Chapter 3

Android Activities, Fragments, Intents and Services

Activity

- represents a single screen with a user interface (UI) and it will act as an entry point for users to interact with an app
- Android apps can contain multiple screens and each screen of our application is an extension of Activity class **(MainActivity class)**
- Example: Contacts App
  - List of contacts
  - Add new contacts
  - Search contacts
- in android there is a minimal dependency between the activities in an app.
- To uses the activities in application
  - we need to register those activities information in our app's manifest file (**AndroidMainfest.xml**) and
  - need to manage activity life cycle properly

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

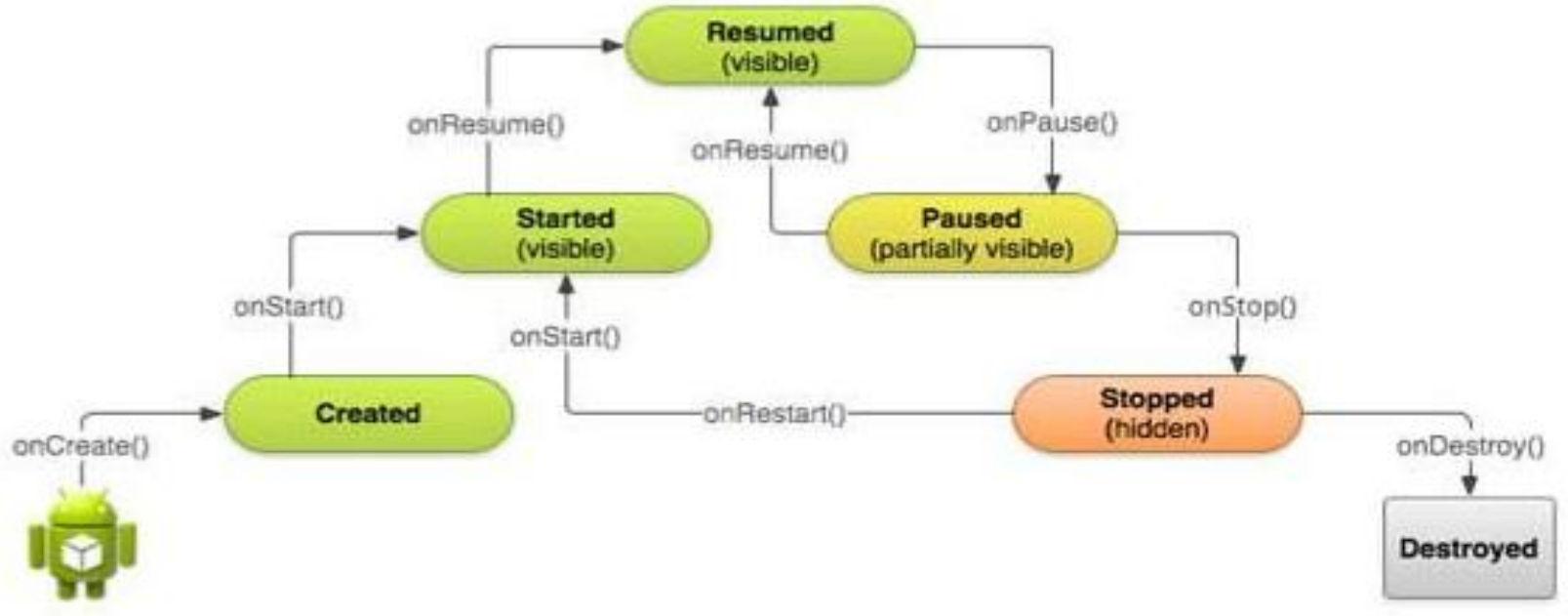
- activity attribute **android:name** will represent the name of class and we can also add multiple attributes like **icon**, **label**, **theme**, **permissions**, etc. to an activity element

- activities can be implemented as a subclass of **Activity** class

```
import android.app.Activity;
import android.os.Bundle;
public class MainActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Life Cycles of an Activity

- **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 onCreate()
- **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



- Use the **onCreate()** method to **create and instantiate the objects** that you will be using in your application.
- Use the **onResume()** method to **start any services or code that needs to run** while your activity is in the **foreground**.
- Use the **onPause()** method **to stop any services or code that does not need to run** when your activity is **not in the foreground**.
- Use the **onDestroy()** method to **free up resources before your activity is destroyed**.

Fragments

- In a small-screen device (such as a smartphone), an **activity typically fills the entire screen** but in a large-screen device, such as on a tablet, it is **somewhat out of place**
- all the views in an activity must be arranged to **make full use of the increased space,** a better approach is to use "**mini-activities**", each containing its own set of views.
- In Android 3.0 and later, these **mini-activities** are known as **fragments.**
- Fragments are always embedded in an **activity**

Fragments

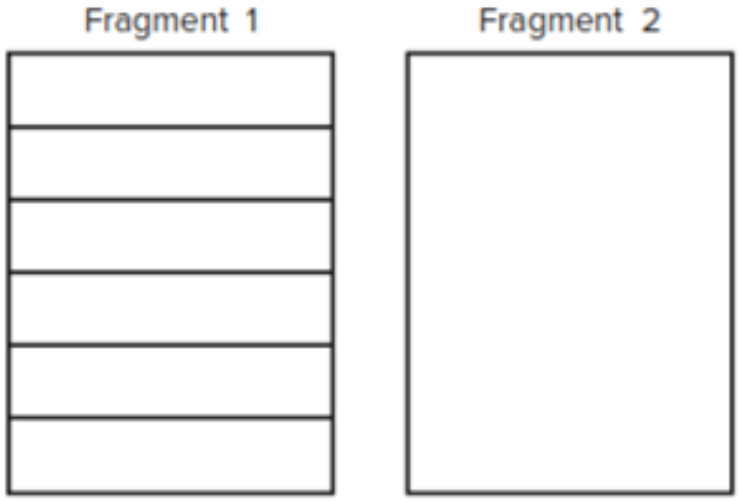


Figure 3.2.1 Fragments in an activity

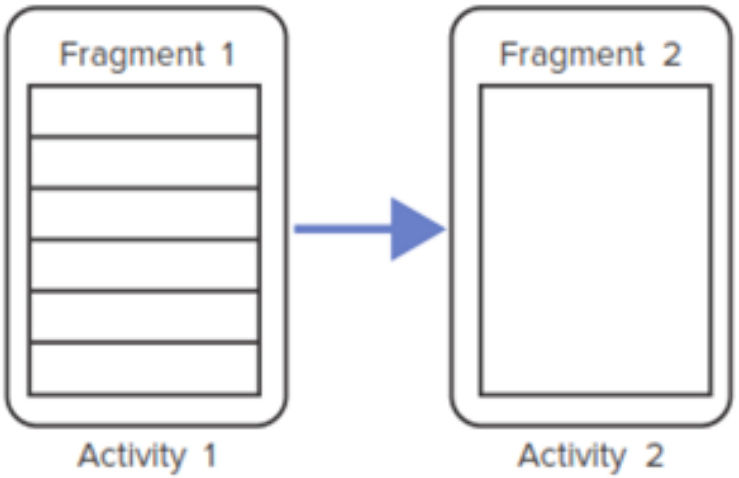


Figure 3.2.2 Fragments in different activities

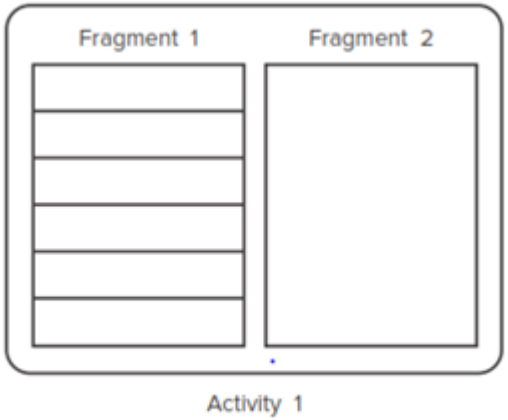


Figure 3.2.3 Fragments in landscape mode

Life Cycle of Fragments

Method	Description
onAttach()	It is called when the fragment has been associated with an activity.
onCreate()	It is used to initialize the fragment.
onCreateView()	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 onStop() method.
onDestoryView()	The view hierarchy which 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.
nDetach()	It is called immediately after the fragment disassociated from the activity.

Intents

- Android uses Intent for communicating between the components (such as activities, services, broadcast receivers and content providers) of an Application and also from one application to another application.
- It helps you to redirect your activity to another activity on occurrence of any event
- For example **startActivity()** you can perform this task.

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

  - foreground activity is getting redirected to another activity i.e. SecondActivity.java.
  - getApplicationContext() returns the context for your foreground activity

Types of Intents

- **Explicit Intent:**
  - Explicit Intents are used to connect the application internally.
  - In Explicit we use the name of component which will be affected by Intent
  - Explicit Intent works internally within an application to perform navigation and data transfer. Example:

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

  - Here **SecondActivity** is the JAVA class name where the activity will now be navigated.

Implicit Intent:

- In Implicit Intents we do need to specify the name of the component.
- We just specify the Action which has to be performed and further this action is handled by the component of another application. The basic example of implicit Intent is to open any web page.

```
Intent intentObj = new Intent(Intent.ACTION_VIEW);
intentObj.setData(Uri.parse("https://www.google.com"));
startActivity(intentObj);
```

- Unlike Explicit Intent you do not use any class name to pass through Intent().

Benefits of Intents

- **For activity**: Intent object helps to start a new activity and passing data to the second activity
- **For Services**: Services work in background, Intents could be used to start a Service
- **For Broadcast Receivers:** Android system initiates some broadcast message on several events, such as System Reboot, Low Battery warning message etc.
- **For Android Applications:** Whenever you need to navigate to another activity of your app or you need to send some information to next activity then we can always prefer to Intents for doing so.

Services

- **Android service** is a component that is *used to perform long running operations on the background* such as playing music, handle network transactions, interacting with content providers etc.
- It doesn't have any UI (user interface)
- service can be bounded by a component to perform interactivity and inter process communication (IPC).

Life Cycle of Android Service

1) **Started Service (foreground or background)**

- A service is started when component (like activity) calls **startService()** method, now it runs in the background indefinitely.
- It runs in the background even if the application that started the service is closed.
- It is stopped by **stopService()** method.
- The service can stop itself by calling the **stopSelf()** method.

2) **Bound Service**

- A service is bound when another component (e.g. client) calls **bindService()** method.
- The client can unbind the service by calling the **unbindService()** method. The service cannot be stopped until all clients unbind the service.