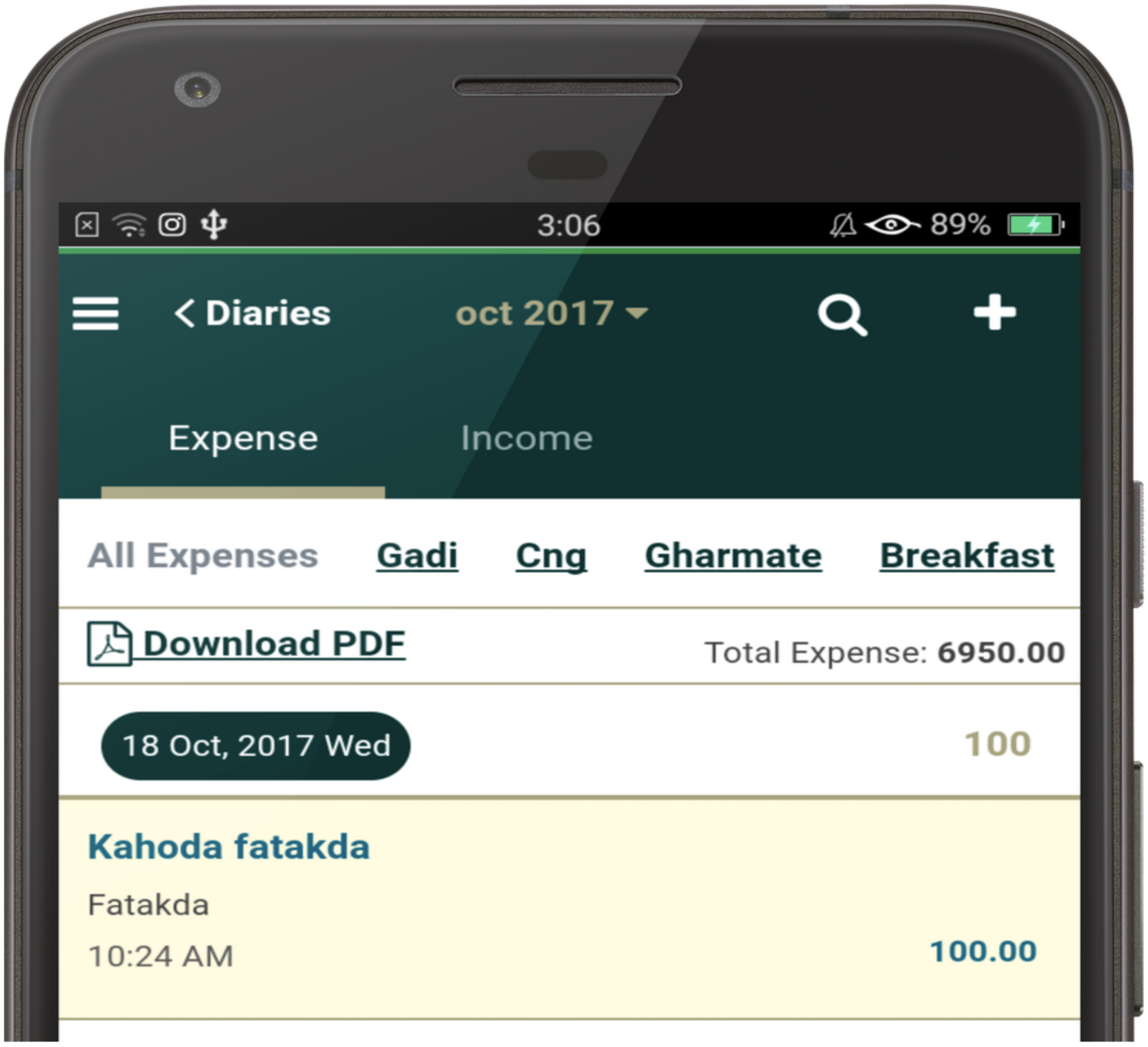
ANDROID APP DEVELOPMENT GUIDE

By AhesanAli Suthar



A saralhisab publication

Android App Development Guide

Ahesanali Suthar

This book is for sale at http://leanpub.com/android-book

This version was published on 2020-10-04

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

© 2020 Ahesanali Suthar

Tweet This Book!

Please help Ahesanali Suthar by spreading the word about this book on Twitter!

The suggested hashtag for this book is #android-book.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

#android-book

Contents

Preface

In the era of the mobility, accessing apps on mobile for day to day operation are very common. Software development drastically move towards mobile application development for each enterprise software. With this high demand of mobile apps in the market creates new opportunity for software developers to come up with new ideas. In mobile domain two major platforms plays important role in the market. One is android and second one is iOS. Being open source android becomes very popular in short amount of time. We are talking about android app development in this book.

This book will help you to learn android app development from setting up development environment to bare minimal android app. You will learn the topics which are required for the day to day life of android app developer. We have organise the book in a way to explain each concept practically, so added one small example after each concept. You can consider this book for you practical training. Book covers topics ranging from basic to advance concepts like list view, tabbed and navigation drawers. We have also covered api implementation using async client http library. Storing data in the device using different techniques like stored preferences and sqlite database is also interesting. Most of the examples in this book are in java. It is expected that you know some java programming.

You can download the source code of this book from github.

Source Code: Github Repositray[1]

[1]https://github.com/ahesanali/aadg

About Author

AhesanAli Suthar is the Co. Foundar of saralhisab.com[2] a digital diary where you can track your daily income/expense transactions.

SaralHisab is a company which develops the apps to facilitate users which digitalise their experience. Till 2019 saralhisab has published 3 apps which are available on google play store. You can visit saralhisab home page in Google play store by visiting this link[3]

Ahesanali Suthar has received Bachelor of Computer Engineering in 2010 from Sardar Patel University , V.V.Nagar, Anand. After completing graduation ahesanali has joined Sibridge Technologies Pvt. Ltd where he has to build GUI for embedded systems for 5.6 years. After leaving Sibridge Technologies Pvt. Ltd AhesanAli has joined Inspire Software Co. where he has to manage software products for hospitality industry.

Ahesanali has very good expertise on PHP, Javascript , Mysql based web applications. He also possess sound knowledge of android application development. He also posses project management skills.

[2]https://saralhisab.com
[3]https://play.google.com/store/apps/dev?id=8018112137376644406

Acknowledgments

Before this book the material was part of ppt which i have arranged at Maktabah Jafariyah Knowledge and Research Academy based on formal talk with friend of mine.
Here is the link of those slides.

1. Presentation-1:[4]
2. Presentation-2:[5]
3. Presentation-3:[6]

Although i couldn't cover all three sessions but the experience was good. Students also find app development subject interesting and somewhat new compare to syllabus.

This academy arrange trade fair every year in January end. One day i was in trade fair with my friend and he was asking that "can you write a book?" . I just stopped and replied that "yes i like, but on which topic i should write i can't decide". After some time i just realised that why i can't use those slides and put updated and well organised content in the book.

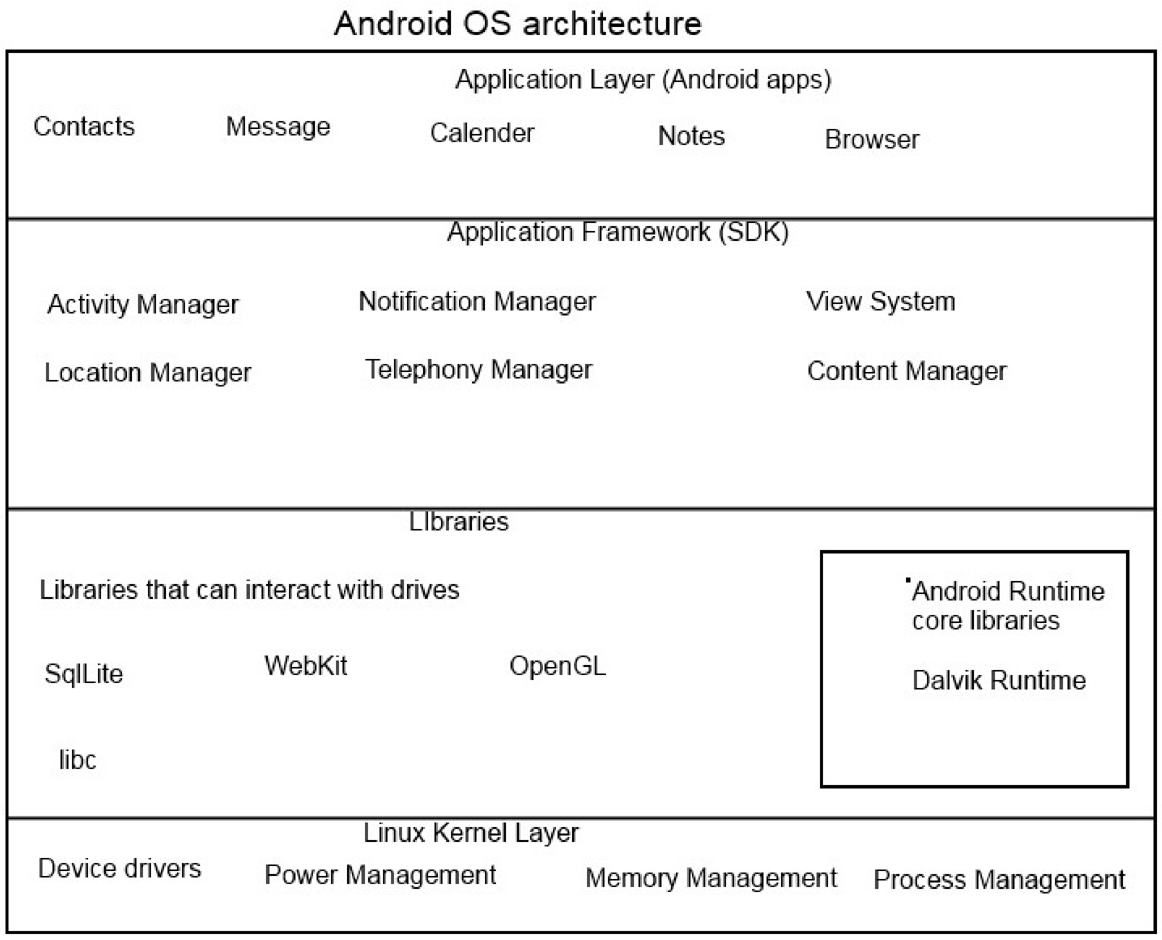
I just plan the index in apple notes and start writing chapter based on index i plan. Day by day chapters were completed and this book comes in picture.

I have tried to explain the concepts based on my experience and time available to me. Examples illustrated in this book are developed by me and meant to demonstrate the relevant concept.
If you have any queries or questions regarding this book you can write an email me on ahesanali.suthar@gmail.com .

[4]https://www.slideshare.net/AhesanaliMomin/android-session-1
[5]https://www.slideshare.net/AhesanaliMomin/android-session-2
[6]https://www.slideshare.net/AhesanaliMomin/android-session-3

1-Introduction

Introduction of android



1.1 Android Operating System

Android is an open source operating system based on Linux kernel specifically designed for smart phones and tablets. The android based mobile phone if you have contains android operating system . Android OS have below main layers. Most of users interacting through app layer and we are also talking about app in this book.

We are not talking much about operating system in this book and i think for a beginner it is not recommended but sure later on you should learn more about android operating system so that you can understand platform very well.

1.2 Linux overview

Linux is an open source operating system that can run on multiple hardware. Linux source code is written in C and freely available on INTERNET. You can build linux image for your specific hardware based on your requirements. Linux is a multi users, multi threaded operating system and have widely adopted in embedded world. Due to it’s maturity and popularity linux kernel is adopted for android operating system base. Though android has much more than on top of core linux.
Most of features required for operating system like process management, memory management, disk management are provided by Linux kernel.

1.3 Android application development building blocks

There are four major building block for android app which are mentioned below:

1.3.1 Activity:

Activity is java class provided by android software development kit where all user interface relies. User’s interaction in app is mainly handled by Activity. An Activity is an application component that provides a screen with which users can interact in order to do something, such as dial the phone, take a photo, send an email, or view a map. Activity handle events in short for a windows .NET programer Activity is a code that is generated when we place button in form and double click on it and Visual Studio IDE generate button1_click() method or function. Activity decides which Screen(in android terminology it’s call View) to be displayed. We are designing the screen in xml format as we seen in Hello word case we have main.xml which is view for hello word activity.

1.3.2 Service:

Service component doesn’t have user interface but service can run independently in background without user interface. Some services can run even your app is not running. So most of the time service will perform background task like network call, playing music. Service is executing in background and provide result to activity so activity updates the user-interface.

1.3.3 Broadcast Receiver:

This is an interesting and very important building block of android app development and i personally likes it. Broadcast Receiver is a callback receiver for a broadcasts that happen in the system. Like if sms received by system you app should perform specific task. Normally broadcast receiver have less time for execution so we should not have to do processing much in broadcast receiver instead we should have to start service and have more processing in service code.

1.3.4 Content Providers:

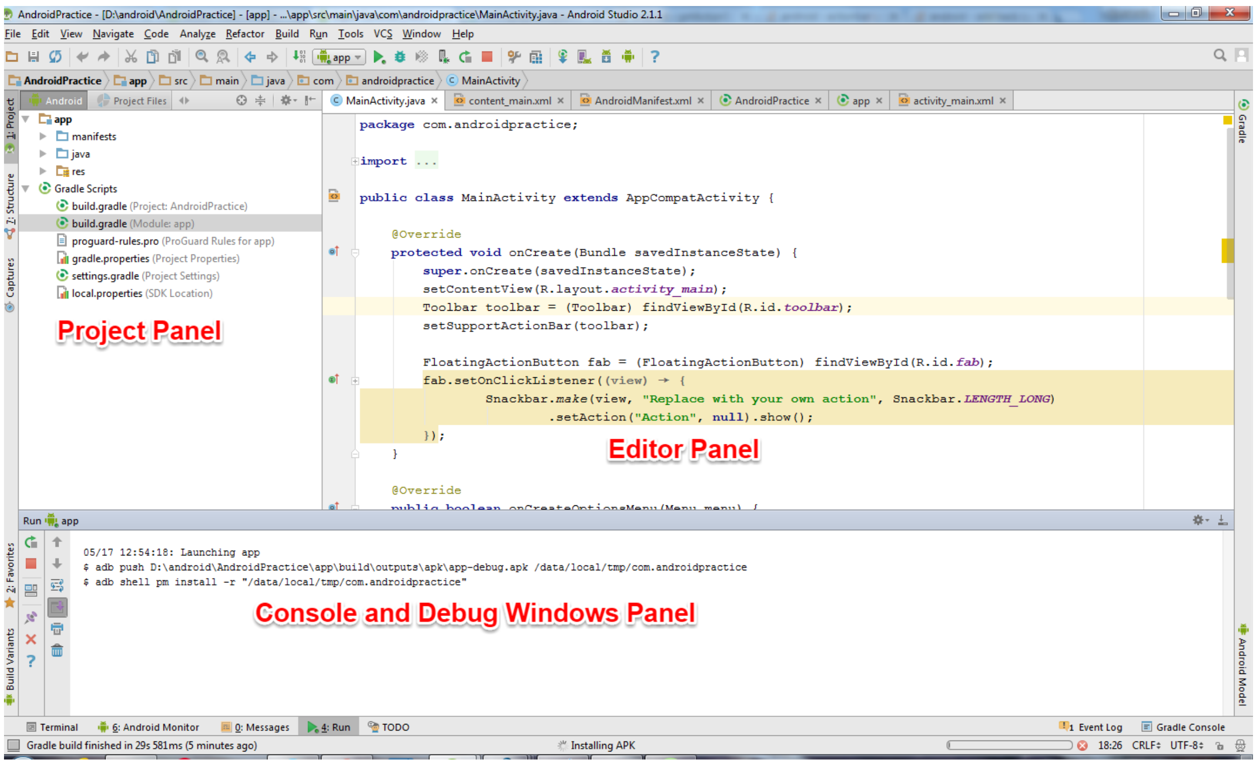
Content Providers are used for data operations like storing and retrieving the data.Content providers manage access to a structured set of data. They encapsulate the data, and provide mechanisms for defining data security. Content providers are the standard interface(not user interface but programming interface) that connects data in one process with code running in another process. In short content provider are more or less same like class any driver which we are normally using database connection.

2-Tools

Android Studio IDE and SDK

2.1 Android studio ide

To develop android app we need Android Studio IDE. Android studio can be downloaded from this link for your suitable platform. Android studio is very power IDE for building android apps. It provides awesome intelisence while writing code. Nice search utilities.


AndroidPractice - [D:\android\AndroidPractice] - [app] - ...\app\src\main\java\com\androidpractice\MainActivity.java - Android Studio 2.1.1
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
AndroidPractice app src main java com androidpractice MainActivity
Android Project Files
app
manifests
java
res
Gradle Scripts
build.gradle (Project: AndroidPractice)
build.gradle (Module: app)
proguard-rules.pro (ProGuard Rules for app)
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)
Project Panel
MainActivity.java content_main.xml AndroidManifest.xml AndroidPractice app activity_main.xml
Editor Panel
Run app
Console and Debug Windows Panel
Terminal 6: Android Monitor 0: Messages 4: Run TODO
Event Log Gradle Console
Gradle build finished in 29s 581ms (5 minutes ago)
Installing APK
18:26 CRLF UTF-8

```
package com.androidpractice;

import ...

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
        fab.setOnClickListener((view) → {
                Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
        });
    }

    @Override
```

```
05/17 12:54:18: Launching app
$ adb push D:\android\AndroidPractice\app\build\outputs\apk\app-debug.apk /data/local/tmp/com.androidpractice
$ adb shell pm install -r "/data/local/tmp/com.androidpractice"
```

Android studio have below major section.

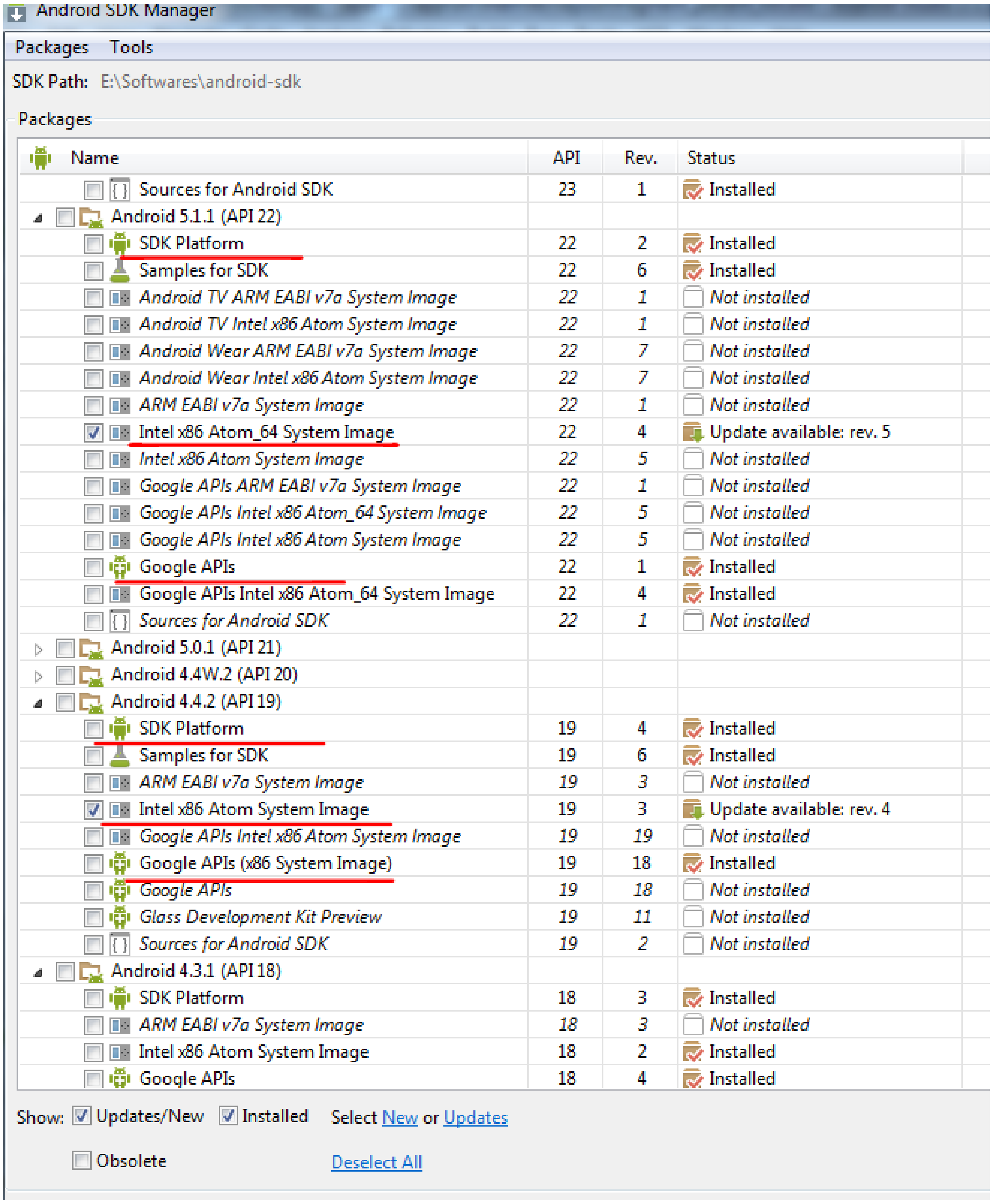
Project Panel: Here we can see project overview, project directory structure and files.

Editor Panel: This section contains files which we have open to write code. Here we have to write code.

Console and Other Windows: Most probably this section of android studio have output debug prints and some times it will show error longs.

2.2 Android sdk introduction and download tips

Android SDK is the library which we will use to compile android app. Android SDK also provides the classes which are helpful to build android app. Most probably android sdk will be downloaded and installed along with Android Studio IDE , but if you do not have android sdk and just android studio you can also download android sdk separately too.
When ever we open sdk manager it offers numbers of things to download but for any API Level we have to at least download below things which are necessary to build app with that Api and run emulator with same api level.



SDK Path: E:\Softwares\android-sdk

Packages

Name	API	Rev.	Status
☐ Sources for Android SDK	23	1	Installed
☐ Android 5.1.1 (API 22)			
☐ SDK Platform	22	2	Installed
☐ Samples for SDK	22	6	Installed
☐ *Android TV ARM EABI v7a System Image*	*22*	*1*	*Not installed*
☐ *Android TV Intel x86 Atom System Image*	*22*	*1*	*Not installed*
☐ *Android Wear ARM EABI v7a System Image*	*22*	*7*	*Not installed*
☐ *Android Wear Intel x86 Atom System Image*	*22*	*7*	*Not installed*
☐ *ARM EABI v7a System Image*	*22*	*1*	*Not installed*
☑ Intel x86 Atom_64 System Image	22	4	Update available: rev. 5
☐ *Intel x86 Atom System Image*	*22*	*5*	*Not installed*
☐ *Google APIs ARM EABI v7a System Image*	*22*	*1*	*Not installed*
☐ *Google APIs Intel x86 Atom_64 System Image*	*22*	*5*	*Not installed*
☐ *Google APIs Intel x86 Atom System Image*	*22*	*5*	*Not installed*
☐ Google APIs	22	1	Installed
☐ Google APIs Intel x86 Atom_64 System Image	22	4	Installed
☐ *Sources for Android SDK*	*22*	*1*	*Not installed*
☐ Android 5.0.1 (API 21)			
☐ Android 4.4W.2 (API 20)			
☐ Android 4.4.2 (API 19)			
☐ SDK Platform	19	4	Installed
☐ Samples for SDK	19	6	Installed
☐ *ARM EABI v7a System Image*	*19*	*3*	*Not installed*
☑ Intel x86 Atom System Image	19	3	Update available: rev. 4
☐ *Google APIs Intel x86 Atom System Image*	*19*	*19*	*Not installed*
☐ Google APIs (x86 System Image)	19	18	Installed
☐ *Google APIs*	*19*	*18*	*Not installed*
☐ *Glass Development Kit Preview*	*19*	*11*	*Not installed*
☐ *Sources for Android SDK*	*19*	*2*	*Not installed*
☐ Android 4.3.1 (API 18)			
☐ SDK Platform	18	3	Installed
☐ *ARM EABI v7a System Image*	*18*	*3*	*Not installed*
☐ Intel x86 Atom System Image	18	2	Installed
☐ Google APIs	18	4	Installed

Show: ☑ Updates/New ☑ Installed Select New or Updates

☐ Obsolete Deselect All

API LEVEL: Api level is different technical name of android version which is used in technical community.

API 8 Android 2.2,2.2.3 Froyo

API 10 Android 2.3-2.3.7 Gingerbread

API 15 Android 4.0-4.0.4 Ice-cream Sandwich

API 16,17,18 Android 4.1-4.3.1 Jelly Bean

API 19 Android 4.4-4.4.4 KitKat

API 20,21 Android 5.0-5.1 Lollipop

API 23 Android 6.0-6.0.1 Marshmallow

Reference WikiPedia[7]

For Any API level you must have to download below mentioned items

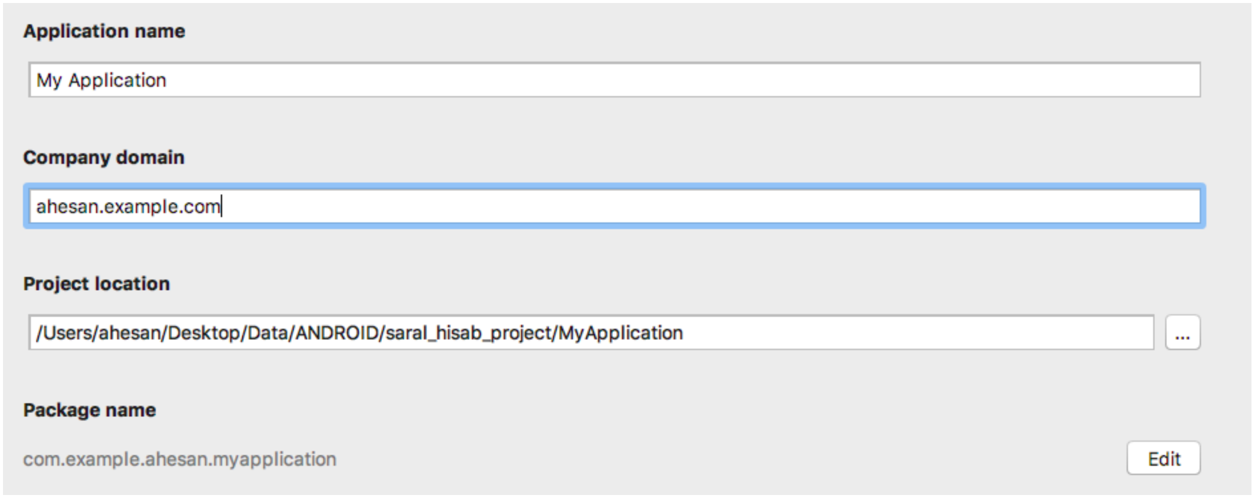
- SDK Platform
- According to your system architecture System Image (This will help to create emulator screen)

You can see from above image for API Level 19 which is android 4.4.2 have SDK Platform, Intel x86 Atom System Image, Google APIs(x86 System Image) installed.

You can also download other items from sdk manager based on your requirements but above list is just to start with app development.

2.3 Creating new project

To create new project you have to click File > New > New Project. You will see below window.



Application name

My Application

Company domain

ahesan.example.com

Project location

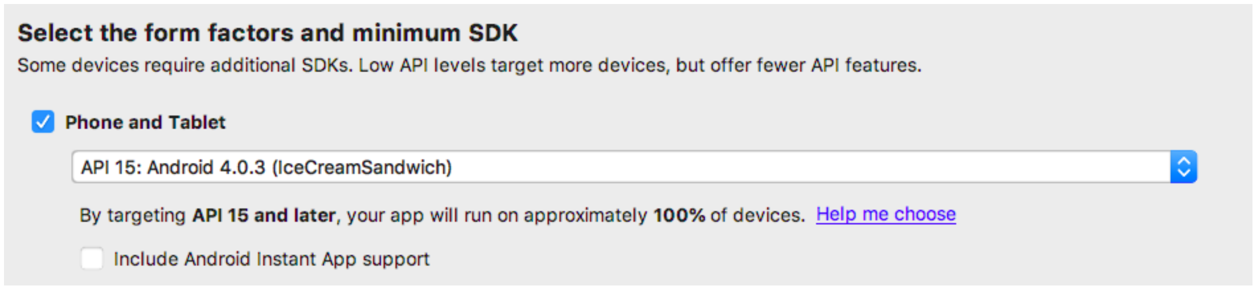
/Users/ahesan/Desktop/Data/ANDROID/saral_hisab_project/MyApplication

Package name

com.example.ahesan.myapplication

[7]https://en.wikipedia.org/wiki/Android_(operating_system)#Platform_usage

Enter application name and company domain name package name will be automatically populated. If you wish to change package name you can click on edit and change package name. Then click on Next Button you will see SDK selection window as seen below.



Select the form factors and minimum SDK

Some devices require additional SDKs. Low API levels target more devices, but offer fewer API features.

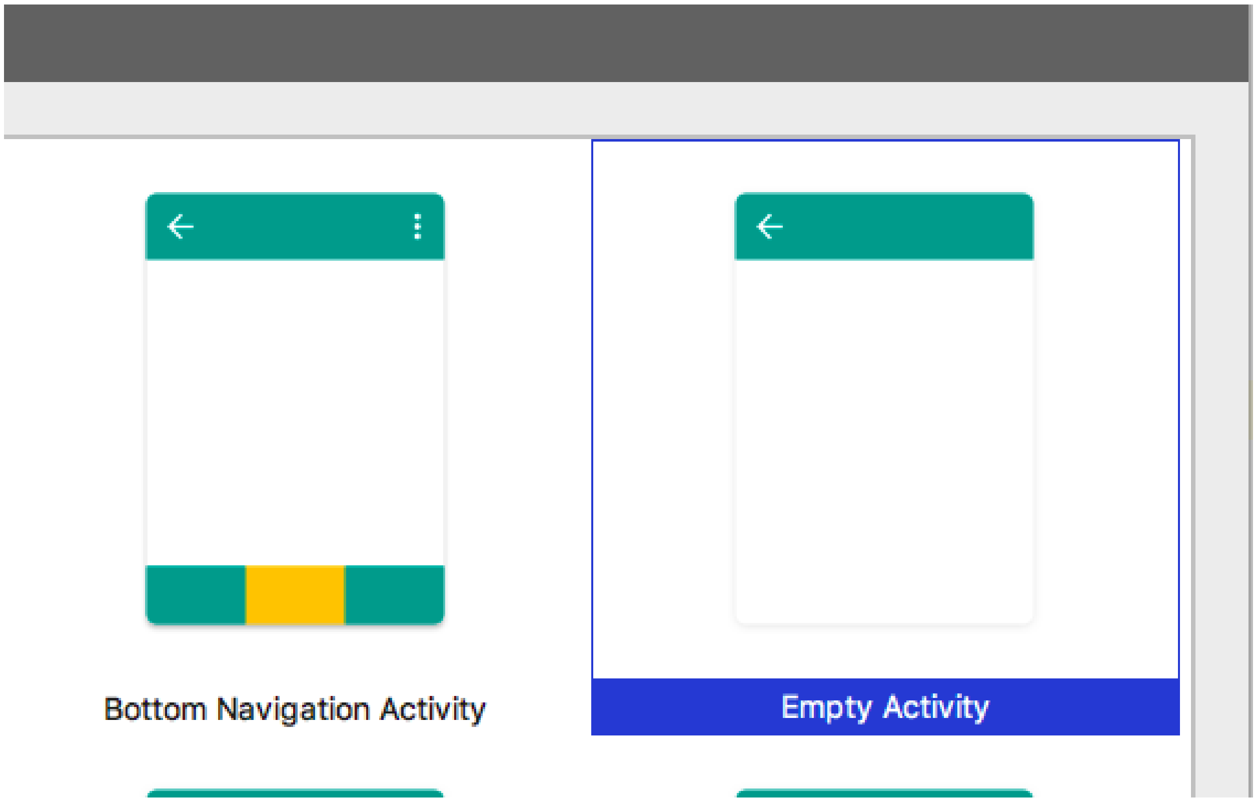
[✓] **Phone and Tablet**

API 15: Android 4.0.3 (IceCreamSandwich)

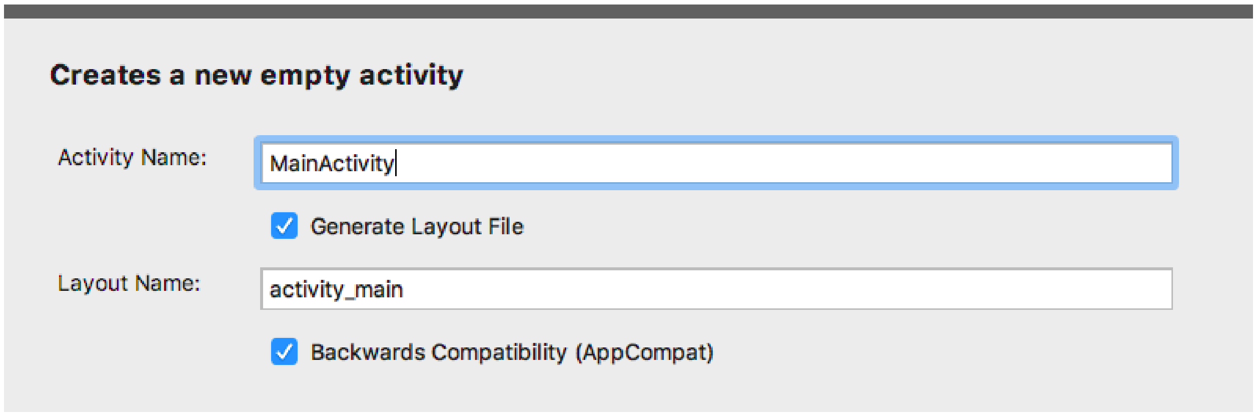
By targeting **API 15 and later**, your app will run on approximately **100%** of devices. Help me choose

[ ] Include Android Instant App support

Here you have to decide minimum SDK version for your app. It means device requires minimum selected sdk to run your app. As of now we are focusing on phone and tablet so we are not selecting wear, tv, auto and glass SDK. Also we will not check these check boxes. Then click on Next. Activity type selection window will appear as seen below. There are numbers of activity type this window will offer but initially we will chose empty activity.



The window will ask the Activity Name and it's layout name. By default it's MainActivity and activity_main respectively. Finally click on Finish button.



Creates a new empty activity

Activity Name: MainActivity

[✓] Generate Layout File

Layout Name: activity_main

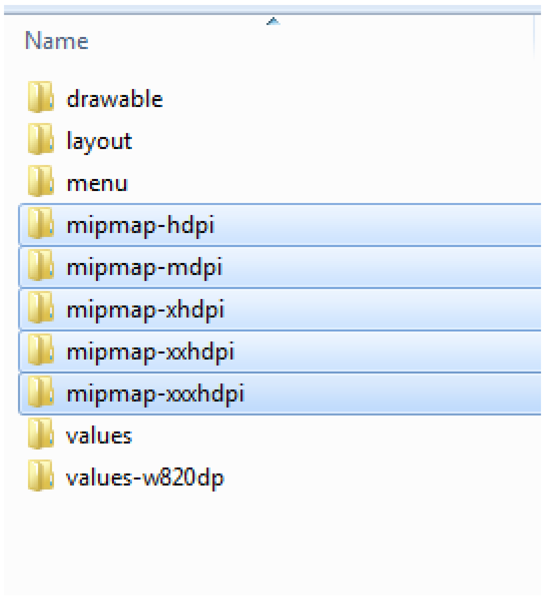
[✓] Backwards Compatibility (AppCompat)

2.4 New project directory structure

Newly created project have below mentioned directories.

- app/src/main/java : Here you will have all java code which is you will write for app. In this folder you have to created different packages as per your need to better organisation of code as recommended in java.
- app/src/main/res : Here you will find all resources files like layout, menu, strings and images which are used in app.
- app/src/main/AndroidManifest

Let's explore res folder in detail



Name

drawable
layout
menu
mipmap-hdpi
mipmap-mdpi
mipmap-xhdpi
mipmap-xxhdpi
mipmap-xxxhdpi
values
values-w820dp

- **drawable**: This folder contains xmls files which are required for background and shapes.
- **layout**: contains all layout used in different activities of app
- **menu**: contains xml files for menu used in app. Menu like action bar menu and nav drawer menu etc.
- **mipmap-hdpi to mipmap-xxxhdpi**: these folder contains images for different size. Android app will pick appropriate image based on device density(most probably device size) from these folder. To know how which folder contains which size of image you can see ic_launcher size in each folder. To put image in this folder you have to keep same name of that image. The Name should be separated with '_' character only.
- **values**: This folder contains different xmls file which contains values for colors, dimensions, string and styles used in app. For colors you will find colors.xml, for strings you will find strings.xml and for styles it is styles.xml .

2.5 Manifest file importance and manifest building blocks

Manifest is the application core. We can say this as a configuration file. Here we have to declare activities, services, broadcast receivers used in app. Also we have to mentioned app's permissions in this file. This is very important file we will see it's usage gradually as we go in this book and see app development. You can see sample manifest file below. Notice how permissions is declared. How activity, service is declared inside application node.

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.doctorapp">
    <uses-permission android:name="android.permission.INTERNET" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/app_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <uses-sdk
            android:maxSdkVersion="23"
            android:minSdkVersion="16"
            android:targetSdkVersion="23"/>
<activity
    android:name=".activity.RegistrationActivity"
    android:label="@string/app_name">
    <intent-filter>
```

```
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity
    android:name=".activity.RegistrationActivity"
    android:label="@string/app_name">
</activity>
<service android:name=".service.BackgroundCheckingService">
        </service>
    </application>
</manifest>
```

2.6 Gradle build system

Android studio mainly uses gradle build system to build the android app. Gradle creates build.gradle file in android project. Android studio creates 2 build.gradle files one at project root level and second is inside app directory. Most probably we have to change one inside app directory. where we can specify project dependencies, build tool versions, version code and version number.

3-UI Elements

Layouts and Widgets

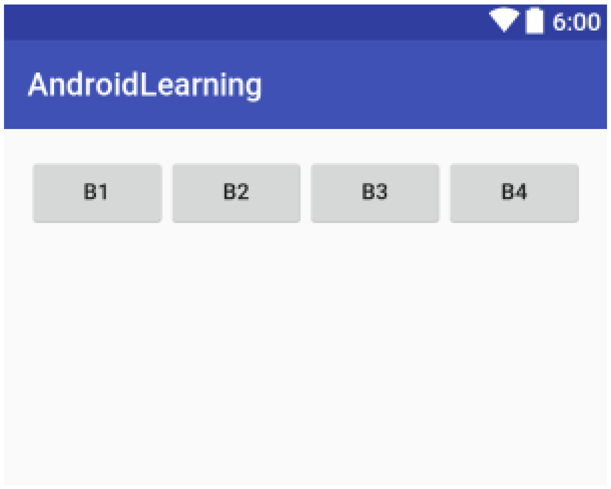
In previous chapter we have seen how to use android studio ide, how to create project in android studio, what are the directories that an android project have. We have also look at some of the elements of manifest file and it's important.
In this chapter we will see different types of layouts to draw UI, some basic ui elements like button, textbox, label, form elements like text area, radio button, checkbox, spinner and some event listener to handle click(tap) or double click(tap) event on these elements.

Basic layouts

3.1. Linear Layout

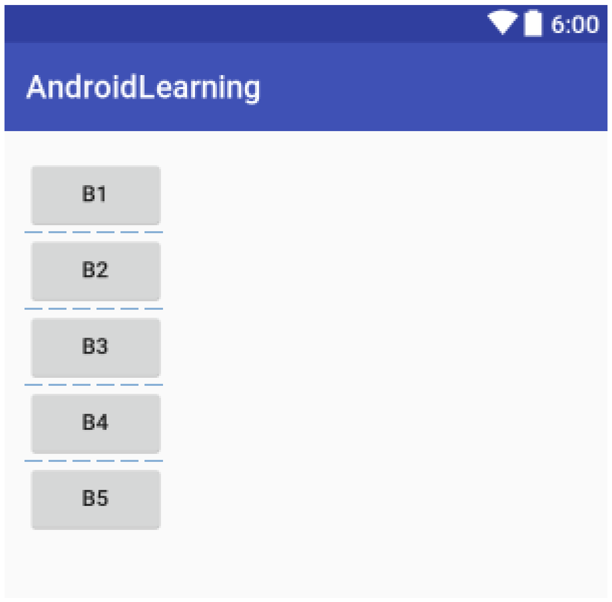
LinearLayout displays all elements linearly one by one. Consider layout is a container and it holds ui elements inside it.

```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:orientation="horizontal">
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="B1"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="B2"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="B3"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="B4"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="B5"/>
</LinearLayout>
```



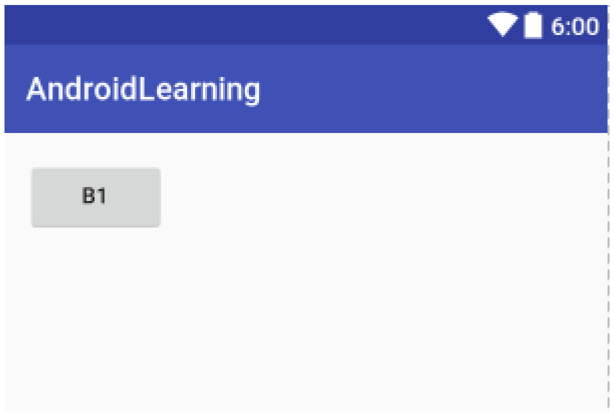
LinearLayout had one important attribute which is android:orientation which indicates elements placed inside LinearLayout displays horizontally or vertically linear. In above image LinearLayout has orientation horizontal. If we put orientation vertical it will display all elements vertically linear.

```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:orientation="vertical">
........
</LinearLayout>
```



The important attribute of all views in android is **layout_width** and **layout_height**. These two attributes have below possible values.

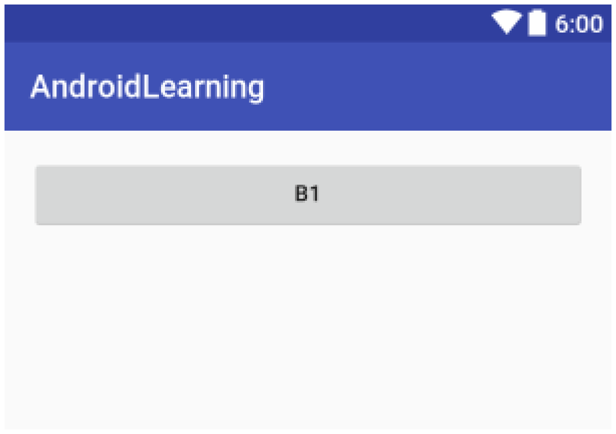
- **wrap_content** : height or width of view expands upto content inside view expands.



```
<Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="B1"/>
```

- **match_parent** : height or width of view expands upto it's parent height or width.

```
<Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="B1"/>
```



- **id attribute**: android:id attribute each view component should have which indicates unique identifier of view element in layout file. It must be unique throughout the project. This id attribute will be used in activity to reference UI element.

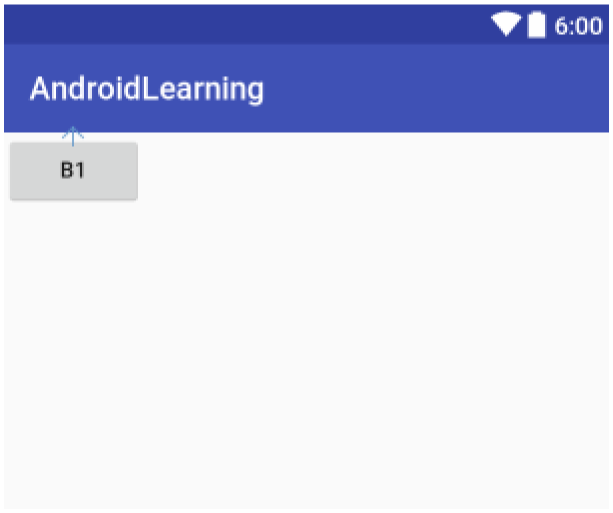
3.2. Relative Layout

RelativeLayout arranges elements in relative to other ui elements position. UI elements will be displayed like right_of or left_of specific element. Relative layout is a very powerful layout and can be used to design complex layout most of the time.

The important attributes of relative layout child elements

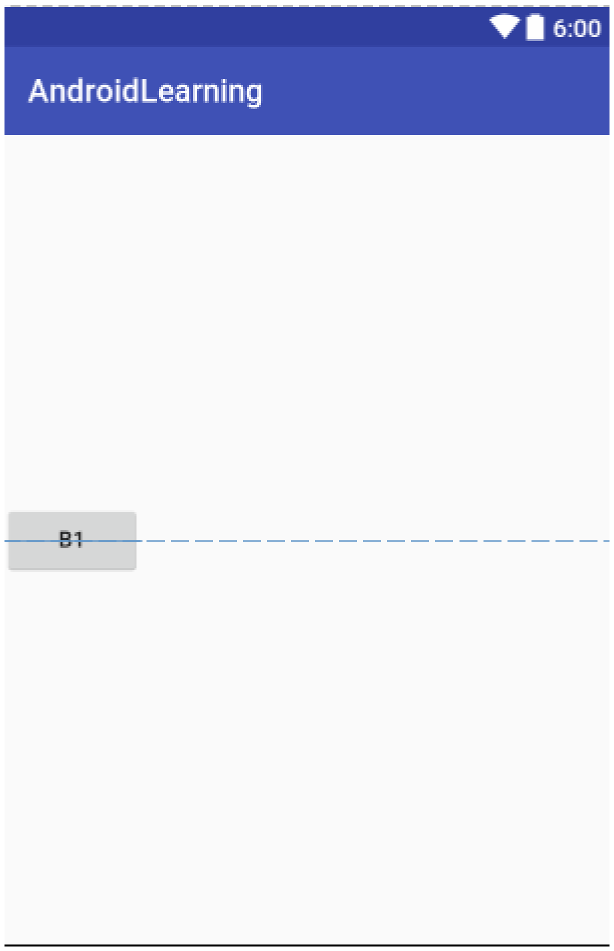
- **android:layout_alignParentTop**: If true child element's top edge will be match top edge of the parent.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_layout_example"
    android:layout_width="match_parent"
    android:layout_height="match_parent"                    tools:context="com.saralhisab.a\
ndroidlearning.LayoutExampleActivity">
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:text="B1"/>

</RelativeLayout>
```



- **android:layout_centerVertical**: If true child element's will be displayed vertically centre of it's parent.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_layout_example"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.saralhisab.androidlearning.LayoutExampleActivity">
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:text="B1"/>

</RelativeLayout>
```

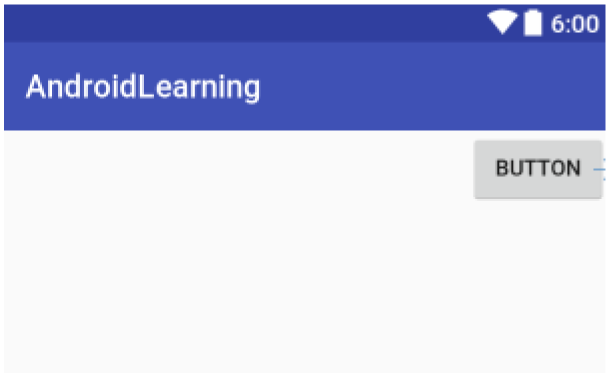


- **android:layout_below**: The element will be displayed below the specified element.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_layout_example"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.saralhisab.androidlearning.LayoutExampleActivity">

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="textPersonName"
        android:text="Product Name"
        android:ems="10"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_marginLeft="13dp"
        android:layout_marginStart="13dp"
        android:layout_marginTop="18dp"
        android:id="@+id/editText2" />

    <Button
        android:text="Buy"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@+id/editText2"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"
        android:layout_marginRight="12dp"
        android:layout_marginEnd="12dp"
        android:id="@+id/button" />

    <TextView
        android:text="125.00"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/button"
        android:layout_alignLeft="@+id/editText2"
        android:layout_alignStart="@+id/editText2"
        android:id="@+id/textView2" />
</RelativeLayout>
```

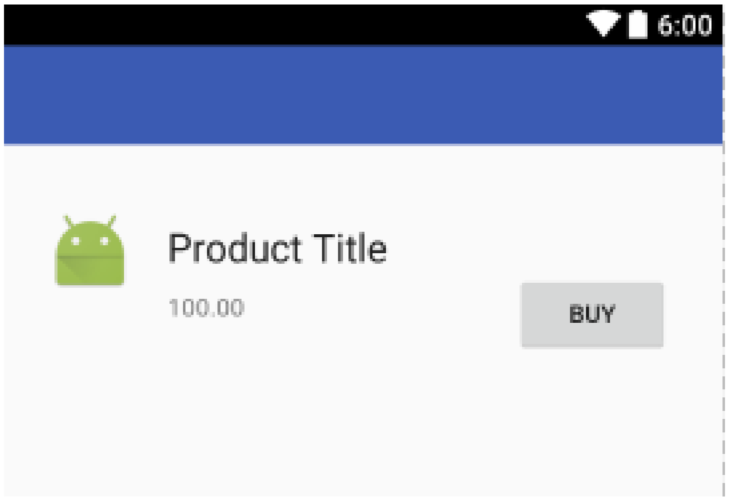
You can notice that the Price text view is below BUY button. You can see bold text which indicates this truth.

- **android:layout_toRightOf**: The top edge of the element will be right align to it's parent element.



```xml
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_layout_example"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.saralhisab.androidlearning.LayoutExampleActivity">
    <Button
        android:text="Button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:id="@+id/button2" />
</RelativeLayout>
```

Now Let's see how we can build this type of layout using RelativeLayout.



```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto" android:layout_width="match_\
parent"
    android:layout_height="match_parent" android:paddingLeft="@dimen/activity_horizo\
ntal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior" tools:showIn="@layo\
ut/app_bar_main"
    tools:context=".MainActivity">
    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="@mipmap/ic_launcher"
        android:layout_marginLeft="12dp"
        android:layout_marginStart="12dp"
        android:layout_marginTop="20dp"
        android:id="@+id/imageView2"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />

    <TextView
        android:text="Product Title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@+id/imageView2"
        android:layout_toRightOf="@+id/imageView2"
        android:layout_toEndOf="@+id/imageView2"
        android:layout_marginLeft="20dp"
        android:layout_marginStart="20dp"
        android:layout_marginTop="10dp"
        android:id="@+id/textView2"
        android:textAppearance="@style/TextAppearance.AppCompat.Large" />

    <TextView
        android:text="100.00"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/imageView2"
        android:layout_alignLeft="@+id/textView2"
        android:layout_alignStart="@+id/textView2"
        android:id="@+id/textView3" />

    <Button
        android:text="Buy"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textView2"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"
        android:layout_marginRight="14dp"
        android:layout_marginEnd="14dp"
        android:id="@+id/button" />
</RelativeLayout>
```

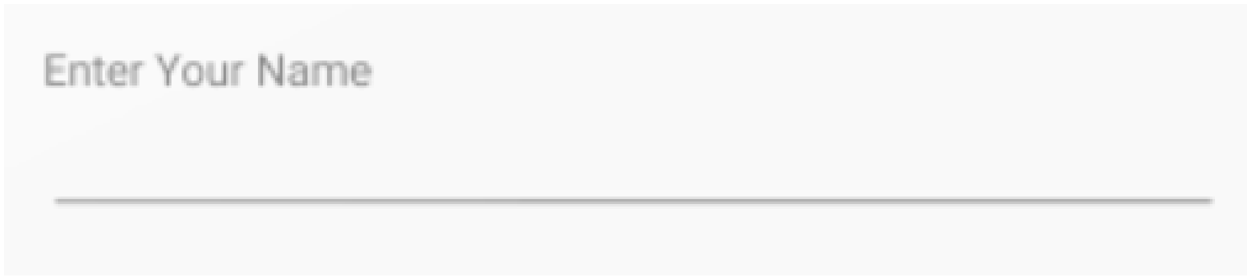
You can see that product title is right of imageview and align top to it's parent. Product price is below image view and right of it. And Buy button is below product name textview and right of it's parent.

3.3. Simple UI Elements Textview, ImageView

Android also gives you input controls like EditText,Radio button for choices and Checkbox for multiple choice selection. Let See how to use these controls and their property.

EditText:

EditText will be used to collect input data from app user. Edit text is essential and frequently used Element among all input widgets.



```
<EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/name"  />
```

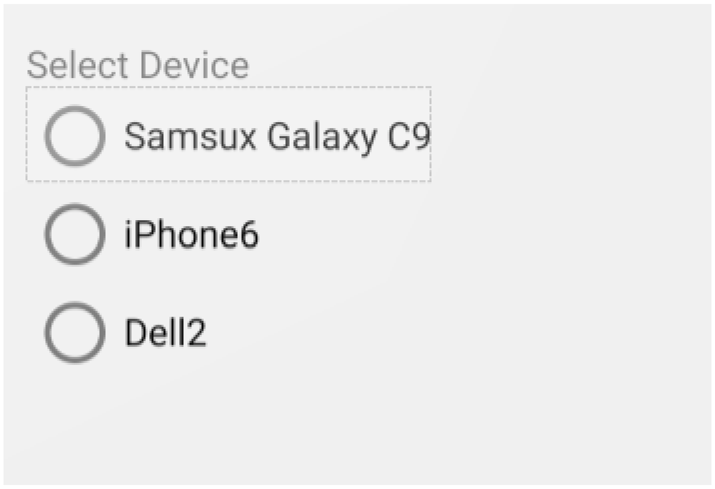
Now if you want to access this EditText in your activity you have to declare EditText in activity. Initialise it in onCrete method of activity. And access edit text .

```java
public class MainActivity extends AppCompatActivity {

    private EditText nameText;

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

        nameText = (EditText) findViewById(R.id.name);
        nameText.setText("Sample text in textview");
        nameText.getText();// to access text from edittext

    }
}
```

If you need edit text for email address you have to add below property in edit text.
android:inputType="textEmailAddress"
For phone number use below attribute.
android:inputType="phone"
For number only
android:inputType="number"

Radio Buttons:

Radio button is useful when you have to design a form in which user can choose only one options among all given. Like Quiz app. For radio button there is RadioButton tag in android which you can use in Layout file. To add behaviour of single choice you have to add all radio buttons in RadioGroup tag.



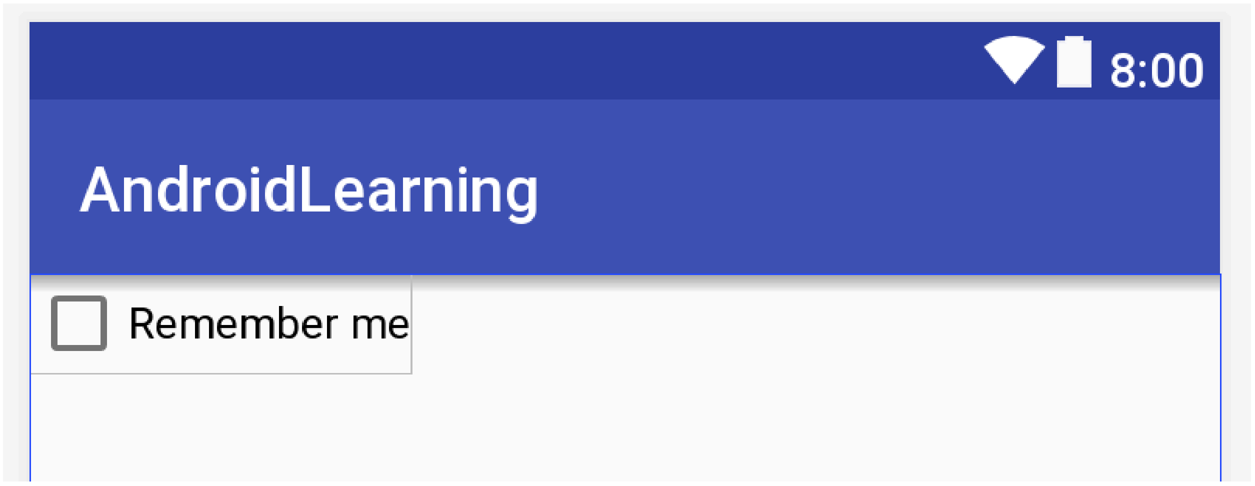
To design layout like above see below code example.

```
<RadioGroup
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/radioGroup"
        android:layout_below="@+id/textView"
        android:layout_centerHorizontal="true">

        <RadioButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Samsung Galaxy C9"
            android:id="@+id/samsung" />
        <RadioButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="iPhone6"
            android:id="@+id/iphone6" />
        <RadioButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Dell2"
            android:id="@+id/dell2" />
    </RadioGroup>
```

To get selected choice from radio button we will see in next section.

Checkbox:

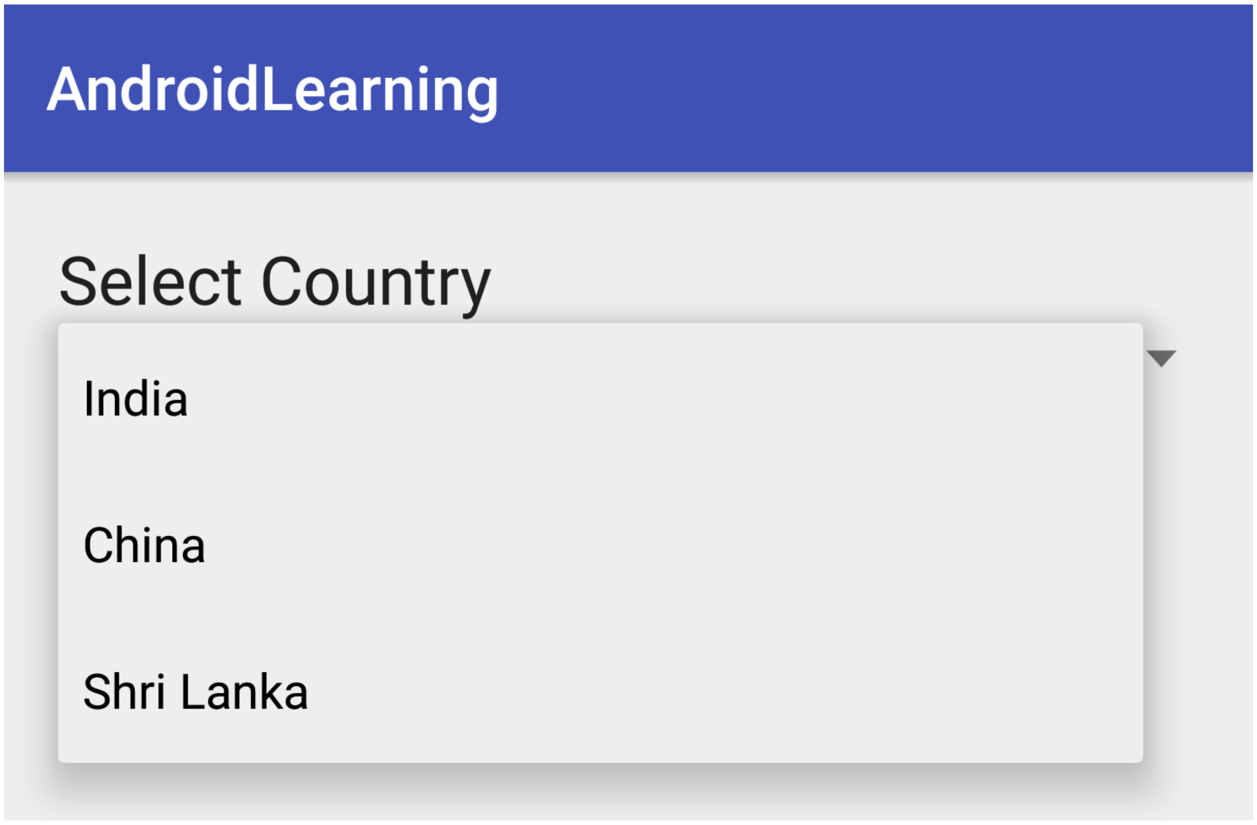
Checkbox are useful when you have multiple choices. Say like you have a multi choice question based app. You can add checkbox in you screen using below tag in xml.



```
<CheckBox android:id="“@+id/checkbox_remember_me”
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="“@string/remember_me”/>
```

Here you notice that two properties which are highlighted android:id and android:text. Using android:id you can reference checkbox in activity for event listening or for fetching the checked state. Using android:text you can set caption (label) for checkbox.

Spinners(Dropdown): spinners are useful when you have to add dropdown in your screen. Spinner needs items array which you can add either by declaring statically in values.xml or you can add items dynamically from activity. Here we will see declaring array items in values.xml



```
<Spinner
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:entries="@array/listCountries"
        ></Spinner>
arrays.xml:
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="listCountries"  >
        <item>India</item>
        <item>China</item>
        <item>Shri Lanka</item>
    </string-array>
</resources>
```

3.5. Event Handlers for Widgets like Button, Checkbox, Radio Button

Until now we have seen widgets which are displayed on screen, but yet not seen how we can process this widgets or how we can interact with this widgets.Like if we have placed button in activity, now we can handle it's click and process data written in edit text. So to understand this event listener comes in picture. Android provides event listener interface for ui widgets like button, radio button or checkbox. Now let see this event listener for these widgets one by one.

For Button:

To listen event first we need to initialise button inside activity in onCreate method.

```
public class Activity2 extends AppCompatActivity {
    private EditText txtName;
    private Button okBtn;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_2);
        txtName = (EditText) findViewById(R.id.txt_fname);
        okBtn = (Button) findViewById(R.id.ok_btn);
        okBtn.setOnClickListener(okBtnClickListener);
    }

    protected View.OnClickListener okBtnClickListener = new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Toast.makeText(Activity2.this, “First name is"+txtName.getText().toStrin\
g(), Toast.LENGTH_SHORT).show();
        }
    };

}
```

In above activity we have take one button reference from layout and added onClick listener. You can see that in onCreate method we have bolded the line which register button onClickListener .

```
okBtn.setOnClickListener(okBtnClickListener);
```

Just after onCreate method we have declared okBtnClickListener inside activity. So inside onClick interface method we have callback when button will clicked. Here in this example we have simply toasted what user have written inside text box.

For Checkbox:

In order to listen onclick event in checkbox first we need to initialise checkbox in oncreate method of activity.

```
rememberMeCheckbox = (Checkbox) findViewById(R.id.remember_me_checkbox);
```

Then after we have to set onclick listener. This line is also inside oncreate method.

```
 rememberMeCheckbox.setOnClickListener(rememberMeClickListener);
```

Now we have to declare checkbox onclick listener as we have declared for button.

```
protected View.OnClickListener rememberMeClickListener = new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if(rememberMeCheckbox.isChecked() )
          Toast.makeText(Activity2.this, "Remember me is checked”, Toast.LENGTH_SHOR\
T).show();
            }
        }
    };
```

In above checkbox listener we have demonstrated that if checkbox is checked then it shows toast message that checkbox is checked other wise it will not show anything.
For checkbox there is another listener which is oncheckedchanged. This listener will gives us checkbox current status in callback in boolean, so we can take decision based on it.

For Radio Button:

Normally for radio button we so not need to listen for radio button clicked but we have to identify which radio button is checked from the radio group. First we have to initialise radio button and radio button group in activity oncreate method.

xml for radiogroup

```xml
<RadioGroup
        android:id="@+id/radio"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" >

        <RadioButton
            android:id="@+id/radioMale"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/radio_male"
            android:checked="true" />

        <RadioButton
            android:id="@+id/radioFemale"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/radio_female" />

    </RadioGroup>
```

java code to access selected radio

```java
radioGroup = (RadioGroup) findViewById(R.id.radio);
int selectedId = radioGroup.getCheckedRadioButtonId();

 // find the radiobutton by returned id
 radioButton = (RadioButton) findViewById(selectedId);
```

Now you can use radiButton to access it's text and process it in your code.

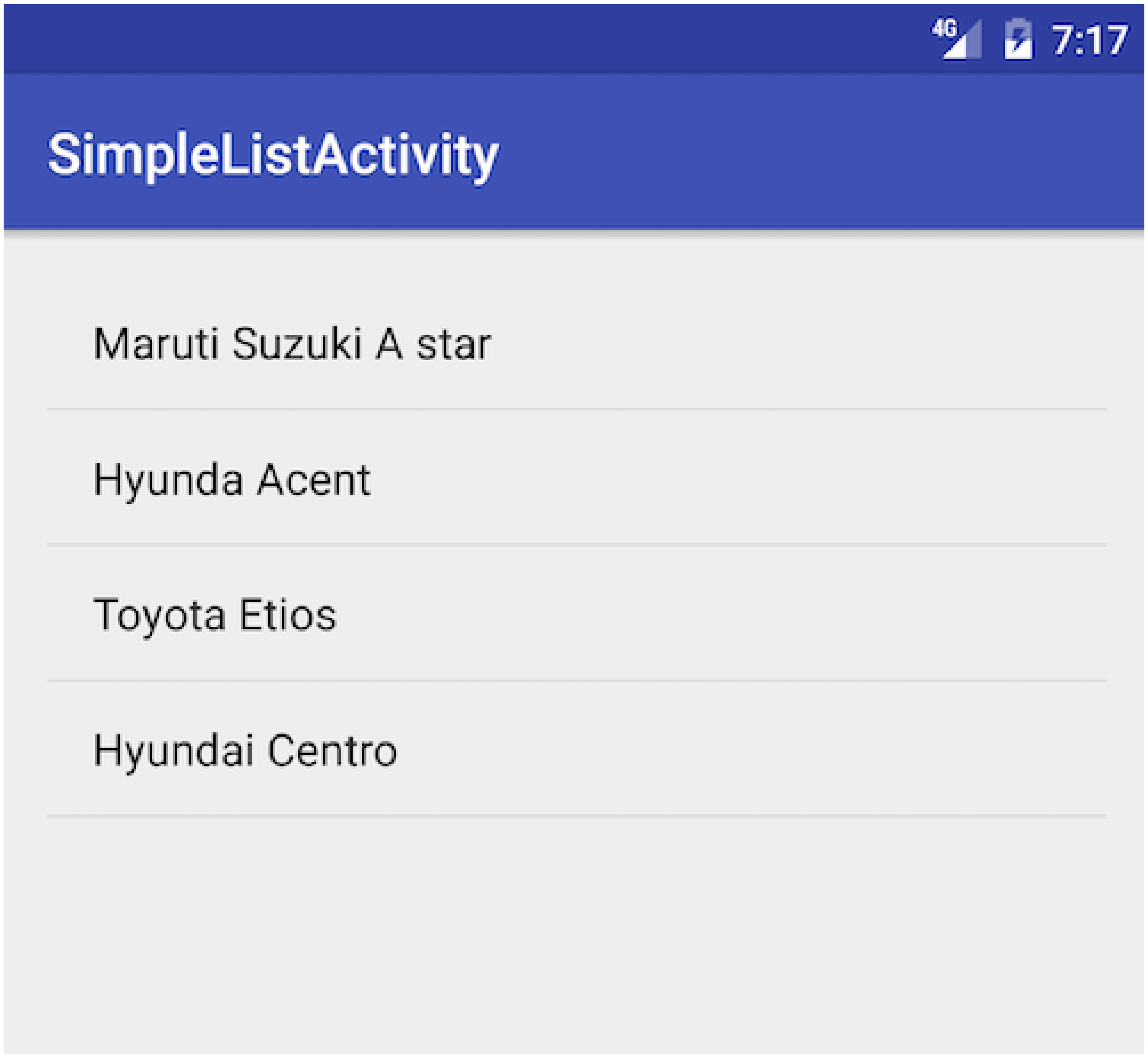
For Spinner:

How ever we can listen onclick event and other events related to spinner but here we will see how to identify which item is selected in spinner. Spinner have various methods like getSelectedItemPosition, getSelectedItem each return different value. We can access selected item value using below mentioned code.

```
spinner.getItemAtPosition(spinner.getSelectedItemPosition()).toString();
```

3.6. ListView

ListView is not actually a layout but view which displays same type of views repetitively. If you have android phone then you will find many apps which uses list view like Contacts, Notes app.
ListView will be used to display collection of data. In Android default ListView have single textview but if you want to display each item with custom design you have to design one custom (row) view which will iterate over all list of data.
A simple ListView will be displayed as below. It is collection of vehicle list.



Maruti Suzuki A star
Hyunda Acent
Toyota Etios
Hyundai Centro

Now we will see how to implement this simple ListView in android.

Layout Code:

```xml
<ListView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/vehicle_list"
        ></ListView>
```

Activity Code:

```java
ListView vehicleList = (ListView) findViewById(R.id.vehicle_list);
        String[] vehicles = {"Maruti Suzuki A star","Hyunda Acent","Toyota Etios","H\
yundai Centro"};
        ArrayAdapter<String> vehicleListAdapter = new ArrayAdapter<String>(this,andr\
oid.R.layout.simple_list_item_1,
                android.R.id.text1,vehicles);
        vehicleList.setAdapter(vehicleListAdapter);
```

findViewById: As we seen earlier in this chapter in android layout each view element have id attribute. This is important attribute to uniquely identify view. In Activity if we have to reference any view we can reference view element using **findViewById** method . This method can reference view in activity's layout using id and return base View class. But we have to type cast it to a specific View class which we want to use in activity.

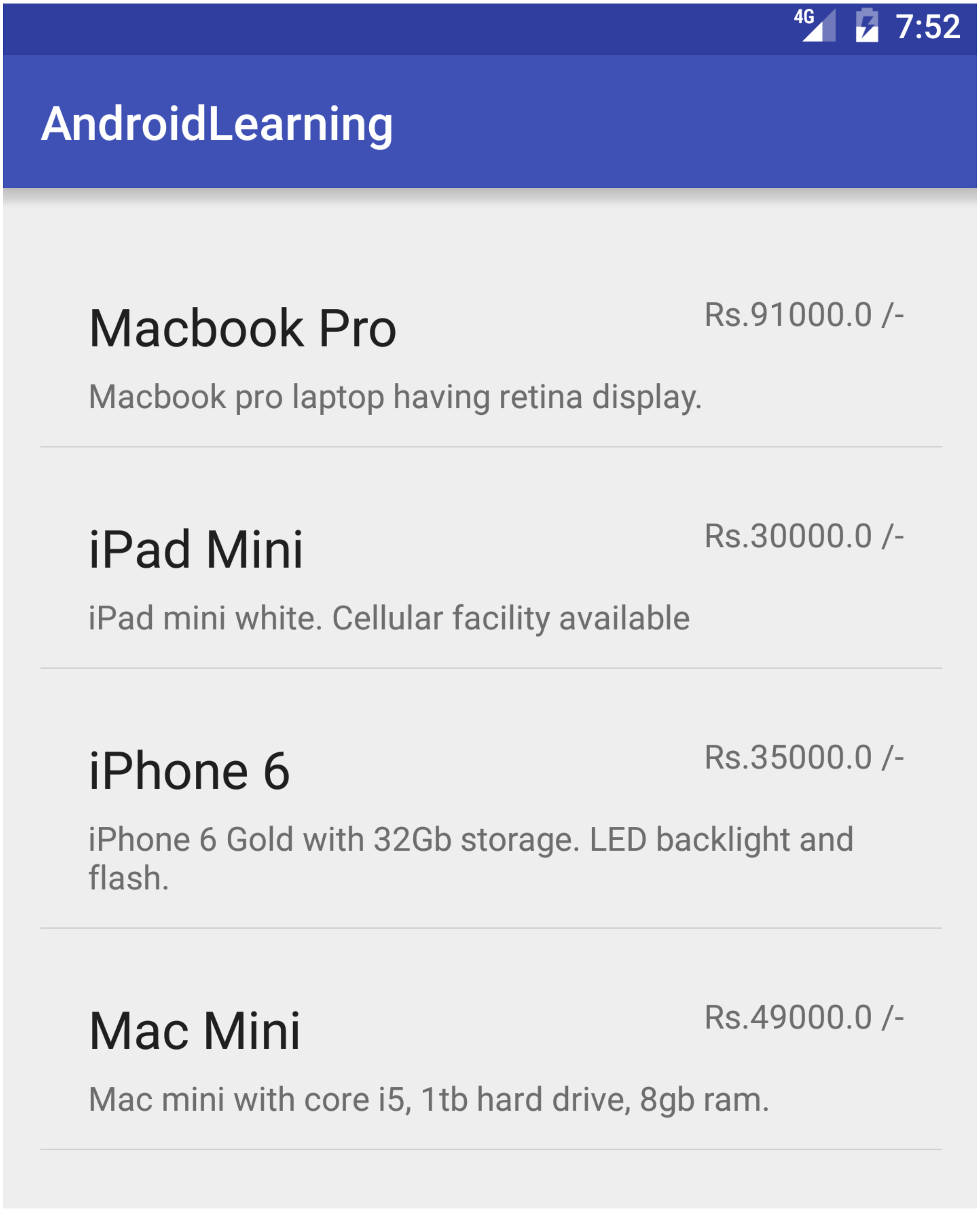
Look at above Layout code, it has ListView element with id **vehicle_list**. Now in activity we have referenced it using **findViewById(R.id.vehicle_list)**. This will return base View class but you can notice that we have type cast it to ListView and in this way we have initialised ListView in Activity.

Adapter: We must have to set adapter in order display items in ListView. Adapter will hold data and type view displayed for each items. For simple ListView we have to use **ArrayAdapter**. You can see that while instantiating new ArrayAdapter class we have passed first argument as this which context(normally activity reference). Second argument **android.R.layout.simple_list_item** this is default layout for simple list view. Third argument is the idea of textview which will use to display list item in list view. And last and fourth argument is a collection of data which will be displayed in list view. You can notice that we have used String type of data for simple list view .

Finally once the adapter is ready we have pass adapter reference in list view using setAdapter method. Once the adapter is set in list view list will displays the data.

Custom Listview

If we have to design UI as mentioned below we have to design list view with custom way using adapter.



Macbook Pro Rs.91000.0 /-
Macbook pro laptop having retina display.

iPad Mini Rs.30000.0 /-
iPad mini white. Cellular facility available

iPhone 6 Rs.35000.0 /-
iPhone 6 Gold with 32Gb storage. LED backlight and flash.

Mac Mini Rs.49000.0 /-
Mac mini with core i5, 1tb hard drive, 8gb ram.

To design this type of list view we have to prepare below mentioned item.

1. **List view activity layout with ListView**
2. **Listview iterator item layout view**
3. **Listview item Model class**
4. **Listview Adapter class**
5. **ListView Activity class**

Now let see step by step guide to implement custom list view.

1. **List view activity layout with ListView**

Create new activity for custom list view in android studio and name it **CustomListActivity** it will create its layout file with named **activity_custom_list.xml** . Open this layout file and write below layout code in it.

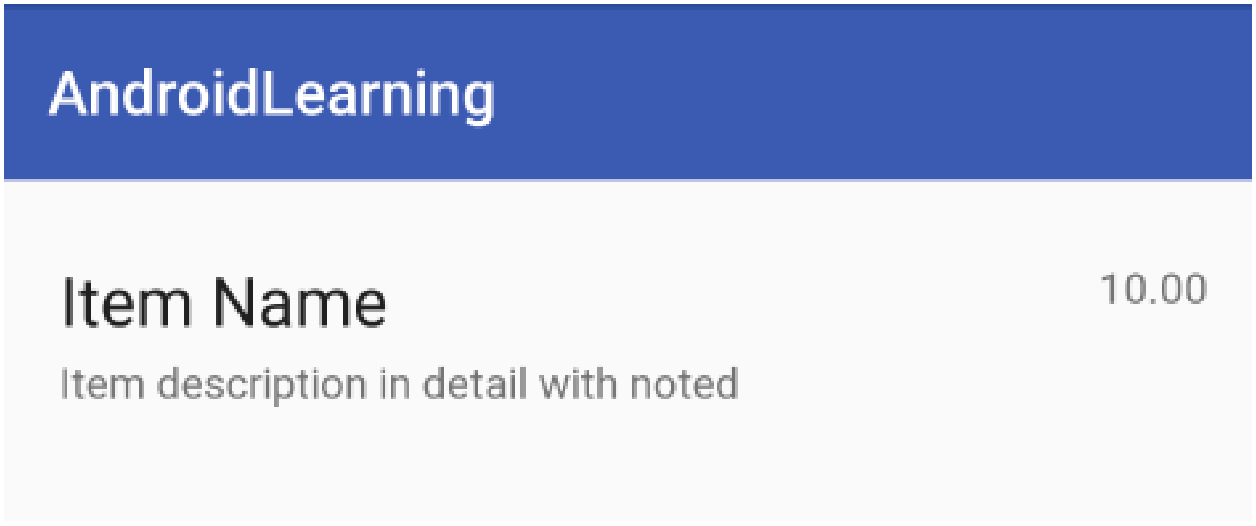
```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_custom_list"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.saralhisab.androidlearning.CustomListActivity">

    <ListView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/product_list"></ListView>

</RelativeLayout>
```

FileName: activity_custom_list.xml

You can notice that in above layout we have added ListView with id product_list . In this layout file we will not have any other UI element apart from list view.

2. **Listview iterator item layout view**

ListView is a collection of one specific type of layout elements. So create one layout file which will represent a single item in list view and name it **custom_list_item.xml** .



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

    <TextView
        android:text="Item Name"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true"
        android:layout_marginStart="20dp"
        android:layout_marginTop="27dp"
        android:id="@+id/item_name" />

    <TextView
        android:text="10.00"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@+id/item_name"
        android:layout_alignParentEnd="true"
        android:layout_marginEnd="16dp"
        android:id="@+id/item_price" />
    <TextView
        android:text="Item description in detail with noted"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/item_name"
        android:layout_alignStart="@+id/item_name"
        android:layout_marginTop="5dp"
        android:layout_marginBottom="12dp"
        android:id="@+id/item_description" />
</RelativeLayout>
```

This layout will iterate for all list view data items.

3. **Listview item Model class**

Now we will create one model class to contain single data item for list view. Create Product.java. This is simple plain java class which contains required data members to hold one product with getters and setters methods.

```java
public class Product {
    private  String itemName;
    private String itemDescription;
    private double itemPrice;

    public Product(String itemName, String itemDescription, double itemPrice) {
        this.itemName = itemName;
        this.itemDescription = itemDescription;
        this.itemPrice = itemPrice;
    }
    public String getItemName() {
        return itemName;
    }
    public void setItemName(String itemName) {
        this.itemName = itemName;
    }

    public String getItemDescription() {
        return itemDescription;
    }

    public void setItemDescription(String itemDescription) {
        this.itemDescription = itemDescription;
    }

    public double getItemPrice() {
        return itemPrice;
    }

    public void setItemPrice(double itemPrice) {
        this.itemPrice = itemPrice;
    }
}
```

4. **Listview Adapter class**

Adapter is important class. This class will process over each data items and prepare views from data items for list view. This class hold ArrayList of model class which we have prepared in previous step. This ArrayList is normally prepared in activity and pass in this class in constructor. In this class getCount and getView are important methods. We have also prepared adapter for simple list view earlier in this chapter. But it is very simple array adapter.

```
public class CustomItemListAdapter extends BaseAdapter {

    private ArrayList<Product> mProducts;
    private Context mContext;

    public CustomItemListAdapter(Context context,ArrayList<Product> products){
        this.mContext = context;
        this.mProducts = products;
    }

    @Override
    public int getCount() {
        return this.mProducts.size();
    }

    @Override
    public Object getItem(int i) {
        return this.mProducts.get(i);
    }

    @Override
    public long getItemId(int i) {
        return 0;
    }

    @Override
    public View getView(int position, View view, ViewGroup viewGroup) {
        TextView productName,productPrice,description;

        LayoutInflater layoutInflater = (LayoutInflater) this.mContext.getSystemServ\
ice(
                Context.LAYOUT_INFLATER_SERVICE);

        view = layoutInflater.inflate(R.layout.custom_list_item, null);

        productName = (TextView) view.findViewById(R.id.item_name);
        productPrice = (TextView) view.findViewById(R.id.item_price);
        description = (TextView) view.findViewById(R.id.item_description);

        Product product = this.mProducts.get(position);
        productName.setText(product.getItemName());
        productPrice.setText("Rs."+product.getItemPrice()+" /-");
        description.setText(product.getItemDescription());

        return view;
    }
}
```

For this adapter class below mentioned points are very important.

- **Model class ArrayList**: We have passed ArrayList of Product class in constructor and initiated mProducts array list.

- **getCount method**: This method will return total numbers of list items presents in list view. By default it is zero but we have to modify and put array list size using ArrayList size method. Here we have put mProducts.size() it means if we have 3 products in mProducts array list 3 list items will be displayed in list view.
- **getView method**: This is important method of adapter. It will return view which backed with data from array list at position from current iteration.
- First argument of this method will indicate position of list item from mProducts. This method will be called 3 times if mProducts will have 3 entries. In general it will call as many times as getCount returns.
- Second argument is top level view element which represent list item. We have designed view for this earlier in step no 2. So After initialising layout inflator we will inflate view with custom_-list_item. Below code indicates this truth.

```
view = layoutInflater.inflate(R.layout.custom_list_item, null);
```

Now we will initialise three textview by inside this custom_list_item by referencing view variable.

```
                productName = (TextView) view.findViewById(R.id.item_name);
```

After completing initialisation we will fetch model class using position argument.

```
Product product = this.mProducts.get(position);
```

Once we will have product model class from array list we will fill textviews initialised earlier from this product class.

```
productName.setText(product.getItemName());
        productPrice.setText("Rs."+product.getItemPrice()+" /-");
```

Finally getView method will return view filled with data and list view will displayed at its position.

5. **ListView Activity class**

We are almost done with most of the components which are required for custom list view that we want. Now we have to use these components in activity. Below code listing shown how we can insatiate adapter class and set it for list view. Here in activity we have initialised list view with sample items but when things are stored in database we have to read this list from database. In later chapters of this book we will see how we can fetch data from database and feel this type of array list.

```
public class CustomListActivity extends AppCompatActivity {

    private ListView productList;
    private CustomItemListAdapter customItemListAdapter;
    private ArrayList<Product> products;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_custom_list);

        productList = (ListView) findViewById(R.id.product_list);
        this.getProducts();
        customItemListAdapter = new CustomItemListAdapter(this,this.products);
        productList.setAdapter(customItemListAdapter);
    }

    private void getProducts()
    {
        this.products = new ArrayList<Product>();
        this.products.add(new Product("Macbook Pro","Macbook pro laptop having retin\
a display.",91000.00));
        this.products.add(new Product("iPad Mini","iPad mini white. Cellular facilit\
y available",30000.00));
        this.products.add(new Product("iPhone 6","iPhone 6 Gold with 32Gb storage. L\
ED backlight and flash.",35000.00));
        this.products.add(new Product("Mac Mini","Mac mini with core i5, 1tb hard dr\
ive, 8gb ram.",49000.00));



    }
}
```

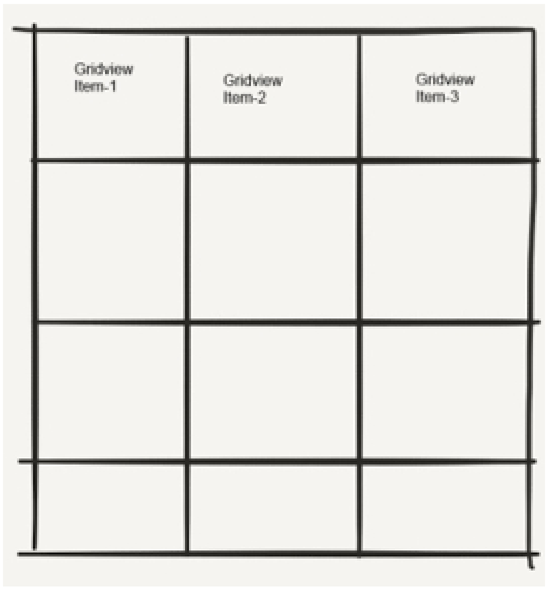
In above code three points are important along with the sequence it wrote.

- First initialised list view with **findViewById** method.
- Second prepare array list for model class. In above code **getProducts** method prepare array list.
- Third create Adapter class object and pass array list prepared in previous step. While creating **CustomItemListAdapter** class we have passed **this.products** which is filled in getProducts method.
- And finally we have set this adapter in list view.

```
       productList.setAdapter(customItemListAdapter);
```

3.7. GridView

Gridview is another type of collection view like listview in android. But Gridview show each items in table like grid. For example a photo gallery application shows thumbnail of all photos which is gridview.



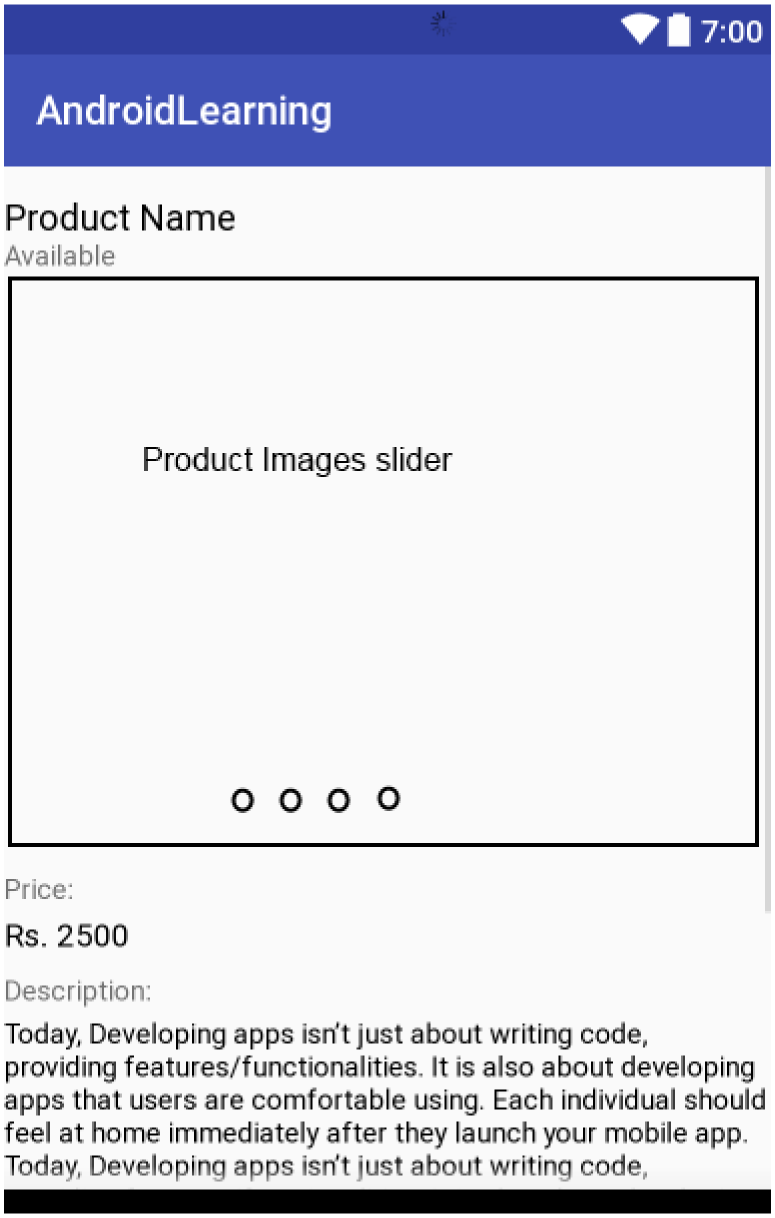
Gridview contains same steps as mentioned in Listview. It means you have to take a model class , a customer adapter for gridview and gridview item layout. Just you need to use GridView tag instead of ListView in layout file.
Gridview will be used for showing products in e-commerce like app, Or you can use it for photo gallery , Or books library app.Ultimately it depends on app.

3.8. ScrollView

Apart from list view and grid view there is scrollview in android which supports scrolling for larger contents. When ever there is a larger form or any content you think will me needed scrolling like description of product add it in scroll view. Remember scrollview has only one child element. But you can add more elements to its child(scrollview child) element. If devices screen is larger enough to hold all contents there will be no scroll bar visible. But if content is larger than device screen then there will be a scrollbar.

To generate below type of layout we must have to use Scrollview so that when product description exceeds height scrollbar will be introduce.

Here is the code listing for below layout using scrollview.



Product Name
Available

Product Images slider

Price:
Rs. 2500
Description:
Today, Developing apps isn't just about writing code, providing features/functionalities. It is also about developing apps that users are comfortable using. Each individual should feel at home immediately after they launch your mobile app. Today, Developing apps isn't just about writing code,

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/content"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.inspire.shopapp.activity.ProductDetailFragment"
    android:background="?android:attr/colorBackground">

    <LinearLayout
```

```
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:theme="@style/AppTheme">

        <android.support.v7.widget.Toolbar
            android:id="@+id/my_toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            android:elevation="4dp"
            android:theme="@style/ThemeOverlay.AppCompat.Dark"
            app:popupTheme="@style/ThemeOverlay.AppCompat.Light"/>


        <TextView
            android:id="@+id/product_name"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="15dp"
            android:layout_alignParentTop="true"
            android:text="Product Name"
            android:textColor="@android:color/background_dark"
            android:textSize="18sp" />

        <TextView
            android:id="@+id/availibility_status"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Available" />

        <android.support.v4.view.ViewPager
            android:id="@+id/product_images_viewpager"
            android:layout_width="match_parent"
            android:layout_height="280dp"
            android:layout_below="@+id/product_name"
            android:layout_marginBottom="10dp"
            android:layout_marginTop="10dp" />


        <TextView
            android:id="@+id/textView11"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/product_images_viewpager"
            android:layout_marginBottom="5dp"
            android:text="Price:" />

        <TextView
            android:id="@+id/product_price"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Rs. 2500"
            android:textColor="@android:color/background_dark"
            android:textSize="16sp"
            android:layout_below="@+id/product_images_viewpager"
            android:layout_alignRight="@+id/product_name" />

        <TextView
            android:id="@+id/textView13"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="5dp"
            android:layout_marginTop="10dp"
            android:text="Description:" />

        <TextView
            android:id="@+id/product_description"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Today, Developing apps isn’t just about writing code, prov\
iding features/functionalities. It is also about developing apps that users are comf\
ortable using. Each individual should feel at home immediately after they launch you\
r mobile app"
            android:textColor="@android:color/background_dark"
            android:textSize="14sp" />

        <Button
            android:id="@+id/contact_seller"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="10dp"
            android:layout_marginTop="20dp"
            android:backgroundTint="@color/colorAccent"
            android:text="Contact Seller"
            android:textAllCaps="true"
            android:textAppearance="@style/TextAppearance.AppCompat.Large"
            android:textColor="@android:color/background_light"
            android:textStyle="bold" />
    </LinearLayout>

    </ScrollView>
```

You can notice that in above code listing <Scrollview> has only one child element which is LinearLayout. But LinearLayout has multiple children as we mentioned earlier. Also you are not seeing Contact Seller button in above screen but when you scroll down you will observe it.

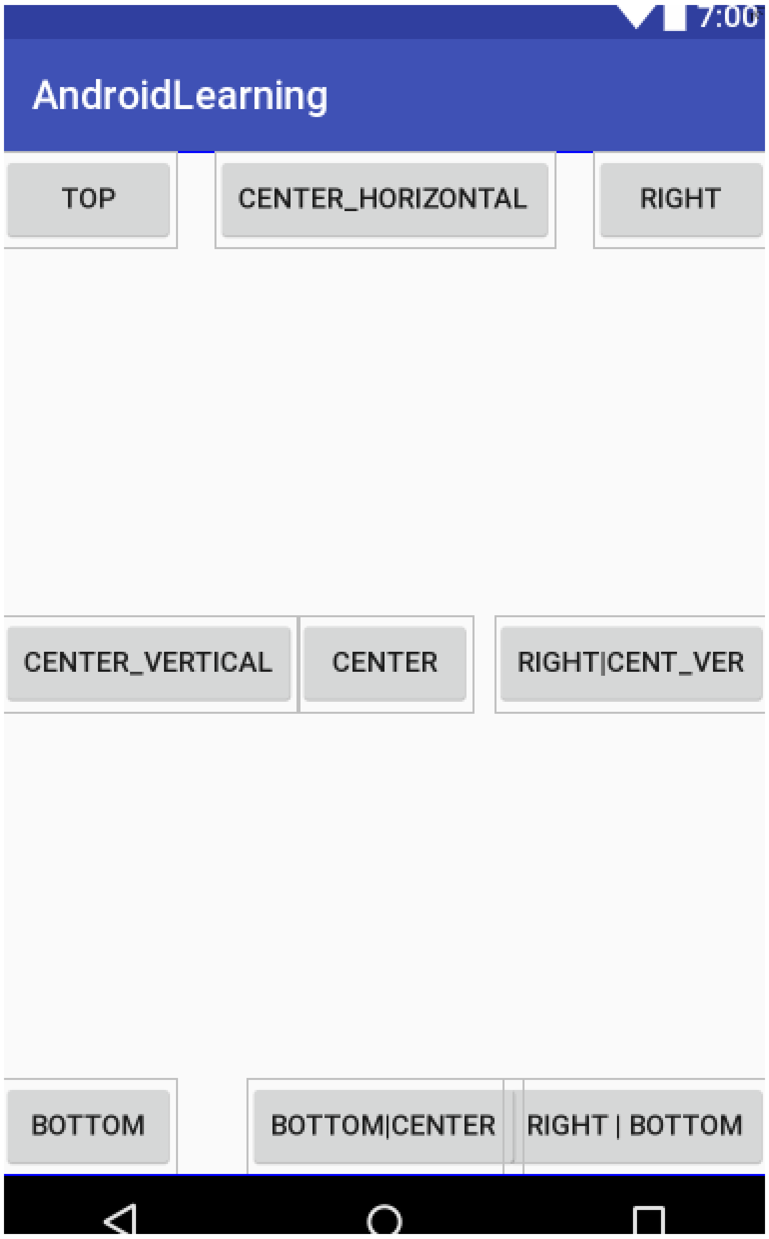
3.9. Frame Layout

Frame Layout is designed to block out an area on the screen to display a single item. Generally, FrameLayout should be used to hold a single child view, because it can be difficult to organise child views in a way that's scalable to different screen sizes without the children overlapping each other. The important attribute of the framelayout is android:layout_gravity through which you can control position of the view controls.

android:foregroundGravity have below possible values.

- bottom: Push object to the bottom of its container, not changing its size.
- centre: Place the object in the centre of its container in both the vertical and horizontal axis, not changing its size.
- center_horizontal: Place object in the horizontal centre of its container, not changing its size.
- center_vertical: Place object in the vertical centre of its container, not changing its size.
- clip_horizontal: Additional option that can be set to have the left and/or right edges of the child clipped to its container's bounds. The clip will be based on the horizontal gravity: a left gravity will clip the right edge, a right gravity will clip the left edge, and neither will clip both edges.
- clip_vertical: Additional option that can be set to have the top and/or bottom edges of the child clipped to its container's bounds. The clip will be based on the vertical gravity: a top gravity will clip the bottom edge, a bottom gravity will clip the top edge, and neither will clip both edges.
- fill: Grow the horizontal and vertical size of the object if needed so it completely fills its container.
- fill_horizontal: Grow the horizontal size of the object if needed so it completely fills its container.
- fill_vertical: Grow the vertical size of the object if needed so it completely fills its container.
- left: Push object to the left of its container, not changing its size.
- right: Push object to the right of its container, not changing its size.
- top: Push object to the top of its container, not changing its size.

Now using above possible attributes we can design below layouts.



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


    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="top"
        android:text="Top" />
    <Button
        android:id="@+id/button4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="right|bottom"
        android:text="Right | Bottom" />
    <Button
        android:id="@+id/button5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="Right" />
    <Button
        android:id="@+id/button6"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="Center" />
    <Button
        android:id="@+id/button7"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:text="Center_Horizontal" />
    <Button
        android:id="@+id/button8"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:text="Center_Vertical" />
    <Button
```

```
        android:id="@+id/button9"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom"
        android:text="Bottom" />
    <Button
        android:id="@+id/button10"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|center"
        android:text="Bottom|Center" />
    <Button
        android:id="@+id/button11"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="right|center_vertical"
        android:text="right|cent_ver" />
</FrameLayout>
```

You can see that we have aligned all buttons only using single attribute android:layout_gravity only using it's possible values.

4-Navigation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/android-book.

Navigation Drawer and Tabs

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/android-book.

4.1 Activity Switching

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/android-book.

4.2 App Navigation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/android-book.

4.3 Navigation Drawer

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/android-book.

4.4 tabbed navigation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/android-book.

5-Persistence-1

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/android-book.

Shared Preferences

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/android-book.

5.1 shared preferences:

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/android-book.

5.2 Initialise Shared Preference object:

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/android-book.

5.3 Storing the value:

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/android-book.

5.4 Retrieve the shared preference value:

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/android-book.

5.5 Best Practices For Enterprise apps

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/android-book.

5.6 preference screen(activity):

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/android-book.

6-Persistence-2

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/android-book.

sqlite database

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/android-book.

6.1 Insert data

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/android-book.

6.2 Data retrieval

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/android-book.

7-Api and network call

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/android-book.

HTTP Web services

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/android-book.

7.1 what is api

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/android-book.

7.2 HTTP background

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/android-book.

7.3 Example of api

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/android-book.

7.4 Libraries to implement api

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/android-book.

7.5 Example implementation api using async client

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/android-book.

7.6 AsyncTask

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/android-book.