FALL 2018 - NARGES NOROUZI

MOBILE APPLICATIONS

LECTURE 5

Resource: https://developers.google.com/training/android/

CONTENT

- Activities
- Defining an activity
- Starting a new activity with an intent
- Passing data between activities with extras
- Navigating between activities
- Activity lifecycle
- Activity lifecycle callbacks

ACTIVITY

WHAT IS AN ACTIVITY?

- An Activity is an application component
- Represents one window, one hierarchy of views
- Typically fills the screen, but can be embedded in other activity or a appear as floating window
- Java class, typically one activity in one file

WHAT DOES AN ACTIVITY DO?

- Represents an activity, such as ordering groceries, sending email, or getting directions
- Handles user interactions, such as button clicks, text entry, or login verification
- Can start other activities in the same or other apps
- Has a life cycle—is created, started, runs, is paused, resumed, stopped, and destroyed

APPS AND ACTIVITIES

- Activities are loosely tied together to make up an app
- First activity user sees is typically called "main activity"
- Activities can be organized in parent-child relationships in the Android manifest to aid navigation

LAYOUTS AND ACTIVITIES

- An activity typically has a UI layout
- Layout is usually defined in one or more XML files
- Activity "inflates" layout as part of being created

IMPLEMENTING ACTIVITIES

IMPLEMENT A NEW ACTIVITY

1. Define layout in XML

2. Define Activity Java class

   - extends AppCompatActivity

3. Connect Activity with Layout

   - Set content view in onCreate()

4. Declare Activity in the Android manifest

CONNECT ACTIVITY WITH YOUR LAYOUT

```
public class MainActivity extends AppCompatActivity {

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);

    }

}
```

DECLARE ACTIVITY IN YOUR ANDROID MANIFEST

```
<activity android:name=".MainActivity">
```

DECLARE MAIN ACTIVITY IN MANIFEST

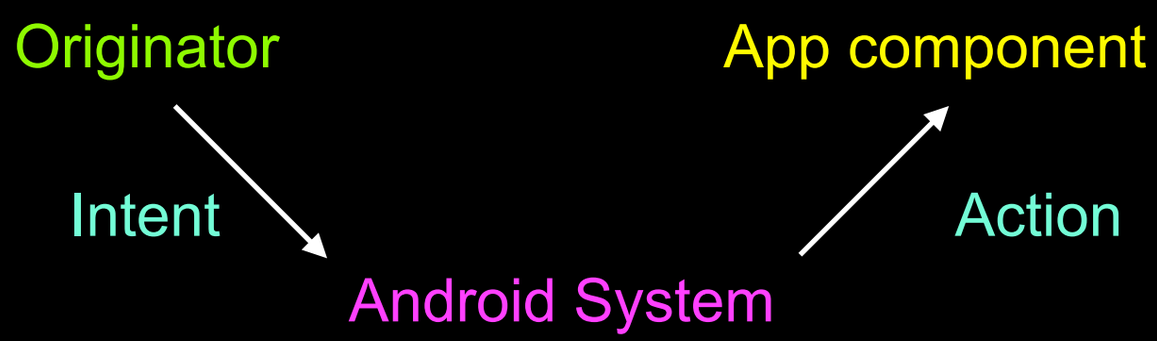
- Main Activity needs to include intent to start from launcher icon

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

INTENT

WHAT IS AN INTENT

- An intent is a description of an operation to be performed.
- An Intent is an object used to request an action from another app component via the Android system.



WHAT CAN INTENTS DO?

- Start activities
  - A button click starts a new activity for text entry
  - Clicking Share opens an app that allows you to post a photo
- Start services
  - Initiate downloading a file in the background
- Deliver broadcasts
  - The system informs everybody that the phone is now charging

EXPLICIT AND IMPLICIT INTENTS

Explicit Intent

- Starts a specific activity
  - Request tea with milk from a specific Starbucks branch
  - Main activity starts the ViewShoppingCart activity

Implicit Intent

- Asks system to find an activity that can handle this request
  - Find an open store that sells green tea
  - Clicking “Share” opens a chooser with a list of apps

STARTING ACTIVITIES

START AN ACTIVITY WITH EXPLICIT INTENT

To start a specific activity, use an explicit intent

1. Create an intent
   - `Intent intent = new Intent(this, ActivityName.class);`
2. Use the intent to start the activity
   - `startActivity(intent);`

START AN ACTIVITY WITH IMPLICIT INTENT

To ask Android to find an Activity to handle your request, use an implicit intent

1. Create an intent
   - `Intent intent = new Intent(action, uri);`
2. Use the intent to start the activity
   - `startActivity(intent);`

IMPLICIT INTENTS - EXAMPLES

Show a web page

```
Uri uri = Uri.parse("http://www.google.com");

Intent it = new Intent(Intent.ACTION_VIEW, uri);

startActivity(it);
```

Dial a phone number

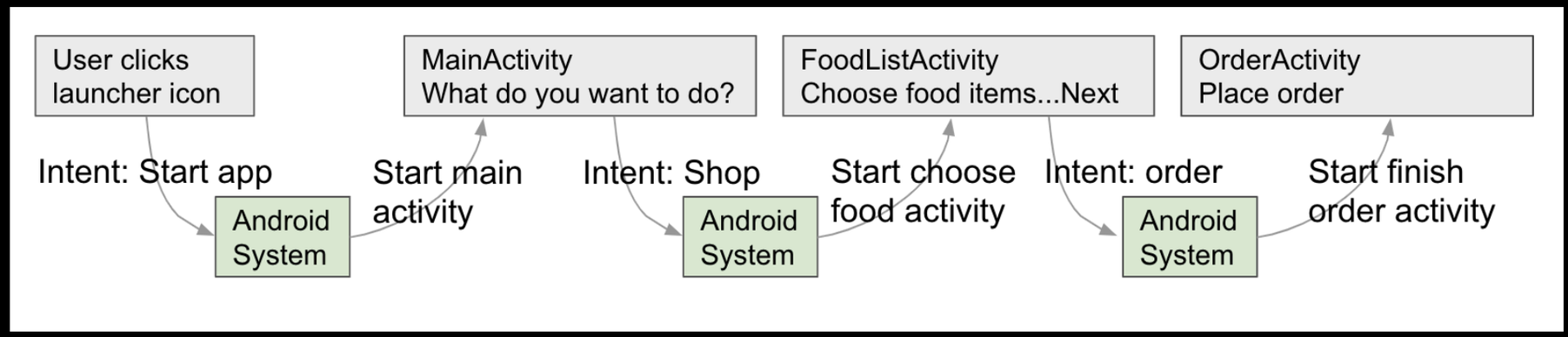
```
Uri uri = Uri.parse("tel:8005551234");

Intent it = new Intent(Intent.ACTION_DIAL, uri);

startActivity(it);
```

HOW ACTIVITIES RUN

- All activities are managed by the Android runtime
- Started by an "intent", a message to the Android runtime to run an activity



SENDING AND RECEIVING DATA

TWO TYPES OF SENDING DATA WITH INTENTS

- Data—one piece of information whose data location can be represented by an URI
- Extras—one or more pieces of information as a collection of key-value pairs in a Bundle

SENDING AND RETRIEVING DATA

In the first (sending) activity:

1. Create the Intent object
2. Put data or extras into that intent
3. Start the new activity with startActivity()

In the second (receiving) activity,:

1. Get the intent object the activity was started with
2. Retrieve the data or extras from the Intent object

PUTTING URI AS INTENT DATA

```
// A web page URL

intent.setData(

    Uri.parse("http://www.google.com"));

// a Sample file URI

intent.setData(

    Uri.fromFile(new File("/sdcard/
sample.jpg")));
```

PUT INFORMATION INTO EXTRAS

- putExtra(String name, int value)
  ⇒ `intent.putExtra("level", 406);`

- putExtra(String name, String[] value)
  ⇒
```
String[] foodList = {"Rice", "Fruit"};
intent.putExtra("food", foodList);
```

- putExtras(bundle);
  ⇒ if lots of data, first create a bundle and pass the bundle.

SENDING DATA TO ACTIVITIES WITH EXTRAS

```
String EXTRA_MESSAGE_KEY =
"com.example.android.twoactivities.extra.MESS
AGE";

Intent intent = new Intent(this,
SecondActivity.class);

String message = "Hello Activity!";

intent.putExtra(EXTRA_MESSAGE_KEY, message);

startActivity(intent);
```

GET DATA FROM INTENTS

- getData();
  ⇒ `Uri locationUri = intent.getData();`

- int getIntExtra (String name, int defaultValue)
  ⇒ `int level =`
  `intent.getIntExtra(“level", 0);`

- Bundle bundle = intent.getExtras();
  ⇒ Get all the data at once as a bundle.

RETURNING DATA TO STARTING ACTIVITY

1. Use startActivityForResult() to start the second activity

2. To return data from the second Activity:

    - Create a new Intent
    - Put the response data in the Intent using putExtra()
    - Set the result to Activity.RESULT_OK or RESULT_CANCELED, if the user cancelled out
    - call finish() to close the activity

3. Implement onActivityResult() in first activity

START ACTIVITY FOR RESULT

```
startActivityForResult(intent, requestCode);
```

- Starts activity (intent), assigns it identifier (requestCode)
- Returns data via intent extras
- When done, pop stack, return to previous activity, and execute onActivityResult() callback to process returned data
- Use requestCode to identify which activity has "returned"

1. STARTACTIVITYFORRESULT() EXAMPLE

```
int CHOOSE_FOOD_REQUEST = 1;
```

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

startActivityForResult(intent,
CHOOSE_FOOD_REQUEST);
```

2. RETURN DATA AND FINISH SECOND ACTIVITY

```
 // Create an intent

Intent replyIntent = new Intent();

// Put the data to return into the extra

replyIntent.putExtra(EXTRA_REPLY, reply);

// Set the activity's result to RESULT_OK

setResult(RESULT_OK, replyIntent);

// Finish the current activity

finish();
```

3. IMPLEMENT ONACTIVITYRESULT()

```
public void onActivityResult(int requestCode,
                             int resultCode, Intent data) {
  super.onActivityResult(requestCode, resultCode, data);
  if (requestCode == TEXT_REQUEST) { // Identify activity
    if (resultCode == RESULT_OK) { // Activity succeeded

      String reply =
data.getStringExtra(SecondActivity.EXTRA_REPLY);

      // … do something with the data

 }}}
```

DEMO