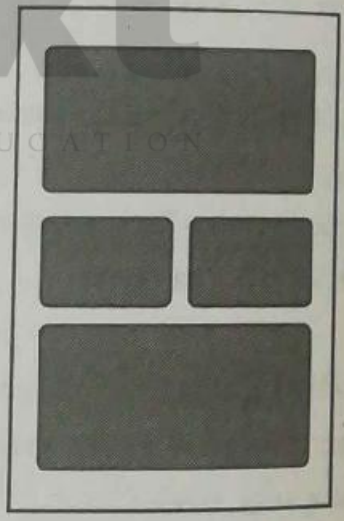


Fig. 1.9.4

1.9.4(A) RelativeLayout Attributes

Following are the important attributes specific to RelativeLayout.

android:id	This ID uniquely identifies the layout.
android:gravity	This attribute specifies how the contents of an object should be positione on the X and Y axes. Values are "top", "bottom", "left", "right", "center "center_vertical", "center_horizontal" etc.
android:ignoreGravity	This indicates what view should not be affected by gravity.

- Using RelativeLayout, you can align two elements by right border, or make one below another, centered in the screen, centered left, and so on.
- By default, all child views are drawn at the top-left of the layout, so you must define the position of each view using the various layout properties available from **RelativeLayout.LayoutParams** and few of the important attributes are given below.

android:layout_above	Aligns the bottom edge of the current view above the given anchor view ID. It needs to be a reference to another resource, of the form "@[+][package:]type:name"
android:layout_alignBottom	Aligns the bottom edge of the current view with the bottom edge of the given anchor view ID. It needs to be a reference to another resource, of the form "@[+][package:]type:name".
android:layout_alignLeft	Aligns the left edge of the current view with the bottom edge of the given anchor view ID. It needs to be a reference to another resource, of the form "@[+][package:]type:name".
android:layout_alignRight	Aligns the right edge of the current view with the right edge of the given anchor view ID. It needs to be a reference to another resource, of the form "@[+][package:]type:name".
android:layout_alignStart	Aligns the start of the current view with the start edge of the given anchor view ID. It needs to be a reference to another resource, of the form "@[+][package:]type:name".
android:layout_alignParentBottom	If true, it makes the bottom edge of this view align with the bottom edge of the parent. This is a boolean value ("true" or "false").
android:layout_alignParentEnd	If true, it makes the end edge of this view align with the end edge of the parent. This is a boolean value ("true" or "false").
android:layout_alignParentLeft	If true, it makes the left edge of this view align with the left edge of the parent. This is a boolean value ("true" or "false").
android:layout_alignParentRight	If true, it makes the right edge of this view align with the right edge of the parent. This is a boolean value ("true" or "false").
android:layout_alignParentStart	If true, it makes the start edge of this view align with the start edge of the parent. This is a boolean value ("true" or "false").
android:layout_alignParentTop	If true, it makes the top edge of this view align with the top edge of the parent. This is a boolean value ("true" or "false").
android:layout_alignTop	Makes the top edge of this view match the top edge of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
android:layout_below	Positions the top edge of this view below the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
android:layout_centerHorizontal	If true, centers this child horizontally within its parent. Must be a boolean value, either "true" or "false".
android:layout_centerInParent	If true, centers this child horizontally and vertically within its parent. Must be a boolean value, either "true" or "false".

android:layout_centerVertical	If true, centers this child vertically within its parent. Must be a boolean value, either "true" or "false".
android:layout_toEndOf	Positions the start edge of this view to the end of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
android:layout_toLeftOf	Positions the right edge of this view to the left of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
android:layout_toRightOf	Positions the left edge of this view to the right of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
android:layout_toStartOf	Positions the end edge of this view to the start of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".

1.9.4(B) Example of RelativeLayout

llowing is the content of the modified main activity file **src/com.example.demo/MainActivity.java**.

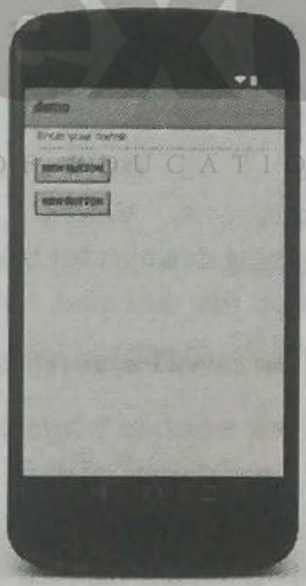
```
package com.example.demo;
import android.os.Bundle;
import android.app.Activity;
public class MainActivity extendsActivity{
@Override
protected void onCreate(BundlesavedInstanceState){
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
} }
```

– Following will be the content of **res/layout/activity_main.xml** file

```
<RelativeLayout
xmlns:android="http:/schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:paddingeft="16dp"
android:paddingRight="16dp">

<EditText
android:id="@+id/name"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:hint="@string/reminder"/>

<LinearLayout
android:orientation="vertical"
```

```
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:layout_alignParentStart="true"
android:layout_below="@+id/name">

<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="NewButton"
android:id="@+id/button"/>

<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="NewButton"
android:id="@+id/button2"/>
</LinearLayout>
</RelativeLayout>
```

- Do not make any changes to the strings.xml and AndroidManifest.xml file.
- To run the app from Android studio, open one of your project's activity files and click Run icon from the toolbar. Android studio 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.



Syllabus Topic : Saving State

1.10 Saving State

☞ **What is saving instance state?**

- All activities have an onSaveInstanceState() method that can be overridden.
- When this method is called, any state-related data should be placed into the outState Bundle.
- This method is called when an Activity is being backgrounded (either after onPause() or onStop(), depending on different factors).

What is the savedInstanceState Bundle?

- The savedInstanceState is a reference to a Bundle object that is passed into the onCreate() method of every Android Activity.
- Activities have the ability, under special circumstances, to restore themselves to a previous state using the data stored in this bundle.
- If there is no available instance data, the savedInstanceState will be null. For example, the savedInstanceState will always be null the first time an Activity is started, but may be non-null if an Activity is destroyed during rotation.

What should be saved?

- The savedInstanceState Bundle should only save information directly related to the current Activity state. Examples of this include:
- User selections – A user selects a tab. In onSaveInstanceState the tab selection gets added to the outState Bundle. During the next onCreate, the selected tab will be available within the Bundle, and the Activity should default to having that tab selected.
- Scroll view positions – A user scrolls half way through a ScrollView. The current position of the ScrollView should be saved in onSaveInstanceState then restored when the Activity is re-created.
- User-submitted data – If a user writes their username into a text box, they would expect the username to still be present when the Activity is resumed.

What should not be saved?

- In general (with few exceptions), the following kinds of data should never be saved into the Bundle:
  1. Files
  2. Database data
  3. Images
  4. Videos
  5. Anything downloaded from the web (feed data)
  6. Models (the data kind, not the people kind)

When is the savedInstanceState useful?

- There is one situation where using the savedInstanceState is almost mandatory: when the Activity gets destroyed during rotation.
  - One way that Android handles rotation is to completely destroy and then re-create the current Activity. When the Activity is being destroyed, any state related information that is saved in onSaveInstanceState will be available when the Activity comes back online.
  - Another situation is when the Activity gets backgrounded. When an Activity is in a backgrounded state (either after onPause or onStop) it can be destroyed at any time with no notice. In case the OS kills your Activity without killing your Application, the OS will first save your outState Bundle so you can later return to your previous state.

Saving Instance State

- Implement onSaveInstanceState() in your activity
  - called by Android runtime when there is a possibility the activity may be destroyed
  - saves data only for this instance of the activity during current session

```
@Override
public void onSaveinstanceState(Bundle outstate){
super.onSaveinstanceState(outState);
//Add information for saving Hello Toast counter
//to  the to the outstate bundle
outState.putString("count",
String.valueOf(mShowCount.getText()));
}
```

☞ **Instance state and app restart**

- When you stop and restart a new app session, the activity instance states are lost and your activities will revert to their default appearance
- If you need to save user data between app sessions, use shared preferences or a database.

Syllabus Topic : Basic Views

1.11 Basic Views

- Users interact with the apps in the following ways:
  - Clicking, pressing, talking, typing, and listening
  - Using user input controls such buttons, menus, keyboards, text boxes, and a microphone
  - Navigating between activities

☞ **Getting input from the user**

There are three ways of getting inputs from the user:

1. Free form : Text and voice input
2. Actions : Buttons, Contextual menus, Gestures, Dialogs
3. Constrained choices : Pickers, Checkboxes, Radio buttons, Toggle buttons, Spinners

☞ **Some user controls**

1.	Button	2.	Text field	3.	Seek bar	4.	Checkboxes
5.	Radio buttons	6.	Toggle	7.	Spinner	8.	Alert Dialog
9.	Date Picker	10.	Time Picker				

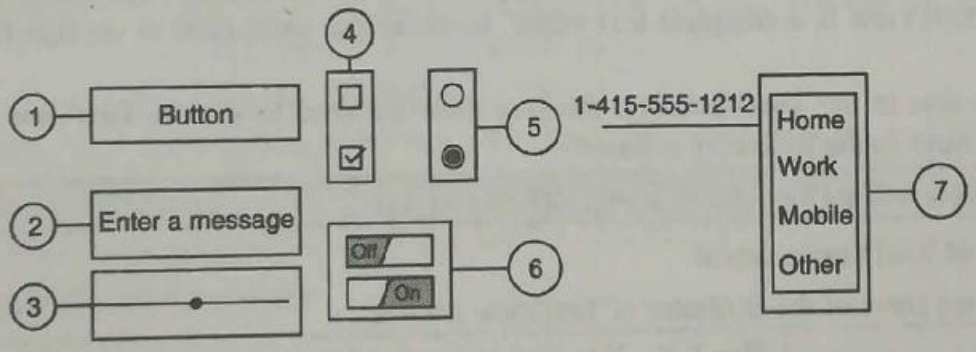


Fig. 1.11.1(Cont...)

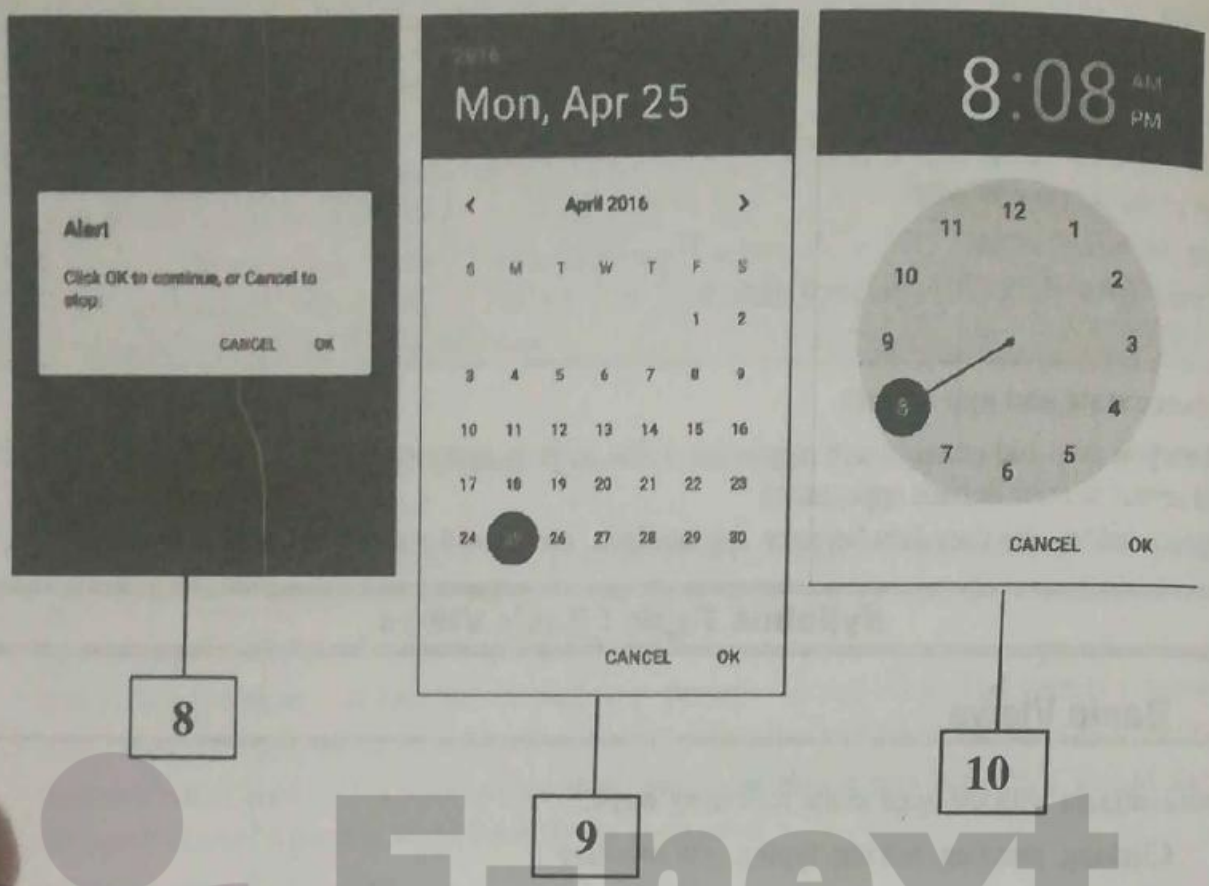


Fig. 1.11.1

1.11.1 View Class

- The View is the base class for all the input controls.
- The View class is the basic building block for all UI components, including input controls
- View is the base class for classes that provide interactive UI components
- View provides basic interaction through android:onClick

Syllabus Topic : TextView

1.11.2 TextView

- TextView is a user interface element that displays text to the user and optionally allows them to edit it. A TextView is a complete text editor, however the basic class is configured to not allow editing.
- To get the text in the TextView into the java code we need to get the TextView object for the TextView view, in the following manner:

```
TextViewtxtView = (TextView)findViewById(R.id.text_id);
```

Attributes of TextView control

- Following are some of the attributes of TextView control:

android:id	This is the ID which uniquely identifies the control.
android:capitalize	If set, specifies that this TextView has a textual input method and should automatically capitalize what the user types.

	Don't automatically capitalize anything - 0 Capitalize the first word of each sentence - 1 Capitalize the first letter of every word – 2 Capitalize every character - 3
android:cursorVisible	Makes the cursor visible (the default) or invisible. Default is false.
android:editable	If set to true, specifies that this TextView has an input method.
android:fontFamily	Font family (named by string) for the text
android:gravity	Specifies how to align the text by the view's x- and/or y-axis when the text is smaller than the view.
android:hint	Hint text to display when the text is empty.
android:inputType	The type of data being placed in a text field. Phone, Date, Time, Number, Password etc.
android:maxHeight	Makes the TextView be at most this many pixels tall.
android:maxWidth	Makes the TextView be at most this many pixels wide.
android:minHeight	Makes the TextView be at least this many pixels tall.
android:minWidth	Makes the TextView be at least this many pixels wide.
android:password	Whether the characters of the field are displayed as password dots instead of themselves. Possible value either "true" or "false".
android:phoneNumber	If set, specifies that this TextView has a phone number input method. Possible value either "true" or "false".
android:text	Text to display.
android:textAllCaps	Present the text in ALL CAPS. Possible value either "true" or "false".
android:textColor	Text color. May be a color value, in the form of "#rgb", "#argb", "#rrggbb", or "#aarrggbb".
android:textColorHighlight	Color of the text selection highlight.
android:textColorHint	Color of the hint text. May be a color value, in the form of "#rgb", "#argb", "#rrggbb", or "#aarrggbb".
android:textIsSelectable	Indicates that the content of a non-editable text can be selected. Possible value either "true" or "false".
android:textSize	Size of the text. Recommended dimension type for text is "sp" for scaled-pixels (example: 15sp).
android:textStyle	Style (bold, italic, bolditalic) for the text. You can use or more of the following values separated by '|'. normal – 0, bold – 1, italic - 2
android:typeface	Typeface (normal, sans, serif, monospace) for the text. You can use or more of the following values separated by '|'. normal – 0 , sans – 1, serif – 2, monospace - 3

1.11.2(A) Example of TextView

– This example will take you through simple steps to show how to create your own Android application using Relative Layout and TextView. Following is the content of the modified main activity file src/com.example.demo/MainActivity.java.

```
package com.examgle.demo;
import android.os.Bundle;
import android.agp.Activit'i;
import android.view.Menu;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends Activity{
@Override
protected void onCreate(Bundle savedInstanceState){
suger.onCreate(savedInstanceState);
setContentView(R.layout.activity_imain);
//--textview---
TextView txtView=(TextView)findViewById(R.id.textid);
}
```

Following will be the content of res/layout/activity_main.xml file

```
<RelativeLayout xmlns:android="http://schemas.androld.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_veritcal_margin"
android :paddingLeft ="@dimen/activity_horizontal_margin"
android :paddingRight="@dimen/activity_horizontal_margin"
android :paddingTop ="@dimen/activity_veritcal_margin"
tools:context=".MainActivity"
<TextView
android :id="@+id/text/id"
android :layout_width="300dp"
android :layout_height="200dp"
android :capotalize ="charatcers"
android :text = "hello_world"
android :textColor="@andoird:color/holo_blur_dark"
android :textColorHighlight="@andoird:color/primary_text_dark"
android :layout_centerVerical ="true"
android :layout_alignParentEnd ="true"
android :textSize ="50dp"/>
</RelativeLayout>
```

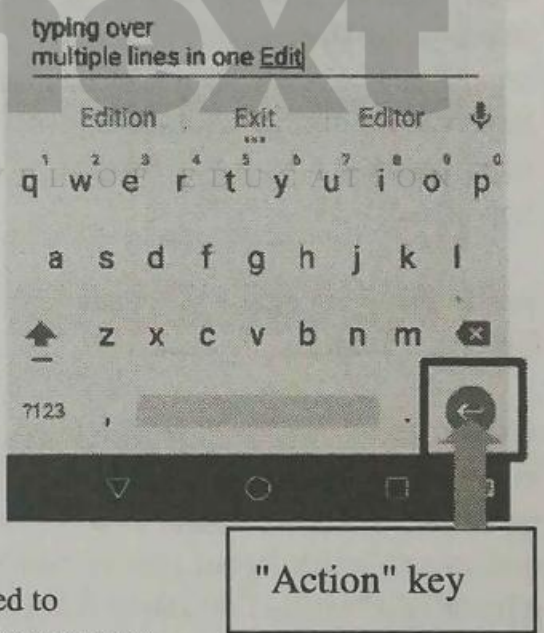
- Do not make any changes to the strings.xml and AndroidManifest.xml file.
- To run the app from Android studio, open one of your project's activity files and click Run icon from the toolbar.
- Android studio installs the app on your AVD and starts it and if everything is fine with your setup and application, it will display Fig. 1.11.2 Emulator window

Fig. 1.11.2

Syllabus Topic : EditText

1.11.3 EditText

- EditText is a user interface element for entering and modifying text.It is an overlay over TextView that configures itself to be editable. It is the predefined subclass of TextView that includes rich editing capabilities.EditText allows:
  - Multiple lines of input
  - Characters, numbers, and symbols
  - Spelling correction
  - Tapping the Return (Enter) key starts a new line
  - Customizable



- To get the text entered by the user into the java code we need to get the EditText object for the EditText view, in the following manner:

```
EditTextsimpleEditText = (EditText)findViewById(R.id.edit_simple);
```

We can Retrieve the CharSequence and convert it to a string in the following manner:

```
String strValue = simpleEditText.getText().toString();
```

Attributes of EditText control

Following are some of the attributes of EditText control:

android:autoText	If set, specifies that this TextView has a textual input method and automatically corrects some common spelling errors.
android:drawableBottom	This is the drawable to be drawn below the text.

android:drawableRight	This is the drawable to be drawn to the right of the text.
android:editable	If set, specifies that this TextView has an input method.
android:text	This is the Text to display.
android:background	This is a drawable to use as the background.
android:contentDescription	This defines text that briefly describes content of the view.
android:id	This supplies an identifier name for this view.
android:onClick	This is the name of the method in this View's context to invoke wh the view is clicked.
android:visibility	This controls the initial visibility of the view.

1.11.3(A) Example of EditText Control

- This example will take you through simple steps to show how to create your own And application using Linear Layout and EditText.
- Following is the content of the modified main activity file src/com.example.demo/MainActivity.java.

```
package com.examele.demo;

import android.cs.Bundle;
import android.aee.Activity;
import android.view.View;
import  android.view.View.OnClicklistener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MalnActivity extendsActivity{
EditText eText;

@Override
protected void onCreate(BundlesavedlnstanceState){
super.onCreate(savedlnstanceState);
setContentView(R.layout.activity main);
eText=(EditText)findViewByid(R.id.edittext);
} }
```

Following will be the content of res/layout/activity_main.xml file

```
<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="matchparent"
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=".MainActivity">

<EditText
android:id="@+id/edittext"
android:layoutwidth="fillparent"
android:layoutheight="wrapcontent"
android:layout aligneft="@+id/button"
android:layout below="@+id/textViewl"
android:layout marginTop="61dp"
android:ems=" O"
android:text="Enter_text"
android:inputType="text"/>
</Relativelayout>
```

- Do not make any changes to the strings.xml and AndroidManifest.xml file.
- To run the app from Android studio, open one of your project's activity files and click Run icon from the toolbar.
- Android studio installs the app on your AVD and starts it and if everything is fine with your setup and application, it will display Fig. 1.11.3 Emulator window.

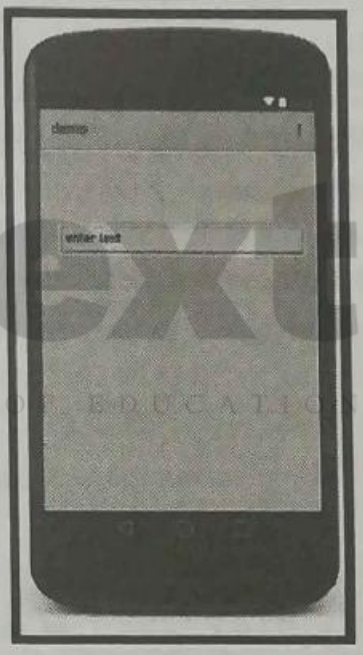
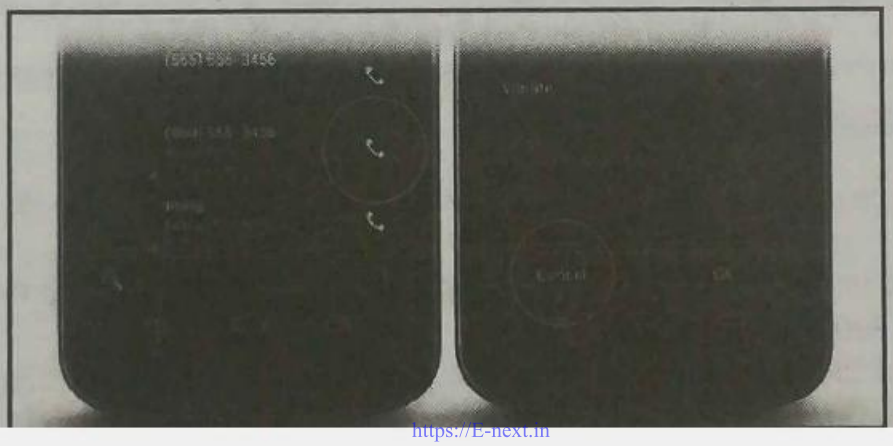


Fig. 1.11.3

Syllabus Topic : Button

1.11.4 Button

A Button is a Push-button which can be pressed, or clicked, by the user to perform an action.



☞ **Attributes of Button control**

Following are some of the attributes of Button control:

android:autoText	if set, specifies that this TextView has a textual input meth automatically corrects some common spelling errors.
android:drawableBottom	This is the drawable to be drawn below the text.
android:drawableRight	This is the drawable to be drawn to the right of the text.
android:editable	if set, specifies that this TextView has an input method.
android:text	This is the Text to display.
android:background	This is a drawable to use as the background.
android:contentDescription	This defines text that briefly describes content of the view.
android:id	This supplies an identifier name for this view.
android:onClick	This is the name of the method in this View's context to invoke wh view is clicked.
android:visibility	This controls the initial visibility of the view.

☞ **Responding to Button Clicks**

- So, whenever a button is pressed some action needs to take place. This action needs to be and that code needs to be invoked. This invocation can be handled in two methods:
- **In your code :** Use OnClickListener event listener.

```
Button button = (Button) findViewById(R.id.button);
button.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
    // Do something in response to button click here
}
});
```

- *In XML:* use android:onClick attribute in the XML layout:

```
<Button
android:id="@+id/button_send"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/button_send"
android:onClick="sendMessage" />
```

→ Android:onClick

Syllabus Topic : ImageButton

1.11.5 ImageButton

- An ImageButton displays a button with an image (instead of text) that can be pressed or click the user.
- By default, an ImageButton looks like a regular Button, with the standard button backgroun changes color during different button states.
- The image on the surface of the button is defined either by the android:src attribute i <ImageButton> XML element or by the setImageResource(int) method.

To remove the standard button background image, define your own background image or set the background color to be transparent.

Attributes of the CheckBox contro

Following are some attributes of the CheckBox control

android:baseline	This is the offset of the baseline within this view.
android:baselineAlignBottom	If true, the image view will be baseline aligned with based on its bottom edge.
android:cropToPadding	If true, the image will be cropped to fit within its padding.
android:src	This sets a drawable as the content of this ImageView
android:background	This is a drawable to use as the background.
android:contentDescription	This defines text that briefly describes content of the view.
android:id	This supplies an identifier name for this view
android:onClick	This is the name of the method in this View's context to invoke when the view is clicked.
android:visibility	This controls the initial visibility of the view.

1.11.5(A) Example of ImageButton

Following is the content of the modified main activity file src/com.example.myapplication /MainActivity.java. In the below example abc indicates the image of Android logo which should be pasted in the **drawable** folder of the **res** folder.

```
package com.example.mypplication;

import android.os.Bundle;
import android.app.Activity;
iomport android.view.View;
import android.view.View.OnClicklistener;
import android.widget.lmageButton;
import android.widgetToast;
public class MainActivity extends Activity{
lmageButton imgButton;
@Override
protected void onCreate(Bundle savedlnstanceState){
super.onCreate(savedlnstanceState);
setContentView(R.layout.activity  main);
imgButton =(lmageButton)findViewByld(R.id.imageButton);
imgButton.setOnClickListener(new View.OnClickListener(){
@Override
public void onClick(View v){
```

```
Toast.makeText(getApplicationContext(),"You download is resumed",Toast.LENGTH
LONG).show();
}
}); }
```

- Following will be the content of res/layout/activity_main.xml file –

```
<?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:lavout_height="match_parent"
android:paddingeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
android:paddingBottom="@dimen/activity_vertical_margin"
tools:context=".MainActivity">

<TextView android:text="ImageButton"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textSize="30dp"
android:layout_alignParentTop="true"
android:layout_alignRight="@+id/imageButton"
android:layout_alignEnd="@+id/imageButton"/>
<ImageButton
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/imageButton"
android:layout_centerVertical="true"
android:layout _centerHorizontal="true"
android:src="@drawable/abc"/>
</RelativeLayout>
```

- Do not make any changes to the strings.xml and AndroidManifest.xml file.
- To run the app from Android studio, open one of your project's activity files and click Run from the toolbar. Android studio installs the app on your AVD and starts it and if everything is with your setup and application, it will display Fig. 1.11.4 Emulator window.

- The Fig. 1.11.5 screen will appear after ImageButton is clicked. It shows a toast message.

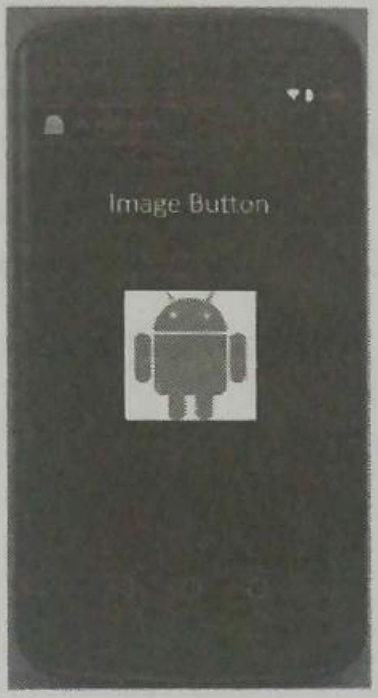
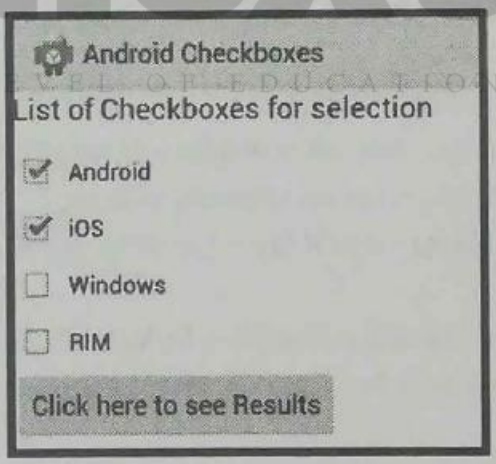


Fig. 1.11.4

Fig. 1.11.5

Syllabus Topic : CheckBox

1.11.6 CheckBox



A CheckBox is an on/off switch that can be toggled by the user. It can be used when there is a group of options that are not mutually exclusive i.e. you may chose more than one options.

- User can select any number of choices
- Checking one box does not uncheck another
- Users expect checkboxes in a vertical list
- Commonly used with a submit button
- Every checkbox is a view and can have an onClick handler

Attributes of the CheckBox control

- Following are some attributes of the CheckBox control

android:autoText	If set, specifies that this TextView has a textual input method and automatically corrects some common spelling errors.
android:drawableBottom	This is the drawable to be drawn below the text.
android:drawableRight	This is the drawable to be drawn to the right of the text.
android:editable	If set, specifies that this TextView has an input method.
android:text	This is the Text to display.
android:background	This is a drawable to use as the background.
android:id	This supplies an identifier name for this view.

android:onClick	This is the name of the method in this View's context to in when the view is clicked.
android:visibility	This controls the initial visibility of the view

1.11.6(A) Example of CheckBox Control

Following is the content of the modified main activity file **src/MainActivity.java**. In the example abc indicates the image of Android logo which should be pasted in the **drawable** folde res folder.

```
package com.example.myapplication;

import android.os.Bundle;
import android.app.Activity;
import android.widget.Button;
import android.view.View;
import android.view.View.OnClickistener;
import android.widget.CheckBox;
import android.widget.Toast;

public class MainActivity extendsActivity{
CheckBox ch1,ch2;
Button b1,b2;

@Override
protected void onCreate(Bundle savedInstanceState){
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);

ch1=(CheckBox)findViewByld(R.id.checkBox );
ch2=(CheckBox)findViewBy  ld(R.id.checkBox2);

b  =(Button)findViewByld(R.id.button);
b2=(Button)findViewByld(R.id.button2);
b2.setOnClicklistener(newView.OnClickListener(){

@Override
public void onClick(View v){finish();
}
});
b .setOnClicklistener(new View.OnClickListener(){
@Override
public void onClick(View v){
```

```
StringBuffer result=new StringBuffer();
result.append("Thanks:").append(chl.isChecked());
result.append("\nThanks: ").append(ch2.isChecked());
Toast.makeText(MainActivity.this,result.toString(),
Toast.LENGTH_LONG).show();
}
});
} }
```

- Following will be the content of **res/layout/activity_main.xml** file

```
<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=".MainActivity">
<TextView android:id="@+id/textViewl"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Exampleofcheckbox"
android:layout_alignParentTop="true"
android:layout_centerHorizontal="true"
android:textSize="30dp"/>

<CheckBox
android:id="@+id/checkBoxl"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="DoyoulikeJava"
android:layout_above="@+id/button"
android:layout_centerHorizontal="true"/>

<CheckBox
android:id="@+id/checkBox2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Doyoulike android"
android:checked="false"
android:layout_above="@+id/checkBoxl"
```

```
android:layout_alignleft="@+id/checkBoxl"
android:layout_alignStart="@+id/checkBoxl"/>
<Button
android:layout_width="wrap_content"
android :layout_height="wrap_content"
android:text="Ok"
android:id="@+id/button"
android:layout_alignParentBottom="true"
android:layout_aligneft="@+id/checkBox"
android:layout_alignStart="@+id/checkBox"/>

<Button
android:layout_width="wrap_content"
android :layout_height="wrap_content"
android:text="Cancel"
android:id="@+id/button2"
android:layout_alignParentBottom="true"
android:layout_alignRight="@+id/textView2"
android:layout_alignEnd="@+id/textView2"/>

<ImageButton
android:layout_width="wrap_content"
android :layout_height="wrap_content"
android:id="@+id/imageButton"
android:src="@drawable/abe"
android:layout_centerVertical="true"
android:layout_centerHorizontal="true"/>
</Relativelayout>
```


- Do not make any changes to the strings.xml and AndroidManifest.xml file.
- To run the app from Android studio, open one of your project's activity files and click
- Run icon from the toolbar.
- Android studio installs the app on your AVD and starts it and if everything is fine with your setup and application, it will display Fig. 1.11.6 Emulator window.

Fig. 1.11.6

Syllabus Topic : ToggleButton

1.11.7 ToggleButton

- A ToggleButton displays checked/unchecked states as a button. It is basically an on/off button with a light indicator.
- Android ToggleButton is a subclass of CompoundButton class.

☞ **Facility for creating ToggleButton class**

- ToggleButton class provides the facility of creating the toggle button.

android:textOff	This is the text for the button when it is not checked.
android:textOn	This is the text for the button when it is checked
android:autoText	If set, specifies that this TextView has a textual input method and automatically corrects some common spelling errors
android:drawableBottom	This is the drawable to be drawn below the text.
android:drawableRight	This is the drawable to be drawn to the right of the text.
android:editable	If set, specifies that this TextView has an input method.
android:text	This is the Text to display.
android:background	This is a drawable to use as the background.
android:contentDescription	This defines text that briefly describes content of the view.
android:id	This supplies an identifier name for this view.
android:onClick	This is the name of the method in this View's context to invoke when the view is clicked
android:visibility	This controls the initial visibility of the view.

1.11.7(A) Example of ToggleButton

- Following is the content of the modified main activity file **src/MainActivity.java**. In the below example abc indicates the image of Android logo which should be pasted in the **drawable** folder of the **res** folder.

```
package com.example.myapplication;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
import android.widget.ToggleButton;

public class MainActivity extends ActionBarActivity{
ToggleButton  tg1,tg2;
Button b;
protected void onCreate(Bundle savedInstanceState){
```

```
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);

tg =(ToggleButton)findViewByld(R.id.toggleButton);
tg2=(ToggleButton)findViewByld(R.id.toggleButton2);

b =(Button)findViewByld(R.id.button2);
b .setOnClicklistener(newView.OnClickListener(){
@Override
public void  onClick(View v){
StringBufferresult=newStringBuffer();
result.append(//YouhaveclickedfirstONButton-:)
//append(tg.getText());
result.append(//YouhaveclickedSecondONButton-:)
//.append(tg2.getText());
Toast.makeText(MainActivity.this,
result.toString(),Toast.LENGTH_SHORT).show();
}
});
} }
```

– Following will be the content of **res/layout/activity_main.**xml file

```
<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:paddingeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
tools:context=".MainActivity">

<TextView
android:id="@+id/textView2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Android"
android:textColor="#ff87ff09"
```

```
android:textSize="30dp"
android:layout_above="@+id/imageButton"
android :layout_centerHorizontal="true"
android:layout_marginBottom="40dp"/>

<ImageButton
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/imageButton"
android:src="@drawable/abe"
android:layout_centerVertical="true"
android:layout_centerHorizontal="true"/>

<ToggleButton
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="On"
android:id="@+id/toggleButton"
android:checked="true"
android:layout_below="@+id/imageButton"
android:layout_toEndOf="@+id/button2/>

<ToggleButton
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Off"
android:id="@+id/toggleButton2"
android:checked="true"
android:layout_alignTop="@+id/toggleButton"/>
<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/button2"
android:text="CIickMe"
android:layout_alignParentBottom="true"
android:layout_centerHorizontal="true"/>
</RelativeLayout>
```

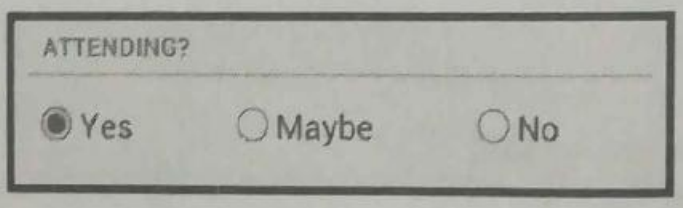
- Do not make any changes to the strings.xml and AndroidManifest.xml file.
- To run the app from Android studio, open one of your project's activity files and click Run icon from the toolbar.
- Android studio installs the app on your AVD and starts it and if everything is fine with your setup and application, it will display Fig. 1.11.7 Emulator window.
- If you click first ON Button, you will get a message on Toast as **You have clicked first ON Button-:)** or else if you clicked on second ON button, you will get a message on Toast as **You have clicked Second ON Button -:)**

Fig. 1.11.7

Syllabus Topic : RadioButton

1.11.8 RadioButton

- A radio button is a two-state button that can be either checked or unchecked.
- When the radio button is unchecked, the user can press or click it to check it. However, contrary to a CheckBox, a radio button cannot be unchecked by the user once checked.
- Radio buttons are normally used together in a RadioGroup. When several radio buttons live inside a radio group, checking one radio button unchecks all the others.
  - User can select one of a number of choices
  - It is advisable to put radio buttons in a RadioGroup
  - Checking one option unchecks another
  - Put radio buttons in a vertical list or horizontally if labels are short
  - Every radio button can have an onClick handler
  - It is commonly used with a submit buttonfor the RadioGroup



1.11.8(A) Example of RadioButton

- Following is the content of the modified main activity file **src/MainActivity.java**.

- In the below example abc indicates the image of Android logo which should be pasted in the **drawable** folder of the **res** folder.

```
package com.example.myapplication;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.Toast;

public class MainActivity extends ActionBarActivity{
RadioGroup rg;
RadioButton rb;
Button b;

Protected void onCreate(Bundle savedInstanceState){
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
addListenerRadioButton();
}

private void addListenerRadioButton(){
rg =(RadioGroup)findViewByld(R.id.radioGroup);
b=(Button)findViewByld(R.id.button2);
b .setOnClickListener(newView.OnClickListener(){
@Override
public void onClick(View v){
int selected=rg.getCheckedRadioButtonId();
rb=(RadioButton)findViewByld(selected);
Toast.makeText(MainActivity.this,rb .getText(),Toast.LENGTH_LONG).show();
}
});
} }
```

- Following will be the content of **res/layout/activity_main.xml** file

```
<?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:paddingeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
tools:context=".MainActivity">

<TextView
android:id="@+id/textView "
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="ExampleofRadioButton"
android:layout_alignParentTop="true"
android:layout_centerHorizontal="true"
android:textSize="30dp"/>

<lmageButton
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/imageButton"
android:src="@drawable/abe"
android:layout_centerVertical="true"
android:layout_centerHorizontal="true"/>

<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/button2"
android:text="ClickMe"
android:layout_alignParentBottom="true"
android:layout_centerHorizontal="true"/>

<RadioGroup
android:id="@+id/radioGroup"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:layout_below="@+id/imageButton"
android:layout_aligneft="@+id/textView2"
android:layout_alignStart="@+id/textView2">
```

```
<RadioButton
andraid:layout_width="142dp"
andraid:layout_height="wrap_content"
android:text="JAVA"
android:id="@+id/radioButton"
android:textSize="25dp"
android:textColor="@android:color/holo_red_light"
android:checked="false"
android:layout_gravity="center_horizontaI"/>

<RadioButton
android:layout_width="wrap_content"
android:layout_ height="wrap_content"
android:text="ANDROID"
android:id="@+id/radioButton2"
android:layout_gravity="center_horizontaI"
android:checked="false"
android:textColor="@android:color/holo_red_dark"
android:textSize="25dp"/>

<RadioButton
android:layout_width="136dp"
andraid:layout_height="wrap_content"
android:text="HTML"
android:id="@+id/radieButton3"
android:layout_gravity="center_ horizontaI"
android:checked="false"
android:textSize="25dp"
android:textColor="@android:color/holo_red_dark"/>
</RadioGroup>
</Relativelayout>
```

- Do not make any changes to the strings.xml and AndroidManifest.xml file.
- To run the app from Android studio, open one of your project's activity files and click Run icon from the toolbar.
- Android studio 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 .
- If User selected any of a Radio Button, It should give same name on Toast message.
- Eg, if User selected JAVA, it gives a toast message as JAVA.

Fig. 1.11.8

Syllabus Topic : RadioGroup

1.11.9 RadioGroup

- A RadioGroup class is used for set of radio buttons.If we check one radio button that belongs radio group, it automatically unchecks any previously checked radio button within the same gro

☞ **Attributes of RadioGroup control**

- Following are the important attributes related to RadioGroup control

android:background	This is a drawable to use as the background.
android:contentDescription	This defines text that briefly describes content of the view.
android:id	This supplies an identifier name for this view
android:onClick	This is the name of the method in this View's context to in when the view is clicked.
android:visibility	This controls the initial visibility of the view.

- Following is the content of the modified main activity file **src/MainActivity.java**.
- In the below example abc indicates the image of Android logo which should be pasted **drawable** folder of the **res** folder.

```
package com.example.myapplication;

import android.app.Activity;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
```

```
import android.view.View;
import android.widget.Button;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.Toast;

public class MainActivity extendsActivity{
private RadioGroup radioSexGroup;
private RadioButton radioSexButton;
private Button btnDisplay;

@Override
protected void onCreate(Bundle savedlnstanceState){
super.onCreate(savedInstanceState);
setContentView(R .layout.activity_main);
radioSexGroup=(RadioGroup)findViewByld(R.id.radioGroup);
btnDisplay=(Button)findVi ewByld(R.id.button);
btnDisplay.setOnClicklistener(newView.OnClicklistener() {
@Override
public void onClick(View v){
int selectedld=radioSexGroup.getCheckedRadioButtonld();
radioSexButton=(Radi oButton)findViewByld(selectedld);
Toast.makeText(MainActivity.this,radioSexButton.getText(),Toast.LENGTH_SHORT).show();
}
});
} }
```

- Following will be the content of res/layout/activity_main.xml file

```
<?xml version="1.0"encoding="utf-8"?>
<Relative:ayout 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:paddingeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
android:paddingBottom="@dimen/activity_vertical_margin"
tools:context=" .MainActivity">

<TextView
android:layout_width="wrap_content"android :layout_height="wrap_content"
android:text="Radiobutton"
```

```
android:id="@+id/textView"
android:layout_alignParentTop="true"
android:layout_centerHorizontal="true"
android:textSize="35dp"/>

<ImageView
android:layout_width="wrap_content"
android :layout_height="wrap_content"
android :id="@+id/imageView"
android:src="@drawable/abe"
android:layout_below="@+id/textView2"
android:layout_alignleft="@+id/textView"
android:layout_alignStart="@+id/textView"
android:layout_alignRight="@+id/textView"
android:layout_alignEnd="@+id/textView"/>

<RadioGroup
android:layout_width="fill_parent"
android:layout_height="90dp"
android:layout_below="@+id/imageView"
android:layout_marginTop="58dp"
android:weightSum="1"
android :id="@+id/radioGroup"
android:layout_alignleft="@+id/textView2"
android:layout_alignStart="@+id/textView2"
android :layout_alignRight="@+id/textView3"
android:layout_alignEnd="@+id/textView3">

<RadioButton
android:layout_width="wrap_content"
android:layout_height="SSdp"
android:text="Male"
android:id="@+id/radioButton"
android:layout_gravity="center_horizontal"
android:checked="false"
android:textSize="25dp"/>
```

```
<RadioButton
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Female"
android:id="@+id/radioButton2"
android:layout_gravity="center_horizontal"
android:checked="false"
android:textSize="25dp"
android :layout_weight="0.13"/>
</RadioGroup>

<TextView
android:layout_width="wrap_content"
android :layout_height="wrap_content"
android:text="Are you?"
android:id="@+id/textView3"
android:textSize="35dp"
android:layout_below="@+id/imageView"
android :layout_alignRight="@+id/textView2"
android:layout_alignEnd="@+id/textView2"
android:layout_alignleft="@+id/imageView"
android:layout_alignStart="@+id/imageView" />

<Button
android:layout_width="wrap_content"
android :layout_height="wrap_content"
android:text="NewButton"
android:id="@+id/button"
android:layout_gravity="center_horizontaI"
android:layout_below="@+id/radioGroup"
android:layout_centerHorizontal="true"/>
</Relativelayout>
```

- Do not make any changes to the strings.xml and AndroidManifest.xml file.
- To run the app from Android studio, open one of your project's activity files and click Run icon from the toolbar.
- Android studio installs the app on your AVD and starts it and if everything is fine with your setup and application, it will display Fig. 1.11.9 Emulator window –
- Need to select male or female radio button then click on new button. if you do above steps without fail, you would get a toast message after clicked by new button.

Fig. 1.11.9

Syllabus Topic : ProgressBar

1.11.10 ProgressBar

- Progress bars are used to show progress of a task.
- For example, when you are uploading or downloading something from the internet, it is bet show the progress of download/upload to the user.
- In Android there is a class called ProgressDialog that allows you to create progress bar.
- In order to do this, you need to instantiate an object of this class. Its syntax is.

```
ProgressDialog progress = new ProgressDialog(this);
```

- Now you can set some properties of this dialog. Such as, its style, its text etc.

```
progress.setMessage("Downloading Music :) ");
progress.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
progress.setIndeterminate(true);
```

Methods provided by ProgressDialog class

- Some other methods provided by ProgressDialog class are as follows.

getMax()	This method returns the maximum value of the progr
incrementProgressBy(int diff)	This method increments the progress bar by difference of value passed as a parameter.
setIndeterminate(boolean indeterminate)	This method sets the progress indicator as determin or indeterminate.
setMax(int max)	This method sets the maximum value of the progr dialog.
setProgress(int value)	This method is used to update the progress dialog w some specific value.
show(Context context, CharSequence title, CharSequence message)	This is a static method, used to display progress dialog

1.11.10(A) Example of ProgressBar

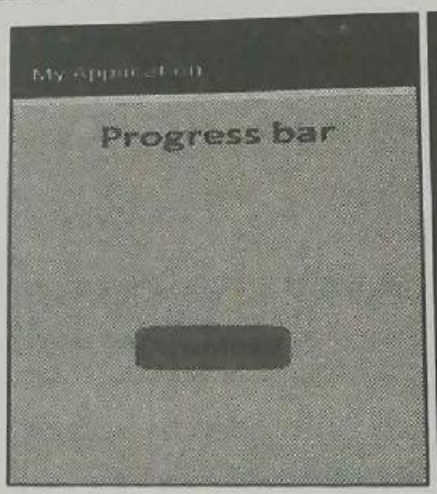
Following is the content of the modified main activity file **src/MainActivity.java**. In the below example abc indicates the image of Android logo which should be pasted in the **drawable** folder of the **res** folder.

```
package com.example.myapplication;

import android.app.ProgressDialog;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
public class MainActivity extends ActionBarActivity{
Button bl;
private ProgressDialog progress;

protected void onCreate(Bundle savedInstanceState){
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
bl=(Button)findViewByld(R.id.button2);
}

public void download(View view){
progress=newProgressDialog(this);
progress.setMessage(" Downloading Music");
progress.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
progress.setIndeterminate(true);
progress.setProgress(O);progress.show();

final int totalProgressTime=100;finalThreadt=newThread(){
@Override
public void run(){intjumpTime=O;
while(jumpTime<totalProgressTime){try{
sleep(200);jumpTime+=5;
progress.setProgress(jumpTime);
}catch(InterruptedException e){
e.printStackTrace();
} }
} };
t.start();
} }
```

- Following will be the content of res/layout/activity_main.xml file

```
<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:paddingeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
android:paddingBottom="@dimen/activity_vertical_margin"
tools:context=".MainActivity">

<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/textView"
android:layout_alignParentTop="true"
android:layout_centerHorizontal="true"
android:textSize="30dp" android:text="Progressbar"/>
<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Download"
android:onClick="download"
android:id="@+id/button2"
android:layout_marginLeft="125dp"
android:layout_marginStart="125dp"
android:layout_centerVertical="true"/>
</Relativelayout>
```

- Do not make any changes to the strings.xml and AndroidManifest.xml file.
- To run the app from Android studio, open one of your project's activity files and click Run icon from the toolbar. If your phone is attached to your computer, Android studio installs the app on your phone and starts it and if everything is fine with your setup and application, it will display following on your phone.



- Just press the button to start the Progress bar. After pressing, following screen would appear –
- It will continuously update itself.

Syllabus Topic : AutoComplete TextView

1.11.11 AutoCompleteTextView

- AutoCompleteTextView provides suggestions when you type in an editable text field. It provides suggestions automatically when the user is typing. The list of suggestions is displayed in a drop down menu from which the user can choose an item to replace the content of the edit box with.
- In order to use AutoCompleteTextView you have to first create an AutoCompletTextView Field in the xml. Its syntax is given below.

```
<AutoCompleteTextView
android:id="@+id/autoCompleteTextView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentTop="true"
android:layout_centerHorizontal="true"
android:layout_marginTop="65dp"
android:ems="10">
```

- Next we need to get a reference of this textview in java. Its syntax is given below.

```
private AutoCompleteTextView actv;
actv=(AutoCompleteTextView)
findViewByld(R.id.autoCompleteTextView1);
```

- The the next thing you need to do is to specify the list of suggestions items to be displayed. You can specify the list items as a string array in java or in strings.xml. Its syntax is given below.

```
String[] countries= getResources().getStringArray(R.array.list_of_countries);
ArrayAdapter<String> adapter=new ArrayAdapter<String>
(this,a ndroid.R.layout.simpie_list_item_1,countries);
actv.setAdapter(adapter);
```

- The array adapter class is responsible for displaying the data as list in the suggestion box of the text field. The **setAdapter** method is used to set the adapter of the autoCompleteTextView.

Methods of Auto Complete

The other methods of Auto Complete are:

getAdapter()	This method returns a filterable list adapter used for auto completion
getCompletionHint()	This method returns optional hint text displayed at the bottom of the the matching list
getDropDownAnchor()	This method returns returns the id for the view that the auto-complete drop down list is anchored to.
getListSelection()	This method returns the position of the dropdown view selection, if there is one
isPopupShowing()	This method indicates whether the popup menu is showing
setText(CharSequence text, boolean filter)	This method sets text except that it can disable filtering
showDropDown()	This method displays the drop down on screen.

1.11.11(A) Example of AutoCompleteTextView

- The below example demonstrates the use of AutoCompleteTextView class. Following is the content of the modified main activity file **src/MainActivity.java**.
- In the below example abc indicates the image of Android logo which should be pasted in the **drawable** folder of the **res** folder.

```
package com.example.myapplication;

import android.app.Activity;
import android.content.Context;

import android.media.AudioManager;
import android.media.MediaPlayer;
import android.media.MediaRecorder;

import android.os.Bundle;
import android.os.Environment;

import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.animation.Animation;
import  android.view.animation.AnimationUtiils;

import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.MultiAutoCompleteTextView;
import android.widget.Toast;
import java.io.IOException;

public class MainActivity extends Activity{AutoCompleteTextViewtext;
MultiAutoCompleteTextViewtext;
String[] Languages={"Android","java","IOS","SQL","JDBC","Webservices"};

@Override
protected void onCreate(Bundle savedInstanceState){
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
```

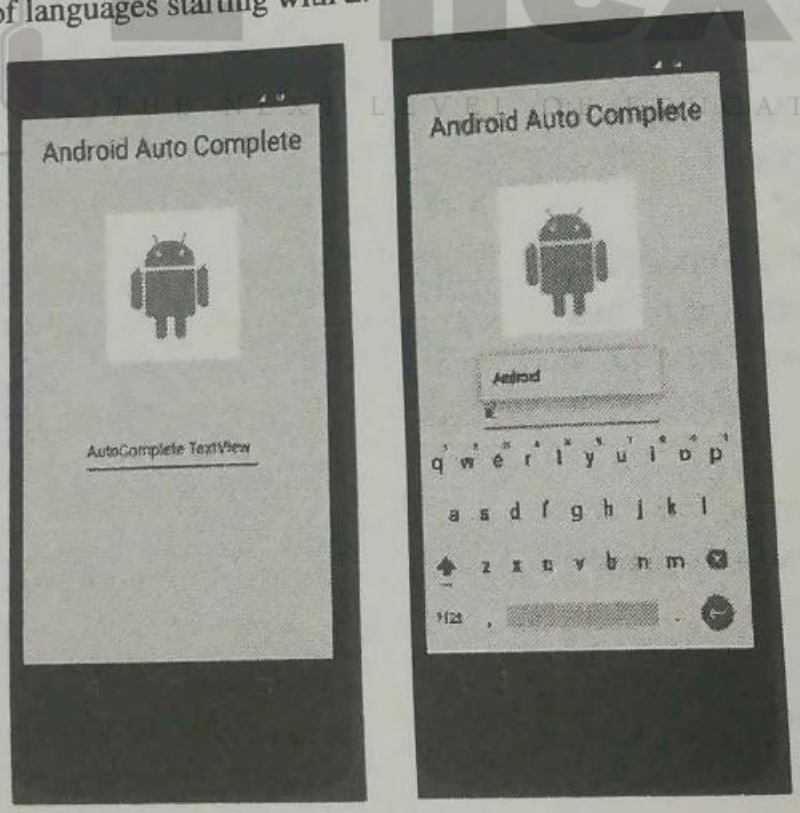
```
text=(AutoCompleteTextView)findViewByld(R.id.autoCompleteTextView);
text1=(MultiAutoCompleteTextView)findViewByld(R.id.multiAutoCompleteTextView1);

ArrayAdapter adapter=new ArrayAdapter(this,android.R.layout.simple_list_item_1,languages);

text.setAdapter(adapter);
text.setThreshold(l);
text.setAdapter(adapter);
text.setTokenizer(new  MultiAutoCompleteTextV iew.CommaTokenizer());
} }
```

- Following will be the content of res/layout/activity_main.xml file

```
<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:paddingeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
android:paddingBottom="@dimen/activity_vertical_margin"
tools:context=".MainActivity">

<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="AndroidAutoComplete"
android:id="@+id/textView"
android:textSize="30dp"
android:layout_alignParentTop="true"
android:layout_alignParentRight="true"
android:layout_alignParentEnd="true"/>

<lmageView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/imageView"
android:src="@drawable/abe"
android:layout_below="@+id/textView2"
android:layout_aligneft="@+id/textView2"
android:layout_alignStart="@+id/textView2"
android:layout_alignRight="@+id/textView2"
android:layout_alignEnd="@+id/textView2"/>
```

```
<AutoCompleteTextView
android:id="@+id/autoCompleteTextView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:ems="10"
android:layout_below="@+id/imageView"
android:layout_align eft="@+id/imageView"
android:layout_alignStart="@+id/imageView"
android:layout_marginTop="72dp"
android:hint="AutoCompleteTextView">
<requestFocus/>
</AutoCompleteTextView>
</Relative:ayout>
```

- Do not make any changes to the strings.xml and AndroidManifest.xml file.
- To run the app from Android studio, open one of your project's activity files and click Run icon from the toolbar. Android studio 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 –
- Now type in the TextView to see suggestions of the Languages. Eg. If you type one letter say, **a**, it shows suggestion of languages starting with **a**.



Syllabus Topic : TimePicker View

1.11.12 TimePicker View

- Android Time Picker allows you to select the time of day in either 24 hour or AM/PM mode. The time consists of hours, minutes and clock format. Android provides this functionality through TimePicker class.
- To be able to use TimePicker class, you first need to define the TimePicker component in the activity_main.xml file. It is defined as below –

```
<TimePicker
android:id="@+id/timePicker1"
android:layout_width="wrap_content"
android:layout_height="wrap_content" />
```

- After that you have to create an object of TimePicker class and get a reference of the above defined xml component. Its syntax is given below.

```
import android.widget.TimePicker;
private TimePicker timePicker1;
timePicker1 = (TimePicker)findViewById(R.id.timePicker1);
```

- In order to get the time selected by the user on the screen, you will use getCurrentHour() and getCurrentMinute() method of the TimePicker Class. Their syntax is given below.

```
int hour = timePicker1.getCurrentHour();
int min = timePicker1.getCurrentMinute();
```

- Apart form these methods, there are other methods in the API that gives more control over TimePicker Component. They are listed below.

is24HourView()	This method returns true if this is in 24 hour view else false
isEnabled()	This method returns the enabled status for this view
setCurrentHour(Integer currentHour)	This method sets the current hour
setCurrentMinute(Integer currentMinute)	This method sets the current minute
setEnabled(boolean enabled)	This method set the enabled state of this view
setIs24HourView(Boolean is24HourView)	This method set whether in 24 hour or AM/PM mode
setOnTimeChangedListener(TimePicker.OnTimeChangedListeneronTimeChangedListener)	This method Set the callback that indicates the time has been adjusted by the user

- Following is the content of the modified main activity file **src/MainActivity.java**.

```
package com.example.timepicker;
import java.util.Calendar;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
```

```
import android.view.View;
import android.widget.TextView;
import android.widget.TimePicker;

public class MainActivity extends Activity{privateTimePickertimePickerl;
private TextView time;
private Calendar calendar;
private String format="";

@Override
protected void onCreate(Bundle savedInstanceState){
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);

timePickerl=(TimePicker)findViewByld(R.id.timePickerl);
time=(TextView)findViewByld(R.id.textViewl);
calendar=Calendar.getInstance();


int hour=calendar.get(Calendar.HOUR_OF_DAY);
int min=calendar.get(Calendar.MINUTE);
showTime(hour,min);
}

Public void setTime(View view){
inthour=timePickerl.getCurrentHour();
intmin=timePickerl.getCurrentMinute();
showTime(hour,min);
}
public void showTime(int hour,int min){if(hour==0){
hour+=12;
format="AM";
}elseif(hour==12){format="PM";
}elseif(hour>12){
hour-=12;format="PM";
}else{
format="AM";
}
time.setText(newStringBuilder().append(hour).append(":").append(min)
.append("").append(format));
} }
```

Following is the modified content of the xml res/layout/activity_main.xml.

```
<?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=".MainActivity">

<TextView
android:id="@+id/textView2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentTop="true"
android :layout_centerHorizontal="true"
android:text="@string/time_pick"
android:textAppearance="?android:attr/textAppearanceMedium"/>

<Button
android:id="@+id/set_button"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentBottom="true"
android :layout_centerHorizontal="true"
android:layout_marginBottom="180dp"
android:onClick="setTime"
android:text="@string/time_save"/>

<TimePicker
android:id="@+id/timePickerl"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_above="@+id/set_button"
android :layout_centerHorizontal="true"
android:layout_marginBottom="24dp"/>

<TextView
android:id="@+id/textView3"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_align eft="@+id/timePickerl"
```

```
android:layout_alignTop="@+id/set_lbutton"
android:layout_marginTop="67dp"
android:text="@string/time_current"
android:textAppearance="?android:attr/textAppearanceMedium"/>

<TextView
android:id="@+id/textView "
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@+id/textView3"
android :layout_centerHorizontal="true"
android:layout_marginTop="50dp"
android:text="@string/time_selected"
android:textAppearance="?android:attr/textAppearanceMedium"/>

</RelativeLayout>
```

- Following is the content of the res/values/string.xml.

```
<?xml version="1.0"encoding="utf-8"?>
<resources>
<stringname="app_name">TimePicker</string>
<stringname="action_settings">Settings</string>
<stringname="time_picker_example">Time PickerExample</string>
<stringname="time_pick">Pick the time and press save button</string>
<stringname="time_save">Save</string>
<stringname="time_selected"></string>
<stringname="time_current">TheTimeis:</string>
</resources
```

- Do not make any changes to the AndroidManifest.xml file.
- To run the app from Android studio, open one of your project's activity files and click Run icon from the toolbar.
- If your phone is attached to your computer, Android studio installs the app on your phone and starts it and if everything is fine with your setup and application, it will display Fig. 1.11.10 on your phone

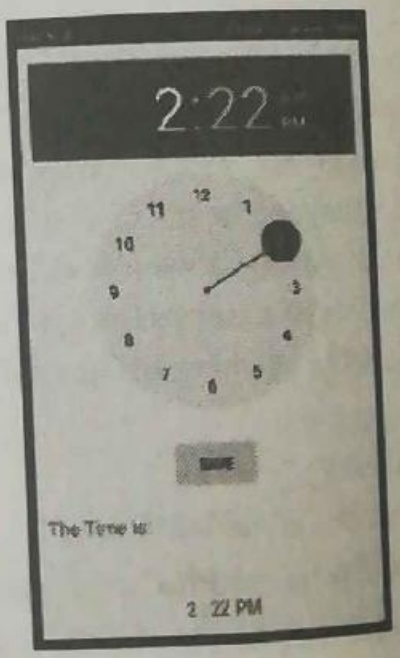


Fig. 1.11.10

Syllabus Topic : DatePicker View

1.11.13 DatePicker View

- Android Date Picker allows you to select the date consisting of day, month and year in your custom user interface. For this functionality Android provides DatePicker and DatePickerDialog components.
- To show DatePickerDialog, you have to pass the DatePickerDialog id to *showDialog(id_of_dialog)* method. Its syntax is given below

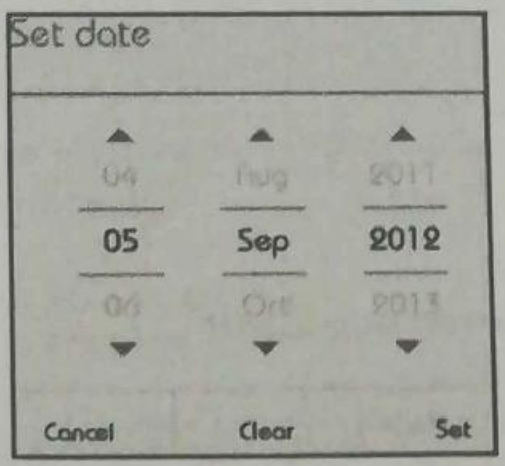
```
showDialog(999);
```

- On calling this *showDialog* method, another method called *onCreateDialog*gets automatically called. So we have to override that method too. Its syntax is given below –

```
@Override
protected Dialog onCreateDialog(int id) {
  // TODO Auto-generated method stub
if (id == 999) {
return new DatePickerDialog(this, myDateListener, year, month, day);
  }
return null;
}
```

- In the last step, you have to register the DatePickerDialog listener and override its onDateSet() method. This onDateSet() method contains the updated day, month and year. Its syntax is given below –

```
privateDatePickerDialog.OnDateSetListenermyDateListener = new
DatePickerDialog.OnDateSetListener() {
  @Override
public void onDateSet(DatePicker arg0, int arg1, int arg2, int arg3) {
   // arg1 = year
   // arg2 = month
   // arg3 = day
  }
};
```



- Apart form date attributes, DatePicker object is also passed into this function. You can use the following methods of the DatePicker to perform further operation.

getDayOfMonth()	This method gets the selected day of month
getMonth()	This method gets the selected month
getYear()	This method gets the selected year
setMaxDate(long maxDate)	This method sets the maximal date supported by this DatePicker in milliseconds since January 1, 1970 00:00:00 in getDefault() time zone
setMinDate(long minDate)	This method sets the minimal date supported by this NumberPicker in milliseconds since January 1, 1970 00:00:00 in getDefault() time zone
setSpinnersShown(boolean shown)	This method sets whether the spinners are shown
updateDate(int year, int month, intdayOfMonth)	This method updates the current date
getCalendarView()	This method returns calendar view
getFirstDayOfWeek()	This Method returns first day of the week

- Following is the content of the modified main activity file **src/com.example.datepicker/MainActivity.java**.

```
package com.example.datepicker;
import java.util.Calendar;
import android.app.Activity;
import android.app.DatePickerDialog;
import android.app.Dialog;

import android.os.Bundle;

import android.view.Menu;
import android.view.View;
import android.widget.DatePicker;
import  android.widget.TextView;
import android.widget.Toast;

public class MainActivity extendsActivity{
private DatePicker datePicker;
private Calendar calendar;
private TextView dateView;
private int year,month,day;

@Override
protected void onCreate(BundlesavedInstanceState){
super.onCreate(savedInstaneeState);
setContentView(R .layout.activity_main);
dateView=(TextView)findViewByld(R.id.textView3);
```

```
calendar=Calendar.getInstance();
year=calendar.get(Calendar.YEAR);

month=calendar.get(Calendar.MONTH);
day=calendar.get(Calendar:DAY_OF_MONTH);
showDate(year, month+l, day);

}
@Suppress Warnings("deprecation")public void setDate(View view){
showDialog(999);
Toast.makeText(getApplicationContext(),"ca",Toast.LENGTH_SHORT).show();

}
@Override
protected Dialog onCreateDialog(intid){//TODO Auto-generated method stub
if(id==999){
return new DatePickerDialog(this, myDatelistener, year, month, day);

}
return null;
}
private DatePickerDialog.OnDateSetListener myDatelistener=new
DatePickerDialog.OnDateSetlistener()
{
@Override
public void onDateSet(DatePickerarg0,int arg ,int arg2,int arg3){
//TODOAuto-generatedmethodstub
//1arg1=year
//arg2=monthllarg3=day
showDate(arg,arg2+1,arg3);
}
};
private void showDate(int year,int month,int day){
dateView.setText(new
StringBuilder().append(day).append("I").append(month).append("I").append(year));
}
}
```

- Following is the modified content of the xml res/layout/activity_main.xml.

```
<?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=".MainActivity">

<Button
android:id="@+id/button 1"
android:layout_width="wrap_content"
android :layout_height="wrap_content"
android:layout_alignParentTop="true"
android:layout_centerHorizontal="true"
android:layout_marginTop="70dp"
android:onClick="setDate"
android:text="@string/date_button_set"/>

<TextView
android:id="@+id/textViewl"
android:layout_width="wrap_content"
android :layout_height="wrap_content"
android:layout_alignParentTop="true"
android:layout_centerHorizontal="true"
android:layout_marginTop="24dp"
android:text="@string/date_labeI_set"
android:textAppearance="?android:attr/textAppearanceMedium"/>

<TextView
android:id="@+id/textView2"
android:layout_width="wrap_content"
android :layout_height="wrap_content"
android:layout_below="@+id/buttonl"
android:layout_marginTop="66dp"
android:layout_toLeftOf="@+id/buttonl"
android:text="@string/date_view_set"
```

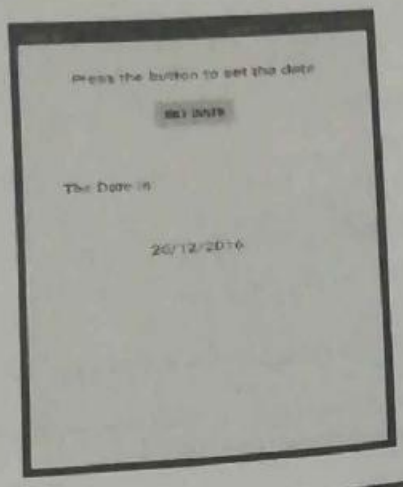
```
android:textAppearance="?android:attr/textAppearanceMedium"/>

<TextView
android:id="@+id/textView3"
android:layout_width="wrap_content"
android :layout_height="wrap_content"
android:layout_alignRight="@+id/button1"
android:layout_below="@+id/textView2"
android:layout_marginTop="72dp"
android:text="@string/date_selected"
android:textAppearance="?android:attr/textAppearanceMedium"/>
</RelativeLayout>
```

- Following is the content of the res/values/string.xml.

```
<?xml version="1.0"encoding="utf-8"?>
<resources>
<string name="app_name">DatePicker</string>
<string name="action_settings">Settings</string>
<string name="hello_world">Helloworld!</string>
<string name="date_label_set">Pressthe button to se tthe date</string>
<string name="date_button_set">SetDate</string>
<string name=;;;"date_view_set">TheDateis:</string>
<string name="date_selected"></string>
</resources>
```

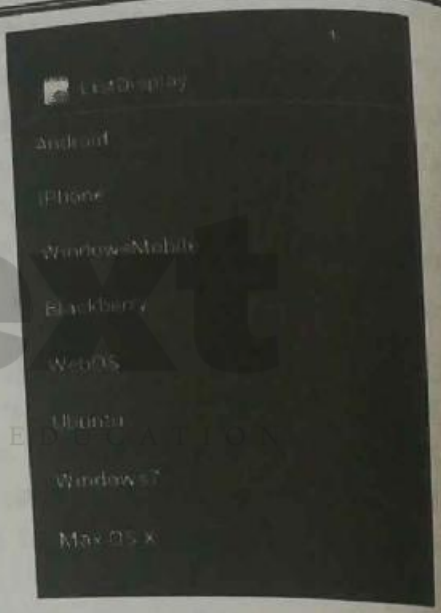
- Do not make any changes to the AndroidManifest.xml file.
- To run the app from Android studio, open one of your project's activity files and click Run icon from the toolbar. If your phone is attached to your computer, Android studio installs the app on your phone and starts it and if everything is fine with your setup and application, it will display following on your phone.
- The date has already been set at the bottom label. To change the date through DatePickerDialog pressing the Set Date button. On pressing the button following screen would appear.

- Now set the required date, and after setting the date, press the Done button. This dialog will disappear and your newly setted date will start showing at the screen.



Syllabus Topic: ListView View

1.11.14 ListView View



- Android **ListView** is a view which groups several items and display them in vertical scrollable list. The list items are automatically inserted to the list using an **Adapter** that pulls content from a source such as an array or database.
- The **ListView** is a subclass of **AdapterView**. A ListView can be populated by binding it to an Adapter.
- An **Adapter** retrieves data from an external source and creates a View that represents each data entry.
- Android provides several subclasses of Adapter that are useful for retrieving different kinds of data and building views for an AdapterView(i.e. ListView or GridView). The common adapters are **ArrayAdapter**,**BaseAdapter**,**CursorAdapter**,**SimpleCursorAdapter**,**SpinnerAdapter** and **WrapperListAdapter**.

☞ **ListView Attributes**

- Following are the important attributes specific to GridView.

android:id	This is the ID which uniquely identifies the layout.
android:divider	This is drawable or color to draw between list items.
android:dividerHeight	This specifies height of the divider. This could be in px, dp, sp, in or mm.
android:entries	Specifies the reference to an array resource that will populate the ListView.

android:footerDividersEnabled	When set to false, the ListView will not draw the divider before each footer view. The default value is true.
android:headerDividersEnabled	When set to false, the ListView will not draw the divider after each header view. The default value is true.

1.11.14(A) ArrayAdapter

- An adapter is like a bridge, or intermediary, between two incompatible interfaces. For example, a memory card reader acts as an adapter between the memory card and a laptop.
- Array adapter can be used when your data source is an array. By default, ArrayAdapter creates a view for each array item by calling toString() on each item and placing the contents in a **TextView**.
- Consider you have an array of strings you want to display in a ListView, initialize a new **ArrayAdapter** using a constructor to specify the layout for each string and the string array –

```
ArrayAdapter adapter = new ArrayAdapter<String>(this, R.layout.ListView, StringArray);
```

- Here are arguments for this constructor –
  - First argument **this** is the application context. Most of the case, keep it **this**.
  - Second argument will be layout defined in XML file and having **TextView** for each string in the array.
  - Final argument is an array of strings which will be populated in the text view.
- Once the array adapter is created, call **setAdapter()** on **ListView** object as follows –

```
ListViewlistView = (ListView) findViewById(R.id.listview);
listView.setAdapter(adapter);
```

- Define list view under res/layout directory in an XML file.
- Following is the content of the modified main_activityfile :

```
package com.example.ListDisplay;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import  android.widget.ArrayAdapter;
import android.widget.ListView;
public class ListDisplay extends Activity{
//Array of strings...
String[] mobileArray= {"Android","IPhone","WindowsMobile","Blackberry",
"WebOS","Ubuntu","Windows7","MaxOSX"};

@Override
protected void onCreate(Bundle savedInstanceState){
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);

ArrayAdapter adapter=new ArrayAdapter<String>(this,R.layout.activity_listview,mobileArray);
```

```
ListView listView=(ListView)findViewByld(R.id.mobile_list);
listView.setAdapter(adapter);
} }
```

- Following will be the content of res/layout/activity_main.xml file

```
<LinearLayout 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:orientation="vertical"tools:context=".ListActivity">
<ListView
android:id="@+id/mobile_list"
android:layout_width="match_parent"
android:layout_height="wrap_content">
</ListView>
</LinearLayout>
```

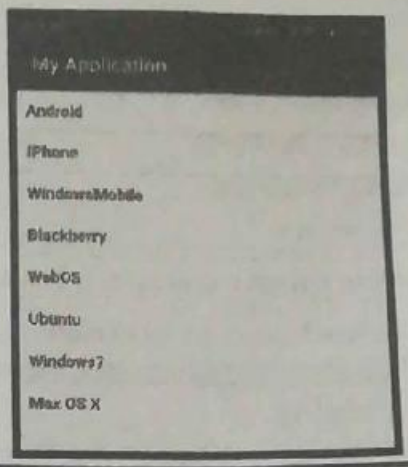
- Following will be the content of res/values/strings.xml to define two new constants

```
<?xmlversion="1.0"encoding="utf-8"?>
<resources>
<stringname="app_name">ListDisplay</string>
<string name="action_settings">Settings</string>
</resources>
```

- Create a Text View file *res/layout/activity_listview.xml*. This file will have setting to display all list items.So you can customize its fonts, padding, color etc. using this file.Following will be content of **res/layout/activity_listview.xml** file

```
<?xmlversion="1.0"encoding="utf-8"?>
<!—Single List Item Design-->

<TextViewxmlns:android="http:/schemas.android.com/apk/res/android"
android:id="@+id/label"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:padding="10dip"
android:textSize="16dip"
android:textStyle="bold">
</TextView>
```

- Do not make any changes to the AndroidManifest.xml file.
- To run the app from Android studio, open one of your project's activity files and click Run icon from the toolbar.
- If your phone is attached to your computer, Android studio installs the app on your phone and starts it and if everything is fine with your setup and application, it will display following on your phone



Syllabus Topic: Spinner View

1.11.15 Spinner View

- Spinner allows you to select an item from a drop down menu.
- Spinners are a quick way to select value from a set. It provides a Drop-down list showing all values, user can select only one. Spinners scroll automatically if necessary

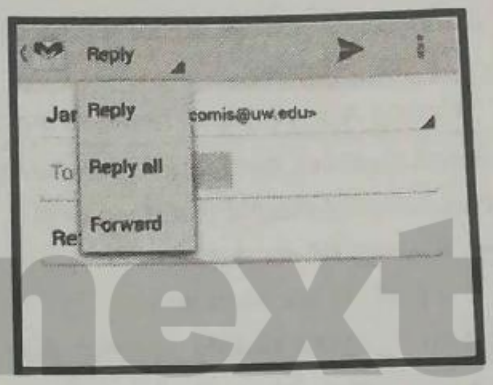


Fig. 1.11.11

- For example. When you are using any mailing application you would get drop down menu as shown in Fig. 1.11.11, you need to select an item from a drop down menu.

Implementing a spinner

1. Create Spinner UI element in the XML layout
2. Define spinner choices in an array
3. Create Spinner and set onItemSelectedListener
4. Create an adapter with default spinner layouts
5. Attach the adapter to the spinner
6. Implement onItemSelectedListener method

Creating Spinner UI in layout XML file

```
<Spinner
android:id="@+id/label_spinner"
android:layout_width="wrap_content"
android:layout_height="wrap_content">
</Spinner>
```

Define array of spinner choices

In arrays.xml resource file

```
<string-array name="labels_array">
<item>Home</item>
<item>Work</item>
<item>Mobile</item>
<item>Other</item>
</string-array>
```

☞ **Create spinner and attach listener.**

```
public class MainActivity extends AppCompatActivity implements
AdapterView.OnItemSelectedListener
// In onCreate()
Spinner spinner = (Spinner) findViewByld(R.id.label_spinner);
if (spinner != null) {
spinner.setOnItemSelectedListener(this);
}
```

☞ **Create adapter**

– Create ArrayAdapter using string arrayand default spinner layout

```
ArrayAdapter<CharSequence> adapter =
ArrayAdapter.createFromResource(
this, R.array.labels_array,
    // Layout for each item
android.R.layout.simple_spinner_item);
```

☞ **Attach the adapter to the spinner**

– Specify the layout for the drop down menu

```
adapter.setDropDownViewResource(
android.R.layout.simple_spinner_dropdown_item);
```

– Attach the adapter to the spinner

```
spinner.setAdapter(adapter);
```

☞ **Implement onItemSelectedListener**

```
public class MainActivity extends AppCompatActivity
implementsAdapterView.OnItemSelectedListener
public void onItemSelected(AdapterView<?>adapterView,
  View view, intpos, long id) {
    String spinner_item =
adapterView.getItemAtPosition(pos).toString();
    // Do something here with the item
}
public void onNothingSelected(AdapterView<?>adapterView) {
  // Do something
}
```

Following is the content of the modified main activity file **src/com.example.spinner/MainActivity.java.**

```
package com.example.spinner;

import java.util.Array ist;
import java.util.List;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.Toast;
import android.widget.AdapterView.OnItemSelectedListener;

class AndroidSpinnerExampleActivity extends Activity implements OnItemSelectedistener
{
@Override
public void  onCreate(Bundle savedInstanceState){
super.onCreate(savedInstanceState);
setContentView(R.layout.main);

//Spinner element
Spinner spinner=(Spinner)findViewByld(R.id.spinner);

//Spinner click listener spinner.setOnItemSelectedlistener(this);
//Spinner Dropdown elements
List<String> categories=new ArrayList<String>();
categories.add("Automobile");
categories.add("BusinessServices");
categories.add("Computers");
categories.add("Education");
categories.add("Personal");
categories.add("Travel");
//Creating adapter for spinner
ArrayAdapter<String> dataAdapter=new ArrayAdapter<String>
(this,android.R.layout.simple_spinner_item,categories);

//Dropdown layoutstyle-list view with radio button
dataAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
```

```
//attaching data dapter to spinner spinner.setAdapter(dataAdapter);
}

@Override
public void onItemSelected(AdapterView<?> parent,View view,int position,long id){
//On selecting a spinner item
String item=parent.getItemAtPosition(position).toString();

//Showing selected spinner item
Toast.makeText(parent.getContext(),"Selected:"+item,Toast.LENGTH_LONG).show();
}
public void onNothingSelected(AdapterView<?>arg0){
//TODO Auto-generated method stub
}
}
```

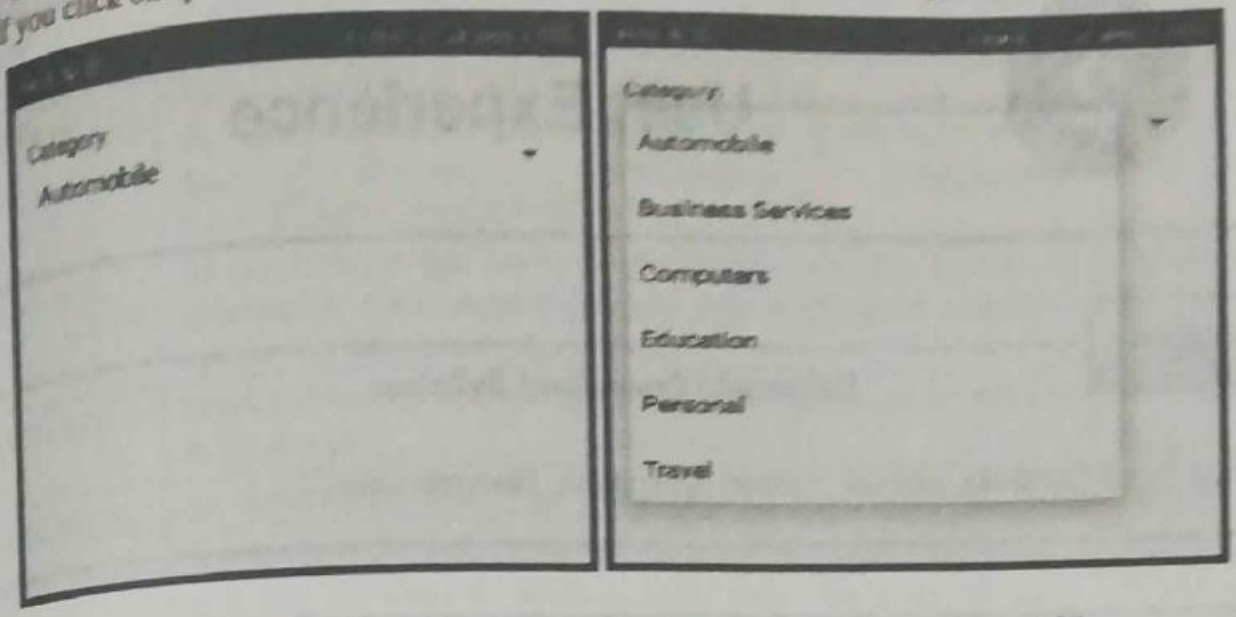
- Modify the content of **res/layout/activity_main.xml** to the following

```
<?xml version="1.0"encoding="utf-8"?>
<LinearLayout
xmlns:android="http:/schemas.android.com/apk/res/android"
android:orientation="vertical"
android:padding="10dip"
android:layout_width="fill_parent"
android:layout_height="wrap_content">

<TextView
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:layout_marginTop="10dp"
android:text="Category:"
android:layout_marginBottom="5dp"/>
<Spinner
android:id="@+id/spinner"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:prompt="@string/spinner_title"/>
</LinearLayout>
```

- Do not make any changes to the strings.xml and AndroidManifest.xml file.
- To run the app from Android studio, open one of your project's activity files and click Run from the toolbar. If your phone is attached to your computer, Android studio installs the app

your phone and starts it and if everything is fine with your setup and application, it will display Fig. 1.11.12(a) on your phone

If you click on spinner button, It will a drop down menu as shown in Fig. 1.11.12(b).



(a) (b)

Fig. 1.11.12

Review Questions

Q. 1 What is Android? **(Refer section 1.1)**

Q. 2 List Android features. **(Refer section 1.1.1)**

Q. 3 Write short note on Dalvik Virtual Machine(DVM). **(Refer section 1.1.3)**

Q. 4 How to create android app? **(Refer section 1.4)**

Q. 5 List and explain adapting display orientation. **(Refer section 1.5)**

Q. 6 Explain concept of activity in Android. **(Refer section 1.7)**

Q. 7 Describe activity lifecycle with help of diagram. **(Refer sections 1.7.2, 1.7.3 and 1.7.4)**

Q. 8 Define intents with its types. **(Refer sections 1.8, 1.8.1 and 1.8.2)**

Q. 9 List android layout types and its attributes. **(Refer sections 1.9.1 and 1.9.2)**

Q. 10 What is saving state? **(Refer section 1.10)**

Q. 11 How to use ImageButton in Android? **(Refer sections 1.11.5 and 1.11.5(A))**

Q. 12 How to use CheckBox in Android? **(Refer sections 1.11.6 and 1.11.6(A))**

Q. 13 Explain how to implement ProgressBar in Android app? **(Refer sections 1.11.10 and 1.11.10(A))**

□□□