Android Application Development, COMP 10073

Mohawk College, Winter 2021

Activities

https://developer.android.com/guide/components/activities/intro-activities

- An Activity is one of the primary Android application components.
- Activity can generally be thought of as a window that fills the screen on the device presenting the user interface, but it is not restricted to that.

The Activity class is a crucial component of an Android app, and the way activities are launched and put together is a fundamental part of the platform's application model. Unlike programming paradigms in which apps are launched with a main() method, the Android system initiates code in an Activity instance by invoking specific callback methods that correspond to specific stages of its lifecycle.

Most apps contain multiple screens, which means they comprise multiple activities. Typically, one activity in an app is specified as the main activity, which is the first screen to appear when the user launches the app. Each activity can then start another activity in order to perform different actions.

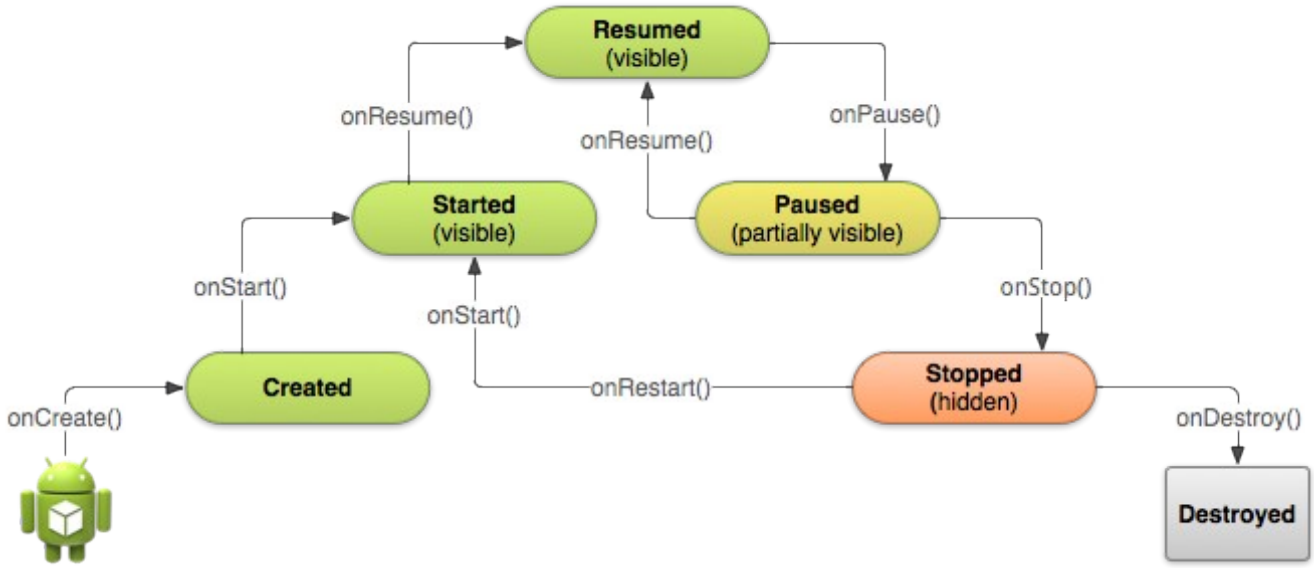
Only one activity can have focus and be responding to events from the user such as touch events. As the user navigates your application activities are started and stopped. A user's interaction with the app doesn't always begin in the same place. Instead, the user journey often begins non-deterministically.

Although activities work together to form a cohesive user experience in an app, each activity is only loosely bound to the other activities; there are usually minimal dependencies among the activities in an app. In fact, activities often start up activities belonging to other apps.

Activity LifeCycle

https://developer.android.com/guide/components/activities/activity-lifecycle

As a user navigates through, out of, and back to your app, the Activity instances in your app transition through different states in their lifecycle. The Activity class provides a number of callbacks that allow the activity to know that a state has changed: that the system is creating, stopping, or resuming an activity, or destroying the process in which the activity resides.



Lifetimes

Entire Lifetime: between onCreate() and onDestroy()

Visible Lifetime: between onStart() and onStop(), your activity is visible but not necessarily in focus

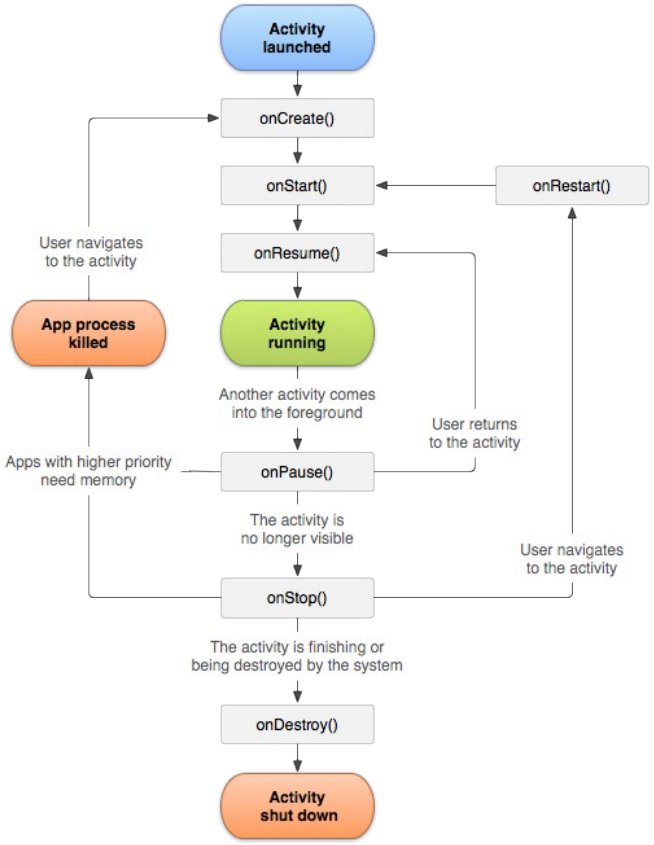
Foreground Lifetime: between onResume() and onPause() activity is in front of everything else and you have focus

Callback Methods

An activity will change state during its lifetime. Callback methods can be defined to respond to those events with appropriate code. For example, onCreate() is called as the activity first comes to life. There are many others such as onPause() and onResume().

Activity LifeCycle Viewed with Log.d()

Use Android's logging facility to trace the application through various callbacks. You can change the applications state by rotating the phone, switching activities or ending the application



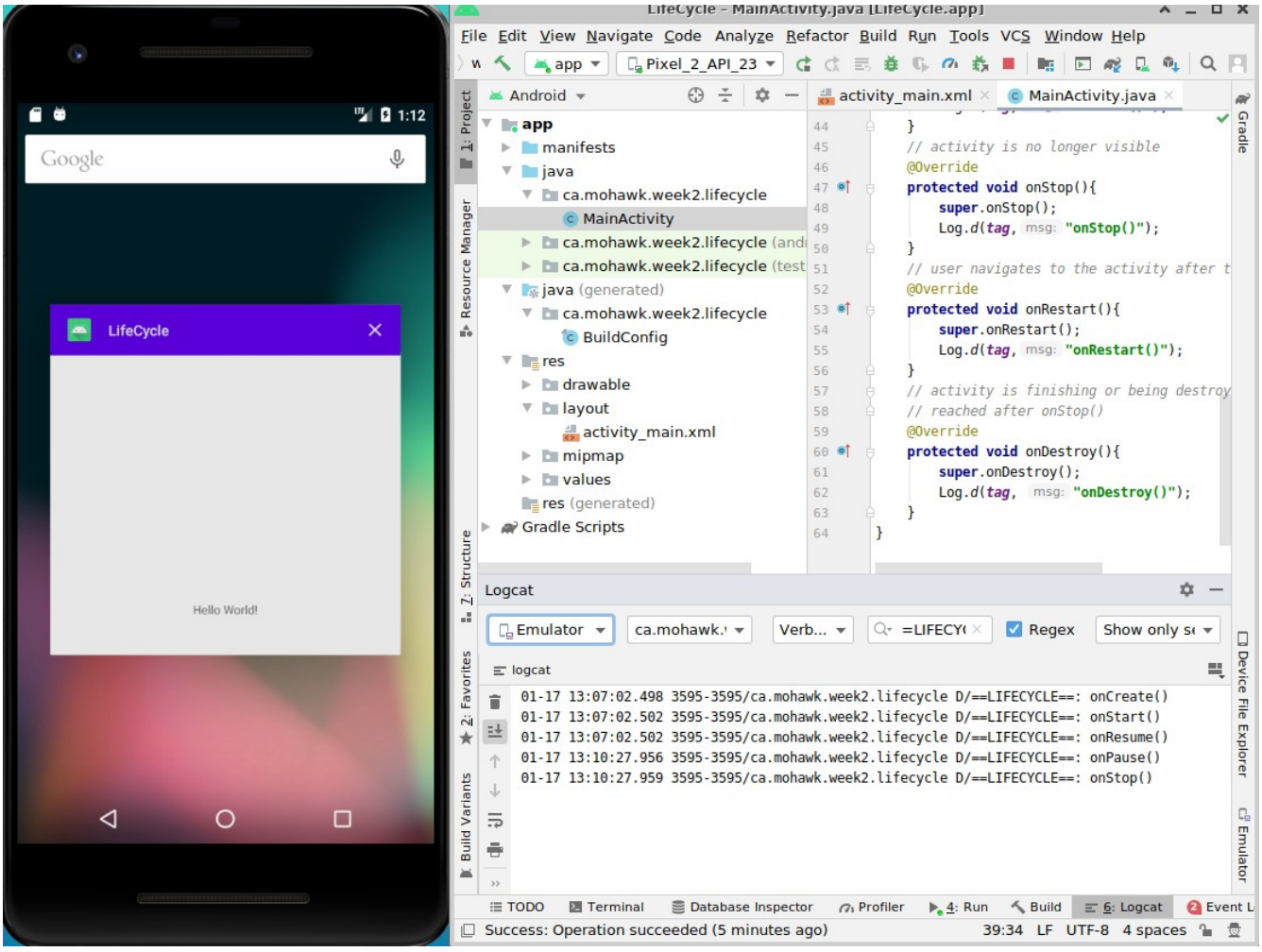

```
package ca.mohawk.week2.lifecycle;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.util.Log;

public class MainActivity extends AppCompatActivity {

    final static String tag = "==LIFECYCLE==";

    // user navigates to the activity, render the display
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.d(tag,"onCreate()");
    }
    // reached after onCreate() or onRestart()
    // called when the activity is about to become visible
    @Override
    protected void onStart() {
        super.onStart();
        Log.d(tag,"onStart()");
    }
    // reached after onPause() or onStart()
    // called when the activity has become visible
    @Override
    protected void onResume() {
        super.onResume();
        Log.d(tag,"onResume()");
    }

    /**
     * ---------------------------------------------
     *  Activity is running after call to onResume()
     * ---------------------------------------------
     */

    // another activity comes into the foreground
    @Override
    protected void onPause(){
        super.onPause();
        Log.d(tag,"onPause()");
    }
    // activity is no longer visible
    @Override
    protected void onStop(){
        super.onStop();
        Log.d(tag,"onStop()");
    }
    // user navigates to the activity after the stop state
    // next state will be onStart()
    @Override
    protected void onRestart(){
        super.onRestart();
        Log.d(tag,"onRestart()");
    }
    // activity is finishing or being destroyed by the system
    // reached after onStop()
    @Override
    protected void onDestroy(){
        super.onDestroy();
        Log.d(tag, "onDestroy()");
    }
}
```

Log.d() tag, message and filter

https://developer.android.com/reference/android/util/Log

Select the Logcat tab at the bottom of android studio. This example uses a unique tag, this makes it easy to filter out the unrelated messages.

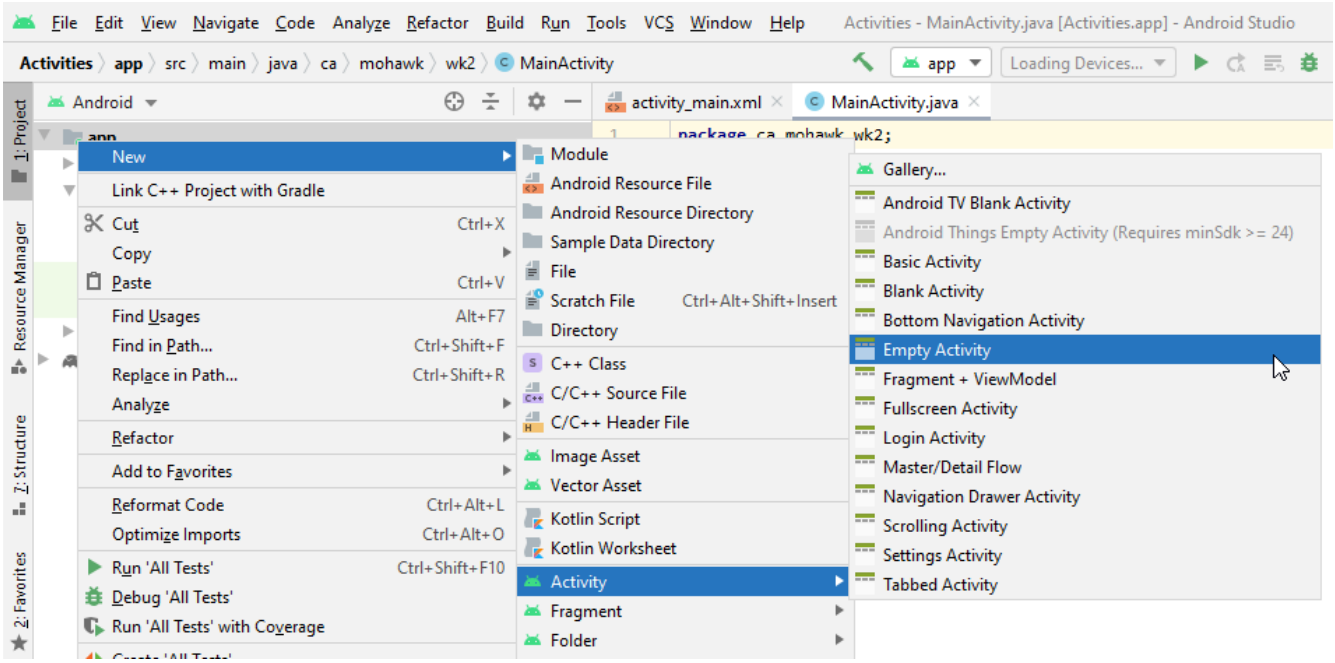
Select the correct emulator instance and running process. If you select an old process you will not see your current set of messages. Debug level messages are useful for understanding the behaviour of your application. Production builds will normally remove debugging code so your users won't have access to these messages and you can be as verbose as you like.



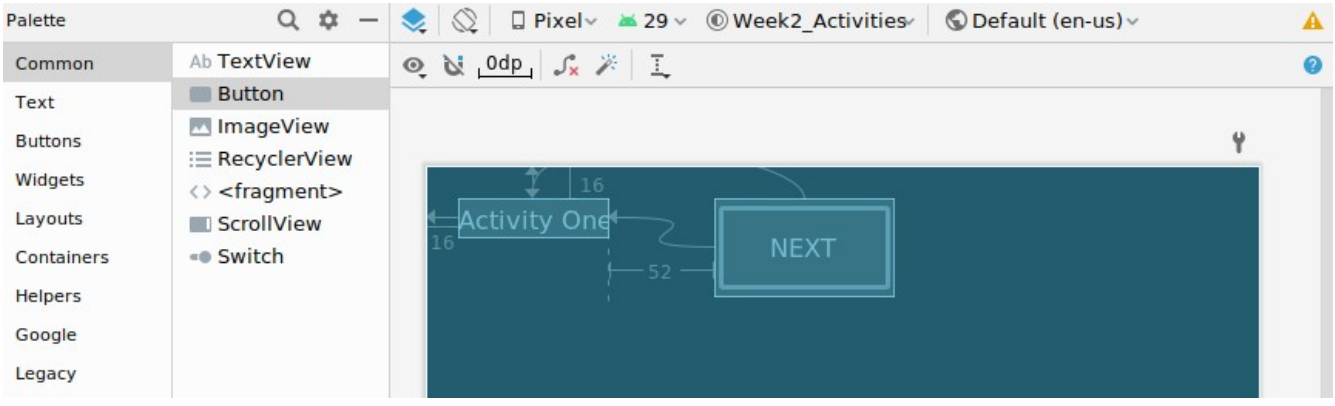
Multiple Activities

Android will start your first (MAIN) activity for you. Once the main activity is up and running you can develop code that switches activites.

Create a new project starting with Android SDK23. Once the application is loaded, right click on the root of the app tree and create a new empty activity. Let it use the default name “MainActivity2”.



Modify the TextView widgets in each of the activities so that they are located at the upper right of the screen and display either “Activity One” or “Activity Two”. Add a button to the first activity and create an empty handler.



startActivity() and Intent

https://developer.android.com/reference/android/content/Context#startActivity(android.content.Intent)

https://developer.android.com/reference/android/content/Intent

You can start up a new activity by specifying an “Intent”. The intent describes what we plan on doing, the startActivity() method does the work using the Intent as its argument. For our application make a button handler that creates an intent describing the new activity we want to start, the code looks like this:

```
public void nextActivity (View view) {
    Intent switch2Activity2 = new Intent(MainActivity.this, MainActivity2.class);
    startActivity(switch2Activity2);
}
```

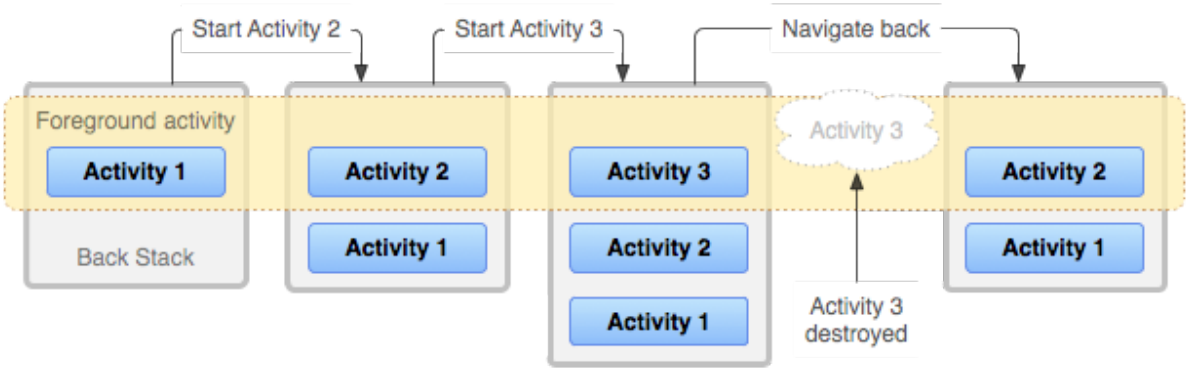
The switch2Activity2 object that we created specifies the current context, “MainActivity.this” and the name of the class that we want to start “MainActivity2.class”.

Set the onClick attribute for the button in the first activity to trigger the new method nextActivity and now we can switch activities. You can use the phone’s back button to return to the first activity.

Activity Stack

https://developer.android.com/guide/components/activities/tasks-and-back-stack

When a new activity is started the previous one is stopped and pushed onto the stack which the Android system maintains. User can press the Back button to go back through those previous activities. The stack is LIFO, Last In First Out. An activity held on the stack can simply be destroyed by Android if resources are low.



Add the calls to Log.d() to both activity classes to track the state changes as you switch between activities. Modify the Tag argument for the second activity.

- Timestamp 11:00:17, the app starts.
  - The first activity switches to the first 3 lifecycle stages.
- Timestamp 11:00:31, press the “Next” button.
  - First activity pauses, the 2nd activity switches to the first 3 lifecycle stages
- Timestamp 11:00:44, press the “Back” button.
  - 2nd activity pauses while waiting for 1st activity to get to onResume()
  - Timestamp 11:00:45, second activity reaches onStop(), then onDestroy()