Android Developer Fundamentals

Practical no. 1.

Aim: Install Android Studio and Run Hello World App.

Theory: Below are the steps to install Android Studio

Task 1 : Install Android Studio

1. Navigate to the Android developers site and follow the instructions to download and install Android Studio.
2. Accept the default configuration for all steps, and ensure that all components are selected for installation.
3. After finishing the install, the setup wizard will download and instal some additional components including the Android SDK. Be patient, this might take some time depending on your internet speed, and some of the steps may seem redundant.
4. When the downloaded components complete, Android studio will start, and you are ready to create your first project.

P.T.O

Task 2 : Create the Hello world App.

1. Open Android Studio, this might need internet connection.
2. Click "Start a new Android Studio Project."
3. Set the Application name as "Hello World."
4. Verify the default project location, set language as Java and minimum SDK as kitkat. or whatever you prefer. Click Finish.
5. Select Empty Activity and verify its name. Click finish.
6. In the activity_main.xml, there would be a textview in that set the android: text property to whatever text you want the screen to have. ex. "Hello world!".
7. Explore the interfaces, project layouts and gradle files.
8. Run the app (two ways):
   1. Using USB debugging on your own phone
   2. Using Emulator.

Methods used: onCreate()

~~Conclusion~~ Classes used: MainActivity.java

Conclusion : Successfully executed Hello world App.

Figure:

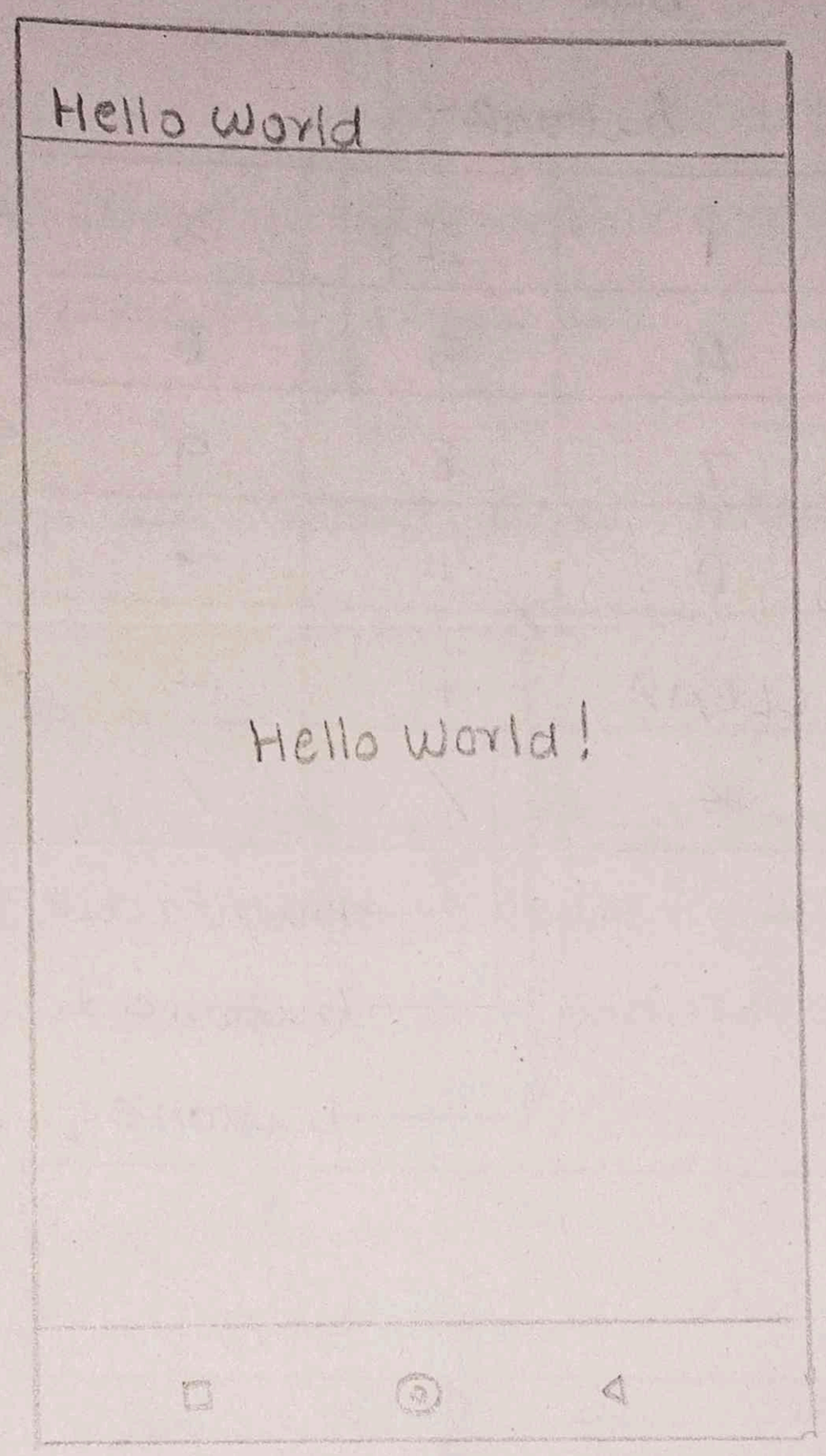
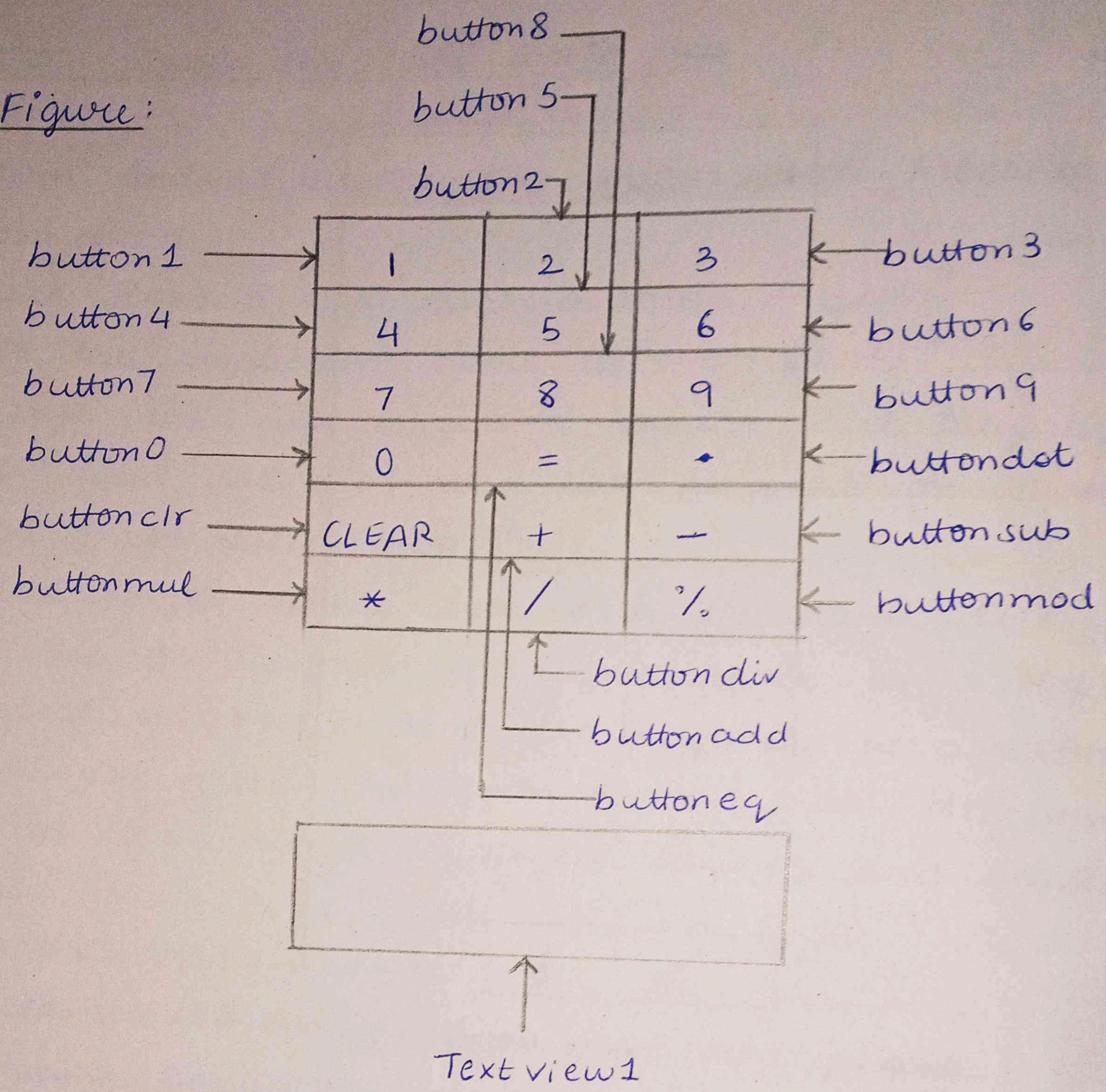


Figure:


button8
button5
button2
button1
button3
button4
button6
button7
button9
button0
buttondot
buttonclr
buttonsub
buttonmul
buttonmod
1
2
3
4
5
6
7
8
9
0
=
.
CLEAR
+
-
*
/
%
buttondiv
buttonadd
buttoneq
Textview1

Practical no. 2.

Aim: Create an android calculator app with Interactive user Interface using appropriate layouts.

Theory: Layouts used are below:

1. **Linear layout:**

Linear layout is a view group that aligns all the children (components) in a single ~~directory~~ direction, vertically or horizontally.

2. **Relative layout:**

- Relative layout is a view group that displays child views in relative positions.
- The position of each view ~~group~~ can be specified as relative to sibling elements (such as left-of or below another view) or in positions relative to the parent Relative Layout area.

Calculator Logic: We create many buttons which when clicked calls the onClick().

This way we replicate a real calculator.

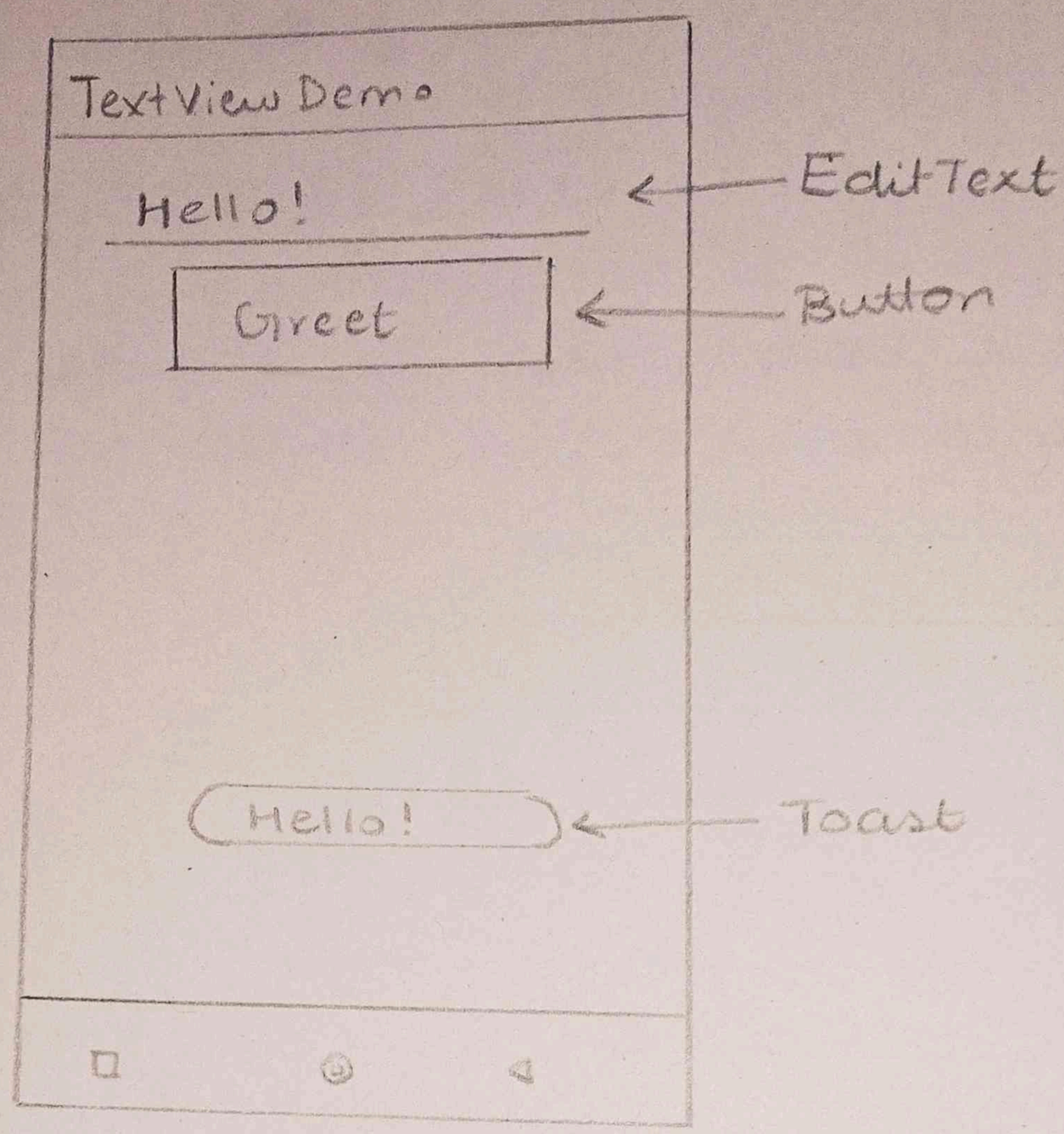
The keys (buttons) are 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 and symbolic labelled buttons are =, -, +, ., /, *, %, CLEAR (this resets the textview and variables)

Classes used: mainActivity.java

Methods used: onCreate(), onClick()

Conclusion: Calculator app was created successfully and desired outputs were obtained.

Figure :



Practical no. 3

Aim: Create an android app that demonstrates working with TextView Elements.

Theory: Text View displays text to the user and ocassionally allows them to edit it. A text view is a complete text editor, however the basic class is configured to not allow editing. Its very similar to labels which only display text to user.

Attribute	Description
• android : id	It is used to uniquely identify the control
• android : width	It makes the Textview be exactly this many pixels wide.
• android : height	It makes the Textview be exactly this many pixels tall.
• android : text	It is used to display the text.
• Android : hint	Similar to Android : text , but this makes text appear to user but doesn't make the value of the text view the given (displayed) text.

Edit Text : This takes value from user which can be further processed to get the desired output. It's attributes are similar to TextView.

Classes used : mainActivity.java

methods used : onCreate(), onClick()

Conclusion : We demonstrated the use of text view elements and displayed the user input as desired.

Practical no. 4

Aim: Create an android app that demonstrates Activity lifecycle and Instance State.

Theory:

Android Activity Lifecycle:

In android, Activity represents a single screen with a user interface (UI) of an application and it will act as an entry point for users to interact with an app.

We generally use many screens i.e. activities so we have one of them as main activity. That is our app's first screen for the user.

Every Activity in android has lifecycle like created, started, resumed, paused, stopped or destroyed. These different states are known as Activity Lifecycle. In other words we can say Activity is a class pre-written in Java programming.

We need to register those activitie's information in our app's manifest file (AndroidManifest.xml)

Example:

```
<? xml version="1.0" encoding="utf-8"?>
<manifest ...>
    <application ...>
        <activity android:name=".MainActivity">
            ... </activity>
    </application>
</manifest>
```

700

Callbacks methods which handle activity transitions between states during lifecycle:

1. On Create()

During the activity creation, activity entered into a Created state. This is first callback method.

2. on Start()

This method will invoke when activity entered into Started State. It makes activity visible to user.

3. on Resume()

It is invoked during Resumed state. Mostly the core functionality of app is implemented in this method.

4. on Pause()

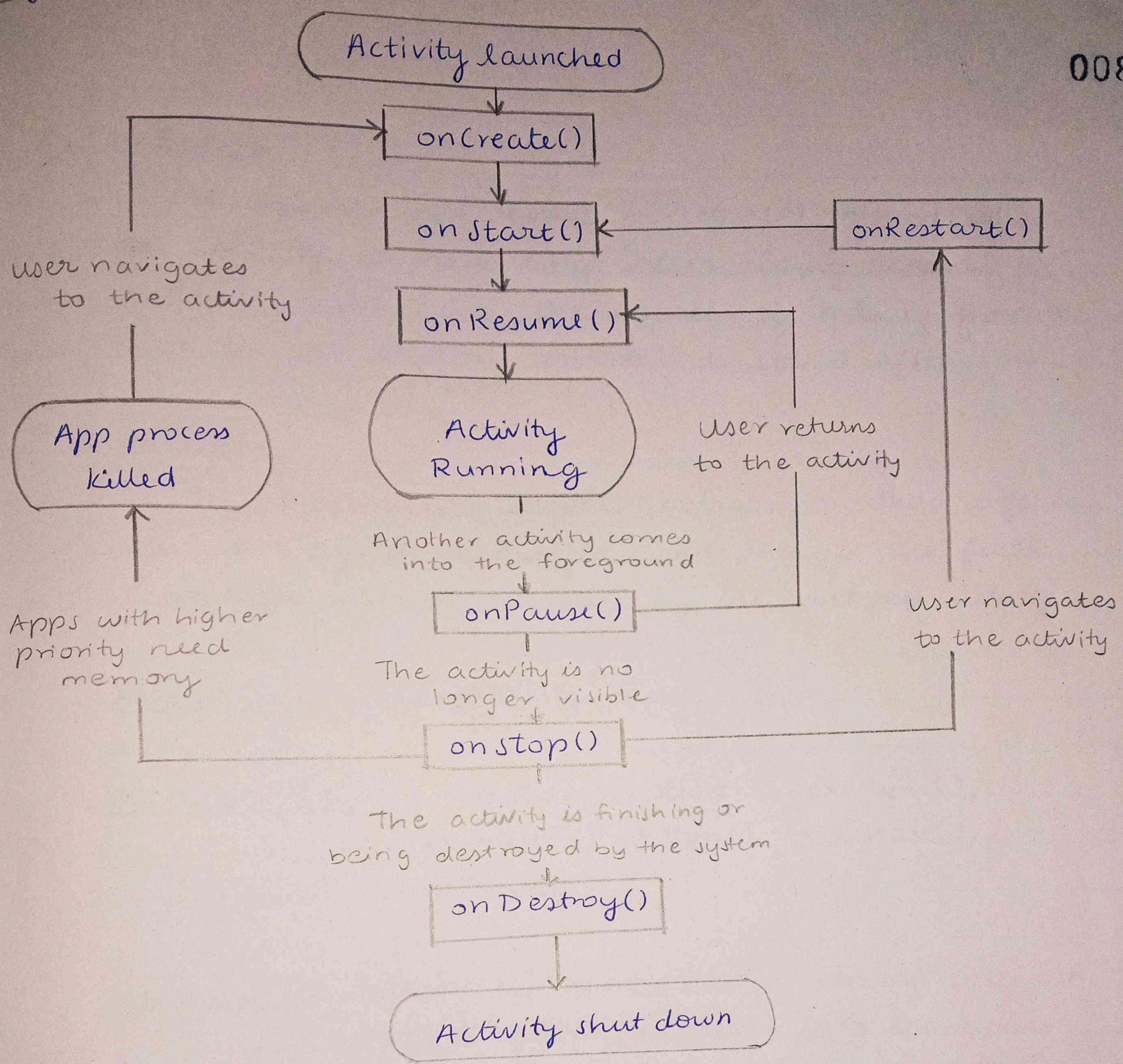
If any interruptions happen during onResume() this method is invoked and system is in Paused state.

5. on stop()

After onPause() either onstop() or onResumed() is called. In onstop() all the app resources which are no longer used are released. Activity will enter stopped state.

P.T.O.

Figure:


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process Killed
Activity Running
User returns to the activity
Another activity comes into the foreground
onPause()
User navigates to the activity
Apps with higher priority need memory
The activity is no longer visible
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

6. onRestart:

The system will invoke onRestart() method when an activity is restarting after being stopped. onRestart will save the state of activity from the time it was stopped. onStart() is called after this.

7. onDestroy:

The system will invoke onDestroy() method before an activity is destroyed. and this is the final callback method recieved by the android activity.

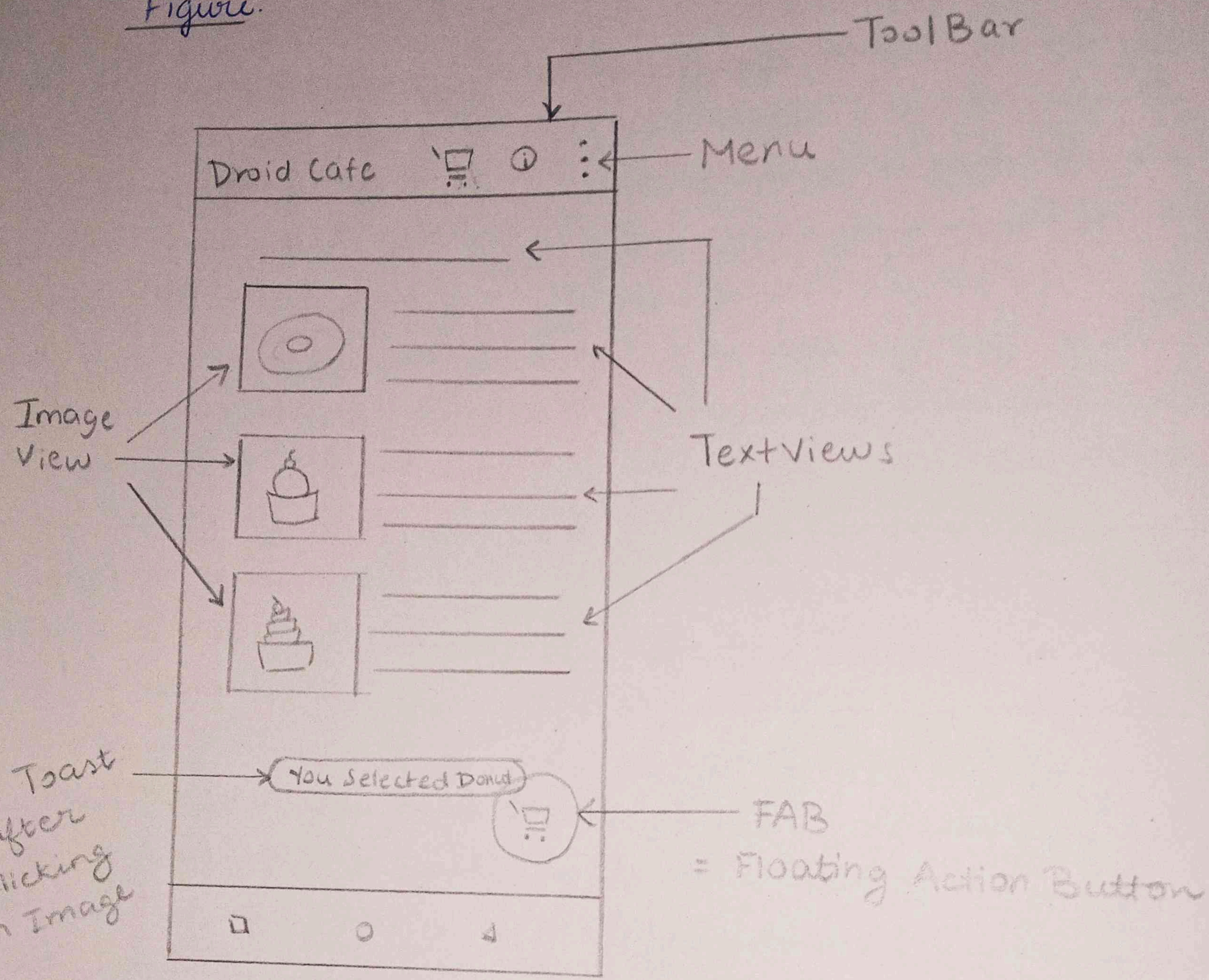
Classes used: MainActivity.java

Methods used: onCreate(), onStart(), onResume(), onPause(), onStop(), onRestart(), onDestroy().

Conclusion: Successfully demonstrated various callback methods which handle activity transitions between states during activity lifecycle.

C10

Figure:


ToolBar
Droid Cafe
Menu
Image View
TextViews
Toast after clicking on Image
You Selected Donut
FAB
= Floating Action Button

Practical no. 5.

Aim: Create an Android app that demonstrates the use of Keyboards, Input Controls, Alerts, and Pickers.

Theory: Layouts used:

1. CoordinatorLayout : It is a superpowered FrameLayout. It coordinates the animations and transitions of child views with one another.
2. AppBarLayout : It is a vertical LinearLayout which implements many of the features of material designs app bar concept, namely scrolling gestures.
3. ConstraintLayout : It is a view group which allows you to position and size widgets in a flexible way.

Components of this application.

1. Tool bar: It is like an action bar and it can contain nav. buttons, brand logo, appicon, title and subtitle.
2. Floating Action button: FAB is a circular icon that hovers over content to promote a primary action in the application.
3. Options menu: This is pre made if we choose primary i.e. basic activity. This is three vertically arranged dots icon which opens list of items.

4. TextView, EditText, RadioButton, RadioGroup and Imageview: This displays image in our view. This image needs to be added to drawables first. right click on Drawables --> new --> vector Asset then select image needed.

5. menus

Syntax: <menu xmlns: ... >

```
    <item>
      // parameters are id, orderInCategory, title, showAs Action.

    </item>
      :
    // other items
      :
</menu>
```

6. Two xml layouts and two java activities respectively.
   1. For Buying selecting item.
   2. For registering user and to order.

Conclusion: We created a desert selling app that takes our order from view and shows it in another activity. Also asks for our order & delivery details successfully displays toast messages.

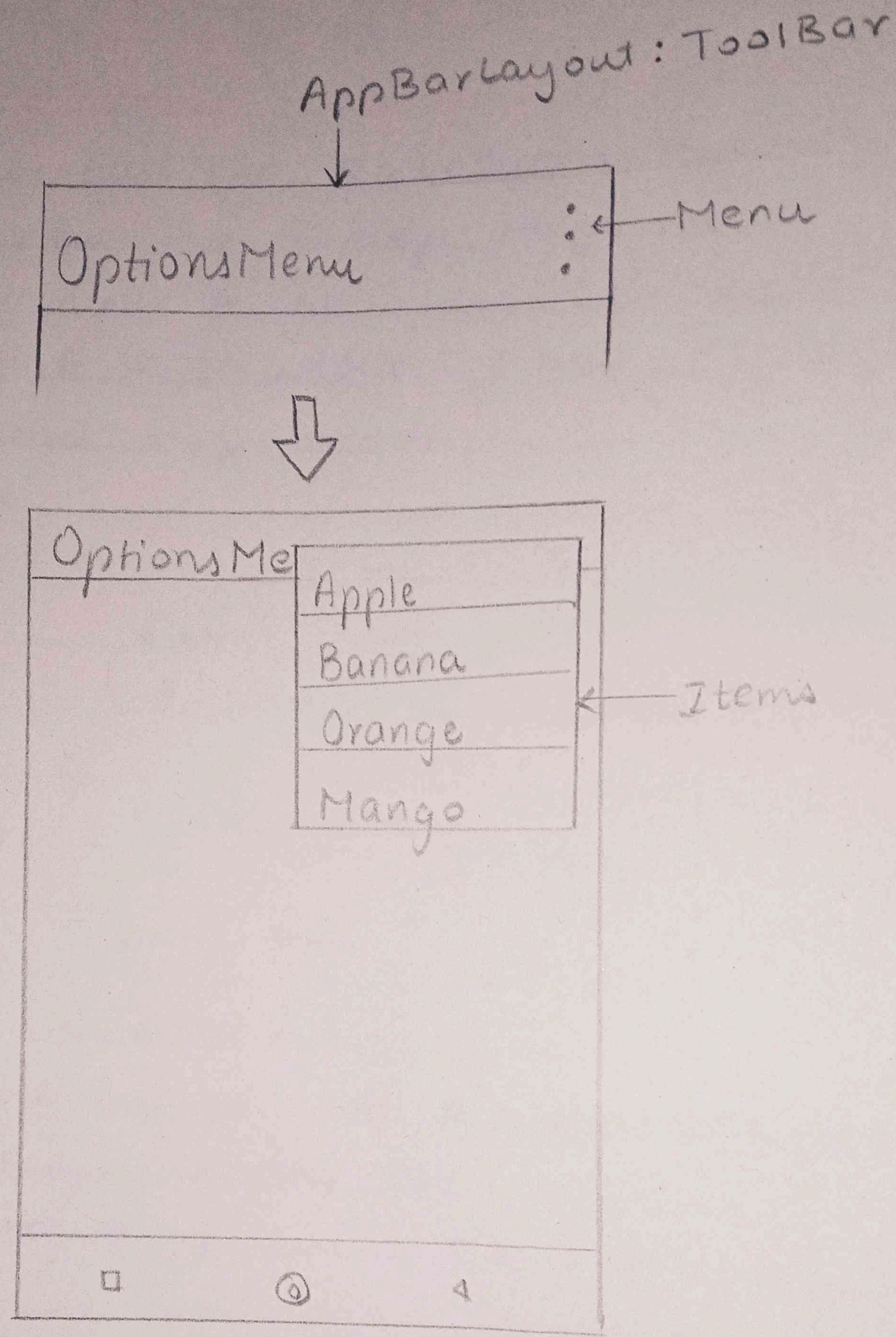
Method used: onCreate(), setOnClickListner(), onClick(), onCreateOptionsMenu(), getMenuInflator(), inflate(), onOptionsItemSelected(), displayToast(), showDonutOrder(), showIceCreamOrder(), showFroyoOrder() onRadioButtonClicked(),

→

Classes used : mainActivity.java, OrderActivity.java

S10

Figure:


AppBarLayout : ToolBar
OptionsMenu
Menu
OptionsMe
Apple
Banana
Orange
Mango
Items

Practical no. 6

Aim: Create an android app that demonstrates the use of an Options menu.

Theory:

- Menu:

Android provides an easy and flexible user interface component known as menu, which is used to handle a set of actions. menu is a set of options, the user can select from to perform an action. Based upon the menu item clicked, a certain action can be performed by the app such as add, search, delete, edit, or refresh etc. Android menu comes from Android 3.0 (API level 11).

- Types of menu.
  1. Options menu
  2. Content menu
  3. Pop up menu.

- Option menu:

Android option menu is the primary collection of menu items for an activity. It is the place where we implement the actions that have a global impact on the app, such as search, Settings and Delete.

→

Options menu appears in the AppBarLayout and provides the primary options.

To add items to the menu, go to menu_main.xml and add them there, its arguments can be changed according to need. The parameters are id, title, etc.

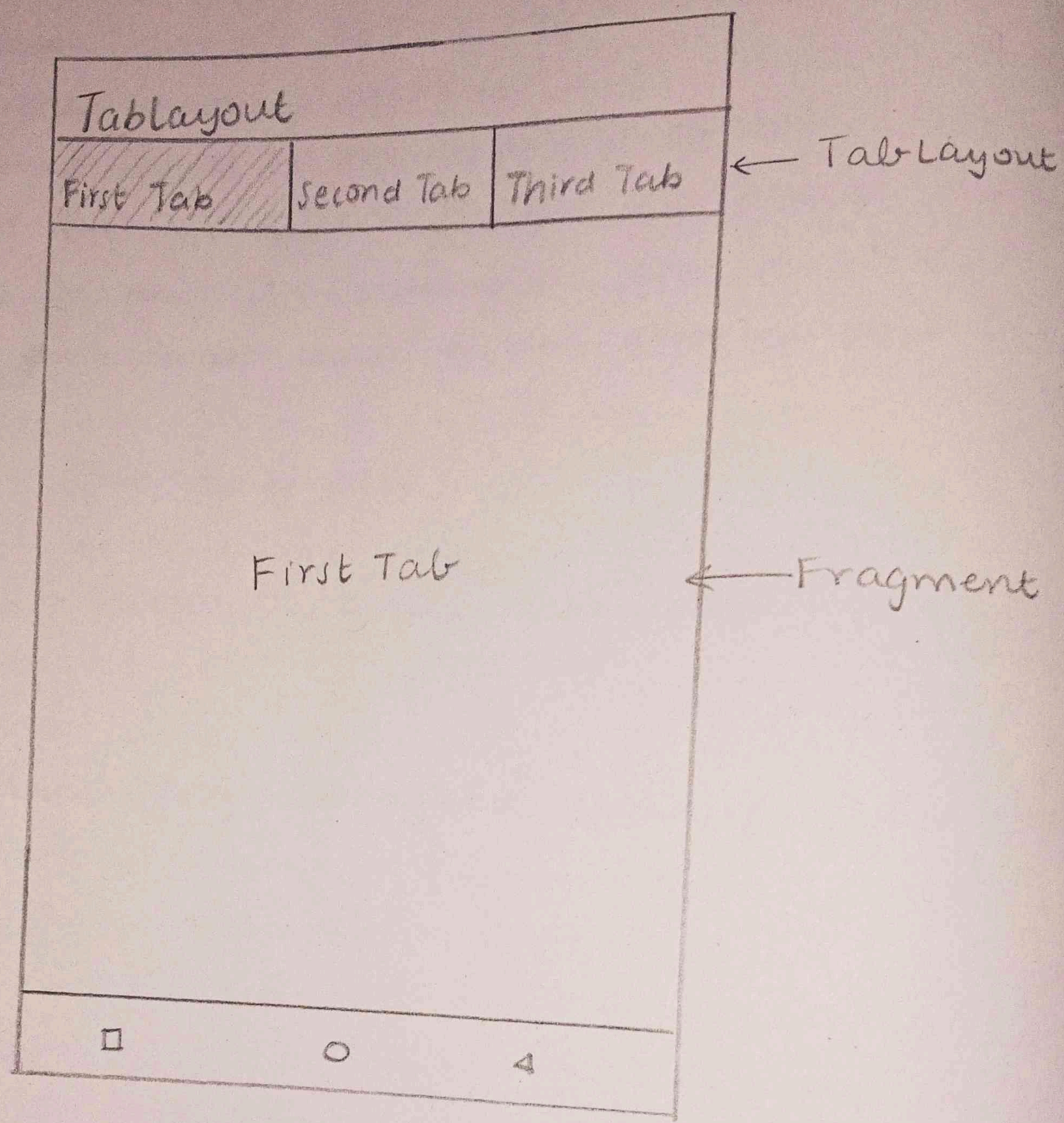
This app gives toast based on the item we select.

Classes used: MainActivity.java,

Methods used: onCreate(), onCreateOptionsMenu(), onOptionsItemSelected().

Conclusion: successfully demonstrated the use of option menu.

Figure:


Tablayout
First Tab
Second Tab
Third Tab
TabLayout
First Tab
Fragment

Practical no. 7.

Aim: Create an android app that demonstrates Screen Navigation using the App Bar and Tabs.

Theory:

App Bar and Tabs:

Tab layout is used to implement horizontal tabs. TabLayout is released by Android after the depreciation of Action Bar.

Tabs are created using newTab() method of TabLayout class. The title and icon of Tabs are set through setText(int) and setIcon(int) method. of TabListner interface respectively. Tabs of layout are attached over TabLayout using the method addTab(Tab) method.

For our project we need to select tabbed activity when prompted.

Layouts used:

- Constraint layout
- AppBarLayout
- RelativeLayout
- TabLayout: It is used to implement horizontal tabs.

ViewPager allows us to swipe to another fragment.

→

Fragments :

A fragment is a part of the application's user interface that is bound to an activity. Fragments have their lifecycle and layouts or UI components. Fragments help enrich your UI design, pass data between different screens, and adapt to different device configurations.

Classes used : MainActivity.java, PagerAdapter.java, FirstFragment.java, SecondFragment.java, ThirdFragment.java

Methods used : onCreate(), TabSelected(), TabUnselected(), TabReselected(), PagerAdapter(), ~~Fragment~~ getItem(), getCount(), onCreateView(),

Conclusion : Successfully demonstrated Screen Navigation using AppBar and Tabs.

Practical no. 8

Aim: Create an android app to Connect to the Internet and use Broadcast Receiver.

Theory: Broadcast Receiver:

Broadcast Receiver is an Android component which you to send or receive Android system or application events. For example, applications can register for various systems events like boot complete, battery low, airplane mode turned on, data connected, data disconnected (internet), etc. Broadcast in android is the system-wide events that can occur when the device when the device starts, when a message is received on the device or when incoming calls are received, etc. Broadcast Receivers are used to respond to these system-wide events.

Example of Broadcast Reciever is an incoming call, notifications, battery low alert, etc.

We need to add receiver tag with exported = true in Android manifest.xml file in main folder.

→

Classes used: NetworkUtil.java, myReceiver.java, mainActivity.java.

Methods used: getConnectivityStatusString(), onReceive(), onClick(), onCreate(), broadcastIntent(), onPause().

Conclusion: Demonstrated use of Broadcast Receiver to check internet connectivity successfully.

Practical no. 9

Aim: Create an android app to show notifications for alarm manager.

Theory:

To create an alarm application we used Broadcast Receiver. It takes our mp3 file from sd card and plays it when our alarm goes off. Here we set the alarm in seconds for the demonstration I have set ~~if~~ it for 5 seconds. To set it for some time say 3 minutes we convert minutes to second like 3 * 60 = 180 and we enter 180 as input in EditText field and run it.

Layouts used: 1. Relative Layout.

Classes used: myBroadcastReceiver.java, mainActivity.java

Methods used: onReceive(), onCreate(), onClick(), startAlert().

Conclusion: Accomplished to run alarm app using Broadcast Receiver.