1)

a.What is the use of intent in android? Explain it with syntax.

Android application components can connect to other Android applications. This connection is based on a task description represented by an `Intent` object.

Intents are asynchronous messages which allow application components to request functionality from other Android components. Intents allow you to interact with components from the same applications as well as with components contributed by other applications. For example, an activity can start an external activity for taking a picture.

Intents are objects of the `android.content.Intent` type. Your code can send them to the Android system defining the components you are targeting. For example, via the `startActivity()` method you can define that the intent should be used to start an activity.

An intent can contain data via a `Bundle`. This data can be used by the receiving component.

In Android the reuse of other application components is a concept known as *task*. An application can access other Android components to achieve a task.

```
Start the activity connect to the
# specified class

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

Android supports explicit and implicit intents. An application can define the target component directly in the intent (*explicit intent*) or ask the Android system to evaluate registered components based on the intent data(*implicit intents*).

b. Explain the differences between Resources and Assets?

The real main difference between the two is that in the `res` directory each file is given a pre-compiled `ID` which can be accessed easily through `R.id.[res id]`. This is useful to quickly and easily access images, sounds, icons...
The `assets` directory is more like a filesystem and provides more freedom to put any file you would like in there. You then can access each of the files in that system as you would when accessing any file in any file system through Java. This directory is good for things such as game details, dictionaries,...etc.

2a. **Android RelativeLayout**

Android RelativeLayout lays out elements based on their relationships with one another, and with the parent container. This is one of the most complicated layout, and we need several properties to actually get the layout we desire. That is, using RelativeLayout we can position a view to be **toLeftOf**, **toRightOf**, **below**or **above** its siblings.

We can also position a view with respect to its parent such as **centered horizontally**, **vertically** or both, or aligned with any of the edges of the parent RelativeLayout. If none of these attributes are specified on a child view then the view is by default rendered to the top left position.

The following are the major attributes used across **RelativeLayout**. They lay across three different categories:

Relative To Container

- android:layout_alignParentBottom : Places the bottom of the element on the bottom of the container
- android:layout_alignParentLeft : Places the left of the element on the left side of the container
- android:layout_alignParentRight : Places the right of the element on the right side of the container

- android:layout_alignParentTop : Places the element at the top of the container
- android:layout_centerHorizontal : Centers the element horizontally within its parent container
- android:layout_centerInParent : Centers the element both horizontally and vertically within its container
- android:layout_centerVertical : Centers the element vertically within its parent container

Relative to Siblings

- android:layout_above : Places the element above the specified element
- android:layout_below : Places the element below the specified element
- android:layout_toLeftOf : Places the element to the left of the specified element
- android:layout_toRightOf : Places the element to the right of the specified element

“@id/XXXXX” is used to reference an element by its id. One thing to remember is that referencing an element before it has been declared will produce an error so @+id/ should be used in such cases.

Alignment With Other Elements

- android:layout_alignBaseline : Aligns baseline of the new element with the baseline of the specified element
- android:layout_alignBottom : Aligns the bottom of new element in with the bottom of the specified element
- android:layout_alignLeft : Aligns left edge of the new element with the left edge of the specified element

- android:layout_alignRight : Aligns right edge of the new element with the right edge of the specified element
- android:layout_alignTop : Places top of the new element in alignment with the top of the specified element

b. FrameLayout is designed to block out an area on the screen to display a single item. Generally, FrameLayout should be used to hold a single child view, because it can be difficult to organize child views in a way that's scalable to different screen sizes without the children overlapping each other. You can, however, add multiple children to a FrameLayout and control their position within the FrameLayout by assigning gravity to each child, using the android:layout_gravity attribute.

Child views are drawn in a stack, with the most recently added child on top. The size of the FrameLayout is the size of its largest child (plus padding), visible or not (if the FrameLayout's parent permits). Views that are GONE are used for sizing only if setConsiderGoneChildrenWhenMeasuring() is set to true.

3. What is Listview? Write the java class to add any 10 items(String data) to a Listview using array adapter

ListView is a view group that displays a list of scrollable items. The list items are automatically inserted to the list using an Adapter that pulls content from a source such as an array or database query and converts each item result into a view that's placed into the list.

```
1. package com.windrealm.android;
2.
3. import java.util.ArrayList;
4. import java.util.Arrays;
5.
6. import android.app.Activity;
7. import android.os.Bundle;
8. import android.widget.ArrayAdapter;
9. import android.widget.ListView;
10.
11.     public class SimpleListViewActivity extends Activity {
12.
13.       private ListView mainListView ;
14.       private ArrayAdapter<String> listAdapter ;
15.
16.       /** Called when the activity is first created. */
17.       @Override
18.       public void onCreate(Bundle savedInstanceState) {
19.         super.onCreate(savedInstanceState);
20.         setContentView(R.layout.main);
21.
22.         // Find the ListView resource.
23.         mainListView = (ListView) findViewById( R.id.mainListVi
   ew );
24.
25.         // Create and populate a List of planet names.
26.         String[] planets = new String[] { "Mercury", "Venus", "
   Earth", "Mars",
27.                                           "Jupiter", "Saturn",
   "Uranus", "Neptune"};
28.         ArrayList<String> planetList = new ArrayList<String>();

29.         planetList.addAll( Arrays.asList(planets) );
```

```
30.
31.          // Create ArrayAdapter using the planet list.
32.          listAdapter = new ArrayAdapter<String>(this, R.layout.s
   implerow, planetList);
33.
34.          // Add more planets. If you passed a String[] instead o
   f a List<String>
35.          // into the ArrayAdapter constructor, you must not add
   more items.
36.          // Otherwise an exception will occur.
37.          listAdapter.add( "Ceres" );
38.          listAdapter.add( "Pluto" );
39.          listAdapter.add( "Haumea" );
40.          listAdapter.add( "Makemake" );
41.          listAdapter.add( "Eris" );
42.
43.          // Set the ArrayAdapter as the ListView's adapter.
44.          mainListView.setAdapter( listAdapter );
45.        }
46.     }
```

4.What is Fragments and explain its life cycle?

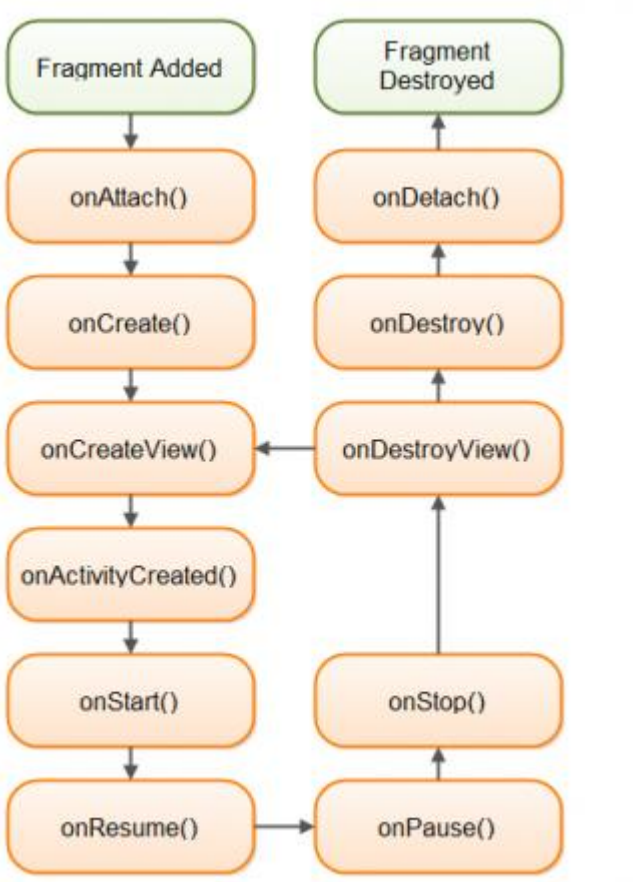
Android Fragment

Fragment class in Android is used to build dynamic User Interfaces. **Fragment** should be used within the Activity. A greatest advantage of fragments is that it simplifies the task of creating UI for multiple screen sizes. A activity can contain any number of fragments.

An Android fragment is not by itself a subclass of View which most other UI components are. Instead, a fragment has a view inside it. It is this view which is eventually displayed inside the activity in which the fragment lives.

Fragment Lifecycle

Android fragment lifecycle is illustrated in below image.



Below are the methods of fragment lifecycle.

onAttach() :This method will be called first, even before onCreate(), letting us know that your fragment has been attached to an activity. You are passed the Activity that will host your fragment

onCreateView() : The system calls this callback when it's time for the fragment to draw its UI for the first time. To draw a UI for the fragment, a View component must be returned from this method which is the root of the fragment's layout. We can return null if the fragment does not provide a UI

onViewCreated() : This will be called after onCreateView(). This is particularly useful when inheriting the onCreateView() implementation but we need to configure the resulting views, such as with a ListFragment and when to set up an adapter

onActivityCreated() :This will be called after onCreate() and onCreateView(), to indicate that the activity's onCreate() has completed. If there is something that's needed to be initialised in the fragment that depends upon the activity's onCreate() having completed its work then onActivityCreated() can be used for that initialisation work

onStart() : The onStart() method is called once the fragment gets visible

onPause() : The system calls this method as the first indication that the user is leaving the fragment. This is usually where you should commit any changes that should be persisted beyond the current user session

onStop() : Fragment going to be stopped by calling onStop()

onDestroyView() : It's called before onDestroy(). This is the counterpart to onCreateView() where we set up the UI. If there are things that are needed to be cleaned up specific to the UI, then that logic can be put up in onDestroyView()

onDestroy() : onDestroy() called to do final clean up of the fragment's state but Not guaranteed to be called by the Android platform.

onDetach() : It’s called after onDestroy(), to notify that the fragment has been disassociated from its hosting activity

5. . Write a short note on following attributes

A)orientation b)weight c)gravity d) padding e)layout_alignParentTop

A.orientation:It can take either horizontal or vertical as value for orientation.In which direction controls are arranged is specified here

B.weight

It can take values between 0 to 1. LinearLayout also supports assigning a weight to individual children with the android:layout_weight attribute. This attribute assigns an "importance" value to a view in terms of how much space it should occupy on the screen. A larger weight value allows it to expand to fill any remaining space in the parent view.

c. android:gravity sets the gravity of the content of the View its used on. Use the gravity attribute to position objects within a larger container. This will only work if the container is larger than the object it contains.

d. Padding

Padding is the extra space inside a view; between the views contents and its boundary. In our examples, it’s the extra space between the button’s text label and outside boundary.

The size of a view includes the padding:for an exact size view, increasing the padding won’t affect the size of the view using wrap_content will increase the size of the view if you increase the size of the padding using match_parent the view will be as big as its parent, minus the padding

e. layout_alignParentTop

android:layout_alignParentTop. If true, makes the top edge of this view match the top edge of the parent.

6. Explain the steps involved in adding and playing an audio in android application (write java code also)?

can play and control the audio files in android by the help of MediaPlayer class.

Here, we are going to see a simple example to play the audio file. In the next page, we will see the example to control the audio playback like start, stop, pause etc.

MediaPlayer class

The android.media.MediaPlayer class is used to control the audio or video files.

Methods of MediaPlayer class

There are many methods of MediaPlayer class. Some of them are as follows:

Method Description

public void setDataSource(String path) sets the data source (file path or http url) to use.

public void prepare() prepares the player for playback synchronously.

public void start() it starts or resumes the playback.

public void stop() it stops the playback.

public void pause()it pauses the playback.

public boolean isPlaying() checks if media player is playing.

public void seekTo(int millis) seeks to specified time in miliseconds.

public void setLooping(boolean looping) sets the player for looping or non-looping.

public boolean isLooping() checks if the player is looping or non-looping.

public void selectTrack(int index) it selects a track for the specified index.

public int getCurrentPosition() returns the current playback position.

public int getDuration() returns duration of the file.

Activity class

Let's write the code of to play the audio file. Here, we are going to play maine.mp3 file located inside the sdcard/Music directory.

File: MainActivity.java

```
1. package com.example.audiomediaplayer1;
2.
3. import android.media.MediaPlayer;
4. import android.net.Uri;
5. import android.os.Bundle;
6. import android.app.Activity;
7. import android.view.Menu;
8. import android.widget.MediaController;
9. import android.widget.VideoView;
10.
11.public class MainActivity extends Activity {
12.
13.    @Override
14.    protected void onCreate(Bundle savedInstanceState) {
15.        super.onCreate(savedInstanceState);
```

```
16.       setContentView(R.layout.activity_main);
17.
18.       MediaPlayer mp=new MediaPlayer();
19.       try{
20.          mp.setDataSource("/sdcard/Music/maine.mp3");//Write your location here

21.          mp.prepare();
22.          mp.start();
23.
24.       }catch(Exception e){e.printStackTrace();}
25.
26.    }
27.
28.    @Override
29.    public boolean onCreateOptionsMenu(Menu menu) {
30.       // Inflate the menu; this adds items to the action bar if it is present.
31.       getMenuInflater().inflate(R.menu.activity_main, menu);
32.       return true;
33.    }
34.
35.}
```

7. What is a spinner control? How can we use spinner control to show 5 integer elements which is stored in strings.xml as an array(write the resource file-strings.xml and xml file for UI)

Spinners in Android are like `select` dropdowns (HTML) of web development. Once tapped, it shows a list of options in a dropdown menu from which one can select a value. It is very simple to add it as a View in the Layout file (just as any other view like `TextView` or `EditView`):

```
1     <Spinner
2             android:layout_width="match_parent"
3             android:layout_height="wrap_content"
4             android:id="@+id/country" />
```

Populating the Spinner with a Set of Choices for the User strings.xml

```
?xml                  version="1.0"                  encoding="utf-8"?>
<resources>
                        <integer-array           name="planets_array">
                                                        <item>1</item>
                                                        <item>2</item>
                                                        <item>3</item>
                                                        <item>4</item>
                                                        <item>5</item>
                                                </    integer    -array>
</resources>
```

The xml file is

```
   <Spinner
      android:id="@+id/spinner"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:prompt="@string/spinner_title"
android:entries="@string/ planets_array "
/>
```

Entries attribute is used for setting an array to spinner control.