MOBILE APPLICATION DEVELOPMENT

Syllabus

Unit-1

GETTING STARTED WITH ANDROID: Android introduction, Versions of android, Features of android, Architecture, Devices in the market, Developer community

Unit 2

ACTIVITIES, FRAGMENTS, INTENTS: Understanding activities, Linking activities using intents, Fragments, Calling built in apps using intents

Unit 3

GETTING TO KNOW ANDROID UI: Understanding the components of screen - Views and view groups, Liner layout, Absolute layout, Table layout, Relative layout, Frame layout, Scroll view

Unit 4

DISPLAY ORIENTATION: Anchoring views, Resizing and repositioning views, Managing changes to screen orientation, Utilizing the action bar, Creating UI programmatically

Unit 5

DESIGNING UI WITH VIEWS: Using basic views - Text view, Button, Image Button, Edit text, check Box, Toggle button, Radio button, and Radio group views, Progress bar view and Auto complete text view

Text book: "Beginning Android 4 Application Development" Author name: Wei-Meng Lee and WILEY publications

UNIT I: GETTING STARTED WITH ANDROID

1.What is Android and History of Android? [IT-REG-2019]

Definition: Android is a mobile platform. It consists of

- Operating system
- Middleware
- Key Applications

- It is first developed by a silicon valley by the name of Android Inc.
- The operating system used in the android is a Linux Kernel, and also is a open source code.
- Development of android done through windows, Linux or Mac. But primarily written in Java
- Instead of to run java programs through the JVM (Java virtual Machine), Google developed DVM (Dalvik virtual machine) for android.
- DVM runs java byte code (.classs) and read as Dalvik byte code and is used to optimize battery power , limited memory and CPU time.
- So it is more suitable for mobile phones, notebooks and tablet Pcs
- Advantage is it is easily developed.
- The first beta version of the Android Software Development Kit (SDK) was released by Google in 2007, it is a commercial version.
- Android 1.0 was released in sep 2008
- Google announced next version is 4.1, Jelly Bean. It is a update of improving user interface in terms of functionality and performance.
- Open Handset Alliance was formed in November 2007.It was created by Google. The OHA is a business alliance that consists of 47 companies for developing open standard platform for mobile devices.
- It includes handset manufactures, chip makers, commercialization companies, software companies, and mobile operators.

2.Discuss about the versions of Android? [IT-REG-2019] [[IT-REG-2021] [IT-SUP-2021]

Android Version	**Release Date**	**API Level**	**Code Name**
Android 1.0		1	BASE
Android 1.1	9th Feb 2009	2	BASE_1_1
Android 1.5	30th Apr 2009	3	CUPCAKE
Android 1.6	15th Sep 2009	4	DONUT
Android 2.0	26th Oct 2009	5	ECLAIR
Android 2.0.1		6	ECLAIR_0_1
Android 2.1.X		7	ECLAIR_MR1
Android 2.2.X	20th May 2010	8	FROYO

Android 2.3 Android 2.3.1 Android 2.3.2	6th Dec 2010	9	GINGERBREAD
Android 2.3.3 Android 2.3.4		10	GINGERBREAD_MR1
Android 3.0.X		11	HONEYCOMB
Android 3.1.X		12	HONEYCOMB_MR1
Android 3.2		13	HONEYCOMB_MR2
Android 4.0 Android 4.0.1 Android 4.0.2		14	ICE_CREAM_SANDWICH
Android 4.0.3 Android 4.0.4		15	ICE_CREAM_SANDWICH_MR1
Android 4.1 Android 4.1.1		16	JELLY_BEAN
Android 4.2 Android 4.2.2		17	JELLY_BEAN_MR1
Android 4.3		18	JELLY_BEAN_MR2
Android 4.4		19	KITKAT
Android 4.4 W		20	KITKAT_WATCH
Android 5.0		21	LOLLIPOP
Android 5.1		22	LOLLIPOP_MR1
Android 6.0		23	MARSHMALLOW

- API level uniquely identifies the framework API version offered by a version of the android platform.
- CUPCAKE : Speech recognition tools, a virtual keyboard, video upload support for YouTube and support for live data feeds and live folders.
- DONUT: Support for CDMA smartphones, additional screen sizes and a text-to-speech engine.
- ECLAIR: Support for multi-touch devices, new browser interface, Microsoft Exchange support, single interface for managing multiple online accounts, soft keys support, and an enhanced camera app (with digital zoom and flash support).
- FROYO: USB tethering support (for turning a smartphone into a Wi-Fi hotspot), significant speed improvements, Flash 10.1 support, voice dialing over Bluetooth, the ability to store apps on external memory cards, updated browser with Google Chrome's V8 JavaScript.
- GINGERBREAD: Debuted in December 2010: Key additions: Google Voice over Wi-Fi, enhanced gaming functionality, improved Google Apps.
- HONEYCOMB: A tablet-centric update that delivered a new interface optimized for devices with larger screen sizes (particularly tablets), video chat support based on Google Talk protocols, new System Bar for global status and notifications and Action Bar for application control, tabbed Web browsing, optimized soft keyboard and a new email interface.

- ICE_CREAM_SANDWICH: A smartphone-centric update based on the Linux kernel v3.0.1 that brings many of Honeycomb's features to smartphones, including Face Unlock facial recognition software, tabbed Web browsing capabilities, unified social networking contacts, 1080p video recording capabilities and video chat support based on Google Talk protocols.
- JELLY_BEAN: Advanced natural language voice command capabilities akin to Apple's Siri, enhanced interface and overall responsiveness via "Project Butter," Google Now support, an improved Web browser, enhanced file management capabilities and more.
- KITKAT: Full-screen immersive mode, new transitions framework, and "Project Svelte," a project initiated to reduce the memory needs of the Android OS. Originally internally referred to as Key Lime Pie, Google announced in early September 2013 that it would be using the iconic candy bar as the codename for the 4.4 Android release.
- LOLLIPOP: Enhanced Material Design user interface, improved continuity across Android devices, multiple user support, a guest user account option, a new notification system, support for 64-bit CPUs, and more.
- MARSHMALLOW: Now on Tap functionality and other Google Now enhancements, native fingerprint authentication support, Android Pay integration, USB Type-C support, improved battery life, better app management and more.
- OREO: Android Go minimized version of Android mobile OS, picture-in-picture video support, Notification Grouping, Bluetooth 5 support, Wi-Fi Aware feature, and more.
- PIE: Adaptive Battery, Digital Well-being Dashboard (monitors screen time usage), App Slices, Intuitive Gestures, Adaptive Brightness, and more.

3. What are the characteristics and benefits of mobile apps? [IT-REG-2021] (OR) What are the Features of Android. [IT-REG-2019] [IT-SUP-2021] [MCA-REG-2022] [BCA-SUP-2021]

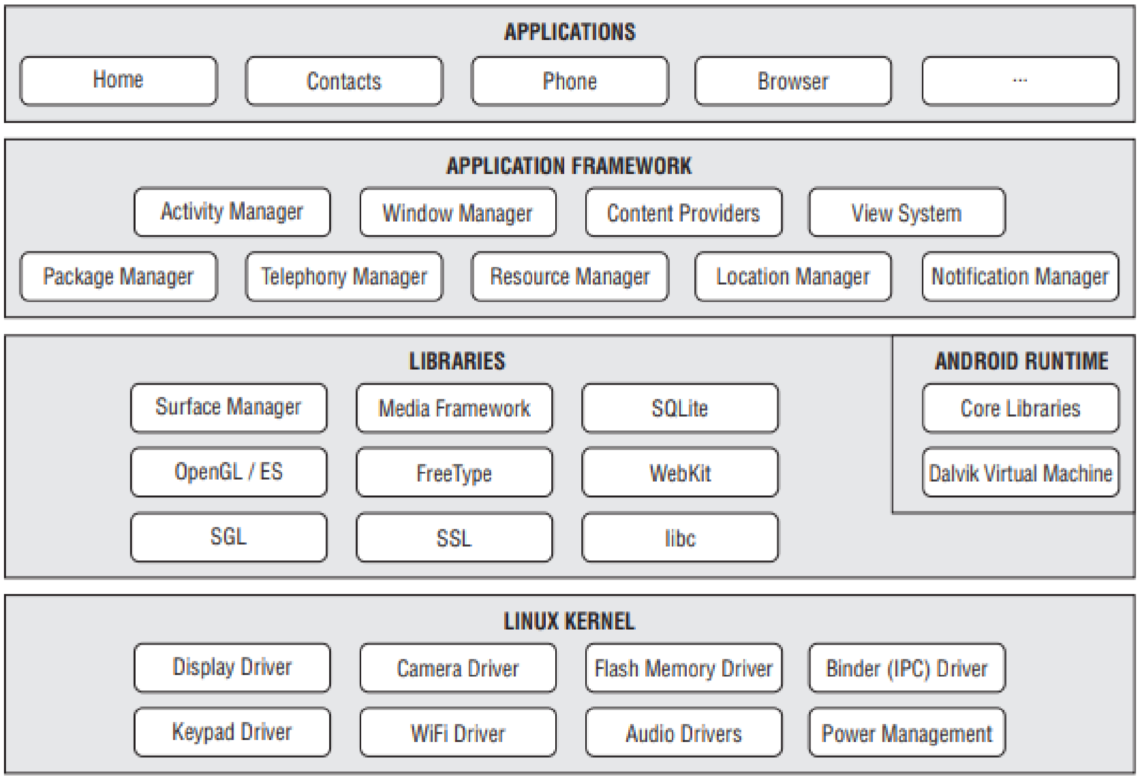
Android is open source and freely available. Android itself supports the following features:

- **Open Source —** Android is an open source operating system .So that anyone can work on android not only the one company developers. Therefore app developers for android devices are able to implement extra features of their apps.
- **Storage** — Uses SQLite, a lightweight relational database, for data storage.
- **Connectivity** — Supports GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth (includes A2DP and AVRCP), WiFi, LTE, and WiMAX.
- **Messaging** — Supports both SMS and MMS.
- **Web browser** — Based on the open-source WebKit, together with Chrome's V8 JavaScript engine
- **Media support** — Includes support for the following media: H.263, H.264 (in 3GP or MP4 container), MPEG-4 SP, AMR, AMR-WB (in 3GP container), AAC, HE-AAC (in MP4 or 3GP container), MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, and BMP.
- **Hardware support** — Accelerometer Sensor, Camera, Digital Compass, Proximity Sensor,and GPS
- **Multi-touch** — It Supports for multi-touch which was initially made available in handsets such as the HTC Hero.
- **Multi-tasking** — Supports multi-tasking applications
- **Flash support** — Android 2.3 supports Flash 10.1.
- **Tethering** — Supports sharing of Internet connections as a wired/wireless hotspot.

- **Accesibity** — Built in text to speech is provided by Talk back for people with low or no vision.
- **External storage** — Most Android devices include microSD slot and can read microSD cards formatted with FAT32, Ext3 or Ext4 file system. To allow use of high-capacity storage media such as USB flash drivers and HDDs, many android tablets include USB 'A' receptacle.
- **Voice based features** —Google search through voice has been available since opening release. Voice activities for navigation, calling, texting, etc. are supported on android 2.2 forwards. Android 4.1 Google has expanded voice Actions with the ability to talk back and read answers from Google's Knowledge Graph when queried with specific commands.

4.Draw and briefly explain about Android Architecture? [IT-REG-2019] [IT-REG-2021] [IT-SUP-2021] OR Explain the application layer components in android Architecture [[IT-REG-2021] OR What is android? Explain with the components of android stack with neat diagram. [BCA-REG-2021] OR Explain the Android Architecture? [MCA-REG-2022] [BCA-SUP-2021]

The Android OS is a software stack of different layers. Each layer contains a group of several program components.
Android has four layers and five sections:
1. Applications
2. Application Framework
3. Libraries
4. Android runtime
5. Linux kernel



- First two layers are in blue colour that means these are written in Java.
- The Libraries are in green color then it goes to c or c++ code libraries. The communication between the java and c,c++ by using the JNI(Java network interface)
- The colour yellow is Android runtime that means then it is developed, which executes a virtual machine, native libraries.
- The linux kernel is red in colour , which includes low level drivers like hardware, networking, USB Driver, Bluetooth driver, camera driver etc.

➤➤Linux kernel — This is the kernel on which Android is based. This layer contains all the lowlevel device drivers for the various hardware components of an Android device.

Android runtime gives .dex file which converted from the class file. .dex file goes to the linux kernel and call the suitable drivers so that it can communicate with corresponding drivers.

The following are the drivers:

- Display Driver
- Camera Driver
- Bluetooth Driver
- Flash memory driver
- USB driver
- WiFi driver
- Keypad Driver
- Audio Driver
- Power management driver

➤➤Libraries — These contain all the code that provides the main features of an Android OS. For example, the SQLite library provides database support so that an application can use it for data storage. The WebKit library provides functionalities for web browsing.

- Open GL/ES: This component provides mostly 3D graphics libraries which have to be implemented by java application it is rendered by the application layer when they want some 3D graphics.
- SQLite: when our application want to connect the database then we can think of some popular RDBMS like MySQL, Oracle. But they are heavy weight database server as our applications are light weight.so better to have light weight RDBMS.
- Media Frame work: Picture quality, Multimedia action should be good which include the sounds
- SGL (scalable graphics libraries): It is responsible for implementing low level graphics by using JNI.
- SSL (secured socket layer): It supports http and tcp protocol. It is mostly for internet application. Suppose online through mobile we want to access our online through mobile then at that time we need some secure socket layer.
- WebKit: It is responsible for browser supports. It supports following browsers
  Google chrome.
  Apple safari
  Javascript, JQuery, HTML, CSS, Ajax.

- Lib c: It stands for C libraries, when we want to access some code c libraries headers then this component is useful.

➢➢ Android runtime — At the same layer as the libraries, the Android runtime provides a set of core libraries that enable developers to write Android apps using the Java programming language. The Android runtime also includes the Dalvik virtual machine, which enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine (Android applications are compiled into the Dalvik executables).

The DVM allows memory management, security, isolation, and threading support. It is developed by Dan Bornstein of Google.

Dalvik is a specialized virtual machine designed specifically for Android and optimized for battery-powered mobile devices with limited memory and CPU.

Core libraries : It provides the functionalities defined in the Java SE libraries.

- Data structures
- File Access
- Network Access
- Utilities
- Graphics

➢➢ Application framework — Exposes the various capabilities of the Android OS to application developers so that they can make use of them in their applications.

This is the second layer. It contains the following components.

1. Activity Manager: This layer manages the lifecycle of applications and provides a common navigation back.
2. Window Manager: It manages the window surface. Then it organizes the screen layout and locates the drawing surface and also it performs other windows related jobs.
3. View Manager: Views the window.
4. Content Manager: Enables applications to access data from other applications or to share their own data.
5. Notification Manager: We get the notification from the system when the battery is low. And also we get the alert messages in the status bar.
6. Package Manager: This manager the other packages if it is instant in the machine. Sometimes we may download we may download some applications which is not in own.
7. Telephony Manager: It is to handle the receiver call or voice calls.
8. Resource Manager: This manager provides access to non-code resources (layout files).
9. Location Manager: This helps to locate the mobile devices at which the location of the mobile device can be searched. Ex. Map, GPs etc.

➢➢ Applications — At this top layer, you will find applications that ship with the Android device (such as Phone, Contacts, Browser, calender etc.), as well as applications that you download and install from the Android Market. Any applications that you write are located at this layer.

5.List out some android devices available in market. Explain. OR Explain android devices in market with suitable example. [BCA-REG-2021] [MCA-REG-2022]

Android devices come in all shapes and sizes. As of late November 2010, the Android OS can be seen powering the following types of devices:

- Smartphones
- Tablets
- E-reader devices
- Netbooks
- MP4 players
- Internet TVs

The Samsung Galaxy S, the HTC Desire HD, and the LG Optimus One smartphones. Another popular category of devices that manufacturers are rushing out is the tablet. Tablet sizes typically start at seven inches, measured diagonally. The Below are Samsung Galaxy Tab and the Dell Streak, which is a five inch phone tablet.

Besides smartphones and tablets, Android is also beginning to appear in dedicated devices, such as e-book readers. The following is the Barnes and Noble's NOOK color, which is a color e-Book reader running the Android OS.

In addition, Android is also slowly finding its way into your living room. People of Lava, a Swedish company, has developed an Android-based TV, call the Scandinavia Android TV. Google has also ventured into a proprietary smart TV platform based on Android and co-developed with companies such as Intel, Sony, and Logitech.

6.Explain about Android Market.

- ➢ One of the main factors determining the success of a smartphone platform is the applications that support it.
- ➢ It is clear from the success of the iPhone that applications play a very vital role in determining whether a new platform swims or sinks.
- ➢ These applications accessible to the general user is extremely important.

- In August 2008, Google announced the Android Market, an online application store for Android devices, and made it available to users in October 2008.
- Using the Market application that is preinstalled on their Android device, users can simply download third-party applications directly onto their devices.
- Both paid and free applications are supported on the Android Market, though paid applications are available only to users in certain countries due to legal issues.
- Similarly, in some countries, users can buy paid applications from the Android Market, but developers cannot sell in that country.
- As an example, at the time of writing, users in India can buy apps from the Android Market, but developers in India cannot sell apps on the Android Market.
- The reverse may also be true; for example, users in South Korea cannot buy apps, but developers in South Korea can sell apps on the Android Market.

7.Mention the List of Android Developer Communities help us to develop mobile apps?

The following communities help us to develop apps, Those are:

1. Stack Overflow
2. Google Android Training
3. Android Discuss

- **Stack Overflow** (www.stackoverflow.com) — Stack Overflow is a collaboratively edited question and answer site for developers. If you have a question about Android, chances are someone at Stack Overflow is probably already discussing the same question and someone else had already provided the answer. Best of all, other developers can vote for the best answer so that you can know which are the answers that are trustworthy.
- **Google Android Training** (http://developer.android.com/training/index.html) — Google has launched the Android Training site that contains a number of useful classes grouped by topics. At the time of writing, the classes mostly contain useful code snippets that are very useful to Android developers once they have started with the basics. Once you have learned the basics in this book, I strongly suggest you take a look at the classes.
- **Android Discuss** (http://groups.google.com/group/android-discuss) — Android Discuss is a discussion group hosted by Google using the Google Groups service. Here, you will be able to discuss the various aspects of Android programming. This group is monitored closely by the Android team at Google, and so this is good place to clarify your doubts and learn new tips and tricks.

8.What is Android Architecture? [BCA-REG-2021]
The Android OS is a software stack of different layers. Each layer contains a group of several program components.
Android has five layers:

1. Applications
2. Application Framework
3. Libraries

4. Android runtime
5. Linux kernel

9. List out at least four versions of Android. [BCA-REG-2021]

- CUPCAKE
- DONUT
- ECLAIR
- FROYO:
- GINGERBREAD
- HONEYCOMD

10.Write differences between VUI and Mobile APP. [BCA-REG-2021]

SNO	Voice User Interface (VUI)	Mobile App
1.	Mobile interfaces are set by designers on behalf of machines and learned by humans.	Voice user interfaces (VUIs) are different because it’s all about teaching computers how to talk to humans, and that’s a huge shift.
2.	Mobile apps are costly to design and costly to build.	The lack of visuals and complex frontend makes voice apps very straightforward, and easy to build.
3.	Mobile apps are heavy. Every single mobile app requires you to download an app which takes up space on your phone, and time to download.	Voice apps are light because they are cloud-based.
4.	Smartphones are meant to have deep engagement that can entertain or help the user for a long duration of time.	Voice apps are not meant to be engaging but highly functional and fast.

UNIT II: ACTIVITIES FRAGMENTS AND INTENTS

1.What do you understand about Activity [IT-REG-2019] [IT-SUP-2021]

Activity: The activity represents a single screen or a interface that allows the user to interact with an application. The applications contains single to multiple activities.
Ex: facebook application have an activity to log into the application, and other activities are such as posting a photograph or activity to send a message to a friend.

```
packagenet.learn2develop.Activities;
import android.app.Activity;
 import android.os.Bundle;
public class MainActivity extends Activity
{
        public void onCreate(Bundle savedInstanceState)
        {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);
        }
}
```

The UI corresponding XML file is located **res/layout** folder. It would be loaded by calling the setContentView(R.layout.main) method.
Every activity in application must be declared in your AndroidManifest.xml file.

```
<?xml version="1.0" encoding="utf-8"?>
<Manifest xmlns:android="http://schemas.android.com/apk/res/android"
        Package="net.learn2develop.Activities"
        android:versionCode="1"
        android:versionName="1.0">
        <application
                android:icon="@drawable/icon"
                android:label="@string/app_name">
                <activity
                        android:name=".MainActivity"
                        android:label="@string/app_name">
                        <intent-filter>
                        <action android:name="android.intent.action.MAIN"/>
                        <category android:name="android.intent.category.LAUNCHER"/>
                        </intent-filter>
                </aactivity>
        </application>
</Manifest>
```

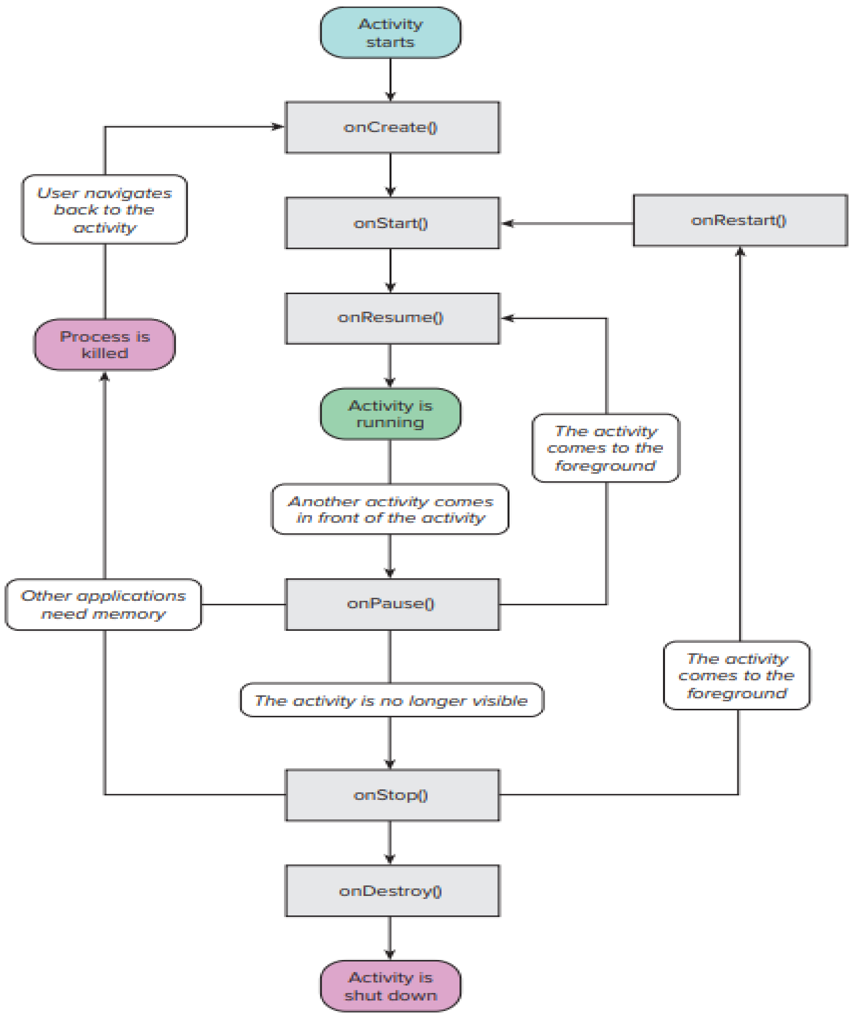
2. Explain the concept of Understanding the Life Cycle of an Activity. Example. [IT-REG-2019] [IT-SUP-2021] [BCA-SUP-2021]

The Activity base class defines a series of events that governs the life cycle of an activity.
The Activity class defines the following events:

- onCreate() — Called when the activity is first created

- onStart() — Called when the activity becomes visible to the user
- onResume() — Called when the activity starts interacting with the user
- onPause() — Called when the current activity is being paused and the previous activity is being resumed
- onStop() — Called when the activity is no longer visible to the user
- onDestroy() — Called before the activity is destroyed by the system (either manually or by the system to conserve memory)
- onRestart() — Called when the activity has been stopped and is restarting again Understanding

By default, the activity created for you contains the onCreate() event.
Within this event handler is the code that helps to display the UI elements of your screen.
The life cycle of an activity and the various stages it goes through from when the activity is started until it ends.


Activity starts
onCreate()
User navigates back to the activity
onStart()
onRestart()
onResume()
Process is killed
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
Other applications need memory
onPause()
The activity is no longer visible
The activity comes to the foreground
onStop()
onDestroy()
Activity is shut down

MainActivity.java
package net.learn2develop.Activities;
import android.app.Activity;
import android.os.Bundle;

```
import android.util.Log;
public class MainActivity extends Activity
{
 String tag = “Events”;
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);
                Log.d(tag, “In the onCreate() event”);
        }
        public void onStart() {
                super.onStart();
                Log.d(tag, “In the onStart() event”);
        }
        public void onRestart(){
                super.onRestart();
                Log.d(tag, “In the onRestart() event”);
        }
        public void onResume() {
                super.onResume();
                Log.d(tag, “In the onResume() event”);
        }
        public void onPause(){
                super.onPause();
                Log.d(tag, “In the onPause() event”);
        }
        public void onStop() {
                super.onStop();
                Log.d(tag, “In the onStop() event”);
        }
        public void onDestroy(){
                super.onDestroy();
                Log.d(tag, “In the onDestroy() event”);
        }
}
```
The execution of lifecycle of an activity is as follows:

When the activity is first loaded

```
In the onCreate() event
In the onStart() event
In the onResume() event
```

Press the back button of an Activity

```
In the onPause()event
In the onStop()event
In the onDestroy()event
```

Click the Home button and hold it

```
In the onCreate()event
In the onStart()event
```

In the onResume()event
Press the Phone button on the Android Emulator so that the activity is pushed to the background.
In the onPause()event
In the onStop()event
The onDestroy() event is not called, indicating that the activity is still in memory. Exit the phone dialer by pressing the Back button. The activity is now visible again.
In the onRestart()event
In the onStart()event
In the onResume()event
The onRestart() event is now fired, followed by the onStart() and onResume() events.

3. Explain about how to Applying Styles and themes to Activity

- An activity occupies the entire screen. To apply a dialog theme to an activity so that it is displayed as a floating dialog.
- For example, you might want to customize your activity to display as a pop-up, warning the user about some actions that they are going to perform. In this case, displaying the activity as a dialog is a good way to get their attention.
- AndroidManifest.xml file by adding the **android: theme"@android:style/Theme.Dailog"** attribute.

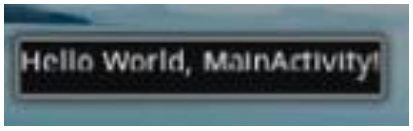
```
<?xml version="1.0" encoding="utf-8"?>
<Manifest xmlns:android="http://schemas.android.com/apk/res/android"
          Package="net.learn2develop.Activities"
          android:versionCode="1"
          android:versionName="1.0">
          <application
                    android:icon="@drawable/icon"
                    android:label="@string/app_name"
                    android:theme="@android:style/Theme.Dailog">
                    <activity
                              android:name=".MainActivity"
                              android:label="@string/app_name">
                              <intent-filter>
                              <action android:name="android.intent.action.MAIN"/>
                              <category android:name="android.intent.category.LAUNCHER"/>
                              </intent-filter>
                    </aactivity>
          </application>
</Manifest>
```

Hiding the Activity Title

To hide the title of an activity use the **requestWindowFeature()** method and pass it the **Window.FEATURE_NO_ TITLE** constant.

```
package net.learn2develop.Activities;
import android.app.Activity;
import android.os.Bundle;
```

```
import android.util.Log;
import android.view.Window;
public class MainActivity extends Activity {
        String tag = “Events”;
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                requestWindowFeature(Window.FEATURE_NO_TITLE);
                setContentView(R.layout.main);
                }
}
```


Hello World, MainActivity!

4. What is Dialog window. Explain different types of dialogs available

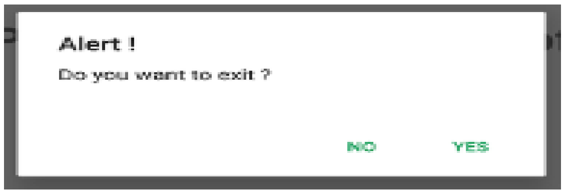
Dialog: A dialog is a small window that prompts the user to make a decision or enter additional information. A dialog does not fill the screen and is normally used for modal events that require users to take an action before they can proceed.

Different types of dialogs are

a. Alert Dialog

✓Alerts are urgent interruptions, requiring acknowledgement, that inform the user about a situation.

✓AlertDialog.Builder


Alert !

Do you want to exit ?

b. Confirmation Dialogs

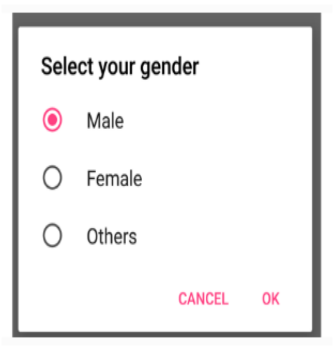
✓Confirmation dialogs require users to explicitly confirm their choice before an option is committed. For example, users can listen to multiple ringtones but only make a final selection upon touching “OK”

i. Single choice: In this only one item can be selected

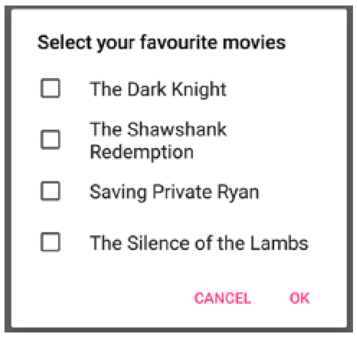
ii. Multiple choice: This dialog is used when we want to select more than one item in a dialog.

iii. Time picker: This dialog picker is used to select a single date.

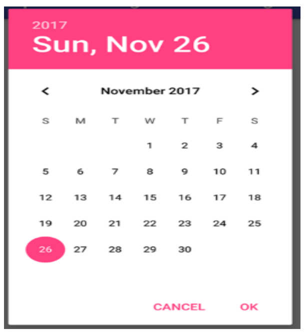
iv. Date picker: The time picker dialog allows the user to pick a time, and adjusts to the user’s preferred time setting, i.e. the 12-hour or 24-hour format



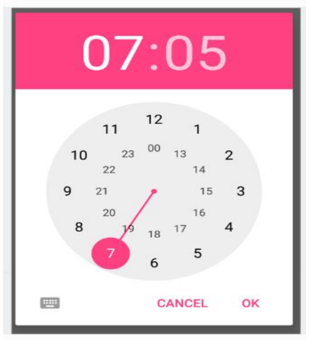
i.Single Choice



ii. Multi choice



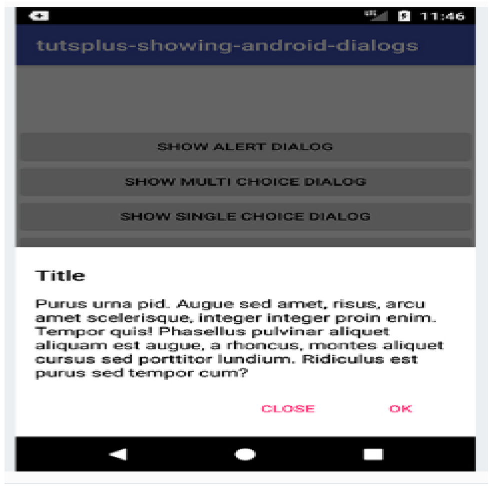
iii. Date picker

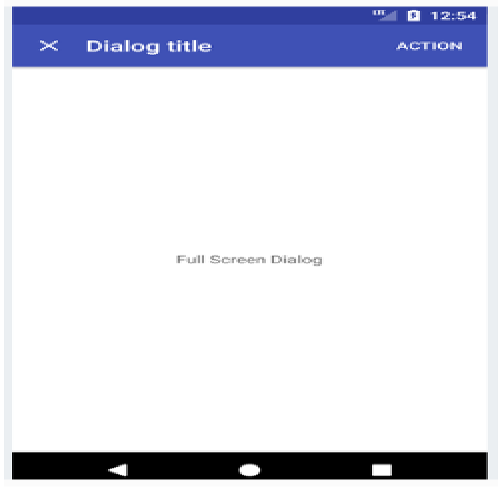


iv.TimePicker

c. **Bottom Sheet Dialog**

✓Bottom sheets slide up from the bottom of the screen to reveal more content.


11:46
tutsplus-showing-android-dialogs
SHOW ALERT DIALOG
SHOW MULTI CHOICE DIALOG
SHOW SINGLE CHOICE DIALOG
Title
Purus urna pid. Augue sed amet, risus, arcu amet scelerisque, integer integer proin enim. Tempor quis! Phasellus pulvinar aliquet aliquam est augue, a rhoncus, montes aliquet cursus sed porttitor lundium. Ridiculus est purus sed tempor cum?
CLOSE
OK



Bottom sheet dialog full screen dialog

d. Full screen dialog

Full-screen dialogs group a series of tasks (such as creating a calendar entry) before they may be committed or discarded. No selections are saved until “Save” is touched. Touching the “X” discards all changes and exits the dialog.

5. How to Displaying a Dialog Window in an Activity.

A Dialog is small window that prompts the user to a decision or enter additional information.

PROPERTIES:

- setTitle()
- setMessage()
- setIcon()
- setPositiveButton()
- setNegativeButton()

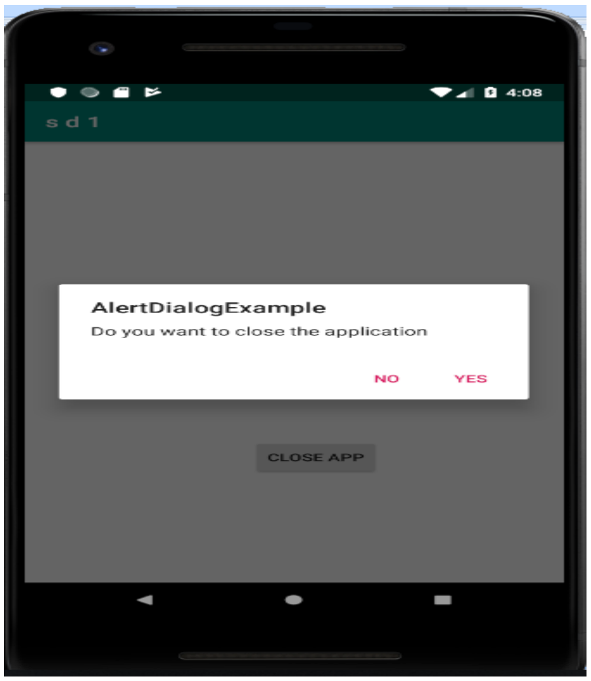
The **onCreateDialog()** protected method defined in the base Activity class to display a dialog window.

Ex 1: MainActivity.xml

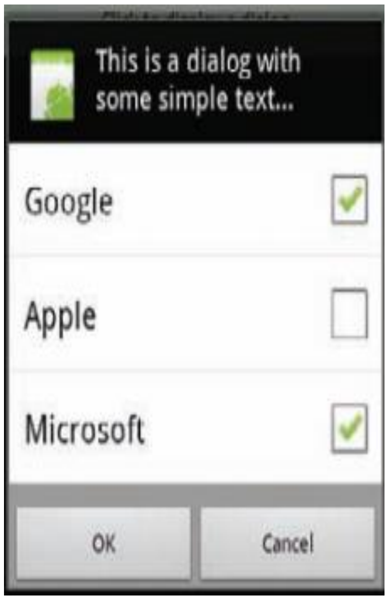
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical">
   <Button
      android:id="@+id/button"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="close app"    />
</LinearLayout>
```

MainActivity.java

```
package com.example.ex1;
import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.DialogInterface;
import android.os.Handler;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
public class MainActivity extends
AppCompatActivity {
   Button closeButton;
   AlertDialog.Builder builder;
   protected void onCreate(Bundle
savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        closeButton = (Button)
findViewById(R.id.button);
        builder = new AlertDialog.Builder(this);
        closeButton.setOnClickListener(new View.OnClickListener() {
           public void onClick(View v) {
                 builder.setMessage("Do you want to close the application");
                 builder.setCancelable(false);
                 builder.setPositiveButton("yes", new DialogInterface.OnClickListener() {
                     public void onClick(DialogInterface dialog, int which) {
                        finish();
                     Toast.makeText(getApplicationContext(), "",Toast.LENGTH_SHORT).show();
                    }
                });
         builder.setNegativeButton("NO", new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int which) {
                        dialog.cancel();
                        Toast.makeText(getApplicationContext(),"you choose the no action in the
                                    alert box",Toast.LENGTH_SHORT).show();
                    }
                });
                AlertDialog alert =builder.create();
                alert.setTitle("AlertDialogExample");
                alert.show();
            }
        });
    }
}
```



Ex 2: MainActivity.java

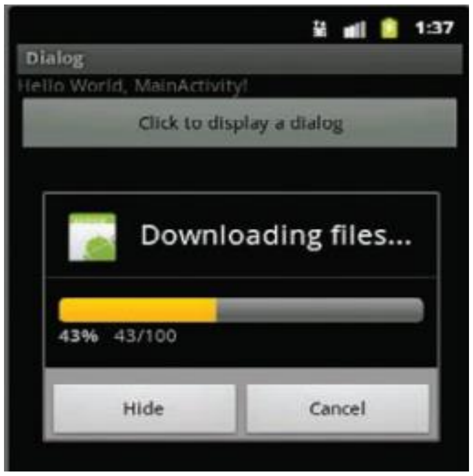


```
import android.app.Activity;
import android.os.Bundle;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.DialogInterface;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
public class MainActivity extends Activity {
         CharSequence[] items = { "Google", "Apple", "Microsoft" };
        boolean[] itemsChecked = new boolean [items.length];
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);
                Button btn = (Button) findViewById(R.id.btn_dialog);
                btn.setOnClickListener(new View.OnClickListener() {
                public void onClick(View v) {
                       showDialog(0);
                }
            });
        }
        protected Dialog onCreateDialog(int id) {
        switch (id) {
        case 0:     return new AlertDialog.Builder(this)
                     .setIcon(R.drawable.icon)
                     .setTitle("This is a dialog with some simple text...")
                     .setPositiveButton("OK",
                     new DialogInterface.OnClickListener() {
                     public void onClick(DialogInterface dialog, int whichButton){
                      Toast.makeText(getBaseContext(), "OK clicked!",  Toast.LENGTH_SHORT).show();
                     }
                     })
                     .setNegativeButton("Cancel",
                     new DialogInterface.OnClickListener() {
                     public void onClick(DialogInterface dialog, int whichButton)
                     {
                     Toast.makeText(getBaseContext(),"Cancel clicked!", Toast.LENGTH_SHORT).show();
                      }
                     })
                     .setMultiChoiceItems(items, itemsChecked,
                     new DialogInterface.OnMultiChoiceClickListener() {
                     public void onClick(DialogInterface dialog, int which, boolean isChecked) {
                     Toast.makeText(getBaseContext(), items[which] + (isChecked ? " checked!": "unchecked!"),
                      Toast.LENGTH_SHORT).show();
                     }
                     } )
                .create();
            }
            return null;
        }
 }
```

The code contains two buttons: OK and Cancel, using the setPositiveButton() and setNegativeButton() methods, respectively. The user also set a list of checkboxes for users to choose via the setMultiChoiceItems() method. For the setMultiChoiceItems() method, you passed in two arrays: one for the list of items to display and another to contain the value of each item to indicate if they are checked. When each item is checked, the Toast class to display a message.

6. Explain Displaying a Progress Dialog with an example.

A progress dialog is useful for showing the progress of some activities, such as the status of a download operation.



Ex 1: MainActivity.java

```
package net.learn2develop.Dialog;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
import android.app.ProgressDialog;
import android.os.Handler;
import android.os.Message;
public class MainActivity extends Activity {
        CharSequence[] items = { “Google”, “Apple”, “Microsoft” };
        boolean[] itemsChecked = new boolean [items.length];
        private ProgressDialog _progressDialog;
        private int _progress = 0;
        private Handler _progressHandler;
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);
                Button btn = (Button) findViewById(R.id.btn_dialog);
                btn.setOnClickListener(new View.OnClickListener() {
                public void onClick(View v) {
                        showDialog(1); _progress = 0;
                        _progressDialog.setProgress(0);
                         _progressHandler.sendEmptyMessage(0);
                }
                });
```

```
                    _progressHandler = new Handler() {
                           public void handleMessage(Message msg) {
                           super.handleMessage(msg);
                            if (_progress >= 100) {
                                          _progressDialog.dismiss();
                           } else {
                                          _progress++;
                                          _progressDialog.incrementProgressBy(1);
                                         _progressHandler.sendEmptyMessageDelayed(0, 100);
                           }
                           }
                            };
                    }
   protected Dialog onCreateDialog(int id) {
   switch (id) {
   case 1:
          _progressDialog = new ProgressDialog(this);
         _progressDialog.setIcon(R.drawable.icon);
         _progressDialog.setTitle("Downloading files...");
         _progressDialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
         _progressDialog.setButton(DialogInterface.BUTTON_POSITIVE, "Hide",
         new  DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int whichButton)
    {
   Toast.makeText(getBaseContext(), "Hide clicked!", Toast.LENGTH_SHORT).show();
   }
    });
   _progressDialog.setButton(DialogInterface.BUTTON_NEGATIVE, "Cancel", new
   DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int whichButton)
   {
   Toast.makeText(getBaseContext(), "Cancel clicked!", Toast.LENGTH_SHORT).show();
   }
   });
    return _progressDialog;
   }
    return null;
   }
   }
```

7. Explain the concept of Linking Activities Using Intents [IT-REG-2019] [IT-SUP-2021] OR Explain Intents? Explain types of Intents [BCA-REG-2021] OR Give the steps in linking Activities with Intents? Example. [MCA-REG-2022]

- An Android application can contain zero or more activities. When your application has more than one activity, you may need to navigate from one activity to another. To navigate between activities is done by using an intent.
- Intents: Android Intent is the *message* that is passed between components such as activities, content providers, broadcast receivers, services etc.The “glue “ that connects different activities

Intents are two types:

a. Implicit Intents

- ✓ Implicit intent doesn’t specify the component.
- ✓ Intent provides the information of available components provided by the system that is to invoked.
- ✓ To navigate between activities present in different application
- ✓ **Ex:** By clicking the you tube link available in what’s app status, navigating to you tube application

Syntax:

```
Intent i=new intent(Intent.ACTION_VIEW);
i.setData(Uri.Parse(“htttp://www.vignan.ac.in”);
startActivity(i);
```

or

```
Intent i=new intent(Intent.ACTION_VIEW, Uri.Parse(“htttp://www.vignan.ac.in”);
startActivity(i);
```

Implicit Intents examples please Refer the concept of Calling Built in Apps Using Intents.

b. Explicit Intents

- ✓ Explicit Intents specifies the component.
- ✓ Intent provides the external class to be invoked.
- ✓ To navigate between activities present in same application
- ✓ **Ex:** In what’s app Navigating between status, calls, and chats.

Syntax:

```
Intent i=new intent(this,SecondActivity.class);
startActivity(i);
```

The following is the example of Explicit Intents:

AndroidManifest.xml

```
<?xmlversion=”1.0”encoding=”utf-8”?>
<manifestxmlns:android=”http://schemas.android.com/apk/res/android”
package=”net.learn2develop.Activities”
android:versionCode=”1”
android:versionName=”1.0”>
<application android:icon=”@drawable/icon”android:label=”@string/app_name”>
<activity android:name=”.MainActivity”
      android:label=”@string/app_name”
      android:theme=”@android:style/Theme.Dialog” >
      <intent-filter>
      <actionandroid:name=”android.intent.action.MAIN” />
      <categoryandroid:name=”android.intent.category.LAUNCHER” />
      </intent-filter>
```

```
</activity>
<activity android:name=”.Activity2”
         android:label=”Activity 2”>
        <intent-filter>
        <action android:name=”net.learn2develop.ACTIVITY2” />
        <category android:name=”android.intent.category.DEFAULT” />
        </intent-filter>
</activity>
</application>
```

Activity2.xml

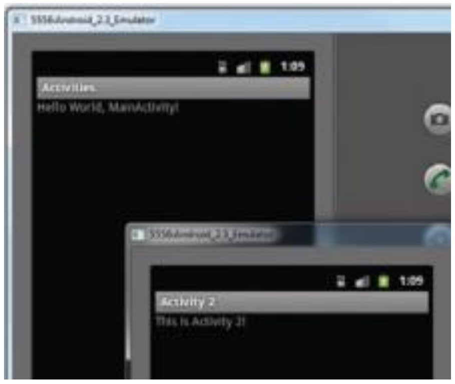
```
<?xml version=”1.0” encoding=”utf-8”?>
<LinearLayout
    xmlns:android=”http://schemas.android.com/apk/res/android”
    android:orientation=”vertical”
    android:layout_width=”fill_parent”
    android:layout_height=”fill_parent”>
    <TextView
            android:layout_width=”fill_parent”
            android:layout_height=”wrap_content”
            android:text=”This is Activity 2!”
    />
</LinearLayout>
```

Activity2.java

```
package net.learn2develop.Activities;
import android.app.Activity;
import android.os.Bundle;
public class Activity2 extends Activity {
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity2);
    }  }
```

MainActivity.java

```
package net.learn2develop.Activities;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.Window;
import android.view.KeyEvent;
import android.content.Intent;
public class MainActivity extends Activity {
    String tag = “Events”;
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      }
```



Hello World, MainActivity!

This is Activity 2!

```
    public boolean onKeyDown(int keyCode, KeyEvent event)
    {
      if (keyCode == KeyEvent.KEYCODE_DPAD_CENTER)
      {
      startActivity(new Intent("net.learn2develop.ACTIVITY2"));
      }
      return false;
    }
}
```

➤➤ The name of the new activity added is "Activity2".

➤➤ The label for the activity is named "Activity 2".

➤➤ The intent filter name for the activity is **"net.learn2develop.ACTIVITY2".** Other activities that wish to call this activity will invoke it via this name.

➤➤ The category for the intent filter is "**android.intent.category.DEFAULT".** You need to add this to the intent filter so that this activity can be started by another activity using the startActivity() method.

8. How to Resolving the Intent Filter Collision [IT-REG-2019] [IT-SUP-2021]

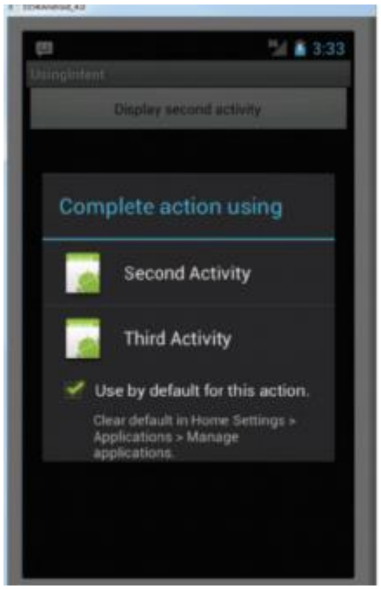
- The <intent-filter> element defines how the activity can be invoked by another activity.

Ex: AndroidManifest.xml

```
<application>
<activity
    android:label="@string/app_name"
    android:name=".UsingIntentActivity" >
    <intent-filter >
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity
    android:label="Second Activity"
    android:name=".SecondActivity" >
    <intent-filter >
    <action android:name="net.learn2develop.SecondActivity" />
    <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
<activity
    android:label="Third Activity"
    android:name=".ThirdActivity" >
    <intent-filter >
    <action android:name="net.learn2develop.SecondActivity" />
    <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
```

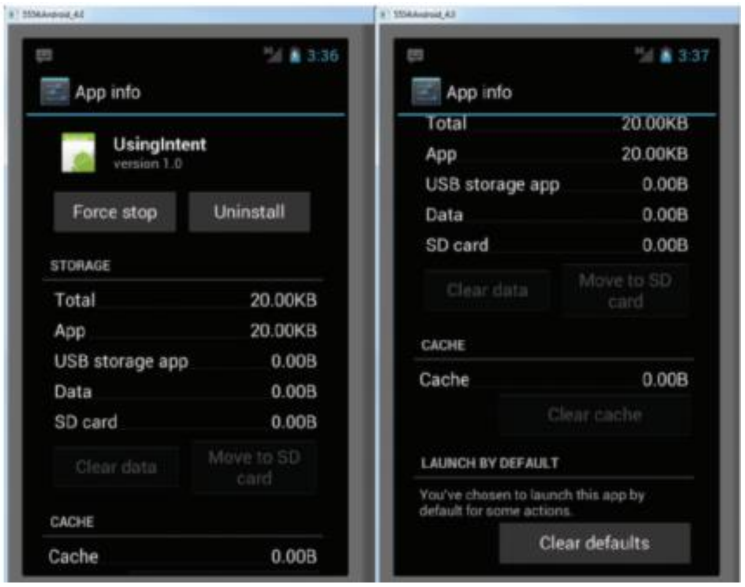
```
</activity>
</application>
```

- If you call the startActivity() method with the following intent, then the Android OS will display a selection of activities.
- startActivity(new Intent("net.learn2develop.SecondActivity"));
- If you check the "Use by default for this action" item and then select an activity, then the next time the intent "net.learn2develop.SecondActivity" is called again, it will launch the previous activity that you have selected.
- To clear this default, go to the Settings application in Android and select Apps ➪ Manage applications, and then select the application name. When the details of the application are shown, scroll down to the bottom and click the Clear defaults button.



✓ Use by default for this action.

Clear default in Home Settings > Applications > Manage applications.



STORAGE

Total	20.00KB
App	20.00KB
USB storage app	0.00B
Data	0.00B
SD card	0.00B

CACHE

Cache	0.00B

LAUNCH BY DEFAULT

You've chosen to launch this app by default for some actions.

9. Explain about Returning Results from an Intent OR Give the steps to get a result from an activity? Example [MCA-REG-2022]

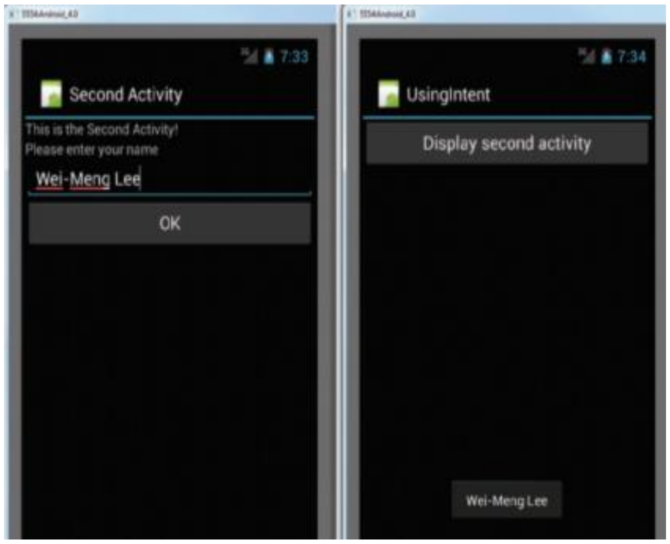
- The **startActivity()** method invokes another activity but does not return a result to the current activity. For example, you may have an activity that prompts the user for user name and password.
- The information entered by the user in that activity needs to be passed back to the calling activity for further processing. If you need to pass data back from an activity, you should instead use the **startActivityForResult()** method.

Mainactivity.xml file

```
<?xml version="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/btn_OK"
            android:layout_width=”fill_parent”
            android:layout_height=”wrap_content”
            android:text=”Display Second Activity”
            android:onClick=”onClick”/>
</LinearLayout>
```

secondactivity.xml file

```
<?xml version=”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” >
    <TextView
            android:layout_width=”fill_parent”
            android:layout_height=”wrap_content”
            android:text=”This is the Second Activity!” />
    <TextView
            android:layout_width=”fill_parent”
            android:layout_height=”wrap_content”
            android:text=”Please enter your name” />
    <EditText
            android:id=”@+id/txt_username”
            android:layout_width=”fill_parent”
            android:layout_height=”wrap_content” />
    <Button
            android:id=”@+id/btn_OK”
            android:layout_width=”fill_parent”
            android:layout_height=”wrap_content”
            android:text=”OK”
            android:onClick=”onClick”/>
</LinearLayout>
```



SecondActivity.java:

```
package net.learn2develop.UsingIntent;
import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
public class SecondActivity extends Activity{
    public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.secondactivity);
    }
    public void onClick(View view) {
            Intent data = new Intent();
            EditText txt_username = (EditText) findViewById(R.id.txt_username);
            data.setData(Uri.parse(txt_username.getText().toString()));
            setResult(RESULT_OK, data);
            finish();
    }
}
```

MainActivity.java

```
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;
public class UsingIntentActivity extends Activity {
    int request_Code = 1;
    public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
    }
    public void onClick(View view) {
      startActivityForResult(new Intent("net.learn2develop.SecondActivity"),request_Code);
    }
    public void onActivityResult(int requestCode, int resultCode, Intent data)
    {
      if (requestCode == request_Code) {
            if (resultCode == RESULT_OK) {
            Toast.makeText(this,data.getData().toString(),Toast.LENGTH_SHORT).show();
            }
       }
    }
}
```

- When the first activity is loaded, click the button. SecondActivity will now be loaded. Enter your name and click the OK button. The first activity will display the name you have entered using the Toast class.
- In order for an activity to return a value to the calling activity, you use an Intent object to send data back via the setData() method:
- The setResult() method sets a result code (either RESULT_OK or RESULT_CANCELLED) and the data (an Intent object) to be returned back to the calling activity.
- The finish() method closes the activity and returns control back to the calling activity.
- The onActivityResult() method, which is called whenever an activity returns.
- It checks for the appropriate request and result codes and display the result that is returned.
- The returned result is passed in via the data argument; and finally obtain its details through the getData() method.

10. How to Passing Data Using an Intent Object

Three different ways to pass the data using an Intent Object

a. putExtra() method
b. putExtras() method
c. setData() method

a. putExtra() method:

- Use the putExtra() method of an Intent object to add a name/value pair
  - **i.putExtra("str1", "This is a string");**
  - **i.putExtra("age1", 25);**
- The preceding statements add two name/value pairs to the Intent object: one of type string and one of type integer.
- To obtain the data sent using the Intent object use the **getIntent()** method. Then, call its getStringExtra() method to get the string value set using the putExtra() method:
- **Toast.makeText(this,getIntent().getStringExtra("str1"),Toast.LENGTH_SHORT).show();**
- For the integer value, use the getIntExtra() method (the second argument is the default value in case no value is stored in the specifi ed name):
- **Toast.makeText(this,Integer.toString(getIntent().getIntExtra("age1",0)), Toast.LENGTH_SHORT).show();**

b. putExtras() method:

- Bundle object as a dictionary object — it contains a set of name/ value pairs
  - **Bundle extras = new Bundle();**
  - **extras.putString("str2", "This is another string");**
  - **extras.putInt("age2", 35);**
  - **i.putExtras(extras);**
- To retrieve the Bundle object, use the getExtras() method:

  Bundle bundle = getIntent().getExtras();
- **Toast.makeText(this, bundle.getString("str2"),Toast.LENGTH_SHORT).show();**
- **Toast.makeText(this,Integer.toString(bundle.getInt("age2")),Toast.LENGTH_SHO RT).show();**

c. setData() method:

- Use the setData() method to return some value.
- **i.setData(Uri.parse("Something passed back to main activity"));**
- To retrieve the data set using the setData() method, use the getData() method.
- **Toast.makeText(this, data.getData().toString(),Toast.LENGTH_SHORT).show();**

Ex: Besides returning data from an activity, it is also common to pass data to an activity. For example, to set some default text in the EditText view before the activity is displayed. Use the Intent object to pass the data to the target activity.

main.xml

```
<?xml version="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/btn_SecondActivity”
                android:layout_width=”fill_parent”
                android:layout_height=”wrap_content”
                android:text=”Click to go to Second Activity”
                android:onClick=”onClick”/>
</LinearLayout>
```

secondactivity.xml

```
<?xml version=”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” >
    <TextView
            android:layout_width=”fill_parent”
            android:layout_height=”wrap_content”
            android:text=”Welcome to Second Activity” />
    <Button
            android:id=”@+id/btn_MainActivity”
            android:layout_width=”fill_parent”
            android:layout_height=”wrap_content”
            android:text=”Click to return to main activity”
            android:onClick=”onClick”/>
</LinearLayout>
```

SecondActivity.java

```
package net.learn2develop.PassingData;
import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;
public class SecondActivity extends Activity {
    public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.secondactivity);
Toast.makeText(this,getIntent().getStringExtra(“str1”), Toast.LENGTH_SHORT).show();
Toast.makeText(this,Integer.toString(getIntent().getIntExtra(“age1”,0)),Toast.LENGTH_SHORT).show();
                Bundle b = getIntent().getExtras();
Toast.makeText(this, b.getString(“str2”),Toast.LENGTH_SHORT).show();
Toast.makeText(this,Integer.toString(bundle.getInt(“age2”)),Toast.LENGTH_SHORT).show();
    }
    public void onClick(View view) {
            Intent i = new Intent();
            i.putExtra(“age3”, 45);
            i.setData(Uri.parse(“Something passed back to main activity”));
```

```
                setResult(RESULT_OK, i);
                finish();
        }
}
```

MainActivity.java

```
package net.learn2develop.PassingData;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;
public class PassingDataActivity extends Activity {
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);
        }
        public void onClick(View view) {
                Intent i = new  Intent("net.learn2develop.PassingDataSecondActivity");
                i.putExtra("str1", "This is a string");
                i.putExtra("age1", 25);
                Bundle extras = new Bundle();
                extras.putString("str2", "This is another string");
                extras.putInt("age2", 35);
                i.putExtras(extras);
                startActivityForResult(i, 1);
        }
        public void onActivityResult(int requestCode,int resultCode, Intent data)
        {
                if (requestCode == 1) {
                        if (resultCode == RESULT_OK) {
Toast.makeText(this, Integer.toString(data.getIntExtra("age3", 0)),Toast.LENGTH_SHORT).show();
Toast.makeText(this, data.getData().toString(),Toast.LENGTH_SHORT).show();
                        }
                }
        }
}
```

11. Define Fragment. [IT-REG-2021] OR What is Fragment? Advantages of Fragments Fragment: [BCA-REG-2021]

- Android Fragment is the part of activity, it is also known as sub-activity.
- There can be more than one fragment in an activity. Fragments represent multiple screen inside one activity.
- Android fragment lifecycle is affected by activity lifecycle because fragments are included in activity.

- Each fragment has its own life cycle methods that is affected by activity life cycle because fragments are embedded in activity.
- The **FragmentManager** class is responsible to make interaction between fragment objects.

Advantage of fragments:

- we can only show a single Activity on the screen at one given point of time so we were not able to divide the screen and control different parts separately. With the help of Fragment's we can divide the screens in different parts and controls different parts separately.
- By using Fragments we can comprise multiple Fragments in a single Activity. Fragments have their own events, layouts and complete life cycle. It provide flexibility and also removed the limitation of single Activity on the screen at a time.

Fragments were added in Honeycomb version of Android i.e API version 11. There are some primary classes related to Fragment's are:
1. FragmentActivity: The base class for all activities using compatibility based Fragment (and loader) features.
2. Fragment: The base class for all Fragment definitions
3. FragmentManager: The class for interacting with Fragment objects inside an activity
4. FragmentTransaction: The class for performing an atomic set of Fragment operations such as Replace or Add a Fragment.
There are two ways to add a fragment to an activity:

a) **Statically (`XML`)**
b) **Dynamically (`Java`)**

12.Explain the concept of statically adding fragments to an activity with example. OR What is fragment? How fragments can be created? [BCA-REG-2021]
Statically using XML:
Steps:

- Create MainActivity .java file and extends Activity.
- Create Fragment1.java and Fragment2.java which extends the Fragment Class.
- Create the respective layouts for Fragment1 and Fragment2.
- Override onCreate() method in Fragment1.java and Fragment2.java, and inflate the Fragment layouts (R.id.Fragment1 and R.id.Fragment2) using LayoutInflater Class object and by passing the ViewGroup argument which is the activity in which the fragment will be embedded.
- Finally attach the two fragments to the Activity UI file (MainActivity.xml).

MainActivity.java

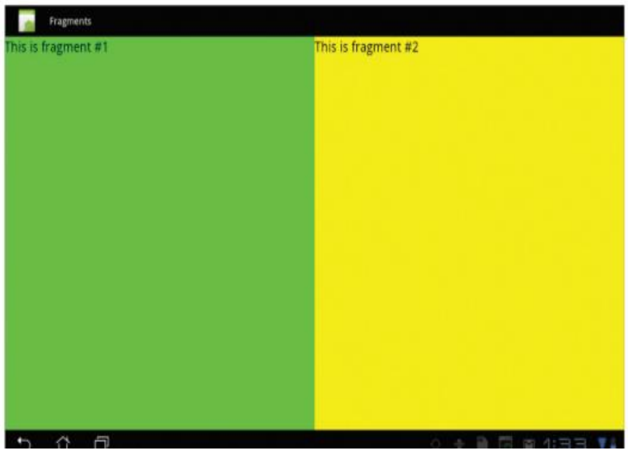
```
<?xml version=”1.0” encoding=”utf-8”?>
<LinearLayout
        xmlns:android=”http://schemas.android.com/apk/res/android”
        android:layout_width=”fill_parent”
        android:layout_height=”fill_parent”
        android:orientation=”horizontal” >
        <fragment
                android:name=”net.learn2develop.Fragments.Fragment1”
                android:id="@+id/fragment1"
                android:layout_weight="1"
                android:layout_width="0px"
                android:layout_height="match_parent" />
        <fragment
                android:name="net.learn2develop.Fragments.Fragment2"
                android:id="@+id/fragment2"
                android:layout_weight="1"
                android:layout_width="0px"
                android:layout_height="match_parent" />
        </LinearLayout>
```

Fragment1.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#00FF00"  >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="This is fragment #1"
        android:textColor="#000000"
        android:textSize="25sp" />
</LinearLayout>
```

Fragment2.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#FFFE00">
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="This is fragment #2"
        android:textColor="#000000"
        android:textSize="25sp" />
</LinearLayout>
```



Fragment1.java

```
package net.learn2develop.Fragments;
import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
public class Fragment1 extends Fragment {
    public View onCreateView(LayoutInflater
    inflater,ViewGroup container, Bundle savedInstanceState) {
            return inflater.inflate(R.layout.fragment1, container, false);
     }
}
```

Fragment2.java

```
package net.learn2develop.Fragments;
import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
public class Fragment2 extends Fragment {
     public View onCreateView(LayoutInflater inflater,ViewGroup container, Bundle
savedInstanceState) {
            return inflater.inflate(R.layout.fragment2, container, false);
     }
}
```

13.Explain the concept of dynamically adding fragments to an activity with example. OR Describe about dynamic linking [BCA-REG-2021]

Dynamically using JAVA:

The second way is by adding the fragment `dynamically` in Java using the `FragmentManager`. The `FragmentManager` class and the `FragmentTransaction class` allow you to add, remove and replace fragments in the layout of your activity at runtime.

Steps:

- Creating a layouts of two fragments. Those are Fragment1.xml and Fragment2.xml
- Create Fragment1.java and Fragment2.java which extends the Fragment Class.
- Create the mainActivity.java and implement the instance of FragmentManager class and also call the FragmmentTransaction methods add/remove/replace of the fragment.
- Finally call the framentTransaction.commit() to save the fragment.

NOTE: Fragment1.xml, Fragment2.xml, Fragments1.java and Fragments2.java refer statically adding fragments example.

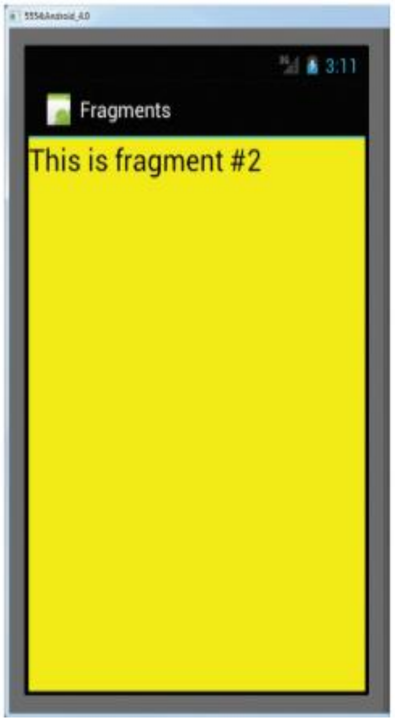
MainActivity.xml

```
<?xml version=”1.0” encoding=”utf-8”?>
<LinearLayout
        xmlns:android=”http://schemas.android.com/apk/res/android”
        android:orientation=”vertical”
        android:layout_width=”fill_parent”
        android:layout_height=”fill_parent”>
        <TextView
                android:layout_width=”fill_parent”
                android:layout_height=”wrap_content”
                android:text=”welcome”/>
</LinearLayout>
```

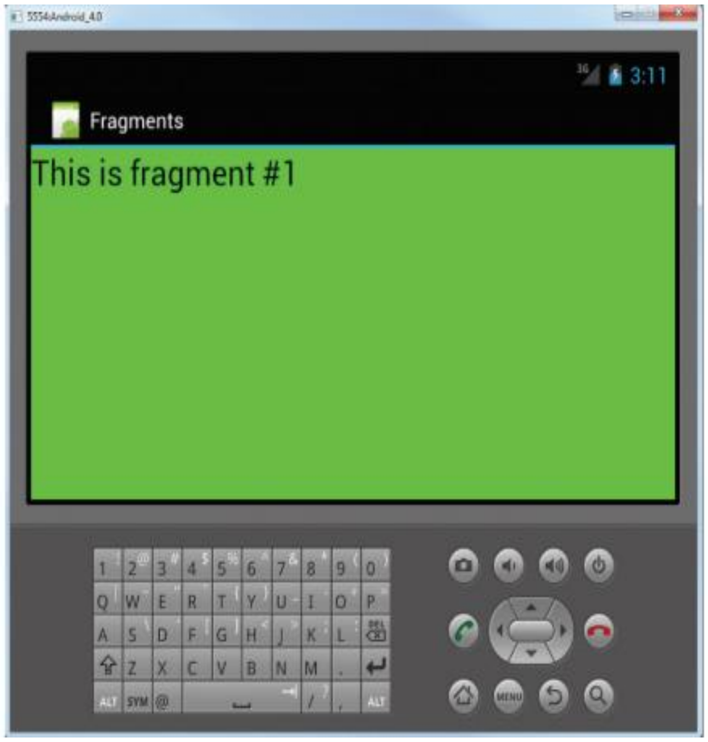
Mainactivity.java

```
package net.learn2develop.Fragments;
import android.app.Activity;
import android.app.FragmentManager;
import android.app.FragmentTransaction;
import android.os.Bundle;
import android.view.Display;
import android.view.WindowManager;
public class FragmentsActivity extends Activity {
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                FragmentManager fm = getFragmentManager();
                FragmentTransaction ft =fm.beginTransaction();
                WindowManager wm = getWindowManager();
                Display d = wm.getDefaultDisplay();
                if (d.getWidth() > d.getHeight())
                {
                Fragment1 f1 = new Fragment1();
                fragmentTransaction.replace(android.R.id.content, f1);
                }
                else
                {
                Fragment2 f2 = new Fragment2();
                fragmentTransaction.replace(android.R.id.content, f2);
                }
          ft.commit();
      }
```

}



This is fragment #2



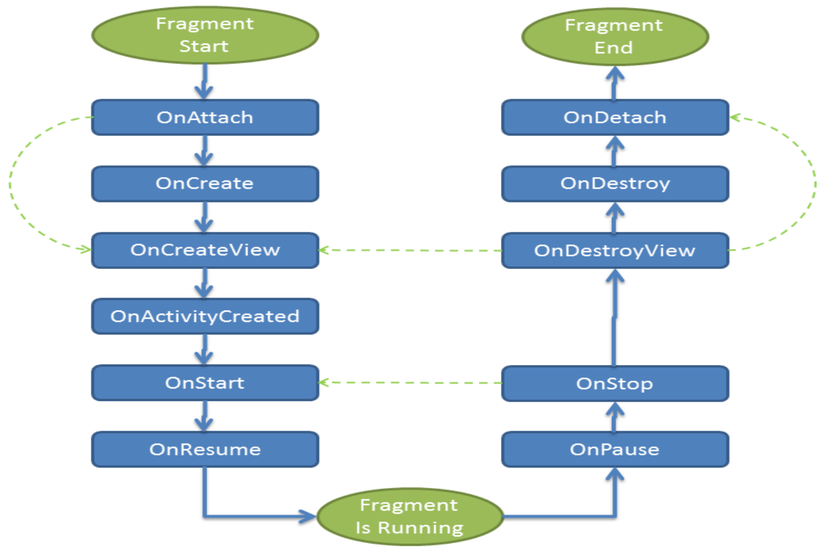
This is fragment #1

14.what do you understand about Life Cycle of a Fragment

Fragment has many methods which can be overridden to plug into the lifecycle (`similar to an Activity`):

- onAttach() is called when a fragment is connected to an activity.
- onCreate() is called to do initial creation of the fragment.
- onCreateView() is called by Android once the Fragment should inflate a view.
- onViewCreated() is called after onCreateView() and ensures that the fragment's root view is non-null. Any view setup should happen here. E.g., view lookups, attaching listeners.
- onActivityCreated() is called when host activity has completed its onCreate() method.
- onStart() is called once the fragment is ready to be displayed on screen.
- onResume() - Allocate "expensive" resources such as registering for location, sensor updates, etc.
- onPause() - Release "expensive" resources. Commit any changes.
- onDestroyView() is called when fragment's view is being destroyed, but the fragment is still kept around.
- onDestroy() is called when fragment is no longer in use.
- onDetach() is called when fragment is no longer connected to the activity.

The lifecycle execution order is mapped out below:



15. Create an application which implement the Life Cycle Methods of an Fragment.

```
package net.learn2develop.Fragments;
import android.app.Activity;
import android.app.Fragment;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
public class Fragment1 extends Fragment {
        public View onCreateView(LayoutInflater inflater,ViewGroup container, Bundle
        savedInstanceState) {
                Log.d("Fragment 1", "onCreateView");
                return inflater.inflate(R.layout.fragment1, container, false);
        }
        public void onAttach(Activity activity) {
                super.onAttach(activity);
                Log.d("Fragment 1", "onAttach");
        }
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                Log.d("Fragment 1", "onCreate");
        }
        public void onActivityCreated(Bundle savedInstanceState) {
                super.onActivityCreated(savedInstanceState);
                Log.d("Fragment 1", "onActivityCreated");
        }
        public void onStart() {
                super.onStart();
                Log.d("Fragment 1", "onStart");
        }
        public void onResume() {
                super.onResume();
                Log.d("Fragment 1", "onResume");
        }
        public void onPause() {
                super.onPause();
                Log.d("Fragment 1", "onPause");
        }
        public void onStop() {
                super.onStop();
                Log.d("Fragment 1", "onStop");
        }
        public void onDestroyView() {
                super.onDestroyView();
                Log.d("Fragment 1", "onDestroyView");
        }
        public void onDestroy() {
                super.onDestroy();
                Log.d("Fragment 1", "onDestroy");
        }
        public void onDetach() {
           super.onDetach();
           Log.d("Fragment 1", "onDetach");
        }
}
```

Like activities, fragments in Android also have their own life cycle. As you have seen, when a fragment is being created, it goes through the following states:

➤ onAttach()

➤ onCreate()

➤ onCreateView()

➤ onActivityCreated()

When the fragment becomes visible, it goes through these states:

➤ onStart()

➤ onResume()

When the fragment goes into the background mode, it goes through these states:

➤ onPause()

➤ onStop()

When the fragment is destroyed (when the activity it is currently hosted in is destroyed), it goes through the following states:

➤ onPause()

➤ onStop()

➤ onDestroyView()

➤ onDestroy()

➤ onDetach()
Like activities, you can restore an instance of a fragment using a Bundle object, in the following states:

➤ onCreate()

➤ onCreateView()

➤ onActivityCreated()

16. Explain the concept of Interaction Between Fragments
An activity contains one or more fragments and to communicate with one another and exchange of data the interaction fragments is required.
Ex: One fragment can access the views contained within another fragment.
Steps:

- Create a fragment layout contains one text view.
- Create second fragment which is having a one text view and button
- In the main layout add the two fragments.
- In the second fragment write the code to interact with the two fragments.

Fragment1.xml
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:background="#00FF00" >
        <TextView
                android:id="@+id/lblFragment1"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:text="This is fragment #1"
                android:textColor="#000000"
                android:textSize="25sp" />
</LinearLayout>
```

Fragments2.xml
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:background="#FFFE00" >
        <TextView
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:text="This is fragment #2"
```

```
            android:textColor=”#000000”
            android:textSize=”25sp” />
    <Button
            android:id=”@+id/btnGetText”
            android:layout_width=”wrap_content”
            android:layout_height=”wrap_content”
            android:text=”Get text in Fragment #1”
            android:textColor=”#000000”
            android:onClick=”onClick” />
</LinearLayout>
```

Main.xml

```
<?xml version=”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=”horizontal” >
    <fragment
            android:name=”net.learn2develop.Fragments.Fragment1”
            android:id=”@+id/fragment1”
            android:layout_weight=”1”
            android:layout_width=”0px”
            android:layout_height=”match_parent” />
    <fragment
            android:name=”net.learn2develop.Fragments.Fragment2”
            android:id=”@+id/fragment2”
            android:layout_weight=”1”
            android:layout_width=”0px”
            android:layout_height=”match_parent” />
</LinearLayout>
```

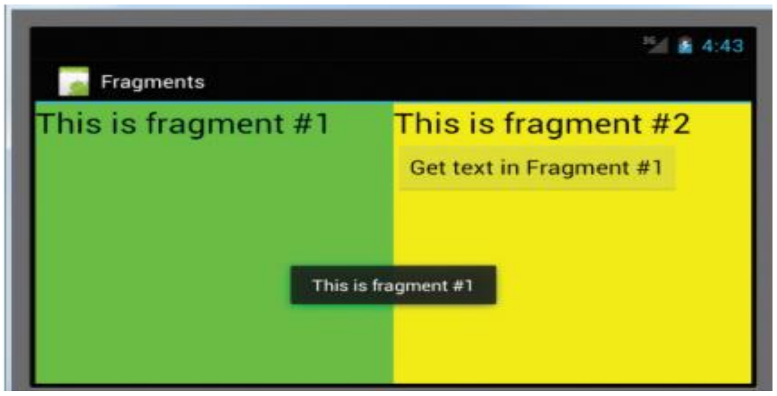
Mainactivity.java

```
public class FragmentsActivity extends Activity {
    public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
    }
}
```

Fragment2.java

```
package net.learn2develop.Fragments;
import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
public class Fragment2 extends Fragment {
        public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
        savedInstanceState) {
                return inflater.inflate(R.layout.fragment2, container, false);
        }
          public void onStart() {
                super.onStart();
                Button btnGetText = (Button) getActivity().findViewById(R.id.btnGetText);
                btnGetText.setOnClickListener(new View.OnClickListener() {
                public void onClick(View v) {
                      TextView lb = (TextView)getActivity().findViewById(R.id.lblFragment1);
                      Toast.makeText(getActivity(),lb.getText(),Toast.LENGTH_SHORT).show();
                      }
                });
          }
}
```

The activity in which a fragment is currently embedded by fi rst using the getActivity() method and then using the findViewById() method to locate the view(s) contained within the fragment. The getActivity() method returns the activity with which the current fragment is currently associated.



17. Explain about Calling built in apps using intents OR Example of Implicit Intents OR Demonstrate about calling built-in apps using Intents [BCA-REG-2021]

One of the advantage of Android Programming is using the intents to call activities from other applications.

Ex: Application needs to load a web page, so use the Intent object to invoke the built-in web browser to display the webpage, instead of building your own web browser.

Main.xml

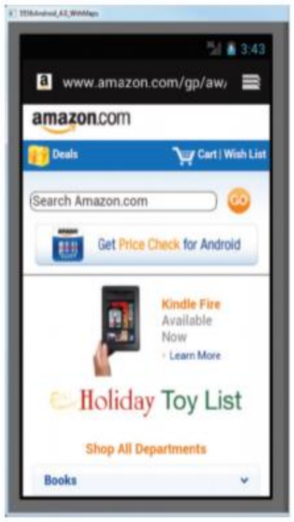
```
<?xml version=”1.0” encoding=”utf-8”?>
<LinearLayout xmlns:android=”http://schemas.android.com/apk/res/android”
        android:layout_width=”fill_parent”
        android:layout_height=”fill_parent”
        android:orientation=”vertical” >
        <Button
                android:id=”@+id/btn_webbrowser”
```

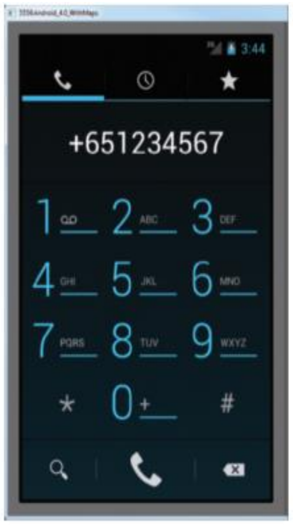
```
                    android:layout_width=”fill_parent”
                    android:layout_height=”wrap_content”
                    android:text=”Web Browser”
                    android:onClick=”onClickWebBrowser” />
          <Button
                    android:id=”@+id/btn_makecalls”
                    android:layout_width=”fill_parent”
                    android:layout_height=”wrap_content”
                    android:text=”Make Calls”
                    android:onClick=”onClickMakeCalls” />
          <Button
                    android:id=”@+id/btn_showMap”
                    android:layout_width=”fill_parent”
                    android:layout_height=”wrap_content”
                    android:text=”Show Map”
                    android:onClick=”onClickShowMap” />
</LinearLayout>
```

MainActivity.java

```
package net.learn2develop.Intents;
import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
public class IntentsActivity extends Activity {
            int request_Code = 1;
            public void onCreate(Bundle savedInstanceState) {
                  super.onCreate(savedInstanceState);
                  setContentView(R.layout.main);
            }
            public void onClickWebBrowser(View view) {
Intent i = new Intent(android.content.Intent.ACTION_VIEW,Uri.parse(“http://www.amazon.com”));
            startActivity(i);
            }
            public void onClickMakeCalls(View view) {
                  Uri u=Uri.parse("tel:9988776655");
                  Intent i=new Intent(Intent.ACTION_DIAL, u);
                  startActivity(i);
            }
            public void onClickShowMap(View view) {
Intent i = new Intent(android.content.Intent.ACTION_VIEW,Uri.parse(“geo:37.827500,-122.481670”));
            startActivity(i);
            }
}
```

- Intents usually come in pairs: action and data.
- The action describes what is to be performed, such as editing an item, viewing the content of an item, and so on.
- The data specifies what is affected, as a person in the Contacts database. The data is specified as an Uri object.
- The actions are **ACTION_VIEW, ACTION_DIAL, and ACTION_PICK.**





18. What do you understand about Intent Object.
The main purpose of the Intent Object is to call other activities. To call the activities we are having different ways.

i. Call another activity by passing its action to the constructor of an Intent Object:
startActivity(new Intent("net.learn2develop.Secondactivity");
The action is also known as component name i.e "net.learn2develop.Secondactivity". This is used to identify the target activity/application that you want to invoke.

ii. Rewrite the component name by specifying the class name of the activity if it resides in your project:
startActivity(new Intent(this, Secondactivity.class);

iii. Create an Intent object by passing in an action constant and data:
Intent i=new Intent(android.content.Intent.ACTION_VIEW, Uri.parse("http://www.vignan.ac.in"));
startActivity(i);
The action portion defines what you want to do, while the data portion contains the data for the activity act upon.

iv. Pass the data to the Intent object using setData() method:
Intent i=new Intent(android.content.Intent.ACTION_VIEW);
i.setData(Uri.parse("http://www.vignan.ac.in"));
startActivity(i);

v. Without specifying the data.
Intent i=new Intent(android.content.Intent.ACTION_PICK);
i.setType(Uri.parse(contactsContract.Contacts.CONTENT_TYPE);
Ex: To select a contact from Contacts application, just specify the action and then indicate MIME type using the setType() method. The setType() method indicate the type of data to return is MIME.

19. What do you understand about Intent Filters.

- An Activity can invoke another activity using the Intent object.
- To invoke activity, another way is to specify the action and category within the **<intent- filter>** element in AndroidManifest.xml file.

```
<intent-filter>
            <action android:name = ”net.learn2develop.SecondActivity”/>
            <category android:name = “android.intent.category.DEFAULT”/>
</intent-filter>
```

- The intent filter name for the new activity is net.learn2develop.SecondActivity. Other activities that wish to call this activity will invoke it via this name. Ideally, you should use the reverse domain name of your company as the intent filter name in order to reduce the chances of another application having the same intent filter name.
- The category for the intent filter is android.intent.category.DEFAULT. To add this to the intent filer so that this activity can be started by another activity using the startActivity() method.

20. Differentiate Activities and fragments. [BCA-REG-2021]

Activities: The activity represents a single screen or a interface that allows the user to interact with an application. The applications contains single to multiple activities.
Ex: facebook application have an activity to log into the application, and other activities are such as posting a photograph or activity to send a message to a friend.
Fragments: Android Fragment is the part of activity, it is also known as sub-activity. There can be more than one fragment in an activity. Fragments represent multiple screen inside one activity. Android fragment lifecycle is affected by activity lifecycle because fragments are included in activity.
Each fragment has its own life cycle methods that is affected by activity life cycle because fragments are embedded in activity.

21. What are benefits of Intents? [BCA-REG-2021]

Intents: Android Intent is the *message* that is passed between components such as activities, content providers, broadcast receivers, services etc. The “glue “ that connects different activities

- Easy to pass the data.
- Navigate easily from one activity to other.
- Handle Multiple Activities in the application.

UNIT – III : GETTING TO KNOW ANDROID UI

1. What do you understand about the Components of a Screen. OR Define and write short note on XML. [IT-REG-2021]

- Activity displays the UI, it contain widgets like buttons, labels, text boxes, and so on.
- The UI designed in XML file which is located in res/layout folder.

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

The UI file is loaded, by using the onCreate() event in the Activity class in runtime, using the setContentView() method.

```
public voidonCreate(BundlesavedInstanceState){
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
}
```

2. What is view and view group? Discuss. [IT-REG-2021] OR
Explain the following concepts
a. VIEWS b. VIEWGROUPS [MCA-REG-2022]

An activity contains Views and ViewGroups.

a.View:

- ➢It is a widget that has an appearance on screen.
- ➢A view derives from the base class **android.view.View.**
- ➢**Ex**: buttons, labels, and text boxes etc.

b.ViewGroups:

- ➢One or more views can be grouped together into a ViewGroup.
- ➢A ViewGroup (which is itself a special type of view) provides the layout in which you can order the appearance and sequence of views.
- ➢A ViewGroup derives from the base class **android.view.ViewGroup**.

Android supports the following ViewGroups:

- LinearLayout
- AbsoluteLayout
- TableLayout
- RelativeLayout
- FrameLayout
- ScrollView

View and ViewGroup has a set of common attributes, some of which are described in below.

i. layout_width --Specifies the width of the View or ViewGroup
   The width of the element fills the entire width of its parent (which is the screen in this case) using the fill_parent constant.

If use the wrap_content constant, which means that its width is only upto its contents aviable in that view.
Sometimes we specify values also,
dp — Density-independent pixel. 160dp is equivalent to one inch of physical screen size.
sp — Scale-independent pixel. This is similar to dpand is recommended for specifying font sizes.
pt — Point. A point is defi ned to be 1/72 of an inch, based on the physical screen size.
px — Pixel. Corresponds to actual pixels on the screen.

ii. layout_height --Specifies the height of the View or ViewGroup.
iii. layout_marginTop --Specifies extra space on the top side of the View or ViewGroup
iv. layout_marginBottom --Specifies extra space on the bottom side of the View or ViewGroup
v. layout_marginLeft --Specifies extra space on the left side of the View or ViewGroup
vi. layout_marginRight --Specifies extra space on the right side of the View or ViewGroupe
vii. layout_gravity --Specifies how child Views are positioned
viii. layout_weight --Specifies how much of the extra space in the layout should be allocated
ix. to the View
x. layout_x --Specifies the x-coordinate of the View or ViewGroup
xi. layout_y --Specifies the y-coordinate of the View or ViewGroup

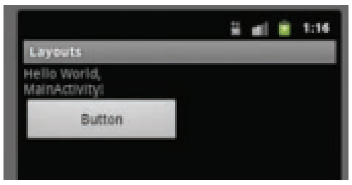
3. Explain about Linear Layout along with attributes. With an example. [IT-REG-2019] [IT-SUP-2021] OR List out different types of layouts? Explain any two layouts. [IT-REG-2021] [MCA-REG-2022] [BCA-SUP-2021]

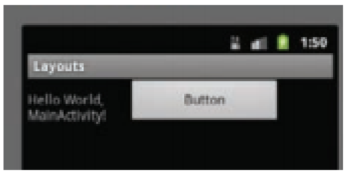
The LinearLayout arranges views in a single column or a single row.
Child views can be arranged either vertically or horizontally.
The root element is the layout and the Childs are the views in the UI.
Attributes in LinearLayout:

i. layout_gravity --Specifies how child Views are positioned
ii. layout_weight --Specifies how much of the extra space in the layout should be allocated to the View.
iii. Orientation --- specifies how child views are arranged (vertical/horizontal)

Ex 1: main.xml (The root element is the <LinearLayout> and childs are <Textview>)

```
<?xml version=”1.0” encoding=”utf-8”?>
<LinearLayout
        xmlns:android=”http://schemas.android.com/apk/res/android”
        android:orientation=”vertical”
        android:layout_width=”fill_parent”
        android:layout_height=”fill_parent”>
        <TextView
                android:layout_width=”fill_parent”
                android:layout_height=”wrap_content”
                android:text=”@string/hello”/>
</LinearLayout>
```

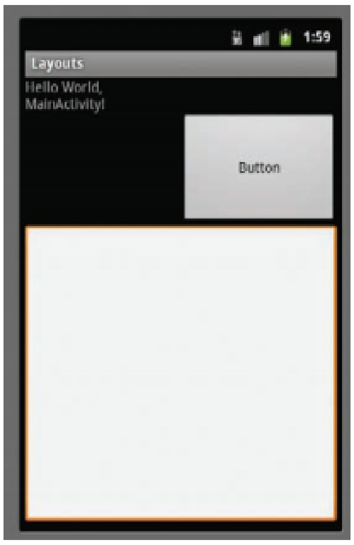




Note: The default orientation layout is **horizontal,** so if you omit the android:orientation attribute, the views will appear as below.

Ex 2:

```
<?xml version=”1.0” encoding=”utf-8”?>
<LinearLayout
        xmlns:android=”http://schemas.android.com/apk/res/android”
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">
        <TextView
                android:layout_width="105dp"
                android:layout_height="wrap_content"
                android:text="@string/hello" />
        <Button
                android:layout_width="160dp"
                android:layout_height="wrap_content"
                android:text="Button"
                android:layout_gravity="right"
                android:layout_weight="0.2" />
        <EditText
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:textSize="18sp"
                android:layout_weight="0.8" />
</LinearLayout>
```



- The button is aligned to the right of its parent using the layout_gravity attribute.
- The layout_weight attribute to specify the ratio in which the Button and EditText views occupy the remaining space on the screen.
- The total value for the layout_weight attribute must be equal to 1.

4. Explain about Absolute Layout & attributes with an example. [IT-REG-2019] [BCA-REG-2021] [IT-SUP-2021]

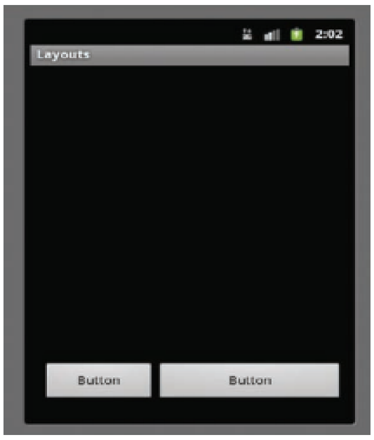
The Absolute Layout enables you to specify the exact location of its children.

Attributes in AbsoluteLayout:

i. layout_x --Specifies the x-coordinate of the View or ViewGroup
ii. layout_y --Specifies the y-coordinate of the View or ViewGroup

Ex: Main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        xmlns:android="http://schemas.android.com/apk/res/android">
        <Button
                android:layout_width="188dp"
                android:layout_height="wrap_content"
                android:text="Button"
                android:layout_x="126px"
                android:layout_y="361px"/>
        <Button
                android:layout_width="113dp"
                android:layout_height="wrap_content"
                android:text="Button"
                android:layout_x="12px"
                android:layout_y="361px"/>
</AbsoluteLayout>
```



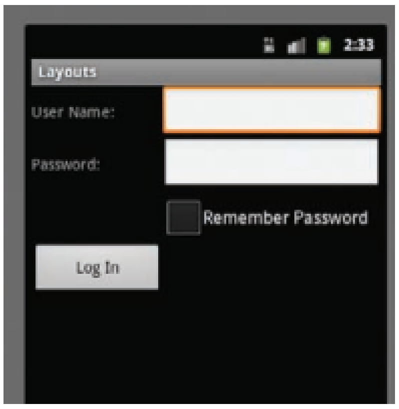
Note: The activity is viewed on a high-resolution screen the alignment of the UI is modified.

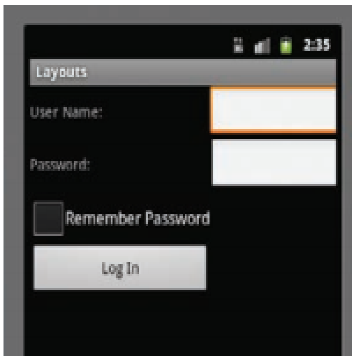
5. Explain about Table Layout & attributes with an example. [IT-REG-2019] [IT-SUP-2021] [BCA-SUP-2021]

- The TableLayout groups views into rows and columns.
- The <TableRow> element is used to designate a row in the table.
- Each row can contain one or more views.
- Each view you place within a row forms a cell.
- The width of each column is determined by the largest width of each cell in that column.

Ex: Main.xml

```
<?xml version=”1.0” encoding=”utf-8”?>
<TableLayout
        xmlns:android=”http://schemas.android.com/apk/res/android”
        android:layout_height=”fill_parent”
        android:layout_width=”fill_parent”>
        <TableRow>
                <TextView
                android:text=”User Name:”
                android:width =”120px”/>
                <EditText
                android:id=”@+id/txtUserName”
                android:width=”200px” />
        </TableRow>
        <TableRow>
                <TextView
                android:text=”Password:”/>
                <EditText
                android:id=”@+id/txtPassword”
                android:password=”true”/>
        </TableRow>
        <TableRow>
                <TextView />
                <CheckBox
                android:id=”@+id/chkRememberPassword”
                android:layout_width=”fill_parent”
                android:layout_height=”wrap_content”
                android:text=”Remember Password”/>
        </TableRow>
        <TableRow>
                <Button
                android:id=”@+id/buttonSignIn”
                android:text=”Log In” />
        </TableRow>
</TableLayout>
```





Note: In the above ex, there are two columns and four rows in the TableLayout. The cell directly under the Password TextView is populated with an <TextView/> empty element. If you don’t do this, the Remember Password checkbox will appear under the Password TextView.

6. Explain about Relative Layout & attributes with an example. [IT-REG-2019] OR what is the use of Relative Layout? Explain any six attributes of the Relative Layout. [BCA-RREG-2021] [IT-SUP-2021] [BCA-SUP-2021]

The Relative Layout enables you to specify how child views are positioned relative to each other.

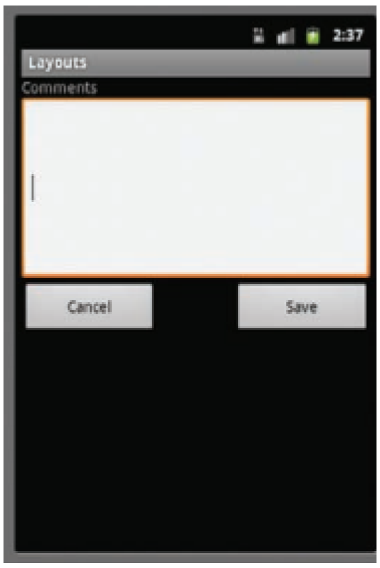
That each view embedded within the RelativeLayout has attributes that enable it to align with another view.
The value for each of these attributes is the ID for the view that you are referencing.

Attributes in RelativeLayout:

i. layout_alignParentTop – Aligns the view to the top of the parent view
ii. layout_alignParentBottom – Aligns the view to the bottom of the parent view
iii. layout_alignParentLeft – Aligns the view to the left of the parent view
iv. Layout_alignParentRight – Aligns the view to the right of the parent view
v. layout_alignLeft – Makes the left edge of this view match the left edge of the given anchor view ID and must be a reference to another resource.
vi. layout_alignRight – Makes the right edge of this view match the right edge of the given anchor view ID and must be a reference to another resource.
vii. layout_below - Positions the top edge of this view above the given anchor view ID and must be a reference to another resource.
viii. layout_above – Positions the bottom edge of this view above the given anchor view ID and must be a reference to another resource.
ix. Layout_centerVertical – Centers the view vertically within its parent view
x. layout_centerHorizontal – Centers the view horizontally within its parent view

Ex: Main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
        android:id="@+id/RLayout"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        xmlns:android="http://schemas.android.com/apk/res/android">
        <TextView
                android:id="@+id/lblComments"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Comments"
                android:layout_alignParentTop="true"
                android:layout_alignParentLeft="true"/>
        <EditText
                android:id="@+id/txtComments"
                android:layout_width="fill_parent"
                android:layout_height="170px"
                android:textSize="18sp"
                android:layout_alignLeft="@+id/lblComments"
                android:layout_below="@+id/lblComments"
                android:layout_centerHorizontal="true"/>
        <Button
                android:id="@+id/btnSave"
                android:layout_width="125px"
                android:layout_height="wrap_content"
                android:text="Save"
                android:layout_below="@+id/txtComments"
                android:layout_alignRight="@+id/txtComments"/>
        <Button
                android:id="@+id/btnCancel"
                android:layout_width="124px"
                android:layout_height="wrap_content"
```



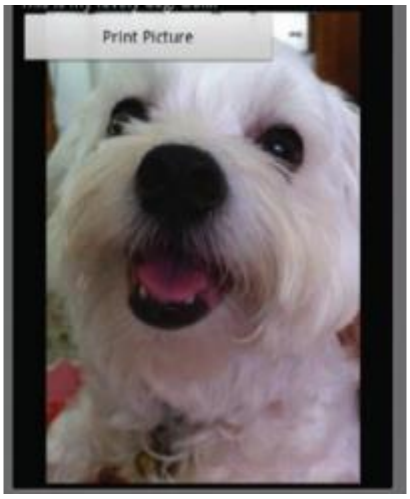
```
                android:text=”Cancel”
                android:layout_below=”@+id/txtComments”
                android:layout_alignLeft=”@+id/txtComments”/>
</RelativeLayout>
```

7. Explain about Frame Layout & attributes with an example.

The FrameLayout is a placeholder on screen that you can use to display a single view.
The view in this FrameLayout are always anchored to the top left of the layout.
If you add more than one view in frame layout, those view will overlap.

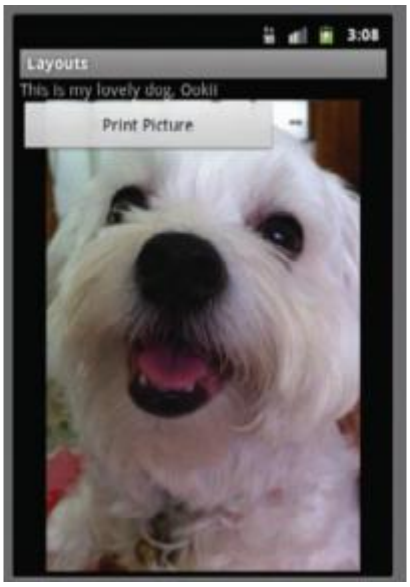
Ex 1: main.xml

```
<?xmlversion=”1.0”encoding=”utf-8”?>
<FrameLayout
    android:layout_width=”wrap_content”
    android:layout_height=”wrap_content”
    android:layout_alignLeft=”@+id/lblComments”
    android:layout_below=”@+id/lblComments”
    android:layout_centerHorizontal=”true”>
    <ImageView
                android:src = “@drawable/ookii”
                android:layout_width=”wrap_content”
                android:layout_height=”wrap_content”/>
    <Button
                android:layout_width=”124dp”
                android:layout_height=”wrap_content”
                android:text=”Print Picture” />
</FrameLayout>
```



Ex 2: main.xml

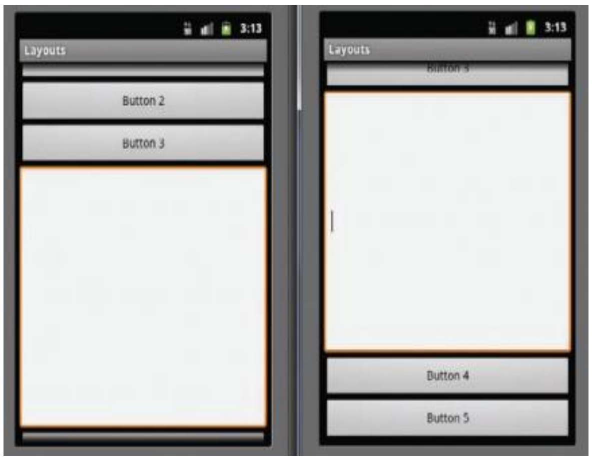
```
<?xmlversion=”1.0”encoding=”utf-8”?>
<RelativeLayout
    android:id=”@+id/Rlayout”
    android:layout_width=”fill_parent”
    android:layout_height=”fill_parent”
    xmlns:android=”http://schemas.android.com/apk/res/android”>
    <TextView
                android:id=”@+id/lblcomments”
                android:layout_width=”wrap_content”
                android:layout_height=”wrap_content”
                android:text=”Hello, Android”
                android:layout_alignParentTop=”true”
                android:layout_alignParentLeft=”true”/>
    <FrameLayout
    android:layout_width=”wrap_content”
    android:layout_height=”wrap_content”
    android:layout_alignLeft=”@+id/lblComments”
    android:layout_below=”@+id/lblComments”
    android:layout_centerHorizontal=”true”>
    <ImageView
                android:src = “@drawable/ookii”
                android:layout_width=”wrap_content”
                android:layout_height=”wrap_content”/>
```



```
        <Button
                android:layout_width=”124dp”
                android:layout_height=”wrap_content”
                android:text=”Print Picture” />
</FrameLayout>
```

8. Write a code for scroll view with an example. [IT-REG-2021] OR Explain the concept of Scrollview with an example.

- A ScrollView is a special type of FrameLayout in that it enables users to scroll through a list of views that occupy more space than the physical display.
- The ScrollView can contain only one child view or ViewGroup, which normally is a LinearLayout.
- The ScrollView enabling the users to drag the screen upward to reveal the views located at the bottom of the screen.



Ex: main.xml

```
<?xmlversion=”1.0”encoding=”utf-8”?>
<ScrollView
        android:layout_width=”fill_parent”
        android:layout_height=”fill_parent”
        xmlns:android=”http://schemas.android.com/apk/res/android”>
        <LinearLayout
                android:layout_width=”fill_parent”
                android:layout_height=”wrap_content”
                android:orientation=”vertical”>
                <Button
                        android:id=”@+id/button1”
                        android:layout_width=”fill_parent”
                        android:layout_height=”wrap_content”
                        android:text=”Button 1”/>
                <Button
                        android:id=”@+id/button2”
                        android:layout_width=”fill_parent”
                        android:layout_height=”wrap_content”
                        android:text=”Button 2”/>
                <Button
                        android:id=”@+id/button3”
                        android:layout_width=”fill_parent”
                        android:layout_height=”wrap_content”
                        android:text=”Button 3”/>
                <EditText
                        android:id=”@+id/txt”
                        android:layout_width=”fill_parent”
                        android:layout_height=”300px”/>
                <Button
                        android:id=”@+id/button4”
                        android:layout_width=”fill_parent”
                        android:layout_height=”wrap_content”
                        android:text=”Button 4”/>
                <Button
```

```
                    android:id=”@+id/button5”
                    android:layout_width=”fill_parent”
                    android:layout_height=”wrap_content”
                    android:text=”Button 5”/>
    </LinearLayout>
</ScrollView>
```

9. What is overriding? Explain Overriding Activity Methods in Android? [MCA-REG-2022]

In object-oriented programming, users can inherit the properties and behaviour of a superclass in subclasses. A subclass can override methods of its superclass, substituting its own implementation of the method for the superclass's implementation.

Sometimes the overriding method will completely replace the corresponding functionality in the superclass, while in other cases the superclass's method must still be called from the overriding method. Therefore most programming languages require that an overriding method must explicitly call the overridden method on the superclass for it to be executed.

```
// base lifecycle
onCreate(Bundle savedInstanceState);
onStart();
onRestart();
onResume();
onPause();
onStop();
onDestroy();
finalize();
onUserLeaveHint();
// instance state
onSaveInstanceState(Bundle outState);
onRestoreInstanceState(Bundle savedInstanceState)
// others
onCreateContextMenu(ContextMenu menu, View v, ContextMenuInfo menuInfo);
onCreateOptionsMenu(Menu menu);
onOptionsItemSelected(MenuItem item);
onContextItemSelected(MenuItem item);
onMenuItemSelected(int featureId, MenuItem item);
onActivityResult(int requestCode, int resultCode, Intent data);
onBackPressed();
```

10. Create an android application which display three button on main activity named RED, GREEN, and BLUE. The background color of the activity should be changed to appropriate color when user click on any of these buttons. [BCA-SUP-2021]

Activity.XML fie

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

```
android:layout_width=”160dp”
android:layout_height=”wrap_content”
android:text=”RED”
android:onClick=”colourred”/>
<Button
android:layout_width=”160dp”
android:layout_height=”wrap_content”
android:text=”GREEN”
android:onClick=”colourred”/>
<Button
android:layout_width=”160dp”
android:layout_height=”wrap_content”
android:text=”BLUE”
android:onClick=”colourred”/>
</LinearLayout>
```

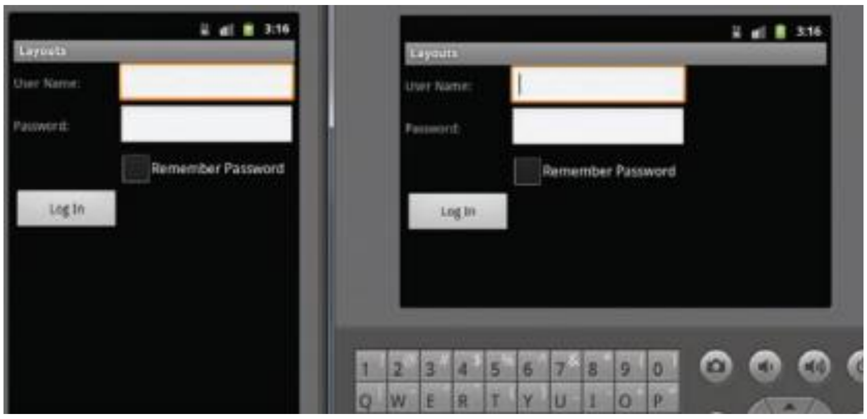
Activity.java file

```
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.RelativeLayout;
import androidx.appcompat.app.AppCompatActivity;
public class MainActivity extends AppCompatActivity {
    protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
         public void colourred(View view) {
                    relativeLayout.setBackgroundResource(R.color.red);
         }
         public void colourblue(View view) {
                    relativeLayout.setBackgroundResource(R.color.blue);
         }
         public void colourgreen(View view) {
                    relativeLayout.setBackgroundResource(R.color.green);
         }
    }
}
```

UNIT – IV: DISPLAY ORIENTATION

1.Display orientation Basics OR Briefly explain about adapting to display orientation [IT-REG-2021]

- Android supports two screen orientations: portrait and landscape.
- By default, when you change the display orientation of your Android device, the current activity that is displayed will automatically redraw its content in the new orientation.
- This is because the onCreate() event of the activity is fired whenever there is a change in display orientation.
- However, when the views are redrawn, they may be drawn in their original locations (depending on the layout selected).
- As you can observe in landscape mode, a lot of empty space on the right of the screen could be used.
- Any additional views at the bottom of the screen would be hidden when the screen orientation is set to landscape.



There are two techniques to handle changes in screen orientation:

a. Anchoring — The easiest way is to “anchor” your views to the four edges of the screen. When the screen orientation changes, the views can anchor neatly to the edges.
b. Resizing and repositioning — Whereas anchoring and centralizing are simple techniques to ensure that views can handle changes in screen orientation, the ultimate technique is resizing each and every view according to the current screen orientation.

2. What is meant by Anchoring Views. Explain with an Example. [IT-REG-2019] [BCA-RREG-2021] [IT-SUP-2021] [MCA-REG-2022]

- Anchoring — The easiest way is to “anchor” your views to the four edges of the screen. When the screen orientation changes, the views can anchor neatly to the edges.
- Anchoring could be easily achieved by using RelativeLayout.
- In this Layout, the screen orientation changes to landscape mode, the four buttons are aligned to the four edges of the screen, and the center button is centered in the middle of the screen with its width fully stretched.

The attributes present in Relative Layout are

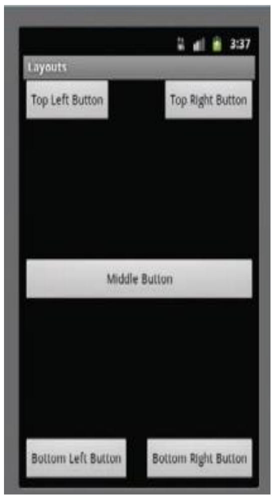
- layout_alignParentLeft — Aligns the view to the left of the parent view
- layout_alignParentRight — Aligns the view to the right of the parent view

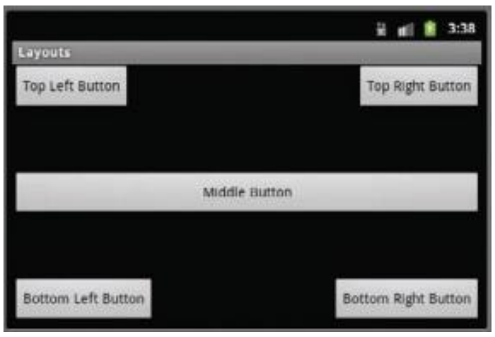
- layout_alignParentTop — Aligns the view to the top of the parent view
- layout_alignParentBottom — Aligns the view to the bottom of the parent view
- layout_centerVertical — Centers the view vertically within its parent view
- layout_centerHorizontal — Centers the view horizontally within its parent view

the below activity is viewed in portrait mode.

Ex: The UI contains five buttons. Whenever changes the orientation also it will occupy the entire space available in Activity.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        xmlns:android="http://schemas.android.com/apk/res/android" >
        <Button
                android:id="@+id/button1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Top Left Button"
                android:layout_alignParentLeft="true"
                android:layout_alignParentTop="true" />
        <Button
                android:id="@+id/button2"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Top Right Button"
                android:layout_alignParentTop="true"
                android:layout_alignParentRight="true"/>
        <Button
                android:id="@+id/button3"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Bottom Left Button"
                android:layout_alignParentLeft="true"
                android:layout_alignParentBottom="true"/>
        <Button
                android:id="@+id/button4"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Bottom Right Button"
                android:layout_alignParentRight="true"
                android:layout_alignParentBottom="true"/>
        <Button
                android:id="@+id/button5"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:text="Middle Button"
                android:layout_centerVertical="true"
                android:layout_centerHorizontal="true"/>
</RelativeLayout>
```

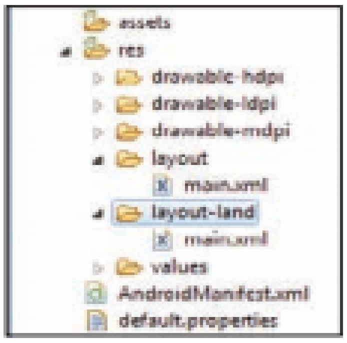




The contains in the UI will occupying the all the space available in both the orientations (landscape and portrait).

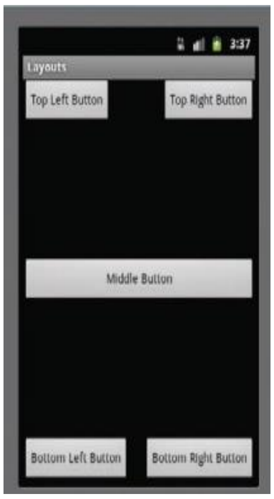
3. What is meant by Resizing and Repositioning. Explain with an Example. [IT-REG-2019] [IT-SUP-2021] [MCA-REG-2022]

- It is the easiest way to customize the UI based on screen orientation. Initially the UI is designed in main.xml file which is present in res/layout folder. It will support the portrait mode.
- To support the Landscape mode, create a new folder in the res folder and name it as layout-land. In this maintain a new main.xml which is to be designed corresponding to the landscape mode
- Basically, the main.xml file contained within the layout folder defines the UI for the activity in portrait mode, whereas the main.xml file in the layout-land folder defines the UI in landscape mode.



Ex: main.xml (portrait mode)

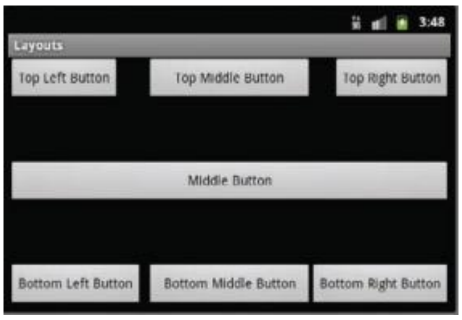
```
<?xml version=”1.0” encoding=”utf-8”?>
<RelativeLayout
        android:layout_width=”fill_parent”
        android:layout_height=”fill_parent”
        xmlns:android=”http://schemas.android.com/apk/res/android” >
        <Button
                android:id=”@+id/button1”
                android:layout_width=”wrap_content”
                android:layout_height=”wrap_content”
                android:text=”Top Left Button”
                android:layout_alignParentLeft=”true”
                android:layout_alignParentTop=”true” />
        <Button
                android:id=”@+id/button2”
                android:layout_width=”wrap_content”
                android:layout_height=”wrap_content”
                android:text=”Top Right Button”
                android:layout_alignParentTop=”true”
                android:layout_alignParentRight=”true”/>
        <Button
                android:id=”@+id/button3”
                android:layout_width=”wrap_content”
                android:layout_height=”wrap_content”
                android:text=”Bottom Left Button”
                android:layout_alignParentLeft=”true”
                android:layout_alignParentBottom=”true”/>
        <Button
                android:id=”@+id/button4”
                android:layout_width=”wrap_content”
                android:layout_height=”wrap_content”
                android:text=”Bottom Right Button”
                android:layout_alignParentRight=”true”
                android:layout_alignParentBottom=”true”/>
        <Button
```



```
                    android:id=”@+id/button5”
                    android:layout_width=”fill_parent”
                    android:layout_height=”wrap_content”
                    android:text=”Middle Button”
                    android:layout_centerVertical=”true”
                    android:layout_centerHorizontal=”true”/>
</RelativeLayout>
```

Main.xml (layout-land folder) – It will contains additional views to display in landscape mode.

```
<?xml version=”1.0” encoding=”utf-8”?>
<RelativeLayout
          android:layout_width=”fill_parent”
          android:layout_height=”fill_parent”
          xmlns:android=”http://schemas.android.com/apk/res/android” >
          <Button
                    android:id=”@+id/button1”
                    android:layout_width=”wrap_content”
                    android:layout_height=”wrap_content”
                    android:text=”Top Left Button”
                    android:layout_alignParentLeft=”true”
                    android:layout_alignParentTop=”true” />
          <Button
                    android:id=”@+id/button2”
                    android:layout_width=”wrap_content”
                    android:layout_height=”wrap_content”
                    android:text=”Top Right Button”
                    android:layout_alignParentTop=”true”
                    android:layout_alignParentRight=”true”/>
          <Button
                    android:id=”@+id/button3”
                    android:layout_width=”wrap_content”
                    android:layout_height=”wrap_content”
                    android:text=”Bottom Left Button”
                    android:layout_alignParentLeft=”true”
                    android:layout_alignParentBottom=”true”/>
          <Button
                    android:id=”@+id/button4”
                    android:layout_width=”wrap_content”
                    android:layout_height=”wrap_content”
                    android:text=”Bottom Right Button”
                    android:layout_alignParentRight=”true”
                    android:layout_alignParentBottom=”true”/>
          <Button
                    android:id=”@+id/button5”
                    android:layout_width=”fill_parent”
                    android:layout_height=”wrap_content”
                    android:text=”Middle Button”
                    android:layout_centerVertical=”true”
                    android:layout_centerHorizontal=”true”/>
```



```
          <Button
                    android:id=”@+id/button6”
                    android:layout_width=”fill_parent”
                    android:layout_height=”wrap_content”
                    android:text=”TopMiddle Button”
                    android:layout_centerVertical=”true”
                    android:layout_alignParentTop=”true”
                    android:layout_centerHorizontal=”true”/>
          <Button
                    android:id=”@+id/button7”
                    android:layout_width=”fill_parent”
                    android:layout_height=”wrap_content”
                    android:text=”Bottom Middle Button”
                    android:layout_centerVertical=”true”
                    android:layout_centerHorizontal=”true”
                    android:layout_alignParentTop=”true”/>
</RelativeLayout>
```

Finally, the activity is loaded in landscape mode, there are now seven buttons, showing that different XML files are loaded when the device is in a different orientation.

In this method, when the orientation of the device changes, Android will automatically load the appropriate XML file for your activity depending on the current screen orientation.

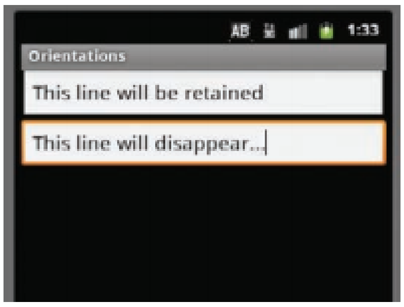
4. Briefly explain about how to Manage the Changes to the Screen Orientation [MCA-REG-2022] [BCA-SUP-2021]

The orientation is changes it will redraw its contents, means initially it will close the present activity UI file and again loads the corresponding UI file. In this process it will loss the state information along with data.

```
<?xml version=”1.0” encoding=”utf-8”?>
<LinearLayout
          android:layout_width=”fill_parent”
          android:layout_height=”fill_parent”
          android:orientation=”vertical”
          xmlns:android=”http://schemas.android.com/apk/res/android” >
          <EditText
                    android:id=”@+id/ed1”
                    android:layout_width=”fill_parent”
                    android:layout_height=”fill_parent” />
          <EditText
                    android:layout_width=”fill_parent”
                    android:layout_height=”fill_parent” />
</LinearLayout>
```

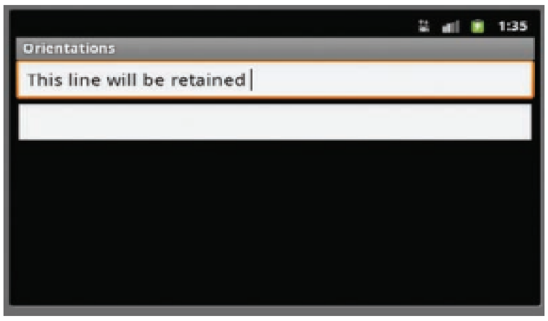
mainActivity.java – to understand the execution of activity lifecycle.

```
package com.example.myactivity101;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
public class MainActivity extends AppCompatActivity {
        String tag=”Life cycle”;
        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_main);
                Log.d(tag,"onCreate() invoked");
        }
        protected void onStart() {
                super.onStart();
                Log.d(tag,"onStart() invoked");
        }
        protected void onResume() {
                super.onResume();
                Log.d(tag,"onResume() invoked");
         }
        protected void onPause() {
                 super.onPause();
                Log.d(tag,"onPause() invoked");
        }
        protected void onStop() {
                 super.onStop();
                Log.d(tag,"onstop() invoked");
        }
        protected void onRestart() {
                 super.onRestart();
                Log.d(tag,"onRestart() invoked");
        }
        protected void onDestroy() {
                 super.onDestroy();
                Log.d(tag,"onDestroy() invoked");
        }
 }
```



This line will be retained

This line will disappear...



This line will be retained

- When the screen orientation changes it display one edittext with information and second edittext is empty.
- We have to preserve the state of activity before it changes orientation.
- The views that are named (via the android:id attribute) in an activity will have their state persisted when the activity they are contained in is destroyed.
- For example, the user may change orientation while entering some text into an EditText view. When this happens, any text inside the EditText view will be persisted and restored automatically when the activity is re-created. In contrast, if you do not name the EditText view using the android:id attribute, the activity will not be able to persist the text currently contained within it.

Persisting State Information during Changes in Configuration

i. Implement the **onSaveInstanceState()** method, it provides a Bundle object as an argument so that you can use it to **save your activity’s state**.

```
public void onSaveInstanceState(Bundle outState) {
        outState.putString(“ID”, “1234567890”);
        super.onSaveInstanceState(outState);
    }
```

When an activity is re-created, the onCreate() event is first fired, followed by the **onRestoreInstanceState()** event, which enables you to **retrieve the state** that you saved previously in the onSaveInstanceState event through the Bundle object in its argument.

```
    public void onRestoreInstanceState(Bundle savedInstanceState) {
        super.onRestoreInstanceState(savedInstanceState);
        String ID = savedInstanceState.getString(“ID”);
    }
```

To save more complex data structures, then this is not an adequate solution.

ii. In this, we use is the **onRetainNonConfigurationInstance()** event. This event is fired when an activity is about to be destroyed due to a configuration change. To save your current data by returning it in this event, like this:

```
      publicObjectonRetainNonConfigurationInstance(){
          return(“Sometexttopreserve”);
      }
```

Note: This event returns an Object type, which pretty much allows you to return any data type. To extract the saved data, you can extract it in the **onCreate()** event, using the **getLastNonConfigurationInstance()** method, like this:

```
      public voidonCreate(BundlesavedInstanceState){
          super.onCreate(savedInstanceState);
          setContentView(R.layout.main);
          Log.d(“StateInfo”,“onCreate”);
          String str = (String) getLastNonConfigurationInstance();
      }
```

5. How to Detect Orientation Changes in an Activity

The device’s current orientation during run time is to be known by the **WindowManager** class. Programmatically detect the current orientation of Activity is as follows.

```
import android.util.Log;
import android.view.Display;
import android.view.WindowManager;
public voidonCreate(BundlesavedInstanceState){
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      WindowManager wm = getWindowManager();
      Display d = wm.getDefaultDisplay();
      if (d.getWidth() > d.getHeight()){
          Log.d(“Orientation”, “Landscape mode”);
      }
      else{
          Log.d(“Orientation”, “Portrait mode”);
      }
 }
```

The **getDefaultDisplay()** method returns a Display object representing the screen of the device. The width and height and deduce the current orientation.

6. How to Control the Orientation of the Activity

i. **setRequestOrientation() method**:

Initially all the apps will be in portrait mode. But some applications like a game that should only be viewed in landscape mode. In this case, you can programmatically force a change in orientation using the setRequestOrientation() method of the Activity class:

```
import android.content.pm.ActivityInfo;
public class MainActivity extends Activity {
        public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
}
```

portrait mode: use the ActivityInfo.SCREEN_ORIENTATION_PORTRAIT
s**etRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);**

ii.android:screenOrientation

AndroidManifest.xml file add **android:screenOrientation** and specify the value according to the orientation

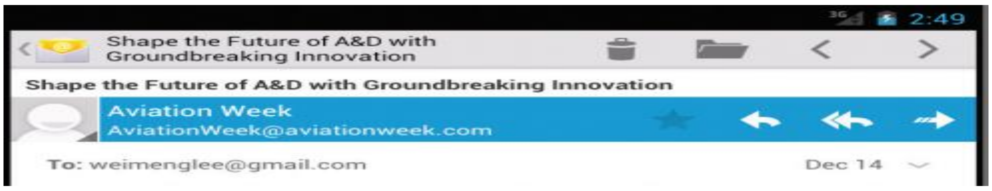
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="net.learn2develop.Orientations"
android:versionCode="1"
android:versionName="1.0">
<application android:icon="@drawable/icon" android:label="@string/app_name">
<activity android:name=".MainActivity"
android:label="@string/app_name"
android:screenOrientation="landscape" >
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
</application>
<uses-sdk android:minSdkVersion="9" />
</manifest>
```

The preceding example constrains the activity to a certain orientation (landscape in this case) and prevents the activity from being destroyed; that is, the activity will not be destroyed and the onCreate() event will not be fired again when the orientation of the device changes.

Following are two other values that you can specify in the **android:screenOrientation** attribute:

- ➤➤ portrait — Portrait mode
- ➤➤ sensor — Based on the accelerometer

7. Explain elaborately about utilizing the action bar with suitable examples. [BCA-REG-2021] [MCA-REG-2022]


2:49
Shape the Future of A&D with Groundbreaking Innovation
Shape the Future of A&D with Groundbreaking Innovation
Aviation Week
AviationWeek@aviationweek.com
To: weimenglee@gmail.com
Dec 14

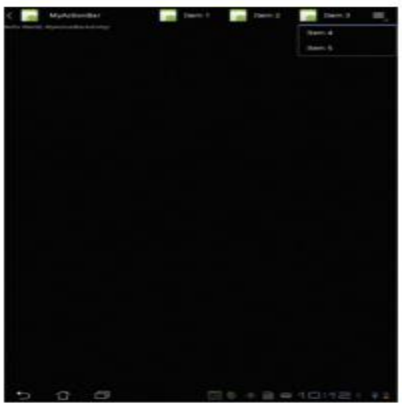
Action Bar: The feature introduced in Android 3 and 4 is the Action Bar. Title bar located at the top of the device's screen, the Action Bar displays the application icon together with the activity title. Optionally, on the right side of the Action Bar are action items.

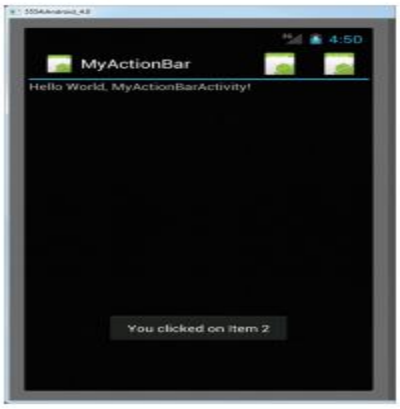
Ex:
Main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">
<TextView
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Welcome to Mobile Application Development lab"/>
</android.support.constraint.ConstraintLayout>
```

Main.java

```
package net.learn2develop.MyActionBar;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Toast;
public class MyActionBarActivity extends Activity {
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
}
public boolean onCreateOptionsMenu(Menu menu) {
super.onCreateOptionsMenu(menu);
CreateMenu(menu);
return true;
}
public boolean onOptionsItemSelected(MenuItem item)
{
return MenuChoice(item);
}
private void CreateMenu(Menu menu)
{
            MenuItem mnu1 = menu.add(0, 0, 0, “Item 1”);
            {
            mnu1.setIcon(R.drawable.ic_launcher);
            mnu1.setShowAsAction(MenuItem.SHOW_AS_ACTION_IF_ROOM);
            }
            MenuItem mnu2 = menu.add(0, 1, 1, “Item 2”);
            {
            mnu2.setIcon(R.drawable.ic_launcher);
            mnu2.setShowAsAction(MenuItem.SHOW_AS_ACTION_IF_ROOM);
            }
            MenuItem mnu3 = menu.add(0, 2, 2, “Item 3”);
            {
            mnu3.setIcon(R.drawable.ic_launcher);
            mnu3.setShowAsAction(MenuItem.SHOW_AS_ACTION_IF_ROOM);
            }
            MenuItem mnu4 = menu.add(0, 3, 3, “Item 4”);
            {
            mnu4.setShowAsAction(MenuItem.SHOW_AS_ACTION_IF_ROOM);
            }
            MenuItem mnu5 = menu.add(0, 4, 4, “Item 5”);
            {
            mnu5.setShowAsAction(MenuItem.SHOW_AS_ACTION_IF_ROOM);
            }
}
private boolean MenuChoice(MenuItem item)
{
switch (item.getItemId()) {
case 0: Toast.makeText(this, “You clicked on Item 1”,Toast.LENGTH_LONG).show();
            return true;
case 1: Toast.makeText(this, “You clicked on Item 2”,Toast.LENGTH_LONG).show();
            return true;
case 2:Toast.makeText(this, “You clicked on Item 3”,Toast.LENGTH_LONG).show();
            return true;
case 3:Toast.makeText(this, “You clicked on Item 4”,Toast.LENGTH_LONG).show();
            return true;
case 4:Toast.makeText(this, “You clicked on Item 5”,Toast.LENGTH_LONG).show();
            return true;
}
return false;         }           }
```



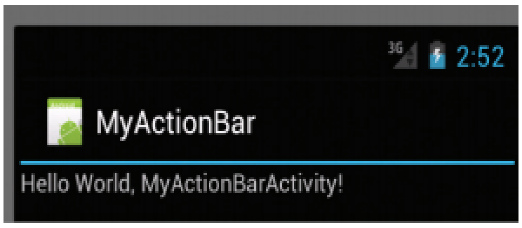


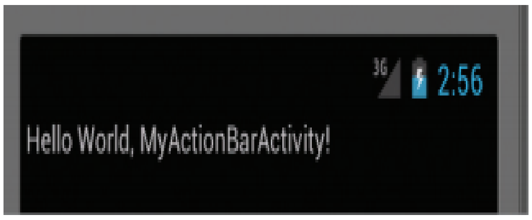
Hello World, MyActionBarActivity!

You clicked on Item 2

8. Write a snippet to customize the Action Items.

a. Hiding the Action Bar:





This is to be achieved in two different ways

i. Specify android:theme="@android:style/Theme.Holo.NoActionBar " in AndroidManifest file.

AndroidManifest.xml file:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="net.learn2develop.MyActionBar"
android:versionCode="1"
android:versionName="1.0" >
<uses-sdk android:minSdkVersion="14" />
<application
android:icon="@drawable/ic_launcher"
android:label="@string/app_name" >
<activity
android:label="@string/app_name"
android:name=".MyActionBarActivity"
android:theme="@android:style/Theme.Holo.NoActionBar" >
<intent-filter >
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
</application>
</manifest>
```

ii. Programmatically remove the Action Bar using the ActionBar class.

Initially remove the android: theme attribute in AndroidManifest file. Create the object of ActionBar class and call the hide method to remove ActionBar.

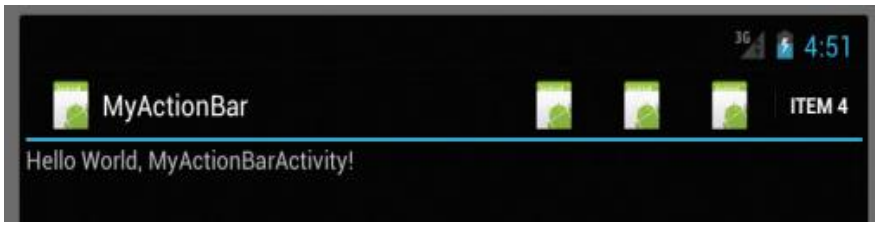
```
ActionBar ab = getActionBar();
ab.hide();
```

The following example is shown how programmatically remove ActionBar.

Ex:

```
package net.learn2develop.MyActionBar;
import android.app.ActionBar;
import android.app.Activity;
import android.os.Bundle;
public class MyActionBarActivity extends Activity {
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
ActionBar actionBar = getActionBar();
actionBar.hide();
//actionBar.show(); //---show it again---          }          }
```

b. Displaying Action items along with Names:



To display the text for the action items together with the icon, you could use the "|" operator together with the MenuItem.SHOW_AS_ACTION_WITH_TEXT constant:

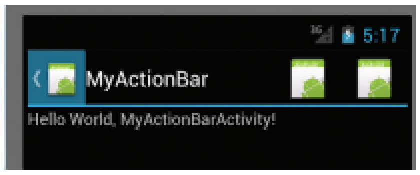
```
MenuItem mnu1 = menu.add(0, 0, 0, "Item 1");
{
        mnu1.setIcon(R.drawable.ic_launcher);
        mnu1.setShowAsAction(MenuItem.SHOW_AS_ACTION_IF_ROOM |
                              MenuItem.SHOW_AS_ACTION_WITH_TEXT);
}
```

This causes the icon to be displayed together with the text of the menu item.

c. Setting Action Bar as clickable option:

To make the application icon clickable, then call the setDisplayHomeAsUpEnabled() method:

```
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ActionBar a = getActionBar();
        a.setDisplayHomeAsUpEnabled(true);
        //a.hide();
        //a.show(); //---show it again---
}
```



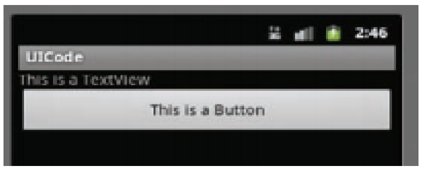
9. How to Create UI programmatically? Explain with a program. [IT-REG-2019] [IT-REG-2021] [IT-SUP-2021] [BCA-SUP-2021]

The UI is designed not only in XML, we can create by using code also. By using approach the UI is dynamically loaded during runtime whenever it requires.

Ex: suppose you are building a cinema ticket reservation system and your application will display the seats of each cinema using buttons. To dynamically generate the UI based on the cinema selected by the user.

MainActivity.java

```
package net.learn2develop.UICode;
import android.app.Activity;
import android.os.Bundle;
import android.view.ViewGroup.LayoutParams;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.TextView;
public class MainActivity extends Activity {
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                LayoutParams params =new LinearLayout.LayoutParams(
                    LayoutParams.FILL_PARENT, LayoutParams.WRAP_CONTENT);

                    LinearLayout LL= new LinearLayout(this);
                    LL.setOrientation(LinearLayout.VERTICAL);

                    TextView tv = new TextView(this);
                    tv.setText("This is a TextView");
                    tv.setLayoutParams(params);

                    Button btn = new Button(this);
                    btn.setText("This is a Button");
                    btn.setLayoutParams(params);

                    LL.addView(tv);
                    LL.addView(btn);

                    LinearLayout.LayoutParams lp=new LinearLayout.LayoutParams(
                    LayoutParams.FILL_PARENT, LayoutParams.WRAP_CONTENT );

                    this.addContentView(LL, lp);
          }
}
```



This is a TextView

- Initially we have to create the layoutparams.
- Linear layout object LL is created and added the orientation of layout.
- Textview object is created with the text message and params
- Button view is created along with text and params.
- Adding the button and Textview to the linear layout object.
- Linear layout params are also created.
- Finally, entire UI is to be added to the activity by using this.addContentView method.

UNIT V: DESIGNING YOUR USER INTERFACE USING VIEWS

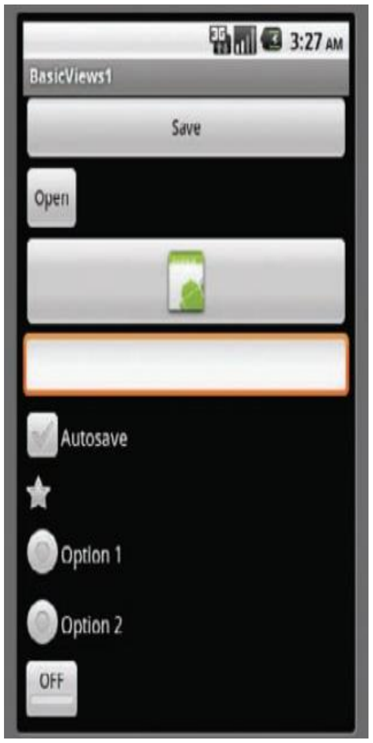
1.Write an android program which uses the basic views to design UI [IT-REG-2019] OR write an android program uses EditText, button, TextView, ToggleButton, and Radio Button. [IT-REG-2021] [IT-SUP-2021] [BCA-SUP-2021] [MCA-REG-2022]

- ❖ TextView – The TextView is used to display text to the user. It is one of the view frequently use when you develop android applications.
- ❖ Button — Represents a push-button widget
- ❖ ImageButton — Similar to the Button view, except that it also displays an image
- ❖ EditText — A subclass of the TextView view, except that it allows users to edit its text content. The EditText view displays a rectangular region where the user can enter some text. If layout _height to wrap_content so that if the user enters a long string of text, its height will automatically be adjusted to fit the content.
- ❖ CheckBox — A special type of button that has two states: checked or unchecked
- ❖ RadioGroup and RadioButton — The RadioButton has two states: either checked or unchecked. Once a RadioButton is checked, it cannot be unchecked. A RadioGroup is used to group together one or more RadioButton views, thereby allowing only one RadioButton to be checked within the RadioGroup.
- ❖ ToggleButton — Displays checked/unchecked states using a light indicator

Basic views: main. Xml is located in res/layout folder

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
          xmlns:android="http://schemas.android.com/apk/res/android"
         xmlns:app="http://schemas.android.com/apk/res-auto"
          xmlns:tools="http://schemas.android.com/tools"
         android:orientation="vertical"
         android:layout_width="match_parent"
          android:layout_height="match_parent"
         tools:context=".MainActivity">
         <TextView
                   android:layout_width="match_parent"
                   android:layout_height="wrap_content"
                   android:text="hello"          />
         <Button
                   android:id= “@id/btnsave”
                   android:layout_width="match_parent"
                   android:layout_height="wrap_content"
                   android:text="Save"           />
         <Button
                  android:id= “@id/btnopen”
                  android:layout_width="wrap_content"
```

```
                android:layout_height="wrap_content"
                android:text="open"             />

        <ImageButton
                android:id= “@id/btnsave”                  <!--The image is set through the src attribute-->
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:src="@drawable/icon"/>
        <EditText
                android:id= “@id/txtname”
                android:layout_width="match_parent"
                android:layout_height="wrap_content"        />
        <Checkbox
                android:id= “@id/chkautosave”
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="AutoSave"     />
        <Checkbox            <!--The default style is square. To apply a star style use style attribute -->
                android:id= “@id/star”
                style=”?android:att/startStyle”
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"    />
        <RadioGroup
                android:id= “@id/rdbgp1”
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:orientation="vertical"            >
                <RadioButton
                        android:id= “@id/rdb1”
                        android:layout_width="match_parent"
                        android:layout_height="wrap_content"
                        android:text=”option1"        />
                <RadioButton
                        android:id= “@id/rdb2”
                        android:layout_width="match_parent"
                        android:layout_height="wrap_content"
                        android:text=”option2"        />
        </RadioGroup>
        <ToggleButton
                android:id= “@id/toggle1”
                android:layout_width="wrap_content"
                android:layout_height="wrap_content”     />
</LinearLayout>
```



Note: The id attribute is an identifier for a view so that it may later be retrieved using the **View.findViewById() or Activity.findViewById()** methods.

2. How to handle the BasicViews. OR Ecplain how to handle basic views in Android? [MCA-REG-2022]

Main.java

```
package net.learn2develop.BasicViews1;
 import android.app.Activity;
 import android.os.Bundle;
 import android.view.View;
 import android.widget.CheckBox;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.Toast;
import android.widget.ToggleButton;
public class MainActivity extends Activity {
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
Button btnOpen = (Button) findViewById(R.id.btnOpen);
btnOpen.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
      DisplayToast("You have clicked the Open button");  }   });

Button btnSave = (Button) findViewById(R.id.btnSave);
btnSave.setOnClickListener(new View.OnClickListener()
{
public void onClick(View v) {
     DisplayToast("You have clicked the Save button"); }   });

CheckBox checkBox = (CheckBox) findViewById(R.id.chkAutosave);
checkBox.setOnClickListener(new View.OnClickListener()
{
public void onClick(View v) {
     if (((CheckBox)v).isChecked())
             DisplayToast("CheckBox is checked");
     else
             DisplayToast("CheckBox is unchecked"); }   });

RadioGroup  rdg1 = (RadioGroup) findViewById(R.id.rdbGp1);
rdg1.setOnCheckedChangeListener(new OnCheckedChangeListener(){
public void onCheckedChanged(RadioGroup group, int checkedId) {
RadioButton rb1 = (RadioButton) findViewById(R.id.rdb1);
      if (rb1.isChecked())
             DisplayToast("Option 1 checked!");
      else
             DisplayToast("Option 2 checked!");  }          });
```

```
            ToggleButton  tb1 = (ToggleButton) findViewById(R.id.toggle1);
            tb1.setOnClickListener(new View.OnClickListener(){
                public void onClick(View v) {
                if (((ToggleButton)v).isChecked())
                        DisplayToast("Toggle button is On");
                else
                        DisplayToast("Toggle button is Off");  }   });
            private void DisplayToast(String msg) {
            Toast.makeText(getBaseContext(), msg, Toast.LENGTH_SHORT).show(); }  }
```

To handle the events fired by each view, you first have to programmatically locate the view that you created during the onCreate() event. You do so using the Activity.findViewById() method, supplying it with the ID of the view:

The onClick() method is called when the view is clicked. For the CheckBox, to determine its state you have to typecast the argument of the onClick() method to a CheckBox and then check its isChecked() method to see if it is checked.

RadioButton, you need to use the setOnCheckedChangeListener() method on the RadioGroup to register a callback to be invoked when the checked RadioButton changes in this group.

When a RadioButton is selected, the onCheckedChanged() method is fired. Within it, you locate individual RadioButtons and then call their isChecked() method to determine which RadioButton is selected. Alternatively, the onCheckedChanged() method contains a second argument that contains a unique identifier of the RadioButton selected.

3. Explain the concept of Progress Bar with an example. [IT-REG-2021][BCA-REG-2021] [BCA-SUP-2021]

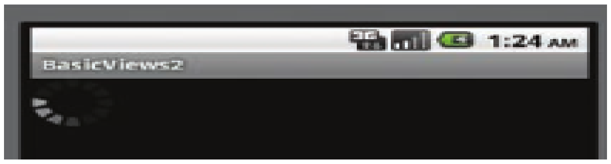
The ProgressBar view provides visual feedback of some ongoing tasks, such as when you are performing a task in the background.

For example, you might be downloading some data from the Web and need to update the user about the status of the download. In this case, the ProgressBar view is a good choice for this task.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
            xmlns:android="http://schemas.android.com/apk/res/android"
             android:orientation="vertical"
            android:layout_width="match_parent"
            android:layout_height="match_parent" >
           <ProgressBar
                    android:id="@+id/progressbar"
                    android:layout_height="wrap_content"
                    android:layout_width="wrap_content"             />
</LinearLayout>
```

MainActivity.java

```
package net.learn2develop.BasicViews2;
import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.widget.ProgressBar;
public class MainActivity extends Activity {
private static int progress;
private ProgressBar progressBar;
private int progressStatus = 0;
 private Handler handler = new Handler();
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
 progress = 0;
 progressBar = (ProgressBar) findViewById(R.id.progressbar);
new Thread(new Runnable() {
     public void run() {
              while (progressStatus < 10){
                  progressStatus = doSomeWork();
             }
         handler.post(new Runnable() {
         public void run(){
              //---0 - VISIBLE; 4 - INVISIBLE; 8 - GONE---
              progressBar.setVisibility(8); } });
         }
private int doSomeWork() {
 try {
     Thread.sleep(500);  }
 catch (InterruptedException e) {
     e.printStackTrace();  }
return ++progress;
}
}).start();
} }
```



The default mode of the ProgressBar view is indeterminate — that is, it shows a cyclic animation. This mode is useful for tasks that do not have a clear indication of when they will be completed, such as when you are sending some data to a web service and waiting for the server to respond.

The run() method starts the execution of the thread, in this it will calls the doSomeWork() method to simulate doing some work. When the simulated work is done (after about five seconds), you use a Handler object to send a message to the thread to dismiss the progressbar.

When the task is completed, you hide the progressbar by setting the value as 8(GONE). The difference between invisible and gone is hides the progressbar. The gone is removes the progressbar from activity.

Customizing the ProgressBar:

i)To change the style of progress bar:

Style = “?android:att/progressBarStyleHorizontal”

The Possible values for style attribute are

Widget.progressBar.Horizontal

Widget.progressBar.Small

Widget.ProgressBar.Large

Widget.ProgressBar.Inverse

Widget.ProgressBar.Small.Inverse

Widget.ProgressBar.Large.Inverse

ii) The maximum value is to be set.

android:max=”100”   <!—sets the maximum value for progressbar-->

When the progress bar is reaches from 0 to 100, the progress bar will automatically close. Otherwise it will continue checking the progress value and update.

4. Explain the concept of AutoCompleteTextView with example. [IT-REG-2021] [BCA-REG-2021]

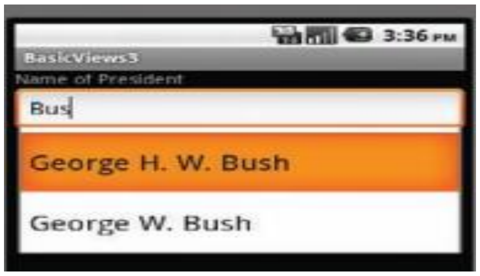
The AutoCompleteTextView is a view, which is similar to EditText, except that it shows a list of completion suggestions automatically while the user is typing.

Example: main.XML – UI design with AutoCompleteTextView

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
          xmlns:android="http://schemas.android.com/apk/res/android"
         android:layout_width="match_parent"
         android:layout_height="match_parent"
         android:orientation="vertical">
         <TextView
                  android:id="@+id/t1"
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"    />
         <AutoCompleteTextView
                  android:id="@+id/actv1"
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"    />
</LinearLayout>
```

MainActivity.java

```
package net.learn2develop.BasicViews3;
import android.app.Activity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
public class MainActivity extends Activity {
String[] presidents = {  "Dwight D.Eisenhower", "John F. Kemnedy", "Lyndon B.Johnson",
Richard Nixon", "Gerald Ford", "Jimmy Carter" }
Public void onCreate(Bundle saved/instanceState){
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
ArrayAdapter<String> adapter = new ArrayAdapter <string> (this,
android.R.layout.simple_dropdown_item_line, presidents);
AutoCompleteTextView tv=(AutoCompleteTextview) findViewByID(R.id.actv1);
tv.setThreshold(3);
tv.setAdapter(adaper);
}
}
```



- ❖ The string array contains the presidents name.
- ❖ The ArrayAdapter object manages the array of strings that will be displayed by the AutoCompleteTextView. AutoCompleteTextView to display in the simpe_dropdown_item_line mode.
- ❖ The setThreshold() method sets the minimum number of characters the user must type before the suggestions appear as a drop- down menu.
- ❖ The list of Suggestions to display for the AutoCompleteTestView is obtained from the ArrayAdapter object.

----------All THE BEST----------