Mobile Application Development

Lesson 5

Dr. Syed Asim Jalal
Department of Computer Science
University of Peshawar

Activity and Activity Life Cycle

Activity

What is an Activity?

An Activity is the screen representation of any application in Android

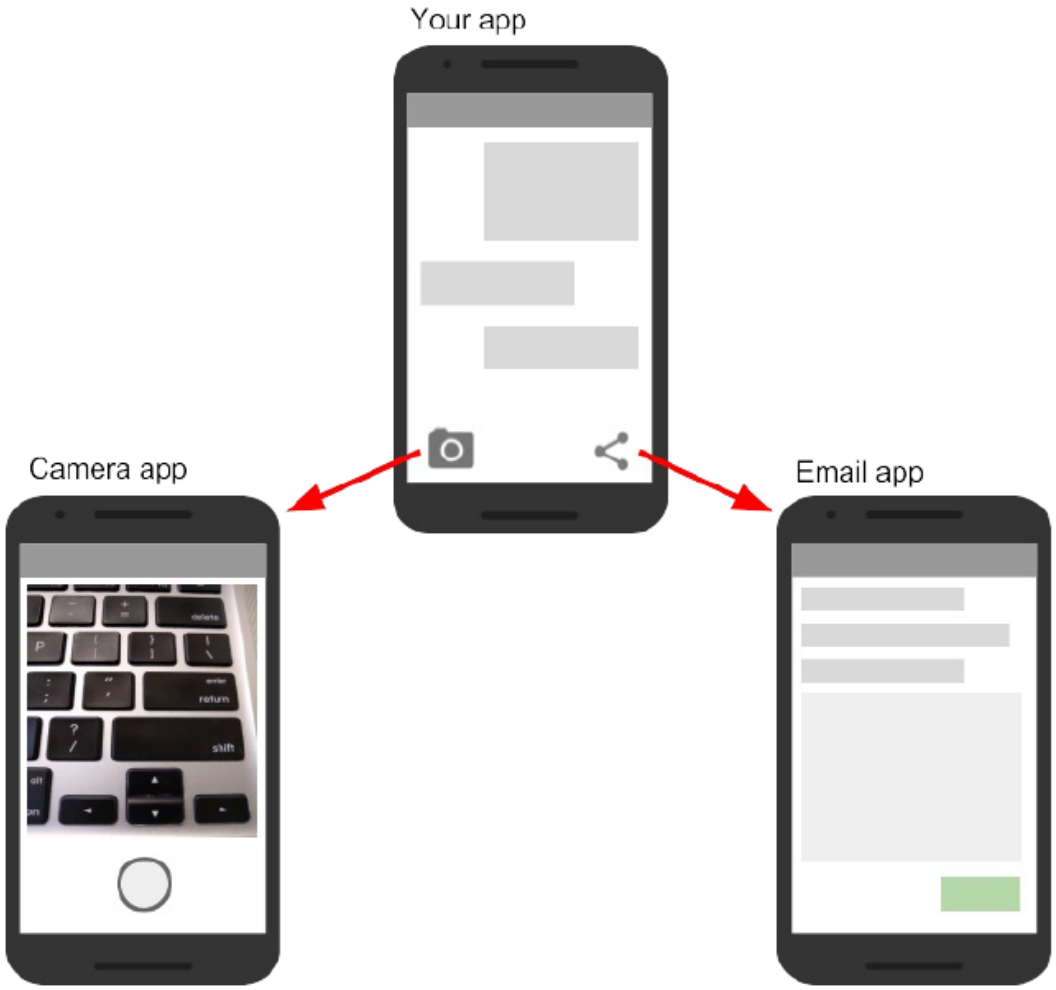
→ It serves as an entry point for user's interaction.

→ Each activity has a layout file where you can place your UI.

→ An application can have different activities.

- An activity represents a **single screen** in your app with an interface the user can interact with.
- For example, an email app might have one activity that shows a **list of new emails**, another activity to **compose** an email, and another activity for **reading** individual messages.
- Your app is a collection of activities that you either create yourself, or that you reuse from other apps.
- Although the activities in your app work together to form a cohesive user experience in your app, each ACTIVITY is **independent** of the others.

You can call other activities of other apps from one activity



Creating activities

- To implement an activity in your app, do the following:
  - Implement a user interface for that activity.
  - Create an activity Java class.
  - Declare that new activity in the app manifest.
- When you create a new project for your app, or add a new activity to your app, in Android Studio (with **File > New > Activity**), template code for each of these tasks is provided for you.

- Activities are subclasses of the Activity class, or one of its subclasses. When you create a new project in Android Studio, your activities are, by default, subclasses of the AppCompatActivity class.
- The AppCompatActivity class is a subclass of Activity that lets you to use up-to-date android app features while still enabling your app to be compatible with devices running older versions of Android.

Here is a skeleton subclass of AppCompatActivity:

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

- The first task for you in your activity subclass is to implement the standard activity lifecycle callback methods (**such as OnCreate()**) to handle the state changes for your activity.
- These state changes include things such as when the activity is *created*, *stopped*, *resumed*, or *destroyed*.

The one required callback your app must implement is the onCreate() method. The system calls this method when it creates your activity, and all the essential components of your activity should be initialized here. Most importantly, the OnCreate() method calls setContentView() to create the primary layout for the activity.

- You typically define the user interface for your activity in one or more XML layout files.
- When the setContentView() method is called with the path to a layout file, the system creates all the initial views from the specified layout and adds them to your activity.
- This is often referred to as **inflating** the layout.

Implement a user interface of an Activity

- The user interface for an activity is provided by a hierarchy of views, which controls a particular space within the activity's window and can respond to user interaction.
- The most common way to define a user interface using views is with an XML layout file stored as part of your app's resources.
- Defining your layout in XML enables you to maintain the design of your user interface separately from the source code that defines the activity's behavior.

- **Declare the activity in the manifest**
  - Each activity in your app must be declared in the Android app manifest with the **<activity>** element, inside **<application>** .
  - When you create a new project or add a new activity to your project in Android Studio, your manifest is created or updated to include skeleton activity declarations for each activity. Here's the declaration for the **main activity**.

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

- **Add more activities to your project**
  - You can add new activities to your project in Android Studio with the **File > New > Activity** menu.
  - Choose the activity template you want to use, or open the Gallery to see all the available templates.

- Android Studio provides these three things for each new activity in your app:

1. A Java file for the new activity with a skeleton class definition and onCreate() method. The new activity, like the main activity, is a subclass of AppCompatActivity.

2. An XML file containing the layout for the new activity. Note that the **setContentView()** method in the activity class inflates this new layout.

3. An additional **<activity>** element in the Android manifest that specifies the new activity. The second activity definition does not include any intent filters.

Activity Life Cycle

- Typically, one activity in an app is specified as the **"main activity”**, which is presented to the user when launching the application for the first time. Each activity can then start other activities in order to perform different actions.
- Each time a **new activity** starts, the **previous** activity is **stopped**, but the system **preserves the activity in a stack** (the **"back stack**").
- When the user is **done** with the current activity and presses the **Back button**, it is popped from the stack (and **destroyed**) and the previous activity resumes.

- When an activity is stopped because a new activity starts, the first activity is notified of that change with the activity's lifecycle callback methods.

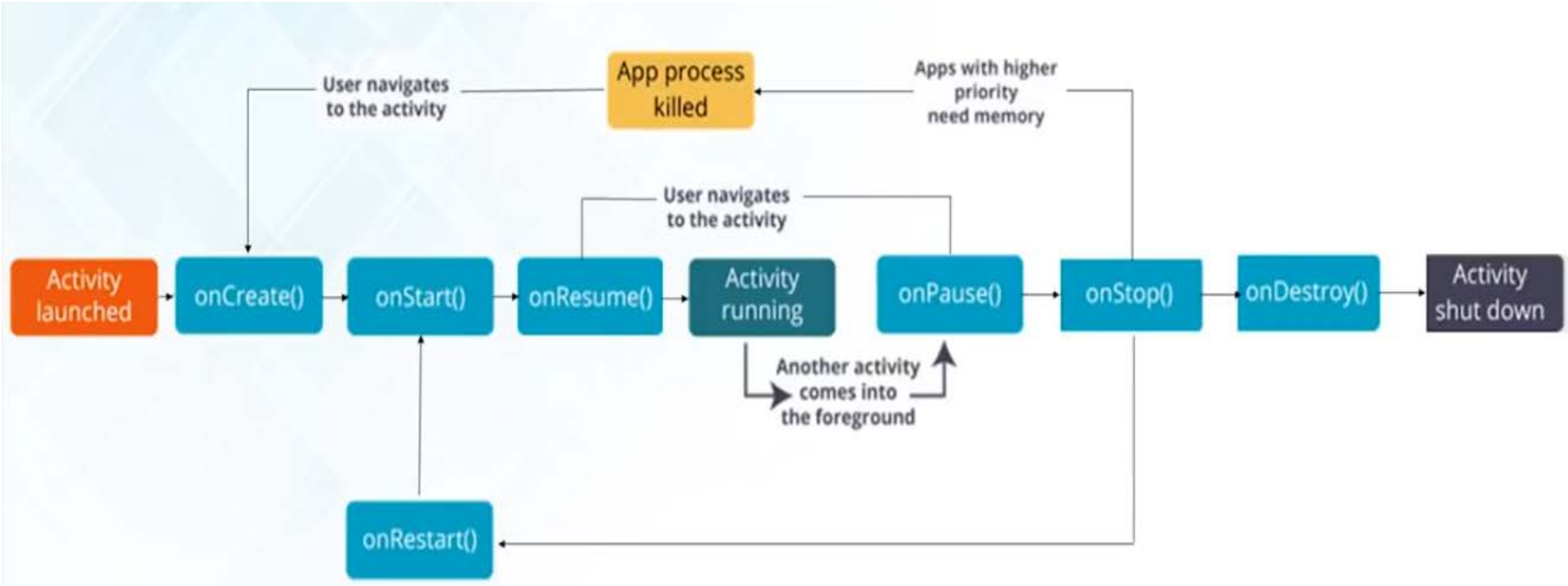
- The **Activity lifecycle** is the set of **states an activity** can be in any particular instance of time, from when it is first created, to each time it is stopped or resumed, to when the system destroys it.

- The lifecycle is the set of states an activity can be in during its entire lifetime, from when it's created to when it's destroyed and the system reclaims its resources.
- As a user navigates between activities in your app (as well as into and out of your app), activities transition between different states in their lifecycles.

- Each stage in an activity's lifecycle has a corresponding callback method:
  - onCreate(), onStart(), onPause(), and so on.
  - When an activity changes state, the associated callback method is invoked.
  - You've already seen one of these methods: onCreate().
  - By overriding any of the lifecycle callback methods in your Activity classes, you can change the activity's default behavior in response to user or system actions.

- The activity state **can also change** in response to **device-configuration changes**, for example when the user rotates the device from portrait to landscape.
- After such changes the activity is destroyed and recreated in its default state, and the user might lose information that they've entered in the activity.
- To avoid confusing your users, it's important that you develop your app to prevent unexpected data loss, by saving all the changes.

Activity life Cycle diagram



Observing Activity Life Cycle

- In order to understand the Activity Life Cycle and the different states an activity can pass through, we can develop an app where we explicitly implement the relevant function for responsible for each state and display some message.
- Then we push the activity to change states by opening other apps and killing this app to observe the messages.
- These functions are
  - onCreate(), onStart(), onResume(), onPause(), onStop(), onDestroy()

An example to understand activity states using Log Messages.

Create

In this state, the activity is created.

Resumed (running state)

In this state, the activity is in the foreground and the user can interact with it.

Stopped

In this state, the activity is completely hidden and not visible to the user. it is considered to be in the background.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Log.i(TAG, "onCreate Called");
}

@Override
protected void onStart() {
    super.onStart();
    Log.i(TAG, "onStart Called");
}

@Override
protected void onResume() {
    super.onResume();
    Log.i(TAG, "onResume Called");
}

@Override
protected void onPause() {
    super.onPause();
    Log.i(TAG, "onPause Called");
}

@Override
protected void onStop() {
    super.onStop();
    Log.i(TAG, "onStop Called");
}

@Override
protected void onDestroy() {
    super.onDestroy();
    Log.i(TAG, "onDestroy Called");
}
```

Paused

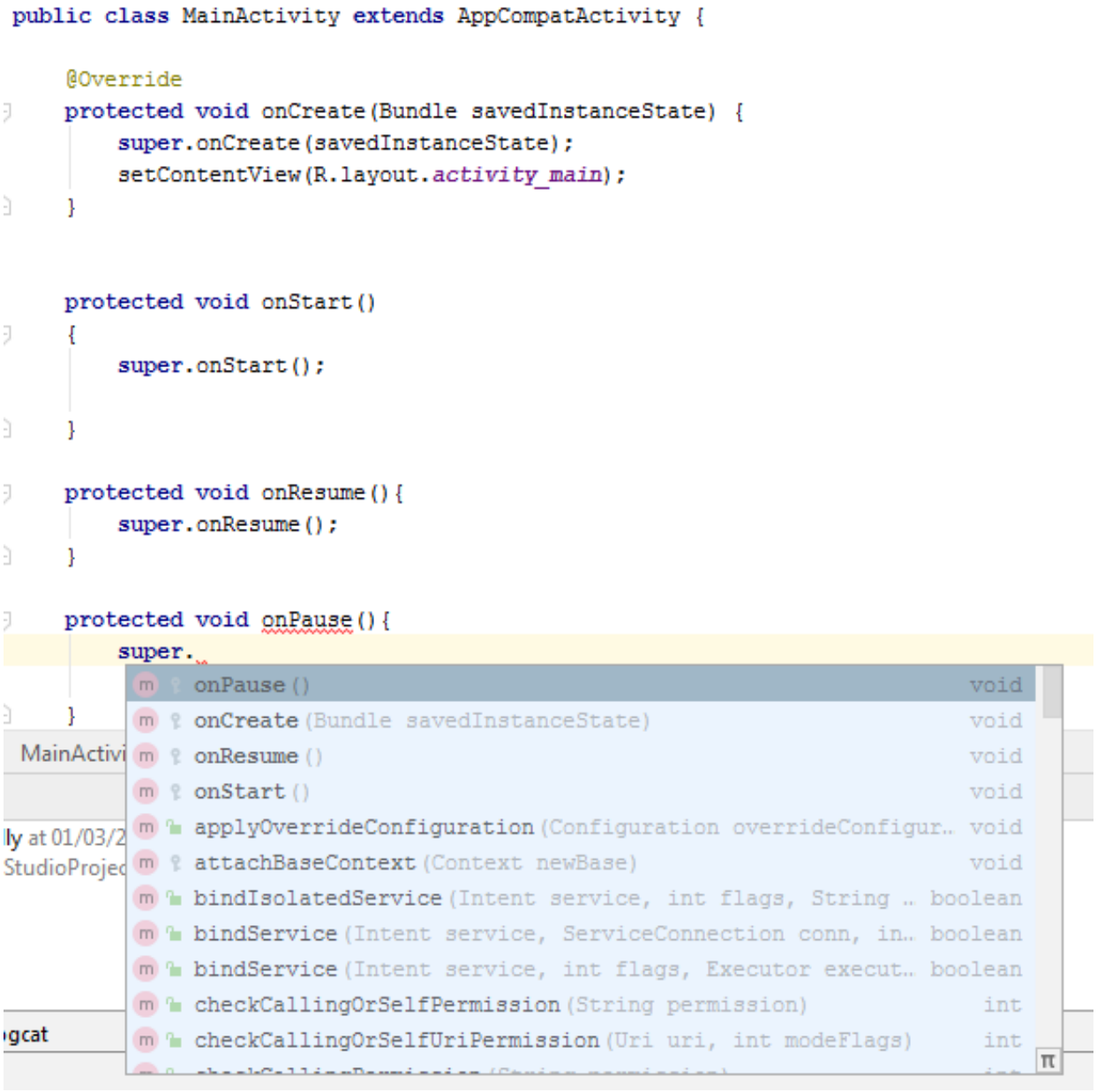
Activity is partially obscured by another activity. The other activity that's in the foreground is semi-transparent.

Destroy

In this case the activity is destroyed and removed from the memory

Example 1

- We implement one example where we display one Toast message in each activity life cycle function to observe the activity life cycle states of every function.
- These functions are implemented in Super class Activity Class.
- We therefore implement these classes, and also execute the execute the original functions in the super class by writing the keyword super.
- E.g. super.onStart().


public class MainActivity extends AppCompatActivity {
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
}
protected void onStart()
{
super.onStart();
}
protected void onResume(){
super.onResume();
}
protected void onPause(){
super.
onPause() void
onCreate(Bundle savedInstanceState) void
onResume() void
onStart() void
applyOverrideConfiguration(Configuration overrideConfigur.. void
attachBaseContext(Context newBase) void
bindIsolatedService(Intent service, int flags, String .. boolean
bindService(Intent service, ServiceConnection conn, in.. boolean
bindService(Intent service, int flags, Executor execut.. boolean
checkCallingOrSelfPermission(String permission) int
checkCallingOrSelfUriPermission(Uri uri, int modeFlags) int
MainActivi
ly at 01/03/2
StudioProjec
gcat

```java
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toast.makeText( context: this, text: "onCreate Function Executed",Toast.LENGTH_LONG).show();
    }
    protected void onStart()
    {
        super.onStart();
        Toast.makeText( context: this, text: "onStart Function Executed",Toast.LENGTH_LONG).show();
    }
    protected void onResume(){
        super.onResume();
        Toast.makeText( context: this, text: "onResume Function Executed",Toast.LENGTH_LONG).show();
    }
    protected void onPause(){
        super.onPause();
        Toast.makeText( context: this, text: "onPause function is called",Toast.LENGTH_LONG).show();
    }
    protected void onStop(){
        super.onStop();
        Toast.makeText( context: this, text: "onStop Function executed NOW",Toast.LENGTH_LONG).show();
    }
    protected void onDestroy(){
        super.onDestroy();
        Toast.makeText( context: this, text: "onDestroy Function is Executed NOW",Toast.LENGTH_LONG).show();
    }
}
```

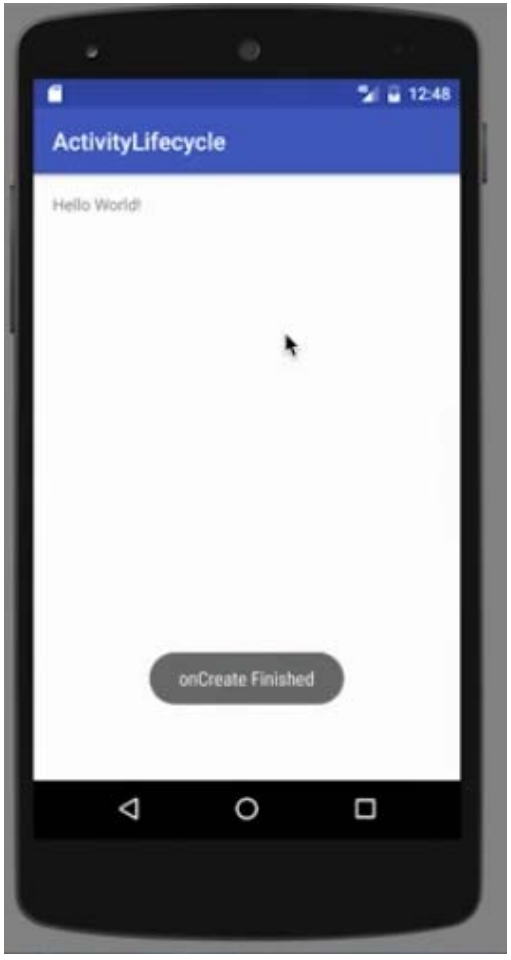
Example 2

This example, displays a Toast message as well as display a log message.

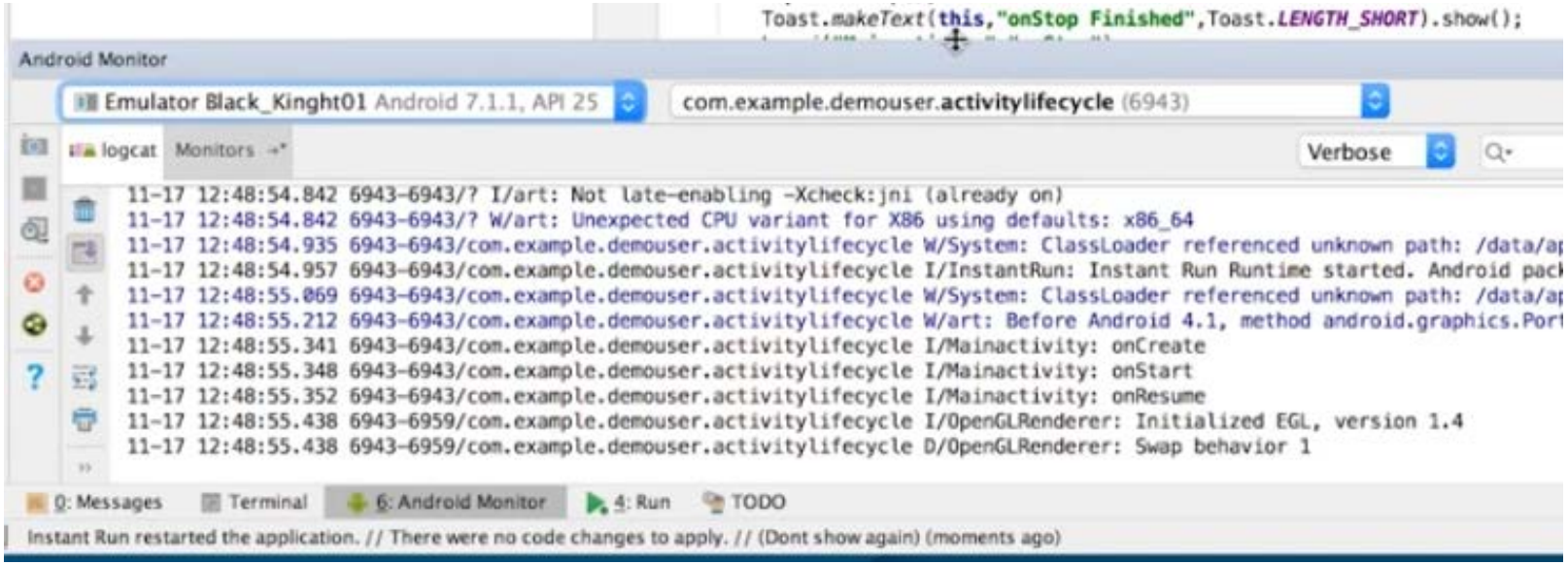
```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toast.makeText(this,"onCreate Finished",Toast.LENGTH_SHORT).show();
        Log.i("Mainactivity","onCreate");
    }

    @Override
    protected void onStart() {
        super.onStart();

    }
```

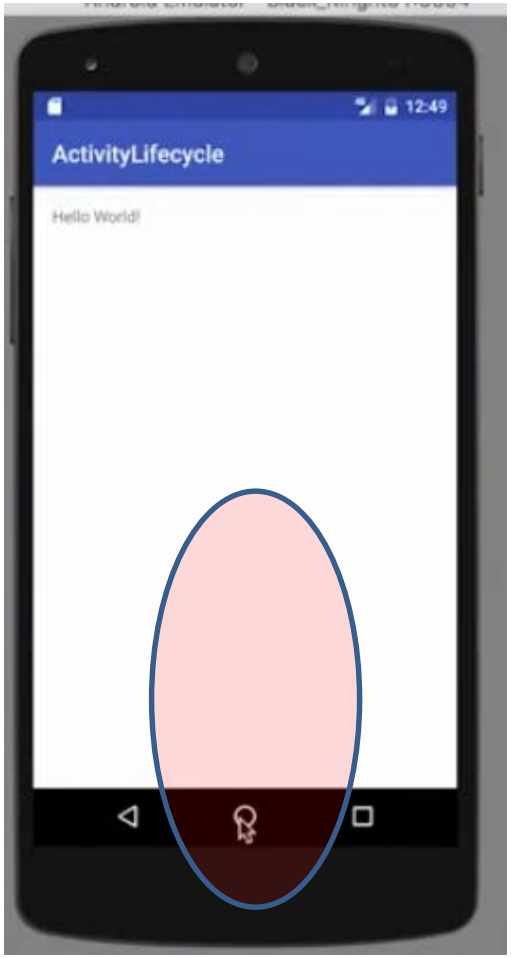

activity_main.xml
MainActivity.java
Toast.makeText(this,"onCreate Finished",Toast.LENGTH_SHORT).show();
Log.i("Mainactivity","onCreate");
}
@Override
protected void onStart() {
super.onStart();
Toast.makeText(this,"onStart Finished",Toast.LENGTH_SHORT).show();
Log.i("Mainactivity","onStart");
}
@Override
protected void onResume() {
super.onResume();
Toast.makeText(this,"onResume Finished",Toast.LENGTH_SHORT).show();
Log.i("Mainactivity","onResume");
}
@Override
protected void onPause() {
super.onPause();
Toast.makeText(this,"onPause Finished",Toast.LENGTH_SHORT).show();
Log.i("Mainactivity","onPause");
}
@Override
protected void onStop() {
super.onStop();
Toast.makeText(this,"onStop Finished",Toast.LENGTH_SHORT).show();
Log.i("Mainactivity","onStop");
}
@Override
protected void onRestart() {
super.onRestart();
Toast.makeText(this,"onRestart Finished",Toast.LENGTH_SHORT).show();
Log.i("Mainactivity","onRestart");
}
@Override
protected void onDestroy() {
super.onDestroy();
Toast.makeText(this,"onDestroy Finished",Toast.LENGTH_SHORT).show();
Log.i("Mainactivity","onDestroy");
}
}

When the application is first started, onCreate(), onStart() and onResume() are executed.



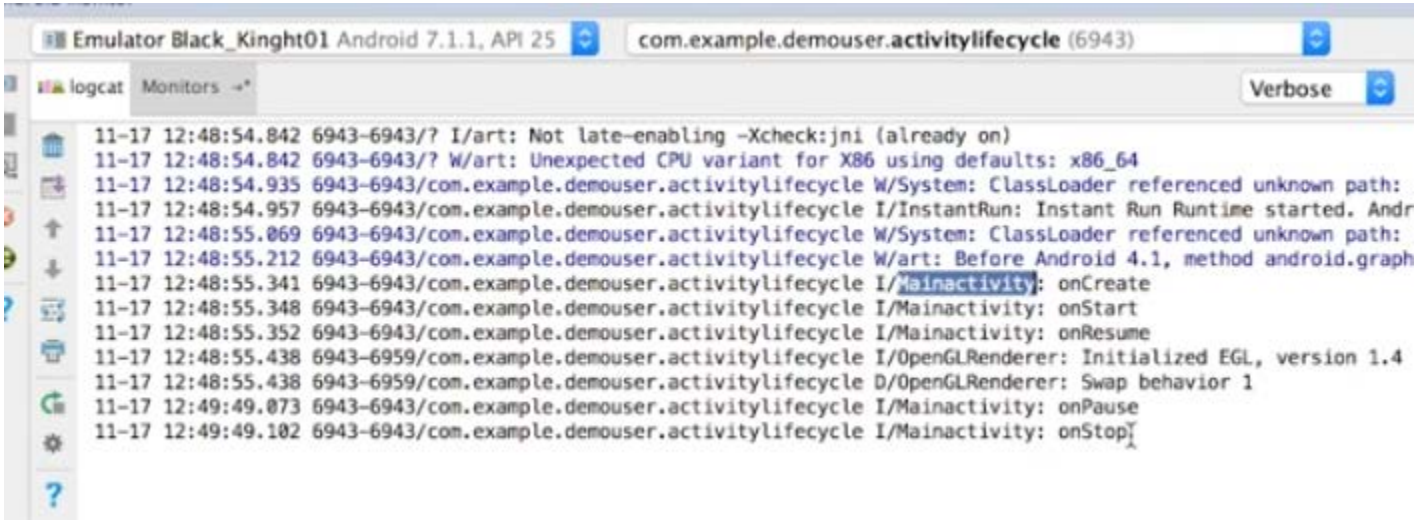
Log messages which are written in each functions are also displayed in the log area.



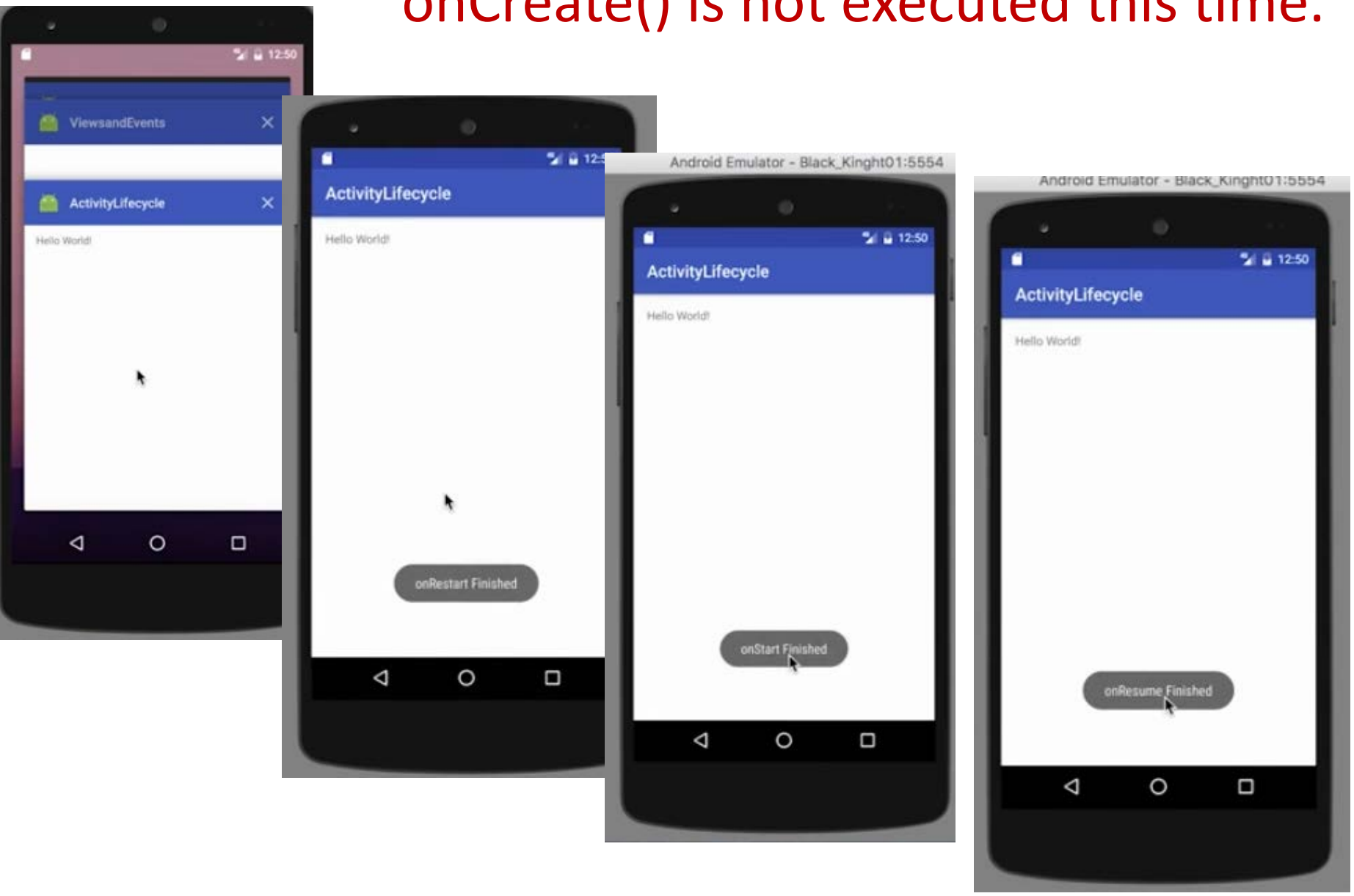

12:49
ActivityLifecycle
Hello World!


12:49
November 17
Maps
onPause Finished

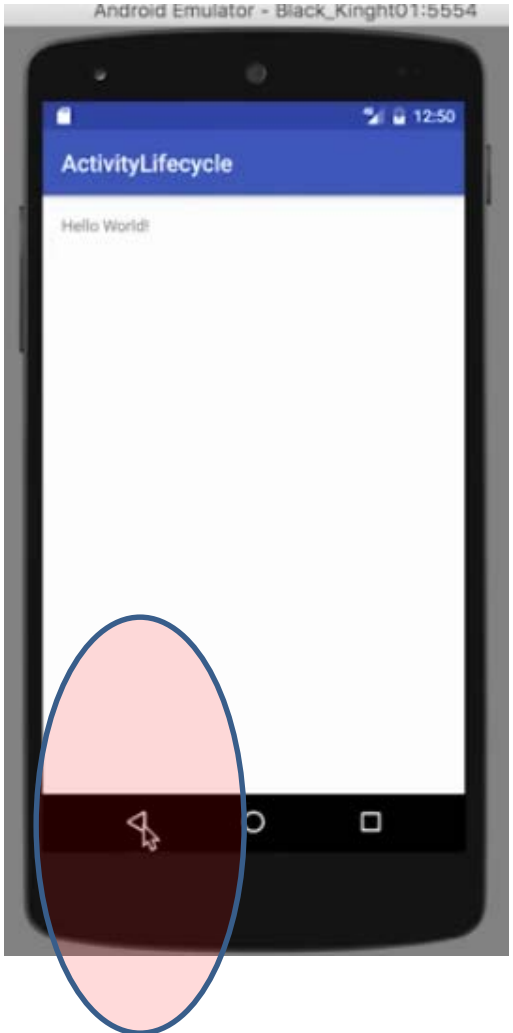

12:49
November 17
Maps
onStop Finished


Emulator Black_Kinght01 Android 7.1.1, API 25
com.example.demouser.activitylifecycle (6943)
logcat
Monitors
Verbose
11-17 12:48:54.842 6943-6943/? I/art: Not late-enabling -Xcheck:jni (already on)
11-17 12:48:54.842 6943-6943/? W/art: Unexpected CPU variant for X86 using defaults: x86_64
11-17 12:48:54.935 6943-6943/com.example.demouser.activitylifecycle W/System: ClassLoader referenced unknown path:
11-17 12:48:54.957 6943-6943/com.example.demouser.activitylifecycle I/InstantRun: Instant Run Runtime started. Andr
11-17 12:48:55.069 6943-6943/com.example.demouser.activitylifecycle W/System: ClassLoader referenced unknown path:
11-17 12:48:55.212 6943-6943/com.example.demouser.activitylifecycle W/art: Before Android 4.1, method android.graph
11-17 12:48:55.341 6943-6943/com.example.demouser.activitylifecycle I/Mainactivity: onCreate
11-17 12:48:55.348 6943-6943/com.example.demouser.activitylifecycle I/Mainactivity: onStart
11-17 12:48:55.352 6943-6943/com.example.demouser.activitylifecycle I/Mainactivity: onResume
11-17 12:48:55.438 6943-6959/com.example.demouser.activitylifecycle I/OpenGLRenderer: Initialized EGL, version 1.4
11-17 12:48:55.438 6943-6959/com.example.demouser.activitylifecycle D/OpenGLRenderer: Swap behavior 1
11-17 12:49:49.073 6943-6943/com.example.demouser.activitylifecycle I/Mainactivity: onPause
11-17 12:49:49.102 6943-6943/com.example.demouser.activitylifecycle I/Mainactivity: onStop

When the app is restarted.
onCreate() is not executed this time.



When the back button is pressed the app is destroyed.



When you click the app icon and restart the app.