Activities and Intents

Activities

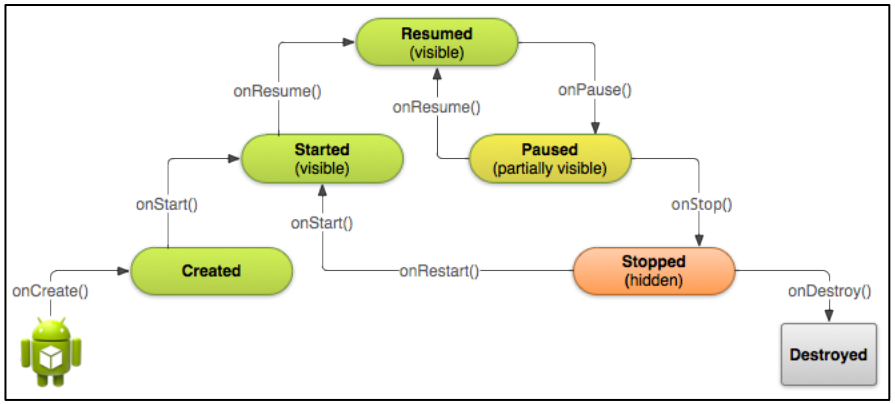
- An **activity** represents a single screen in the app where the user can perform a single focused task such as sending an e-mail. It is usually presented to the user as a full-screen window.
- To implement an activity:
  1. Create an `Activity` Java class.

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

  2. Implement a basic UI for the `Activity` in an associated XML layout file.
  3. Declare the new `Activity` in AndroidManifest.xml.

  To automate the three (3) tasks in Android Studio, choose **File > New > Activity** to start from a template.
- The **main activity** (MainActivity.java) is presented to the user when the app is launched. It can then start other activities to perform different actions.
- The **activity life cycle** is the set of states an activity can be in during its entire lifetime.



- The callback methods used during the activity lifecycle are the following:

Callback Method	Description
`onCreate()`	It is invoked when the app is launched for the first time. It happens only once for the entire life of the activity.
`onStart()`	It is invoked before the activity becomes visible to the user. It is followed by either: • `onResume()` – if the activity comes to the foreground • `onStop()` – if the activity becomes hidden
`onResume()`	It is invoked before the activity starts interacting with the user.
`onPause()`	It is invoked when the system is about to start resuming another activity. It is followed by either: • `onResume()` – if the activity returns to the background • `onStop()` – if the activity becomes invisible to the user
`onStop()`	It is invoked when the activity is no longer visible to the user. It is followed by either: • `onRestart()` – if the activity is coming back to interact with the user • `onDestroy()` – if the activity is about to end
`onDestroy()`	It is invoked when: • the activity is finishing (due to the user completely dismissing the activity or due to `finish()` being called on the activity) • the system is temporarily destroying the activity due to a configuration change (such as device rotation or multi-window mode)

`onRestart()`	It is invoked if the activity comes back after being stopped. It is always followed by `onStart()`.

- Each time a new activity starts, the previous activity is stopped, but the system preserves the activity in a stack called the **back stack**.

Intents

- An activity is started or activated with an **intent**. An **`Intent`** is an asynchronous message that is used in an activity to request an action from another activity, or from some other app component.
- An intent can be used to start one activity from another activity, and to pass data between activities.
- The parts of an intent are following:
  - **Target activity** – the activity that will receive the intent.
  - **Intent data/object** – contains a reference to the data you want the receiving activity to operate on.
  - **Intent extras** – carry information the receiving activity requires to accomplish the requested action (optional).
  - **Intent flags** – may instruct the Android system how to launch an Activity or how to treat it after it's launched (optional).
- The two (2) types of intent are as follows:
  - **Explicit intent**: The target of the intent (the class name of the activity) is already identified.
  - **Implicit intent**: The target of the intent is not yet identified but there is a general action to perform. It also includes an action, category, and data type.
- To add an intent after creating a new activity:
  1. The `<activity>` elements in the AndroidManifest.xml should look like these:

```
<activity android:name=".Main2Activity"
    android:parentActivityName=".MainActivity">
</activity>
<activity android:name=".MainActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

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

The **`parentActivityName`** attribute indicates that the main activity is the parent of the second activity. This enables a left-facing arrow that allows the user to navigate from the current activity to its parent activity.

2. Add the `android:onClick` attribute to the `Button` element that will be used to start another activity.

```
android:id="@+id/button_next"
android:onClick="launchNextActivity"
```

3. Define the method that will be used to start another activity.

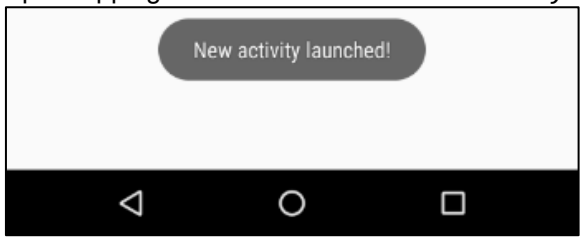
```
public void launchNextActivity(View view) {
    Intent intent = new Intent(this, Main2Activity.class);
    startActivity(intent);
}
```

The intent added is an explicit intent because the receiving activity is specified (`Main2Activity`). The `this` keyword represents the current activity.

To verify if the intent works, add a toast to `Main2Activity`.

```
Toast.makeText(getApplicationContext(),"New activity launched!",
        Toast.LENGTH_SHORT).show();
```

Upon tapping the `Next` button in `MainActivity`:



References:

DiMarzio, J. (2017). *Beginning Android programming with Android Studio*. Indiana: John Wiley & Sons, Inc.

Google Developers Training Team. (2018). *Android developer fundamentals (version 2)*. Retrieved from https://google-developer-training.github.io

Android Developers (n.d.). Citing sources. Retrieved from https://developer.android.com/