Lecture 5: Structure of Android Applications

Jianjun Chen (Jianjun.Chen@xjtlu.edu.cn)

Application Components

- An app comprises components that the system can instantiate and run as needed, Key components:
  - Activities <- covered in this lecture
  - Services
  - BroadcastReceiver
  - ContentProvider

Services

- A service is an app component that runs at the background.
  - It does not have UI.

- Services may still run when the UIs of their corresponding apps are not shown.

- E.g. music play at the background.

BroadcastReceiver

- Receives broadcasts from Android system and other apps and responses with pre-coded actions.
- Example 1: The mechanism of “startup services” in the Android system is achieved by doing a broadcast when the system has finished starting up.
- Example 2: When the battery level changes, the system will also broadcast a message. BroadcastReceiver can then prompt the user to save his files.

ContentProvider

- In Android, apps have separate storages for files, database etc.
  - An app cannot access other apps' data directly.

- ContentProvider can be used to share and exchange data between apps.

Activity

Activity life circle, Creating activities, Starting activities, Closing activities, Transferring data

Activity

- Activity: Primary class for interacting with user.
  - In principle, each activity is associated with one UI.

- An Android app may contain multiple activities.
  - But only one <u>main activity</u>, which is the first activity presented to users. (like `main()` function)

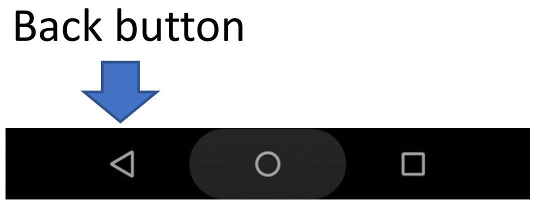
- Activities in an app first register themselves in the system.
  - Then they can be called by other activities.

Activity

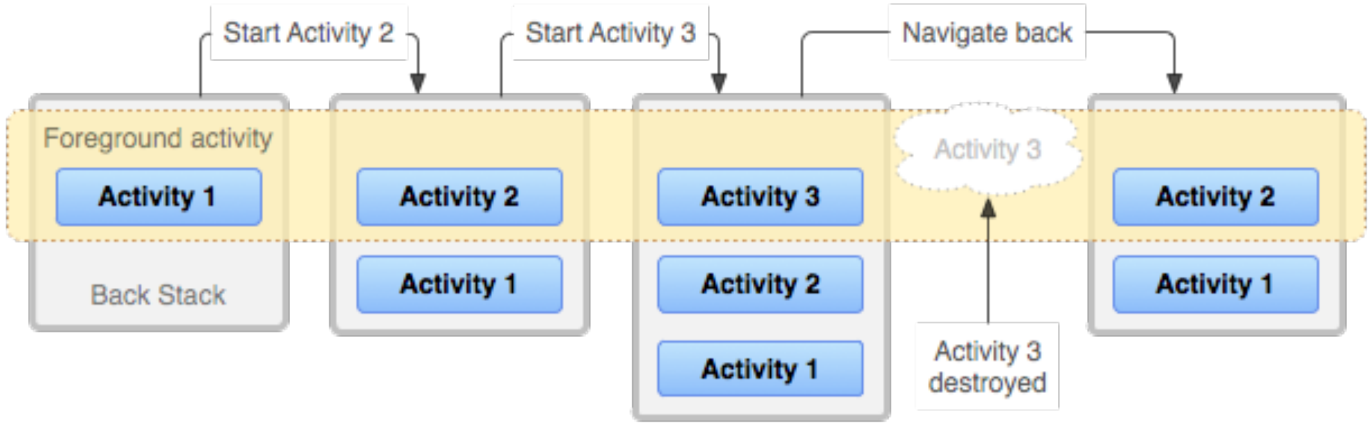
For example, Wechat:

- When you click on the app launcher, its corresponding greeting page will be shown to you.
  - Android system invokes the main activity of the Wechat.
- Apps that request for online payment (such as railway ticket payment) can directly reach the payment page.
  - Another app invokes the payment activity of Wechat.
- Apps that request for “share on moments” can directly invoke the moments sharing page of Wechat.
  - Another app invokes the “share on moments” activity of Wechat

Activity "Back Stack"



- When a new activity is launched, the previous activity will be paused and sent to the top of the back stack.
- Activities in the back stack follows the rule of last-in-first-out.
    - When the user clicks "back" button, the current activity will be destroyed.
    - The top activity in the back stack will be activated and shown.



Activity Life Cycle

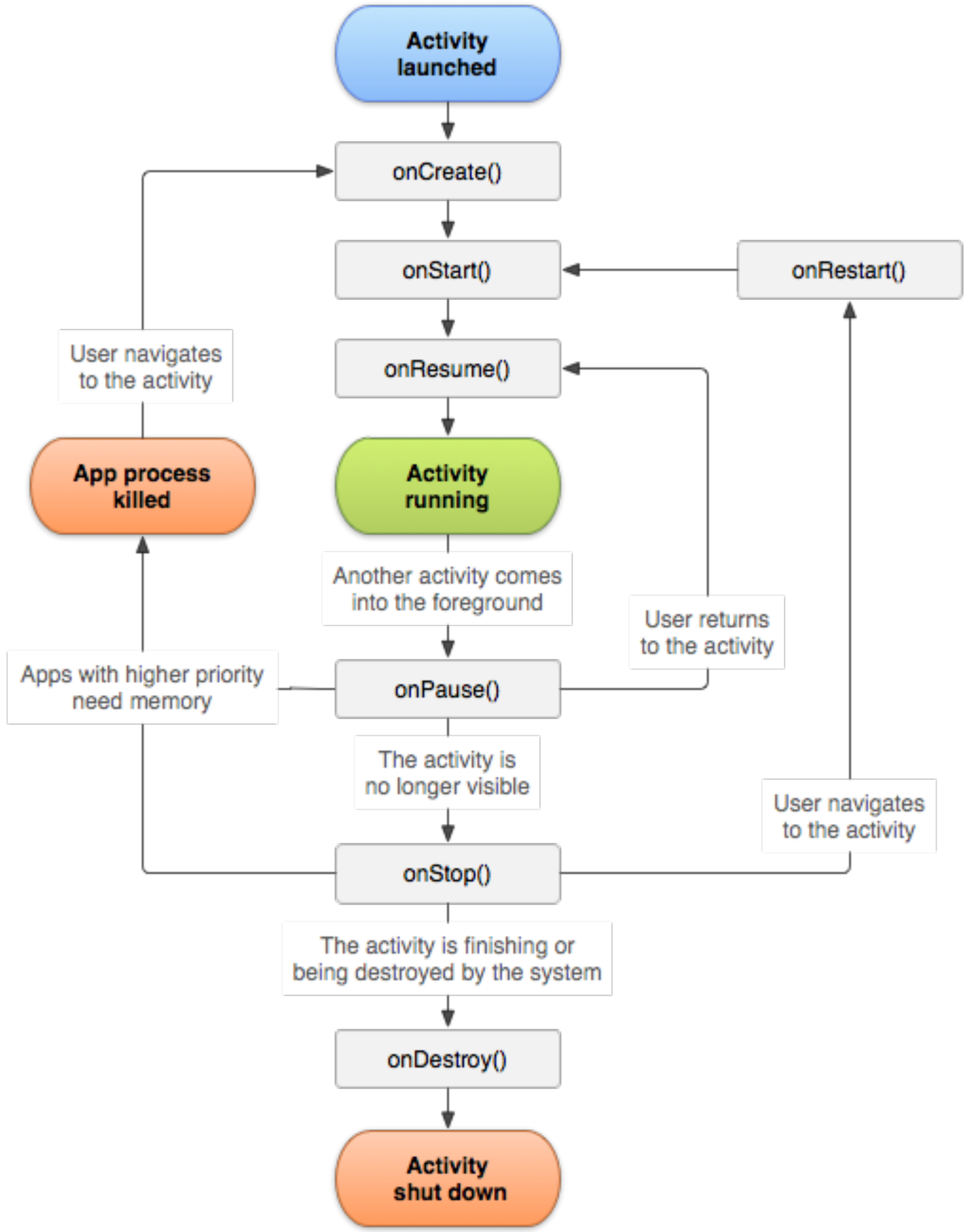
- An activity has four <u>states</u> in its life cycle.
  - **Running**: the activity is on the top of the screen and gained focus.
  - **Paused**: the activity is partially covered by other activities.
  - **Stopped**: the activity is completely covered by another running activity.
  - **Destroyed**.

- When running short of memory, a stopped activity is more likely to get killed than a paused/running activity.

Activity Life Cycle

- Each time the state of an activity is changed, its corresponding callback functions will be invoked.
- When an activity is started:
  - OnCreate -> onStart -> onResume.
- Loses focus and become invisible:
  - onPause -> onStop.
- Recaptured focus:
  - OnRestart -> onStart -> onResume.
- Closed:
  - onPause -> onStop -> onDestroy.

Activity Life Cycle (full version)


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 Life Cycle

- Good implementation of callback functions can make your app more robust and performant.
- Possible issues with a bad implementation:
  - Crashing if the user receives a phone call or switches to another app while using your app.
  - Consuming valuable system resources when the user is not actively using it.
  - Losing the user's progress if they leave your app and return to it at a later time.
  - Crashing or losing the user's progress when the screen rotates between landscape and portrait orientation.

CallBack Functions: Typical Uses

- onCreate(): Initial setup, load persistent state.
- onRestart(): read cached state
- onStart(): reset application
- onResume(): start foreground-only behaviors
- onPause(): shutdown foreground-only behaviors
- onStop(): cache state
- onDestroy(): save persistent state

*The above points are very general. Carefully design your app and keep the life cycle graph in mind.

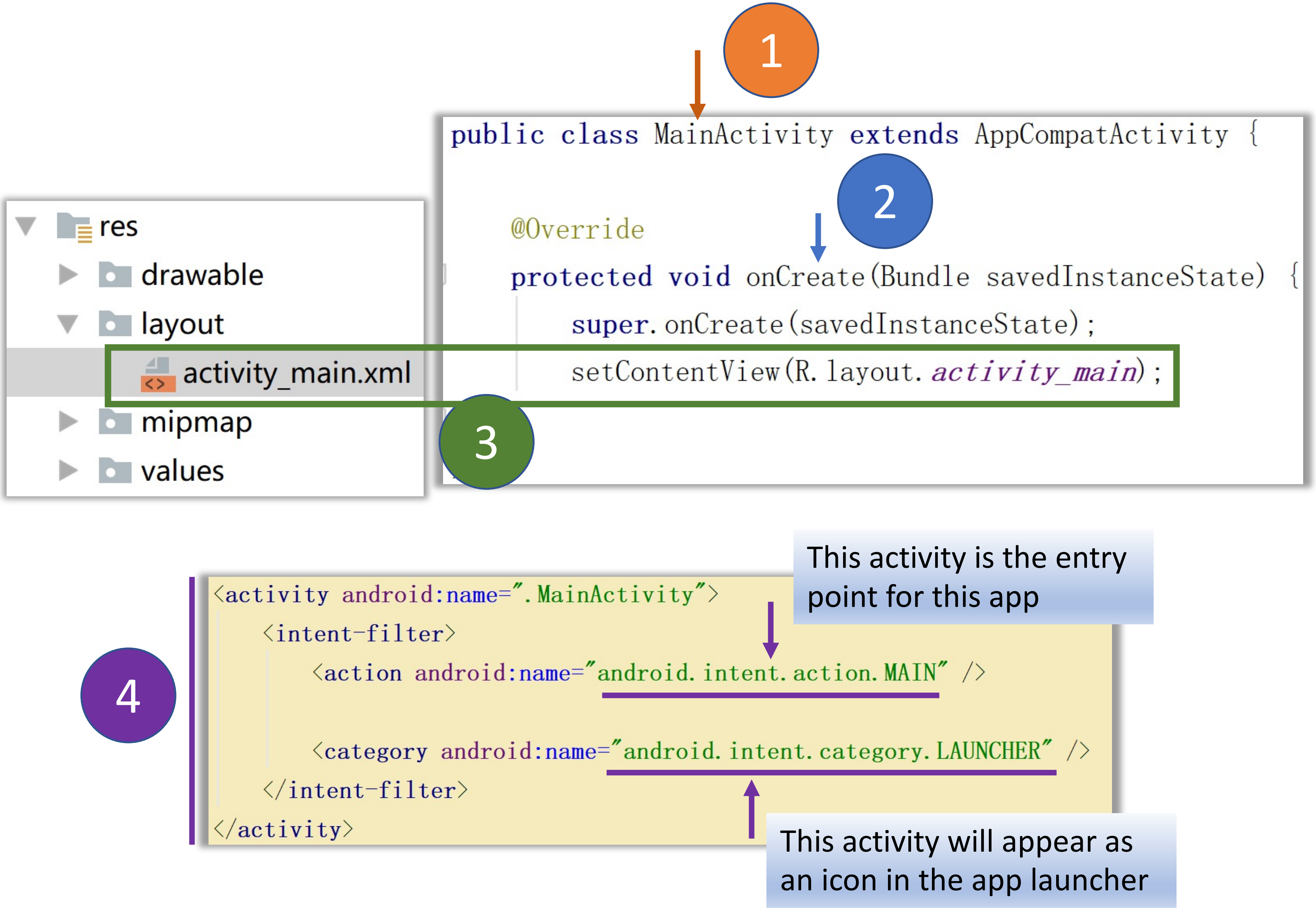
Creating Activities

1. Create a new `Activity` class. Which either inherits `Android.app.Activity` or its subclasses.

2. Override Activity.onCreate().

3. Create a layout XML file in `res/layout` and use `setContentView()` to load this layout.

4. Register the new activity in `AndroidManifest.xml`.
    1. If it is a main activity, you need to add a special `<intent-filter>` section in the manifest file.



```
res
  drawable
  layout
    activity_main.xml
  mipmap
  values
```

```java
public class MainActivity extends AppCompatActivity {

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

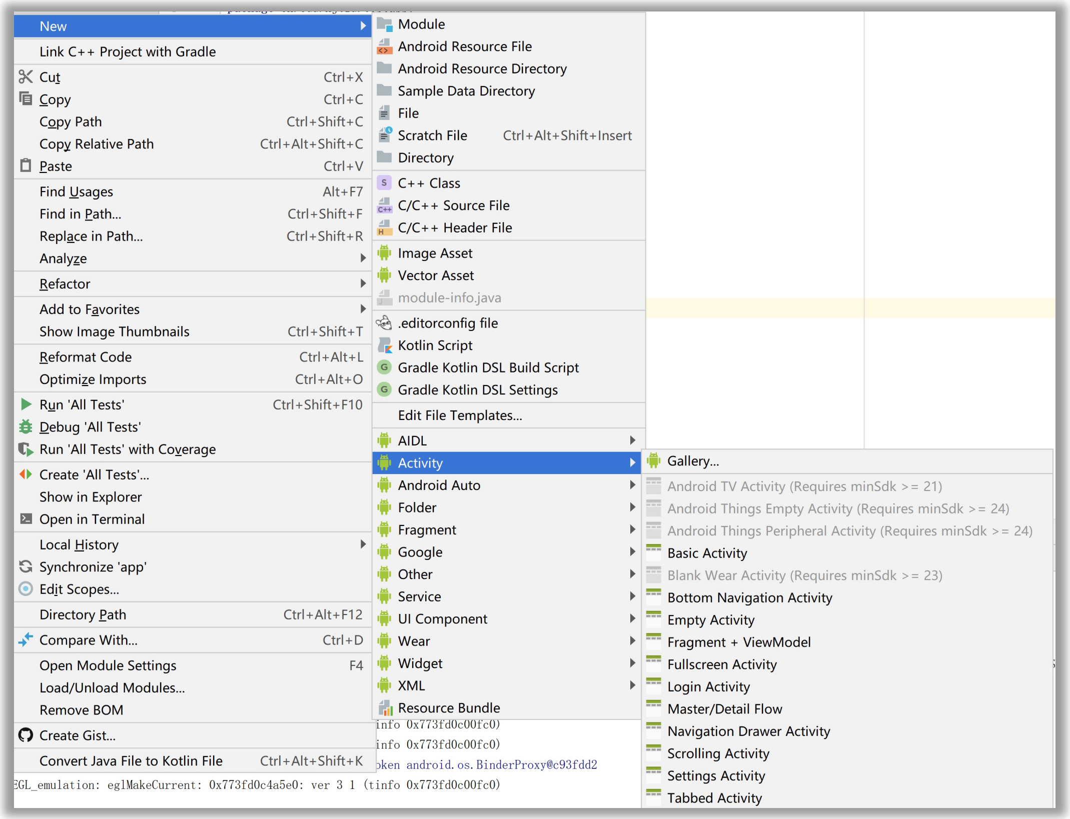
```xml
<activity android:name=".MainActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

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

This activity is the entry point for this app

This activity will appear as an icon in the app launcher

Create in Android Studio

- File->New->Activity
  - Gradle Sync must succeed first before you can see these options.



```
info 0x773fd0c00fc0)
info 0x773fd0c00fc0)
oken android.os.BinderProxy@c93fdd2
EGL_emulation: eglMakeCurrent: 0x773fd0c4a5e0: ver 3 1 (tinfo 0x773fd0c00fc0)
```

Starting Activities

- Activities can be started by calling the function `startActivity(Intent intent)`
  - To call Activity2 Inside an activity, do:

```
Intent intent = new Intent(this, Activity2.class);
startActivity(intent);
```

- The name of the target activity is not always explicitly specified. For instance, to let Android system choose an suitable activity for sending email (in Lecture 9):

```
Intent intent = new Intent(Intent.ACTION_SEND);
Intent.putExtra(Intent.EXTRA_EMAIL, recipientArray);
startActivity(intent);
```

Starting Activities with Return Val

- Sometimes we wish to obtain results from another activity. We need to start that activity using `startActivityForResult().`

- You must also implement the function `onActivityResult()` in your own activity in order to get the return result.

- Example: You want to start the People app in order for the user to select a contact and you'll receive the contact details as a result.

- To start the activity:

```
static final int PICK_CONTACT_REQUEST = 1;  // The request code
...
private void pickContact() {
    Intent pickContactIntent = new Intent(Intent.ACTION_PICK, Uri.parse("content://contacts"));
    pickContactIntent.setType(Phone.CONTENT_TYPE); // Show user only contacts w/ phone numbers
    startActivityForResult(pickContactIntent, PICK_CONTACT_REQUEST);
}
```

You may send requests to different activities, request code can help you identify different cases.

- To receive the result:

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    // Check which request we're responding to
    if (requestCode == PICK_CONTACT_REQUEST) {
        // Make sure the request was successful
        if (resultCode == RESULT_OK) {
            // The user picked a contact.
            // The Intent's data Uri identifies which contact was selected.

            // Do something with the contact here (bigger example below)
        }
    }
}
```

Returning Results

- To return the result in the activity that get called:

```
Intent data = new Intent();
String text = "Result to be returned....";
data.setData(Uri.parse(text));
setResult(RESULT_OK, data);
finish();
```

Closing Activities

- Android will automatically manage the life cycles of your activities.

- You can destroy the current activity manually by calling `finish()`.
- To finish an activity that you previously invoked with `startActivityForResult(Intent, int)`, use `finishActivity(int requestCode)`.

- Can be handy when you want to make sure that the user won’t return to this activity in the future.

Passing Data between Activities

- Method 1: `Intent`.

Activity 1:

```
Intent a = new Intent(this, Activity2.class);
a.putExtra("key", "value");
startActivity(a);
```

Activity 2:

```
Intent a = getIntent();
String val = a.getStringExtra("author");
```

- Method 2: `Bundle`.
- Method 3: `startActivityForResult()`.
  - Already discussed.

Passing Data between Activities

- Method 1: `Intent`.
- Method 2: `Bundle`.

Activity 1:

```
Intent a = new Intent(this, Activity2.class);
Bundle myBundle = new Bundle();
myBundle.putString("key", "value");
a.putExtras(myBundle);
startActivity(a);
```

Activity 2:

```
Intent a = getIntent();
Bundle myBundle = a.getExtras();
String val = myBundle.getString("author");
```

- Method 3: `startActivityForResult()`.
    - Already discussed.

Lab Session Task:

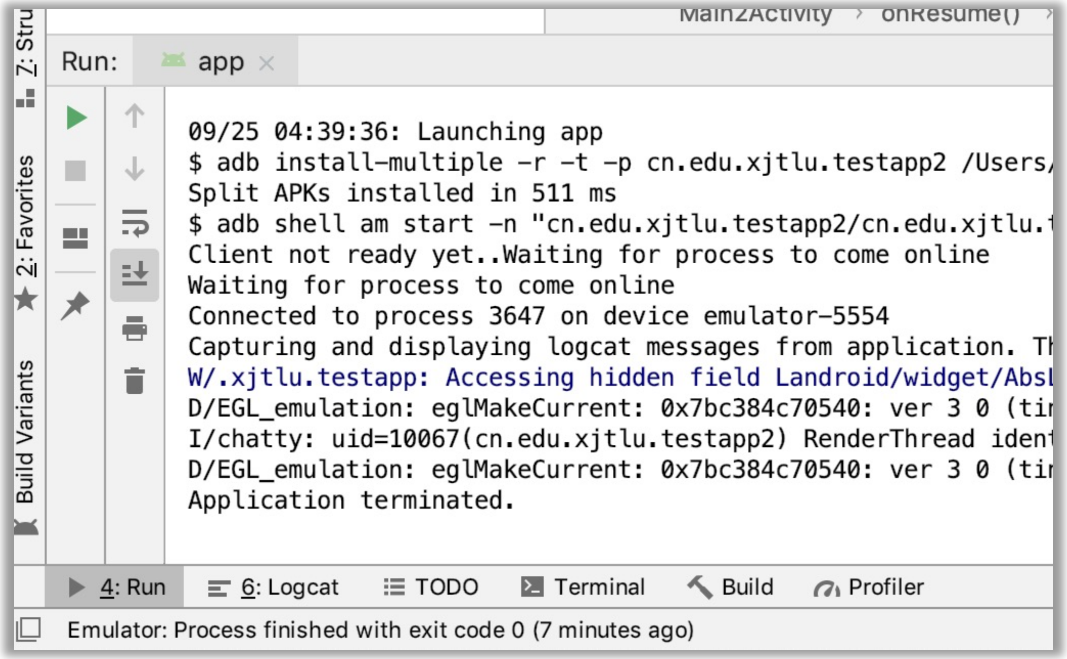
- Write a simple app that:
  - Has two activities with different UIs.
  - Shows the main activity with some greeting messages for 5 seconds.
  - Then automatically jumps to its second activity.

  - Now press “back”, does it still jump to the second activity?
  - What can be done to prevent this?

  - Implement onStop() and let it print out some messages.
  - Try to rotate your screen. What will happen?
  - Try to turn the screen on and off. What will happen?

* Console output

- Your console output (System.out) can be seen from the “run” window in Android Studio.



```
09/25 04:39:36: Launching app
$ adb install-multiple -r -t -p cn.edu.xjtlu.testapp2 /Users/
Split APKs installed in 511 ms
$ adb shell am start -n "cn.edu.xjtlu.testapp2/cn.edu.xjtlu.t
Client not ready yet..Waiting for process to come online
Waiting for process to come online
Connected to process 3647 on device emulator-5554
Capturing and displaying logcat messages from application. Th
W/.xjtlu.testapp: Accessing hidden field Landroid/widget/AbsL
D/EGL_emulation: eglMakeCurrent: 0x7bc384c70540: ver 3 0 (tin
I/chatty: uid=10067(cn.edu.xjtlu.testapp2) RenderThread ident
D/EGL_emulation: eglMakeCurrent: 0x7bc384c70540: ver 3 0 (tin
Application terminated.
```

Emulator: Process finished with exit code 0 (7 minutes ago)