

Activities and Intents

Lesson 2



2.1 Activities and Intents



Contents

- Activities
- Defining an **Activity**
- Starting a new **Activity** with an **Intent**
- Passing data between activities with extras
- Navigating between activities



Activities (high-level view)

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 appear as floating window
- Java class, typically one Activity in one file

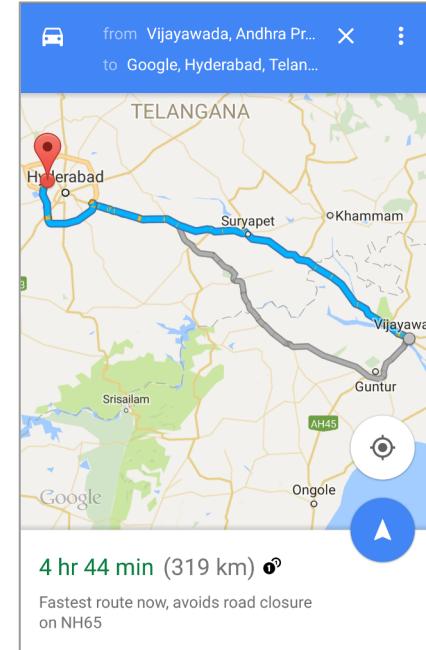
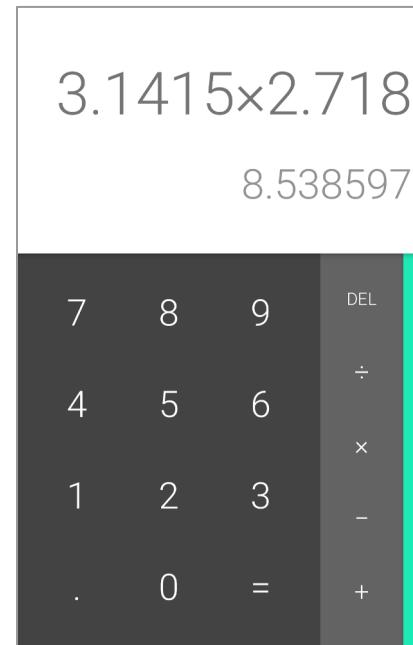


What does an Activity do?

- 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



Examples of activities



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 new activities

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

1. Define layout in XML

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Let's Shop for Food! " />
</RelativeLayout>
```



2. Define Activity Java class

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



3. Connect activity with layout

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

Resource is layout in this XML file



4. Declare activity in Android manifest

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

4. Declare main activity in manifest

MainActivity needs to include **intent-filter** to start from launcher

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

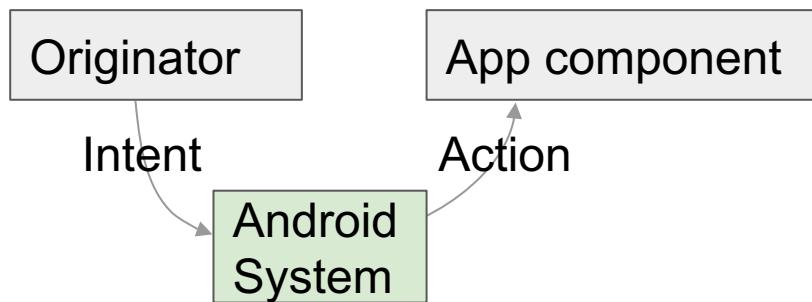


Intents

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

Explicit and implicit intents

Explicit Intent

- Starts a specific **Activity**
 - Request tea with milk delivered by Nikita
 - 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 an 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

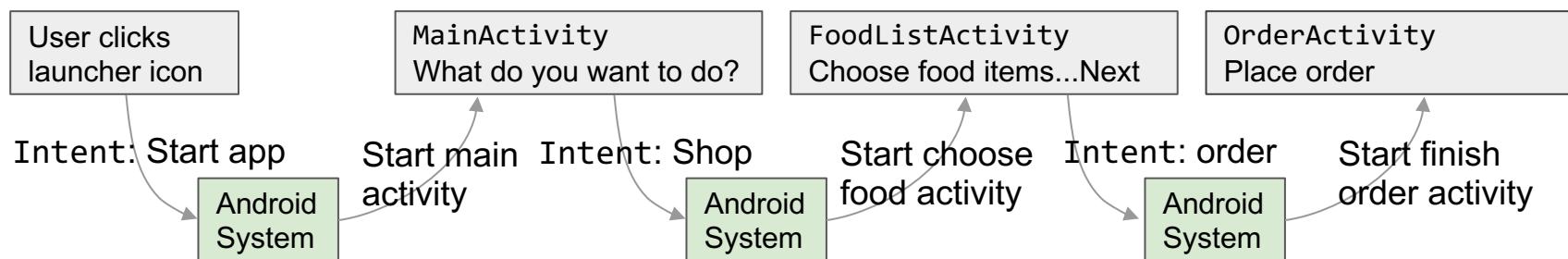
```
Uri uri = Uri.parse("http://www.google.com");  
Intent it = new Intent(Intent.ACTION_VIEW,uri);  
startActivity(it);
```

Dial a phone number

```
Uri uri = Uri.parse("tel:8005551234");  
Intent it = new Intent(Intent.ACTION_DIAL, uri);  
startActivity(it);
```

How Activities Run

- All **Activity** instances 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 a 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 intent extras

- `putExtra(String name, int value)`
⇒ `intent.putExtra("level", 406);`
- `putExtra(String name, String[] value)`
⇒ `String[] foodList = {"Rice", "Beans", "Fruit"};`
`intent.putExtra("food", foodList);`
- `putExtras(bundle);`
⇒ if lots of data, first create a bundle and pass the bundle.
- See [documentation](#) for all

Sending data to an activity with extras

```
public static final String EXTRA_MESSAGE_KEY =  
    "com.example.android.twoactivities.extra.MESSAGE";  
  
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.
- See [documentation](#) for all

Returning data to the 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**
1. Implement `onActivityResult()` in first **Activity**

startActivityForResult()

[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

```
public static final 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
        }
    }
}
```



Navigation

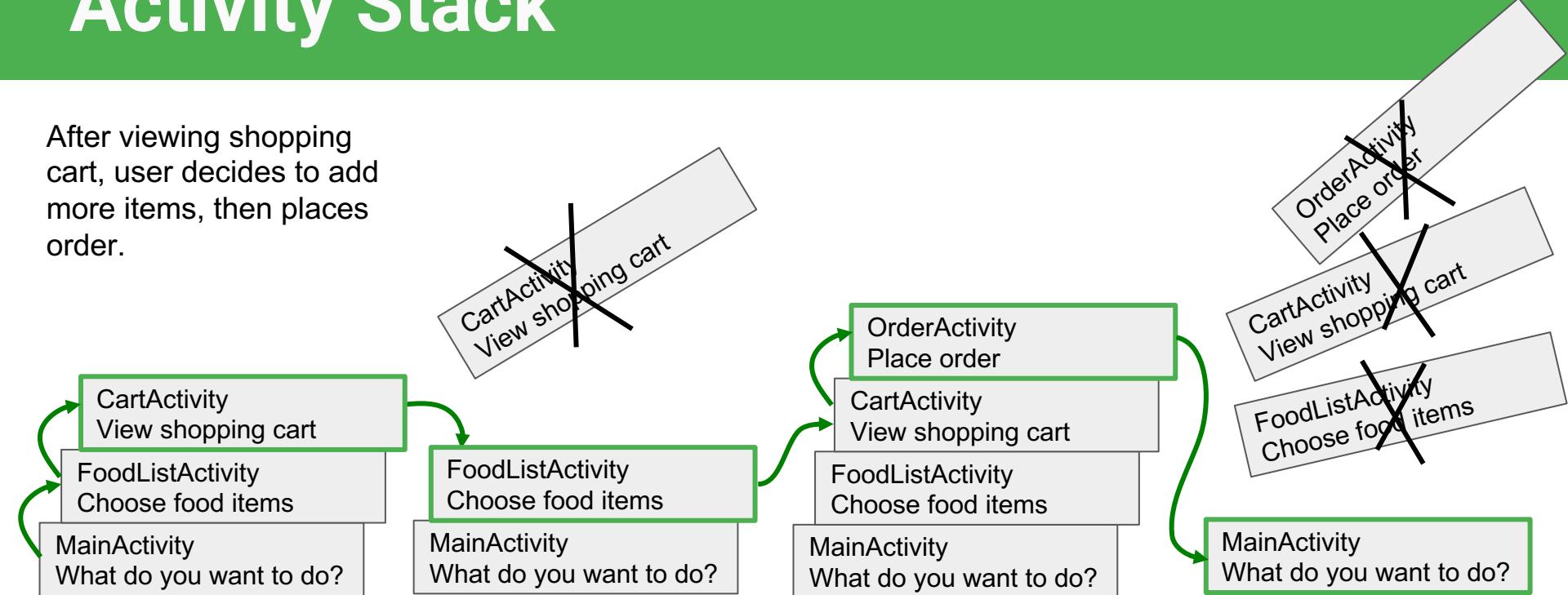
Activity stack

- When a new **Activity** is started, the previous **Activity** is stopped and pushed on the **Activity** back stack
- Last-in-first-out-stack—when the current **Activity** ends, or the user presses the Back button, it is popped from the stack and the previous **Activity** resumes



Activity Stack

After viewing shopping cart, user decides to add more items, then places order.



Two forms of navigation

◀ Temporal or back navigation

- provided by the device's Back button
- controlled by the Android system's back stack

⬅ Ancestral or up navigation

- provided by the Up button in app's action bar
- controlled by defining parent-child relationships between activities in the Android manifest



Back navigation

- Back stack preserves history of recently viewed screens
- Back stack contains all the **Activity** instances that have been launched by the user in reverse order *for the current task*
- Each task has its own back stack
- Switching between tasks activates that task's back stack



Up navigation

- Goes to parent of current **Activity**
- Define an **Activity** parent in Android manifest
- Set **parentActivityName**

```
<activity  
    android:name=".ShowDinnerActivity"  
    android:parentActivityName=".MainActivity" >  
</activity>
```



Learn more



Learn more

- [Android Application Fundamentals](#)
- [Starting Another Activity](#)
- [Activity](#) (API Guide)
- [Activity](#) (API Reference)
- [Intents and Intent Filters](#) (API Guide)
- [Intent](#) (API Reference)
- [Navigation](#)



What's Next?

- Concept Chapter: [2.1 Activities and Intents](#)
- Practical: [2.1 Activities and intents](#)



END

