Chapter-2 :

Using Activities - Fragments and Intents in Android: Working with activities, Using Intents, Fragments, Using the Intent Object to Invoke Built–in Application

Introduction to Activities in Android:

- An activity is a class that represents a single screen in android. It is like window or frame of Java.
- By the help of activity, you can place all your UI components (button, label, text field etc.) or widgets in a single screen.
- Activity is one of the most important components for any android app.
- It is similar to the main () function in different programming languages.
- It is the main entry point for user interaction.
- Any application, don't matter how small it is (in terms of code and scalability), has at least one Activity class. You can have multiple activities in your app.
- All your activities must be declared in the manifest file, with their attributes.
- Android Activity Lifecycle is controlled by 7 methods of android.app.Activity class. The android Activity is the subclass of ContextThemeWrapper class.

Lifecycle of an android activity:

Every activity has different functions throughout its life, onCreate (), onStart (), onResume (), onPause (), onStop (), onRestart (), onDestroy ().

Figure 3-2 shows the life cycle of an activity and the various stages it goes through—from when the activity is started until it ends.

- **The Activity class defines the following Methods:**

- **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 **or** activity is becoming visible to the user and is ready to start receiving user interactions.
- **OnPause()** — Called when the current activity is being paused and the previous activity is being resumed. This is where you typically pause ongoing processes or save transient data.
- **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.

➢ 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.

Demo Android App to Demonstrate Activity Lifecycle in Android

```
package com.example.activity101;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
public class MainActivity extends AppCompatActivity
{
String tag = "Lifecycle Step";
@Override
protected void onCreate(Bundle savedInstanceState)
{
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_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");
}
}
```

Explanation:

1. package com. example.activity101;

This declares the package name for the Java file (activity101.java).

2. import android.support.v7.app.AppCompatActivity;
   import android.os.Bundle;
   import android.util.Log;

These import statements bring in classes from the Android framework that is necessary for this activity. AppCompatActivity is the base class for activities that use the Support Library action bar features. Bundle is used for passing data between activities. Log is used for logging messages.

3. public class MainActivity extends AppCompatActivity
{

When a class extends another class in Java, it means that the subclass (in this case, MainActivity) inherits all the fields and methods from the super class (AppCompatActivity).

4. String tag = "Lifecycle Step";

This declares a string variable tag and initializes it with the value "Lifecycle Step". This tag will be used in logging messages to identify them.

5. @Override
protected void onCreate(Bundle savedInstanceState)
{
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);
   Log.d(tag, "In the onCreate() event");
}

- This method is called when the activity is first created. It sets the content view to a layout defined in activity_main.xml file. Then it logs a debug message indicating that onCreate() event has occurred.
- When an activity is destroyed and recreated due to configuration changes (such as screen rotation), Android preserves certain data from the activity's previous state when it's recreated. This preserved data is stored in the savedInstanceState bundle.

6.
```
@Override
public void onStart() { /* onStart code */ }

@Override
public void onRestart() { /* onRestart code */ }

@Override
public void onResume() { /* onResume code */ }

@Override
public void onPause() { /* onPause code */ }

@Override
public void onStop() { /* onStop code */ }

@Override
public void onDestroy() { /* onDestroy code */ }
```

These methods are overrides of the lifecycle callback methods provided by the AppCompatActivity class. Each of these methods is called by the Android system at specific points in the activity's lifecycle.

Logging Messages:

In each of the overridden methods, there's a call to Log.d() to log a debug message indicating the current lifecycle event.

Intents or Linking Activities Using Intents:

- An Android application can contain zero or more activities. When your application has more than one activity, you often need to navigate from one to another.
- In Android, you navigate between activities through what is known as intent.
- Intent is a messaging object used to request any action from another app component. Intent's most common use is to launch a new activity from the current activity.
- Intent facilitates communication between different components. Intent object is used to call other activities.
- The intent is used to launch an activity, start the services, broadcast receivers, display a web page, dial a phone call, send messages from one activity to another activity, and so on.

Common Use cases for Intents include:

1. **Starting Activities:** Use intents to launch new activities within your application or to launch activities from other applications.

For example, imagine you have a button in your app that says "Open Camera". When a user taps that button, you can use intent to ask the Android system to open the camera app.

2. **Broadcasting Messages:** Use intents to send broadcast messages within your application or to other applications, allowing them to receive and respond to events.
   Ex: let's say you have a music player app and you want to notify other apps whenever a new song starts playing.

- You would use a broadcast intent to send a message saying "Hey, a new song is playing!
- Other apps, like maybe a notification app or a social media app, can listen for this message. When they receive it, they can do things like show a notification saying what song is playing or share the song information on social media.
- Sent when the device finishes booting up, allowing apps to start up services or perform initialization tasks.

3. **Invoking Services:** Use intents to start services that perform background tasks or handle long-running operations.
    - Services: Background workers in your app that perform tasks without needing to be in the foreground.
    - Intents: Messages used to start services.

    Example
    - Create a service to download a file.
    - Register the service in the manifest.
    - Start the service using intent from an activity.

    Using services and intents allows your app to perform tasks like downloading files or playing music in the background, ensuring the main thread remains responsive and providing a smoother user experience.

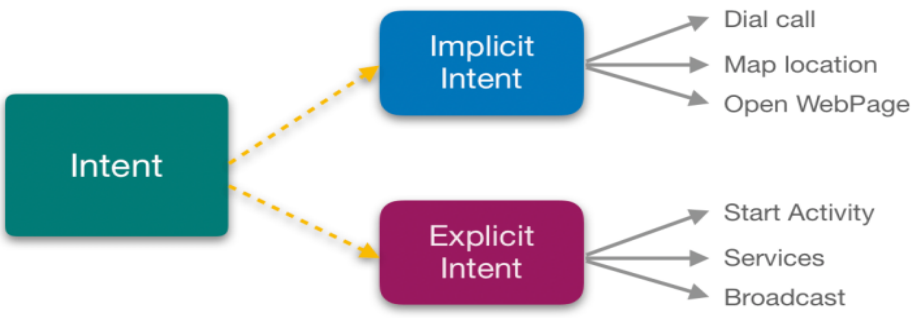
4. **Passing Data:** Intents can carry data (extra information) as key-value pairs, allowing components to exchange information, such as passing data between activities or between different parts of your application.
    - For example, let's say you have an app with two screens, and you want to send a message from the first screen to the second screen. You would use intent to carry that message.
    - **First Screen (Calling Screen):** You create intent and put the information about the call, like the caller's name, as extras.
    - **Second Screen (Receiving Screen):** You get the information about the call from the intent.

Methods and their Description:

Methods	Description
Context.startActivity()	This is to launch a new activity or get an existing activity to be action.
Context.startService()	This is to start a new service or deliver instructions for an existing service.
Context.sendBroadcast()	This is to deliver the message to broadcast receivers.

TYPES OF INTENT

- Intents are of two types:



- Explicit Intent is used to invoke a specific target component. It is used to switch from one activity to another activity in the same application. It is also used to pass data by invoking the external class.

- Explicit Intents specify the target component by providing the exact class name of the component to be invoked.

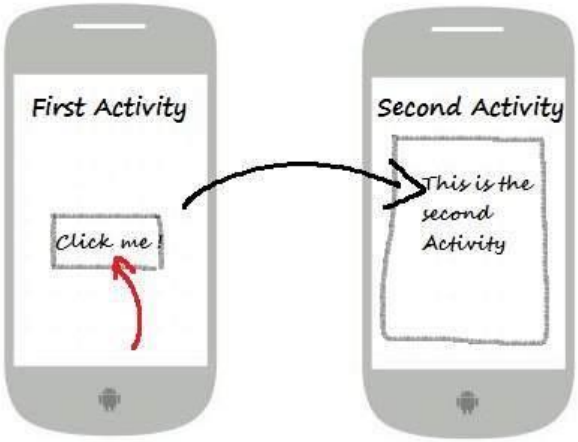
Example:-

1. We can use explicit intent to start a new activity when the user invokes an action or plays music in the background or on click button go to another activity.

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

   - MainActivity.this specifies the current context, which is the MainActivity.
   - SecondActivity.class specifies the target activity to which you want to navigate.
   - In Explicit we use the name of component which will be affected by Intent. **For Example:** If we know class name then we can navigate the app from One Activity to another activity using Intent.

2. In amazon app if you go to Home page you can see prime, fresh, mobiles, electronics etc. If you click prime it transit to prime page activity likewise other tab also works.



IMPLICIT INTENT:

- An implicit intent is used when you want to perform an action, but you don't care which component performs it.

- The system will determine the appropriate component to handle the intent based on the available components that can respond to it.
- Example: Suppose you want to open a website URL in a web browser. You don't know which browser the user prefers, so you'll use an implicit intent:

```
//Intent object and open the webpage
Intent intent = new
Intent(Intent.ACTION_VIEW,Uri.parse("https://example.com"));
startActivity(intent); //call a webpage
```

- Intent.ACTION_VIEW is the action you want to perform, which is viewing content.
- Uri.parse("https://example.com") specifies the data you want to view, which is a website URL.
  - Explicit intents are used for navigating within your own by specifying the target component, while implicit intents are used for performing actions where the system determines the appropriate component to handle the request.

Using the Intent Object to Invoke Built–in Application:

- One of the key aspects of Android programming is using the intent to call activities from other Applications. An Application can call many built-in Applications, which are included with an Android device.
- Suppose, you want to load a Web page, which is not part of your Application. You can use the Intent object to invoke the built-in Web Browser to display the Web page, instead of building your own Web Browser for this purpose.
- Add three buttons are in activity_main.xml file as Web Browser. Call here and display Map.

Code:

```
<Button
android:id="@+id/buttonwebbrowser"
android:text="Web Browser"
```

```
android:onClick="onClickBrowseWeb"/>
<Button
android:id="@+id/buttondocall"
android:text="Call Here"
android:onClick="onClickCallHere"/>
<Button
android:id="@+id/buttondisplaymap"
android:text="Display Map"
android:onClick="onClickDisplayMap"/>
```

Add three methods are in MainActivity.java class to display the Website in the Web Browser. Call on any mobile number and display the map by Google maps.

Code:

```
public void onClickBrowseWeb(View view) {
    Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.google.co.in"));
    startActivity(intent);
  }
  public void onClickCallHere(View view) {
    Intent intent = new Intent(Intent.ACTION_DIAL, Uri.parse("tel:+91xxxxxxxxxx"));
    startActivity(intent);
  }

                    Or
setContentView(R.layout.activity_main);
String[] phoneNumbers = {
      "+91xxxxxxxxxx", // India
      "+1xxxxxxxxxx",  // USA/Canada
      "+44xxxxxxxxxx", // UK
      "+61xxxxxxxxxx", // Australia
      "+81xxxxxxxxxx"  // Japan
    };

private void startCallIntent(String phoneNumber) {
    Intent intent = new Intent(Intent.ACTION_CALL);
    intent.setData(Uri.parse("tel:" + phoneNumber));
```

```
    startActivity(intent);
}
```

To display the Map based upon lat,long Co ordinates

```
public void onClickDisplayMap(View view) {
    Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("geo:28.7041,77.1025"));
    startActivity(intent);
}
```

Or

To display the Map based upon Location name

```
public void onClickDisplayMap(View view) {
    String location = editTextLocation.getText().toString();
    if (!location.isEmpty()) {
        Uri geoLocation = Uri.parse("geo:0,0?q=" + Uri.encode(location));
        showMap(geoLocation);
    }
}
```

In the first button, create an object of Intent and then pass the two arguments to its constructor. The action and the data will be,

```
1. Intent intent=new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.google.co.in"));


2. startActivity(intent);
```

The action here is represented by the Intent.ACION_VIEW constant. We used the parse() method of the URL class to convert a URL string into a URL object.

For the second button, we can dial a specific number by passing in the telephone number in the portion.

```
1.  Intent intent=new Intent(Intent.ACTION_DIAL, URL.parse("tel:+91xxxxxxxxxx"));
```

```
2. startActivity(intent);
```

In this case, the dialler will display the number to be called. The user must still click the dial button to dial the number. If you want to directly call the number without the user intervention, change the action, given below,

```
1. Intent intent=new Intent(Intent.ACTION_CALL, URL.parse("tel:+91xxxxxxxxxx"));
2. startActivity(intent);
```

For this, you need to assign the Android.permission.CALL_PHONE permission to your Application, defined in the manifest file.

You can simply omit the data portion to display the dialer without specifying any number.

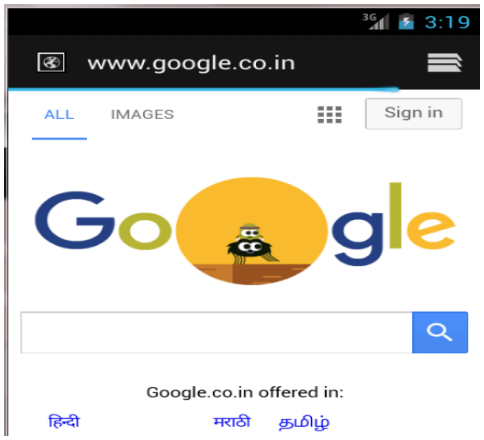
```
1. Intent intent=new Intent(Intent.ACTION_DIAL);
2. startActivity(intent);
```

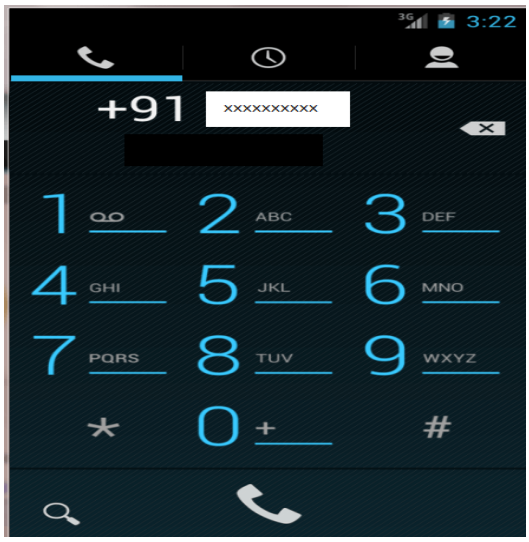
The third button displays a map using the ACTION_VIEW constant. Here we use “geo” in place of “http”.

```
1. Intent intent=new Intent(Intent.ACTION_VIEW,Uri.parse("geo:28.7041,77.1025"));
2. startActivity(intent);
```

In this app, we have used the Intent class to invoke some of the built-in Applications in Android like Browser, Phone, and Maps).

- Now, click on the first option. Google page will open in the Browser.
- Click the second option. It will open a dial-up option to make a call to the specified mobile number.
- To load the Maps Application, click the display map button. To display the Maps Application, you need to run the Application on an AVD, which supports the Google APIs or you have to run this app on your device.





Intent filter

- Implicit intent uses the intent filter to serve the user request.
- The intent filter specifies the types of intents that an activity, service, or broadcast receiver can respond to.
- Intent filters are declared in the Android manifest file.
- Intent filter must contain <action>

1. **<actions>** In this, you can keep all the actions you wish your intent to accept.

Constant	Target Component	Action
ACTION_CALL	Activity	Initiate a phone call
ACTION_EDIT	Activity	Display data for the user to edit
ACTION_BATTERY_LOW	broadcast receiver	A warning that the battery is low
ACTION_HEADSET_PLUG	broadcast receiver	A headset has been plugged into the device, or unplugged from it.

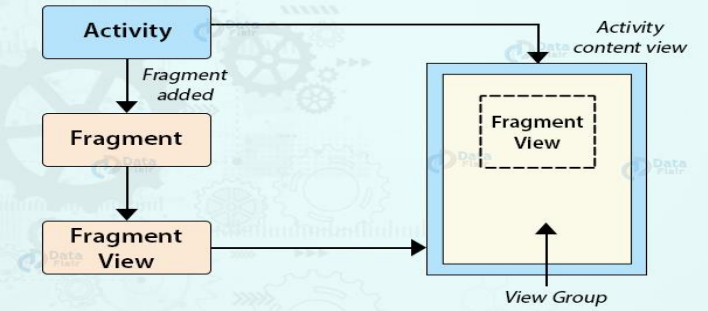
2. **<data>** It defines the type of data that the intent will take.

- Example: I f the action field is ACTION_EDIT, the data field would contain the URI of the document to be displayed for editing.

3. **<Category>** It represents the name of the category that the intent will accept.A string containing additional information about the kind of component that should handle the intent.

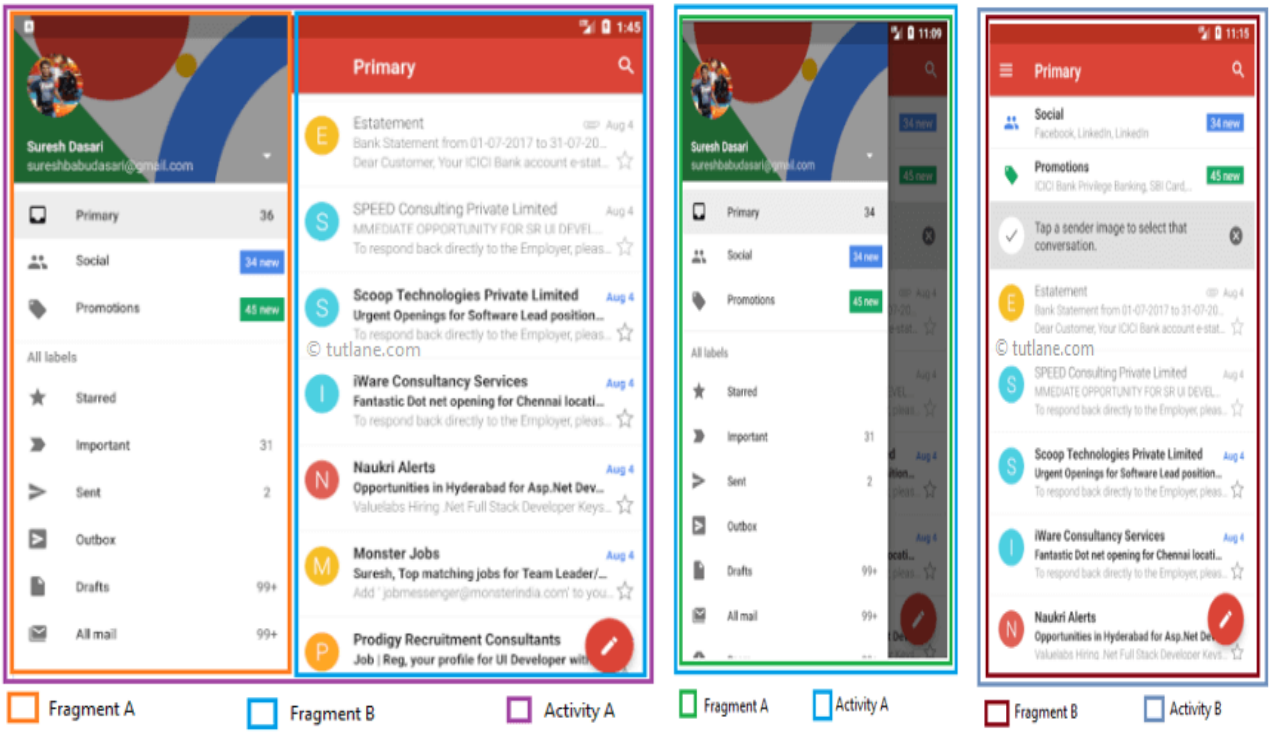
Android Fragments:

- Android Fragment is a Graphical User Interface component of Android. It resides within the Activities of an Android application. It represents a portion of UI that the user sees on the screen.
- Android Fragments cannot exist outside an activity. Another name for Fragment can be **Sub-Activity** as they are part of Activities.
- Fragments are always embedded in Activities; Multiple Fragments can be added to single activity.

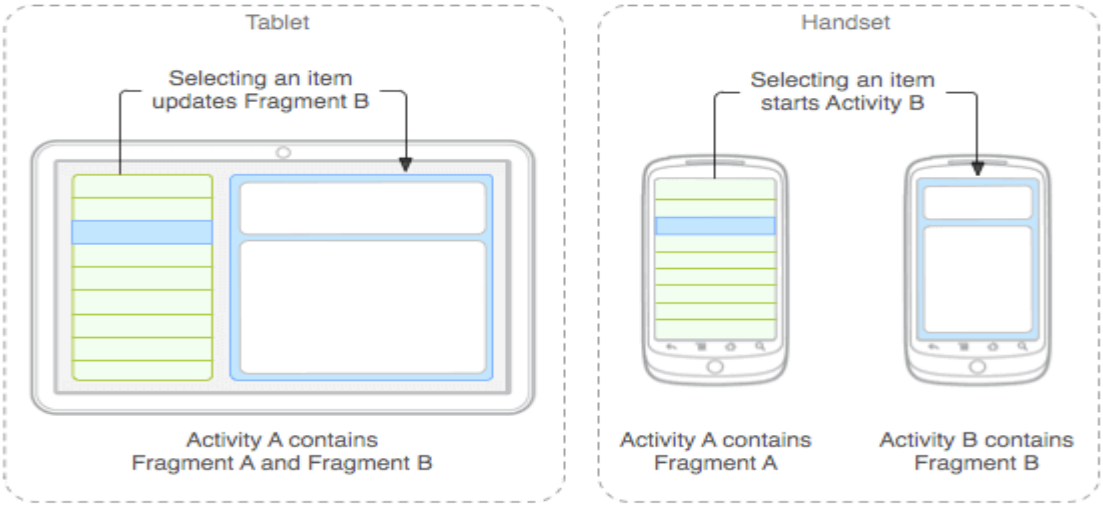


How Fragment Interacts with Activity in Different Devices:

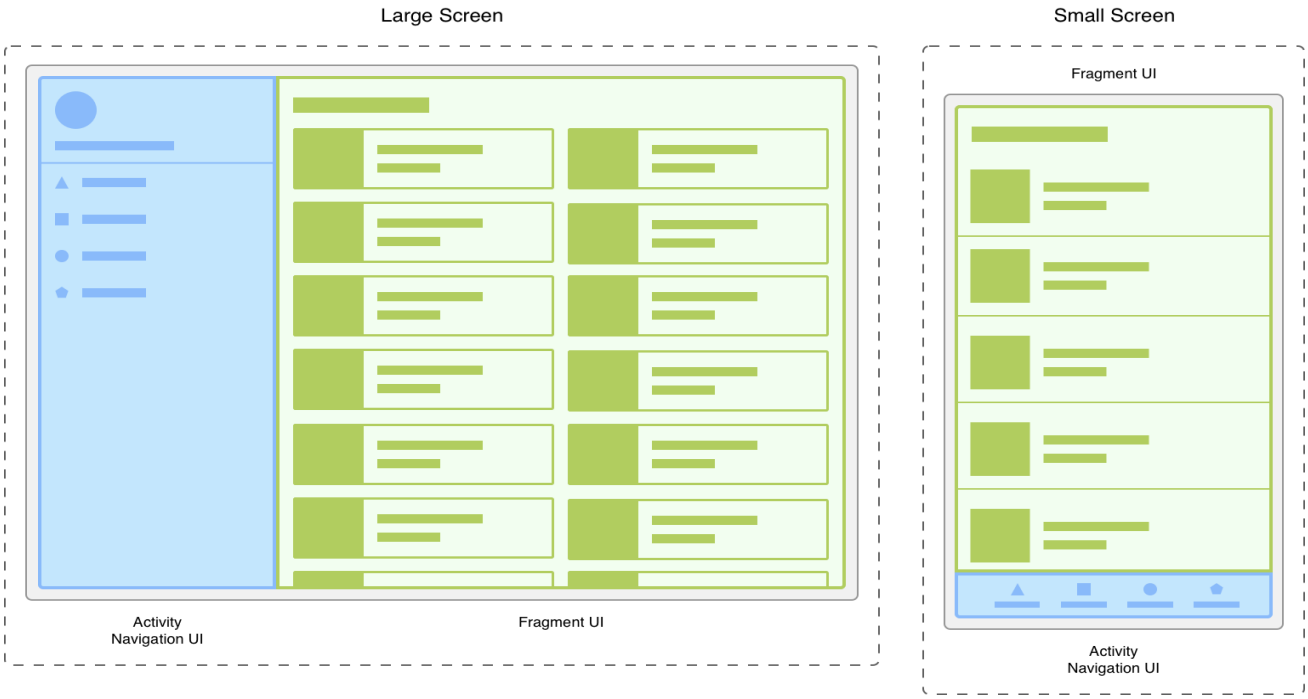
- If you observe above example for **Tablet** we defined an **Activity A** with two fragments such as one is to show the list of items and second one is to show the details of item which we selected in first fragment.
- For **Handset** device, there is no enough space to show both the fragments in single activity, so the **Activity A** includes first fragment to show the list of items and the **Activity B** which includes another fragment to display the details of an item which is selected in **Activity A**.
- For example, **GMAIL app** is designed with multiple fragments, so the design of GMAIL app will be varied based on the size of device such as tablet or mobile device.



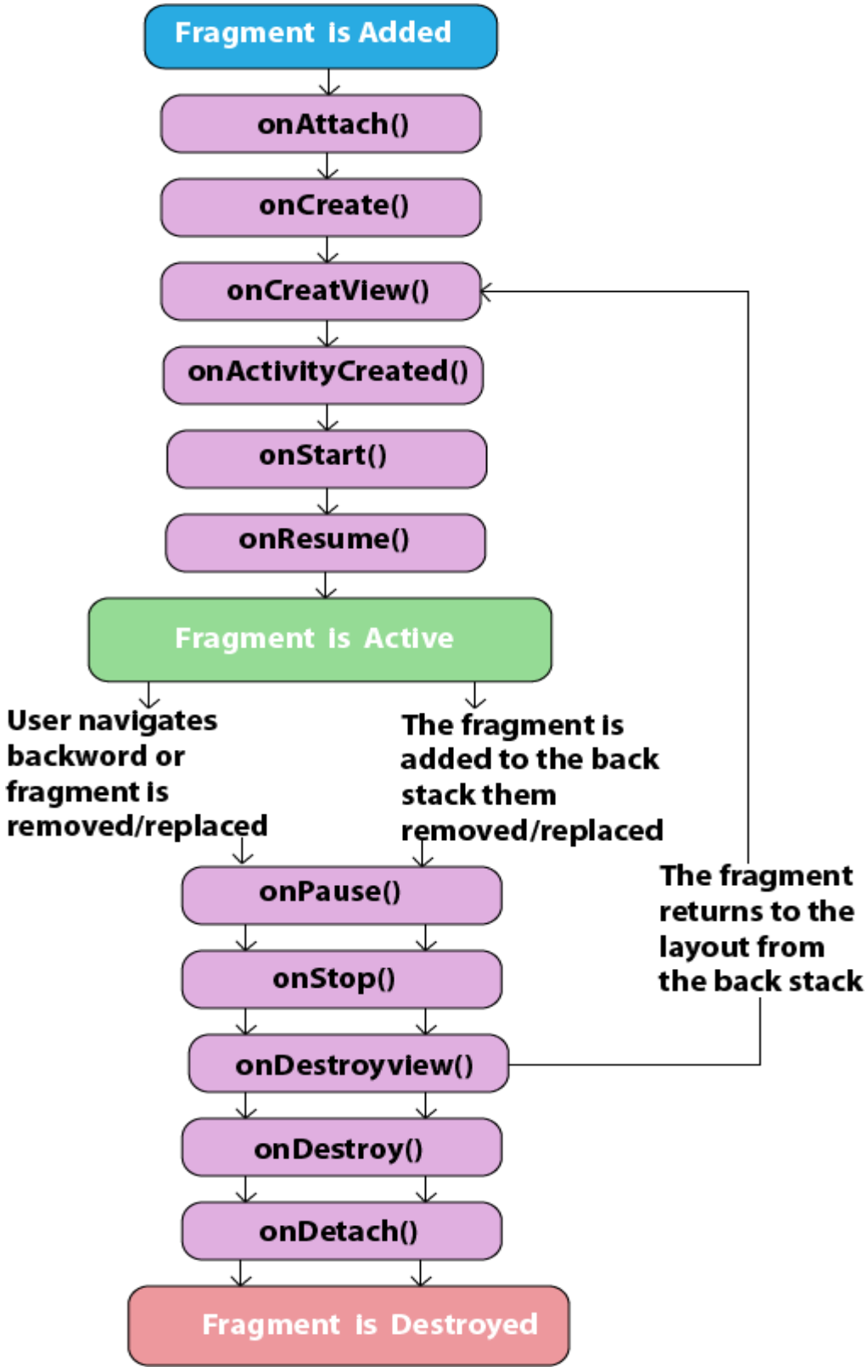
Tablet View **Mobile View**



- **In the case of mobiles, there are two activities that are:**
  - Activity 1 with Fragment A and Activity 2 with Fragment B. When we select an item from Fragment A, it gets open in the Fragment B of Activity 2.
- **In tablets, there is only one activity that is Activity 1.**
  - In **Activity 1**, there are two fragments, **Fragment A** and **Fragment B**. When we select an item from Fragment A, it gets open in Fragment B of the same activity.
- On larger screens, you might want the app to display a static navigation drawer and a list in a grid layout. On smaller screens, you might want the app to display a bottom navigation bar and a list in a linear layout.



Android Fragment Lifecycle:



Method	Description
onAttach()	It is called when the fragment has been associated with an activity.
onCreate()	It is used to initialize the fragment.
onCreteView()	It is used to create a view hierarchy associated with the fragment.
onActivityCreated()	It is called when the fragment activity has been created and the fragment view hierarchy instantiated.
onStart()	It is used to make the fragment visible.
onResume()	It is used to make the fragment visible in an activity.
onPause()	It is called when fragment is no longer visible and it indicates that the user is leaving the fragment.
onStop()	It is called to stop the fragment using the onStop() method.
onDestoryView()	The view hierarchy associated with the fragment is being removed after executing this method.
onDestroy()	It is called to perform a final clean up of the fragments state.
onDetach()	It is called immediately after the fragment disassociated from the activity.

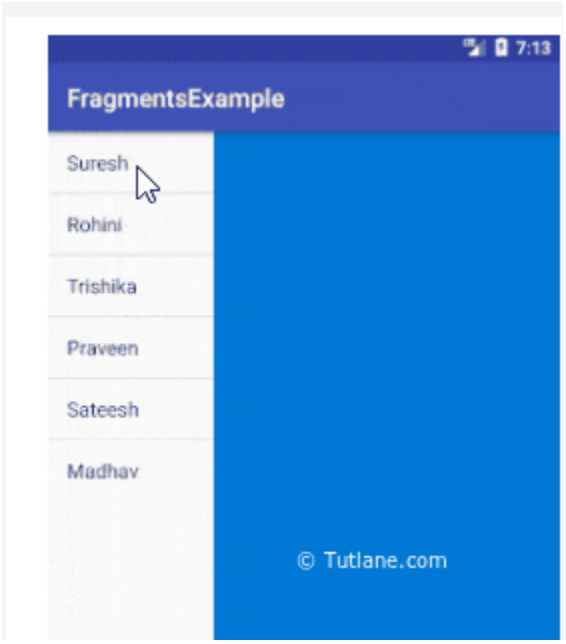
ListMenuFragment.java

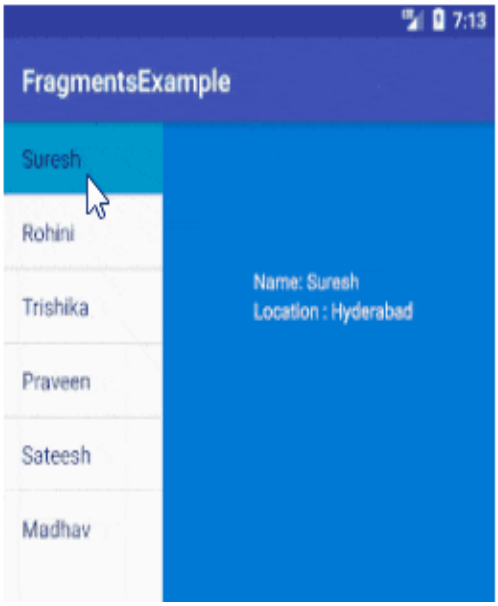
```
public class ListMenuFragment extends ListFragment {
    String[] users = new String[]
{ "Suresh","Rohini","Trishika","Praveen","Sateesh","Madhav" };

String[] location = new String[]{"Hyderabad","Guntur","Hyderabad","Bangalore","Vizag","Nag
pur"};
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        View view =inflater.inflate(R.layout.listitems_info, container, false);
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(getActivity(),
                android.R.layout.simple_list_item_1, users);
```

activity_main.xml:

```
<fragment
        android:layout_height="match_parent"
        android:layout_width="350px"
        class="com.tutlane.fragmentsexample.ListMenuFragment"
        android:id="@+id/fragment"/>
    <fragment
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        class="com.tutlane.fragmentsexample.DetailsFragment"
        android:id="@+id/fragment2"/>
```





Adding Fragments with Activities:

Embedding Fragments with Activities means adding the Fragments to the respective Activity Layout. Now, there are two ways for adding multiple Fragments in one Activity:

1. Statically

To add the fragment statically, we need to mention it ourselves in the these fragments can't be replaced during the execution as they are static. Defined in XML, fixed during compile-time, and cannot be changed at runtime.

2. Dynamically

In this, we embed our Fragment in Activities dynamically using **Fragment Manager**.

Unlike Static Fragment, in this, we can add, remove or replace the Fragments at the runtime itself. Managed in code, flexible, can be added, removed, or replaced at runtime.

Types of Android Fragments

1. Single Fragments

Single fragments show only a single view for the user on the screen. These are for handheld devices such as mobile phones.

2. List Fragments

List fragments are those that have a special list view feature. In this, there's a list and the user can choose to see a Sub-Activity.

3. Fragment Transactions:

Fragment transactions are for the transition from one fragment to another. It supports switching between two fragments.