Andhra Pradesh State Skill Development Corporation

Skill AP
APSSDC

ANDROID APPLICATION DEVELOPMENT

ACTIVITY LIFECYCLES

Activity Lifecycle

Introduction

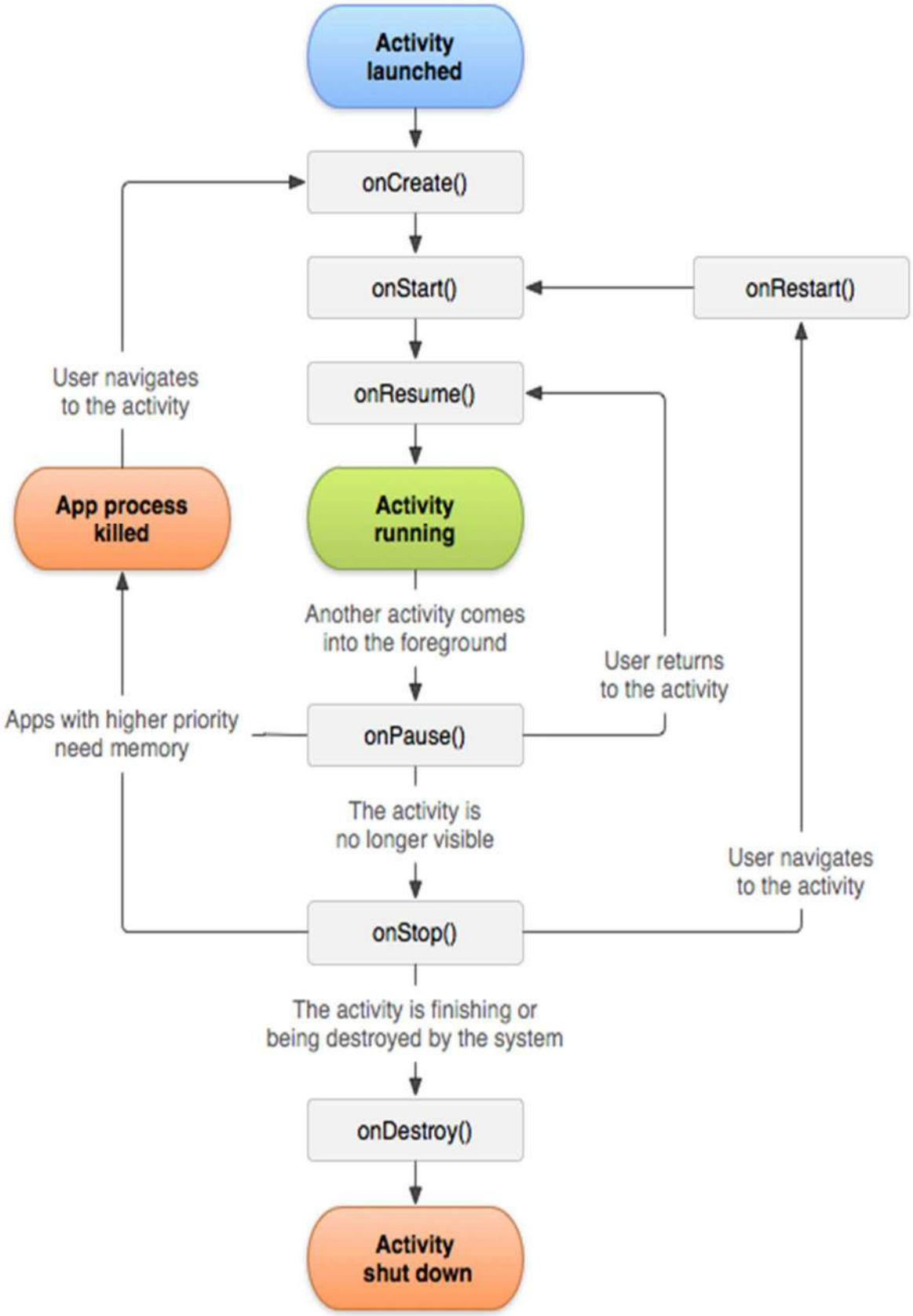
- In this you learn about the activity lifecycle, the callback events you can implement to perform tasks in each stage of the lifecycle, and how to handle Activity instance states throughout the activity lifecycle..,

About the activity lifecycle: The activity lifecycle is the set of states an activity can be in during its entire lifetime, from the time it's created to when it's destroyed and the system reclaims its resources. As the user interacts with your app and other apps on the device, activities move into different states.

For example:

- When you start an app, the app's main activity ("Activity 1" in the figure below) is started, comes to the foreground, and receives the user focus.
- When you start a second activity ("Activity 2" in the figure below), a new activity is created and started, and the main activity is stopped.
- When you're done with the Activity 2 and navigate back, Activity 1 resumes. Activity 2 stops and is no longer needed.
- If the user doesn't resume Activity 2, the system eventually destroys it. Activity lifecycle

Activity states and lifecycle callback methods * When an Activity transitions into and out of the different lifecycle states as it runs, the Android system calls several lifecycle callback methods at each stage. All of the callback methods are hooks that you can override in each of your Activity classes to define how that Activity behaves when the user leaves and re-enters the Activity. Keep in mind that the lifecycle states (and callbacks) are per Activity, not per app, and you may implement different behavior at different points in the lifecycle of each Activity.



This figure shows each of the Activity states and the callback methods that occur as the Activity transitions between different states:

onCreate():

- Called when the activity is first created. This is where you should do all of your normal static set up: create views, bind data to lists, etc. This method also provides you with a Bundle containing the activity's previously frozen state, if there was one. Always followed by onStart().

onRestart():

- Called after your activity has been stopped, prior to it being started again. Always followed by onStart()

onStart():

- Called when the activity is becoming visible to the user. Followed by onResume() if the activity comes to the foreground.

onResume():

- Called when the activity will start interacting with the user. At this point your activity is at the top of the activity stack, with user input going to it. Always followed by onPause().

onPause ():

- Called as part of the activity lifecycle when an activity is going into the background, but has not (yet) been killed. The counterpart to onResume(). When activity B is launched in front of activity A, this callback will be invoked on A. B will not be created until A's onPause() returns, so be sure to not do anything lengthy here.

onStop():

- Called when you are no longer visible to the user. You will next receive either onRestart(), onDestroy(), or nothing, depending on later user activity. Note that this method may never be called, in low memory situations where the system does not have enough memory to keep your activity's process running after its onPause() method is called.

onDestroy():

The final call you receive before your activity is destroyed. This can happen either because the activity is finishing (someone called finish() on it, or because the system is temporarily destroying this instance of the activity to save space. You can distinguish between> these two scenarios with the isFinishing() method.

- When the Activity first time loads the events are called as below:

onCreate() onStart() onResume()

- When you click on Phone button the Activity goes to the background and the below events are called:

onPause() onStop()

- Exit the phone dialer and the below events will be called:

onRestart() onStart() onResume()

- When you click the back button OR try to finish() the activity the events are called as below:

onPause() onStop() onDestroy()

Activity States

- The Android OS uses a priority queue to assist in managing activities running on the device. Based on the state a particular Android activity is in, it will be assigned a certain priority within the OS. This priority system helps Android identify activities that are no longer in use, allowing the OS to reclaim memory and resources. The following diagram illustrates the states an activity can go through, during its lifetime:
- These states can be broken into three main groups as follows:
- Active or Running - Activities are considered active or running if they are in the foreground, also known as the top of the activity stack. This is considered the highest priority activity in the Android Activity stack, and as such will only be killed by the OS in extreme situations, such as if the activity tries to use more memory than is available on the device as this could cause the UI to become unresponsive.
- Paused - When the device goes to sleep, or an activity is still visible but partially hidden by a new, non-full-sized or transparent activity, the activity is considered paused. Paused activities are still alive, that is, they maintain all state and member information, and remain attached to the window manager. This is considered to be the second highest priority activity in the Android Activity stack and, as such, will only be killed by the OS if killing this activity will satisfy the resource requirements needed to keep the Active/Running Activity stable and responsive.
- Stopped - Activities that are completely obscured by another activity are considered stopped or in the background. Stopped activities still try to retain their state and member information for as long as possible, but stopped activities are considered to be the lowest priority of the three states and, as such, the OS will kill activities in this state first to satisfy the resource requirements of higher priority activities.

Sample activity to understand the life cycle

Implenting Activity Life Cycles

Step 1:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    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:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
```

```
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="30sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.449"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.483" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Step :2

```
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.util.Log;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
   TextView tv;

   //First or Default Lifecycle Activity
   @Override
   protected void onCreate(Bundle savedInstanceState)
   {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      tv=findViewById(R.id.textView);
      Log.i("MainAcivity","onCreate() invoked");
      tv.append("onCreate() \n");
      Toast.makeText(this, "App is Created", Toast.LENGTH_SHORT).show();
   }

   @Override
   protected void onStart() {
      super.onStart();
      Log.i("MainAcivity","onStart() invoked");
      tv.append("onStart() \n");
      Toast.makeText(this, "App is Started", Toast.LENGTH_SHORT).show();
   }

   @Override
   protected void onResume() {
      super.onResume();
      Log.i("MainAcivity","onResume() invoked");
      tv.append("onResume() \n");
      Toast.makeText(this, "App is Resumed", Toast.LENGTH_SHORT).show();
```

```
    }

    @Override
    protected void onPause() {
        super.onPause();
        Log.i("MainAcivity","onPause() invoked");
        tv.append("onPause() \n");
        Toast.makeText(this, "App is Paused", Toast.LENGTH_SHORT).show();
    }

    @Override
    protected void onStop() {
        super.onStop();
        Log.i("MainAcivity","onStop() invoked");
        tv.append("onStop()\n");
        Toast.makeText(this, "Now App is Stopped", Toast.LENGTH_SHORT).show();
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        Log.i("MainAcivity","onDestroy() invoked");
        tv.append("onDestroy()");
        Toast.makeText(this, "Now App is Destroyed", Toast.LENGTH_SHORT).show();
    }
}
```