ALGONQUIN COLLEGE

CST2335 GRAPHICAL INTERFACE PROGRAMMING

Week 4

Application Lifecycle

Introduction

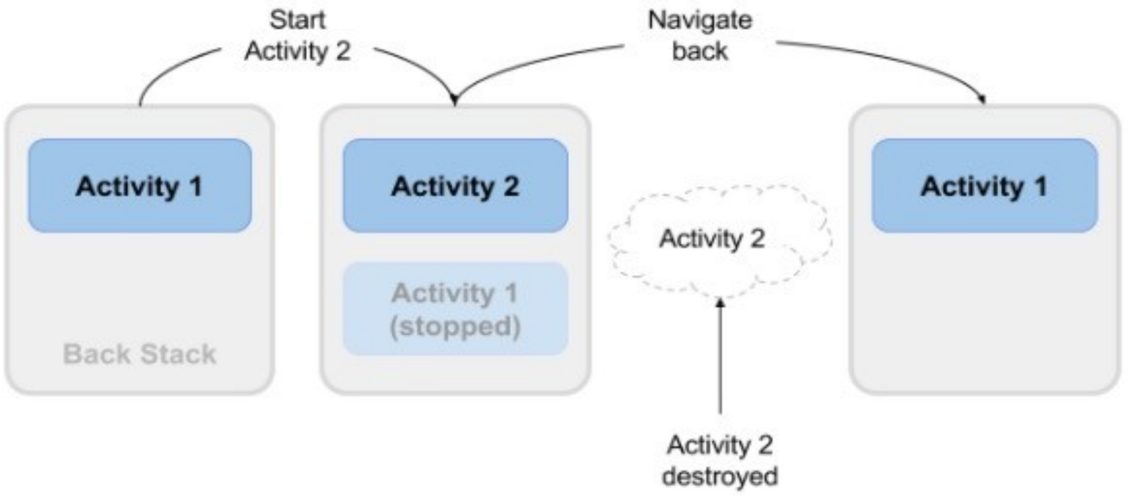
- This week, you will learn how to start an Activity in code.
- We will also learn how to pass information to other Activities at the start, and end of an Activity

Android Activity Lifecycle

- An Android application consists of one or more activities.
- These activities are visual representations of an application in transitioning flow while performing the task, taking user inputs, and showing results to the user.

- Android keeps all the activities in a back stack following the **last in, first out** rule.
- Whenever a new activity is started, the current activity is pushed in the back stack. Thus, Android gives focus to the new activity. The activity can take up the whole screen of the device, or it can also take part of the screen, or it can be dragged.
- When any existing activity is stopped, it is pushed into the back stack, which in turn results in the next top activity being focused.

Android Activity Lifecycle



Android Activity Lifecycle

- When a new activity is created, it is pushed in the stack, and when any existing activity is destroyed, it is pulled out of the stack.
- This process of being pushed in the stack and pulled out of the stack is managed by the activity lifecycle in Android.
- This lifecycle is called Activity lifecycle.
- The lifecycle manages the activities in the stack and notifies about the changes in the state in the activities through the callback methods of the cycle.

Android Activity Lifecycle

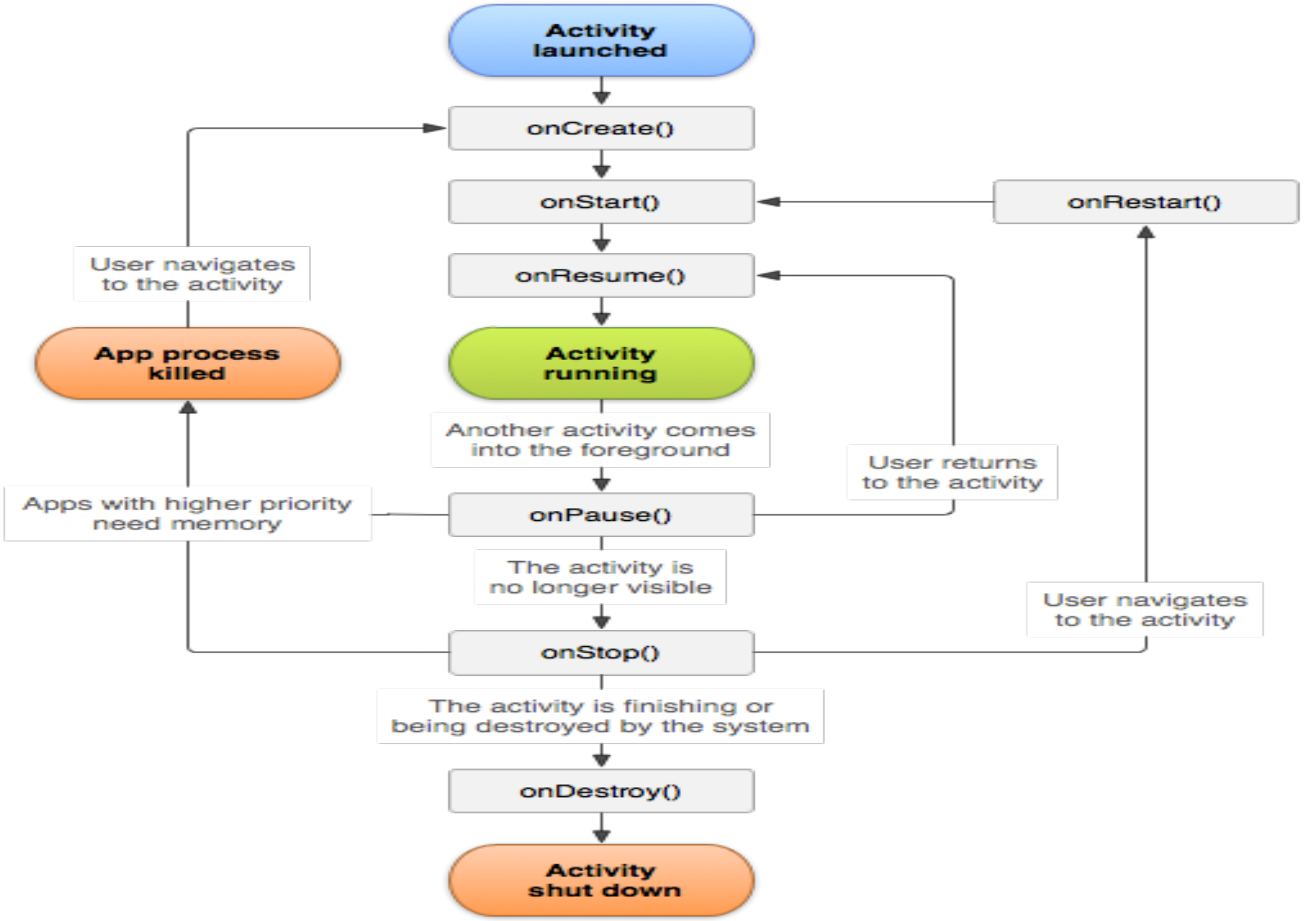
- An activity receives different types of states such as activity created, activity destroyed, and so on, due to a change in the state.
- A developer overrides these callback methods to perform the necessary steps for the respective change of state.
- For example:
    - When an activity is started, the necessary resources should be loaded.
    - When an activity is destroyed, those resources should be unloaded for better performance of the app.
- All these callback methods play a crucial role in managing the Activity lifecycle.
- It is the developer's choice to override none, some, or all methods.

Activity States

- An activity remains in three states: **Resumed, Paused, and Stopped**.
- When an activity is **resumed**, it is shown on the screen and gets the focus of the user.
- When another activity is **started** and becomes visible on the screen, then this activity is paused.
- This activity remains on the foreground task, and it is still alive.
- The activity comes in the **stopped** state when it becomes completely invisible from the screen and is replaced by another activity in the foreground.
- In this **stopped** state, it is in the background section of the back stack. But the activity is still alive.

Activities

- After an Activity is initialized with onCreate(), the system calls the onStart() method, and the Activity is in the started state.
- The onStart() method is also called if a stopped Activity returns to the foreground.
- While onCreate() is called only once when the Activity is created, the onStart() method may be called many times during the lifecycle of the Activity as the user navigates around the application.


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Activity Lifecycle

- There are 3 functions that are called by Android when starting an Activity:
    - onCreate(Bundle savedInstanceState)
    - onStart( ) – the app is on screen
    - onResume( ) – the app responds to touch
- For stopping an activity:
    - onPause() – another Activity is being launched
    - onStop() – an activity is no longer visible.
    - onDestroy() – an activity is being destroyed by the system.

Implementing inherited functions

- To find all the methods that that can be overridden for a class, select the class name, then type Control+O and you will see a list of all the inherited functions.

- For example, if you select MainActivity class and hit Control+O, you could add these functions to your code:

  onStart, onResume, onPause, onStop, onDestroy.

Activities

- Create an app with two Activities ”FirstActivity” and ”SecondActivity” listed.
- Look at the AndroidManifest.xml. There are two <activity > tags, with the parameter android:name=”...”. This tells Android which Java classes to load.
- Which is the first one to load when you start?

Activities

The one with:

```
<intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

Starting / stopping Activity

- The function to go to another Activity is

  startActivity(Intent i)
- An Intent object specifies the page where you are, and then what page you want to go:

  Intent nextPage =new Intent(Activity.this, NextActivity.class);
- Now to go to that page, call:

  startActivity(nextPage);
- In the file FirstActivity.java, there is a button that goes to the second page.

Finish an Activity

- finish() – this function stops an Activity and goes back to the previous Activity (back 1 page in the history)
- Once we go back, FirstActivity.java will call

  onStart() - The page is visible

  onResume( ) – The app is listening for button pushes
- Look at SecondActivity.java. Clicking on the button with id="previousPageButton" will go back to the previous page on the phone. It should go back to "FirstActivity".

Sending data to next page

- You can attach information to an Intent object to send to the next page.

- The Intent class has functions:

  putExtra(String name, type dataToSend); // type is int, long, String

- This creates a map which reserves the dataToSend under a name.

```
intent.putExtra("name", "me");
intent.putExtra("age", 20);
intent.putExtra("typed", editText.getText())
```

Getting data from previous page

- In Android, you can retrieve the Intent object that caused the transition with the function ***getIntent()***
- To get the values that were reserved under names, use getIntExtra(), getStringExtra(), getBooleanExtra(), etc.

- In ***SecondActivity.java***:

```
intent = getIntent()
intent.getStringExtra("name") -> "me"
intent.getIntExtra("age", 0) -> 20
intent.getStringExtra("typed") -> whatever the user typed in editText
```