COMP 2160
Mobile App Development I

MODULE 3 part 2: Intent

Handling Runtime Changes

Screen Orientation (FAQ)

Why did this code not work?

```
<activity android:name=".MyActivity"
          android:configChanges="orientation|keyboardHidden"
          android:label="@string/app_name">

@Override
public void onConfigurationChanged(Configuration newConfig) {
    super.onConfigurationChanged(newConfig);

    // Checks the orientation of the screen
    if (newConfig.orientation == Configuration.ORIENTATION_LANDSCAPE) {
        Toast.makeText(this, "landscape", Toast.LENGTH_SHORT).show();
    } else if (newConfig.orientation ==
Configuration.ORIENTATION_PORTRAIT){
        Toast.makeText(this, "portrait", Toast.LENGTH_SHORT).show();
    }
}
```

Answer: https://stackoverflow.com/questions/6968105/orientation-change-in-honeycomb

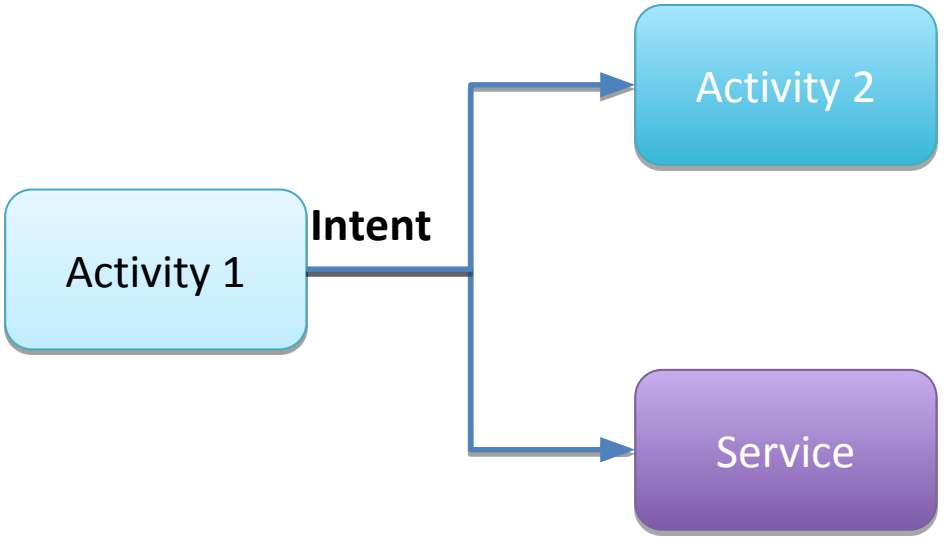
Intent and Intent Filters

Common Intents

Intents

Use Cases

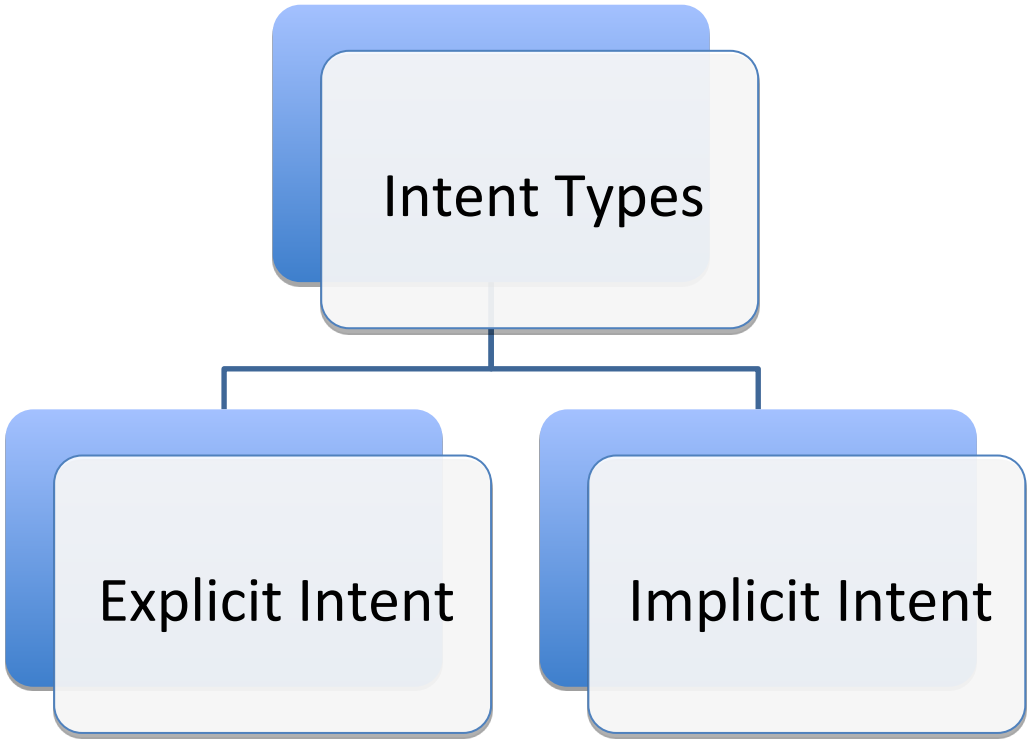
- An Intent is a messaging object you can use to request an action from another app component.
- **Use Cases**
  - **To start an activity**
  - **To start a service**
  - **To deliver a broadcast**



Ref: https://developer.android.com/guide/components/intents-filters.html

Intents

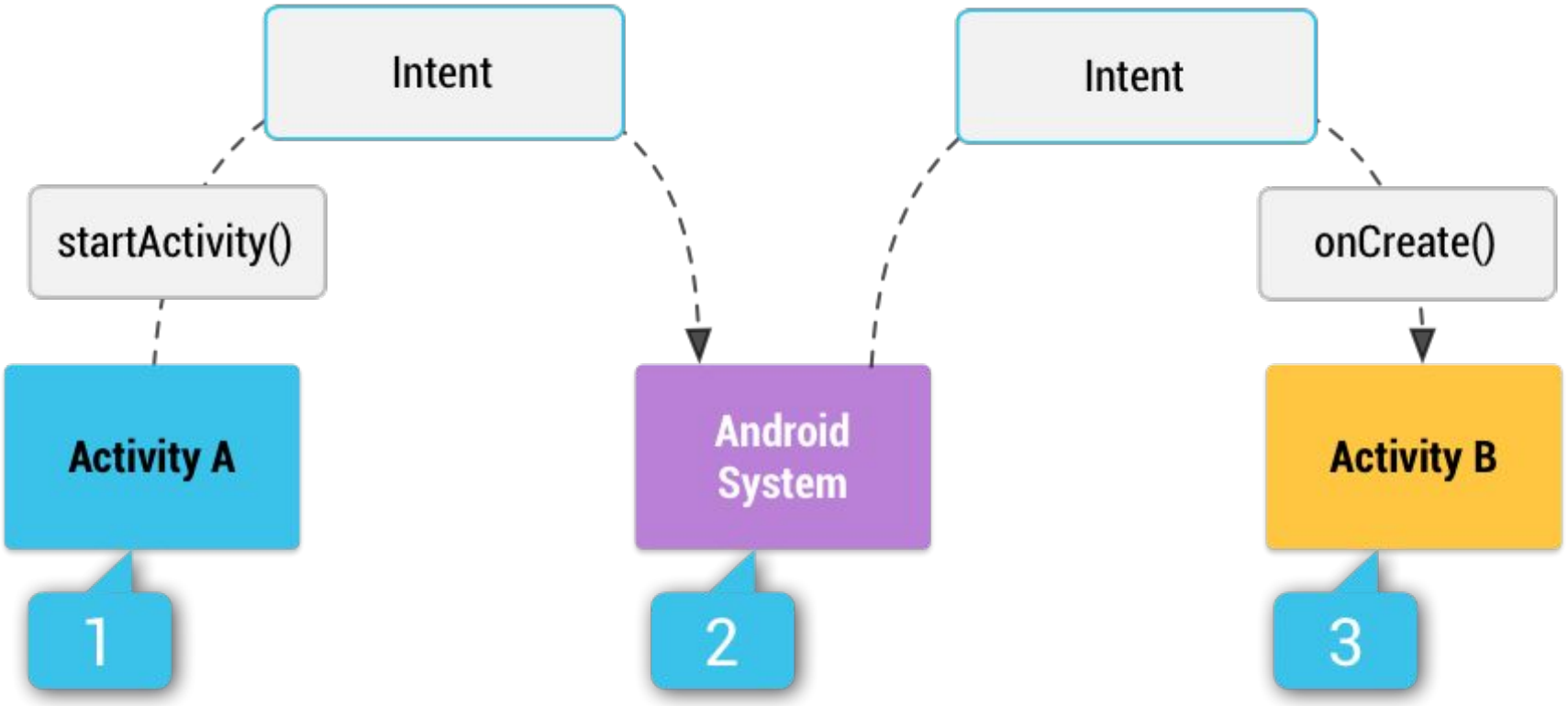
Types



Ref: https://developer.android.com/guide/components/intents-filters.html

Intents

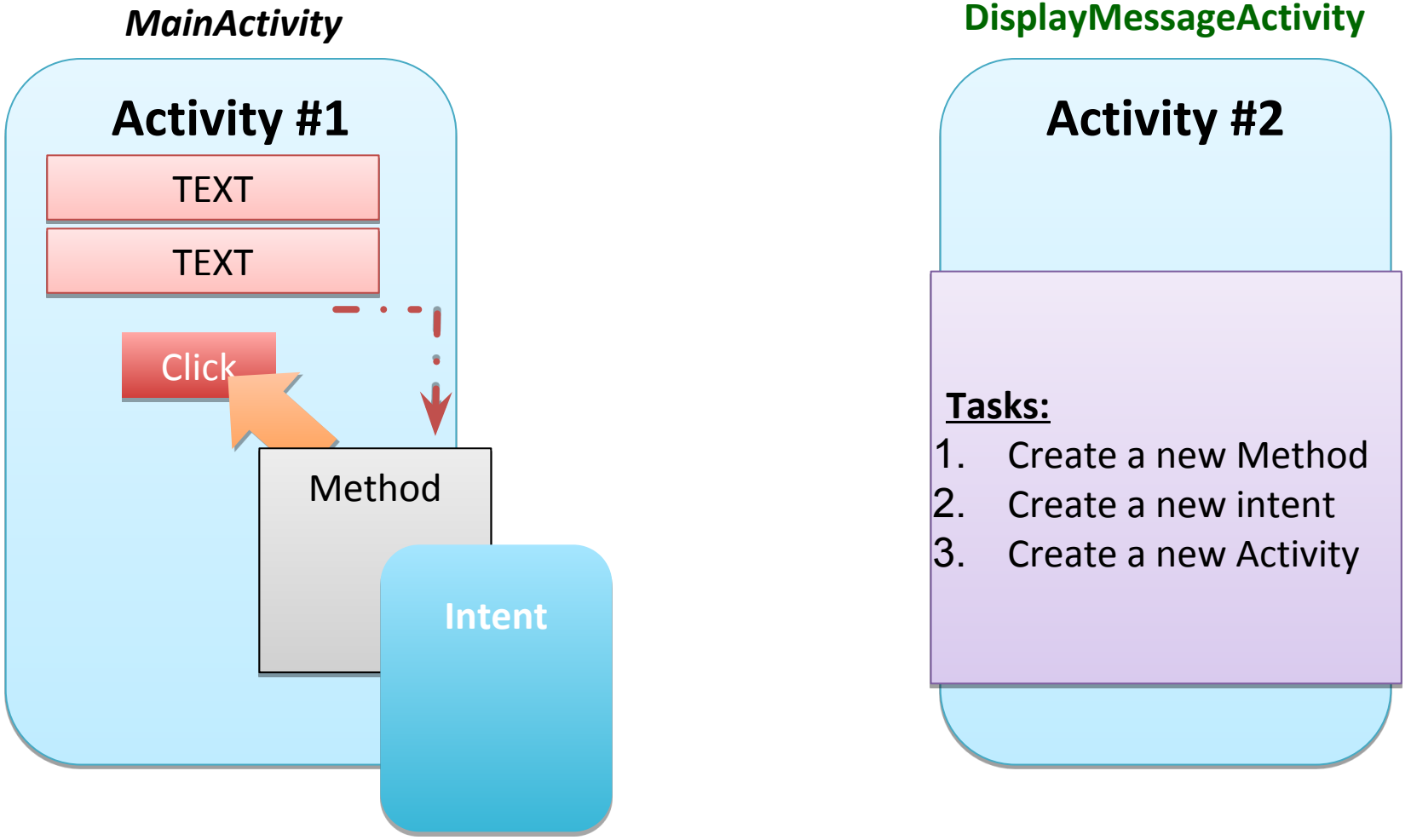
Implicit Intent



Ref: https://developer.android.com/guide/components/intents-filters.html

Intents

Building an Intent

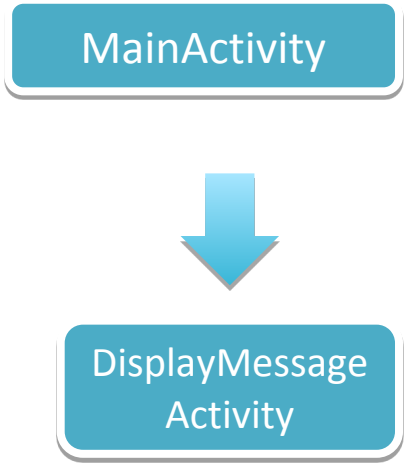


Ref: https://developer.android.com/guide/components/intents-filters.html

Intents

Building an Intent to start an activity called DisplayMessageActivity

```
public class MainActivity extends AppCompatActivity {
    public final static String EXTRA_MESSAGE = "com.example.myfirstapp.MESSAGE";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    /** Called when the user clicks the Send button */
    public void sendMessage(View view) {
        Intent intent = new Intent(this, DisplayMessageActivity.class);
        EditText editText = (EditText) findViewById(R.id.edit_message);
        String message = editText.getText().toString();
        intent.putExtra(EXTRA_MESSAGE, message);
        startActivity(intent);
    }
}
```



* It's a good practice to define keys for intent extras using your app's package name as a prefix

Ref: https://developer.android.com/training/basics/firstapp/starting-activity.html

Intent Filters

An Example

An intent filter is an expression in an app's manifest file that specifies the type of intents that the component would like to receive.

```
<activity android:name="ShareActivity">
   <intent-filter>
      <action android:name="android.intent.action.SEND"/>
      <category android:name="android.intent.category.DEFAULT"/>
      <data android:mimeType="text/plain"/>
   </intent-filter>
</activity>
```

An activity declaration with an intent filter to receive an ACTION_SEND intent when the data type is text

Ref: https://developer.android.com/guide/components/intents-filters.html

Intents and Intent Filters

Intents

Activity 2

Which intent type will be typically used to start a component in its own app? Explicit or Implicit?

You'll typically use an explicit intent to start a component in your own app, because you know the class name of the activity or service you want to start. For example, start a new activity in response to a user action or start a service to download a file in the background.

Common Intents

Actions

Task	Action
Create an alarm	ACTION_SET_ALARM
Create a timer	ACTION_SET_TIMER
Show all alarms	ACTION_SHOW_ALARMS
Add a calendar event	ACTION_INSERT
Capture a picture or video and return it	ACTION_IMAGE_CAPTURE or ACTION_VIDEO_CAPTURE
Start a camera app in still image mode	INTENT_ACTION_STILL_IMAGE_CAMERA
Start a camera app in video mode	INTENT_ACTION_VIDEO_CAMERA

Example: capturing a photo using a camera app

```
public void capturePhoto() {
        Intent intent = new Intent(MediaStore.INTENT_ACTION_STILL_IMAGE_CAMERA);
        if (intent.resolveActivity(getPackageManager()) != null) {
                startActivityForResult(intent);
        }
}
```

https://developer.android.com/training/camera/photobasics.html#TaskCaptureIntent

Common Intents

Handling Intents

Activity 3

When you call startActivity() or startActivityForResult() and pass it an implicit intent, the system resolves the intent to an app that can handle the intent and starts its corresponding Activity.
What happens if there's more than one app that can handle the intent?

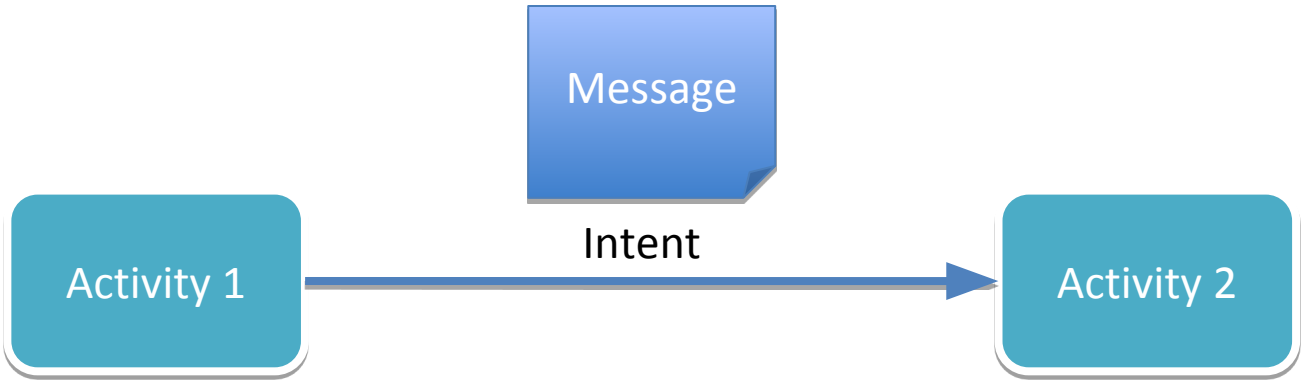
If there's more than one app that can handle the intent, the system presents the user with a dialog to pick which app to use.

Sharing data using Intents

Sharing Data

put/get Extra

- An intent not only allows you to start another activity, but it can carry a bundle of data to the activity as well.



Ref: https://developer.android.com/training/basics/firstapp/starting-activity.html

Sharing Data

Building an Intent to start an activity called DisplayMessageActivity and sending a message to it.

```
public class MainActivity extends AppCompatActivity {
    public final static String EXTRA_MESSAGE = "com.example.myfirstapp.MESSAGE";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    /** Called when the user clicks the Send button */
    public void sendMessage(View view) {
        Intent intent = new Intent(this, DisplayMessageActivity.class);
        EditText editText = (EditText) findViewById(R.id.edit_message);
        String message = editText.getText().toString();
        intent.putExtra(EXTRA_MESSAGE, message);
        startActivity(intent);
    }
}
```

Ref: https://developer.android.com/training/basics/firstapp/starting-activity.html

Sharing Data

Receiving the Intent and Displaying the Message

onCreate() *method for* ***DisplayMessage*** *should look like this:*

```
TextView TextView10;
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

        TextView10 = (TextView)findViewById(R.id.TextView10);


        Intent intent02 = getIntent();
        String message = intent02.getStringExtra(
        MainActivity.EXTRA_MESSAGE);
        TextView10.setText(message);

}
```

Ref: https://developer.android.com/training/basics/firstapp/starting-activity.html

Sharing Data

Receiving an Intent

In this code, the activity receives a message from an intent and display it in a textview. The textview is created dynamically and its text size property is set to 40.

Activity 4

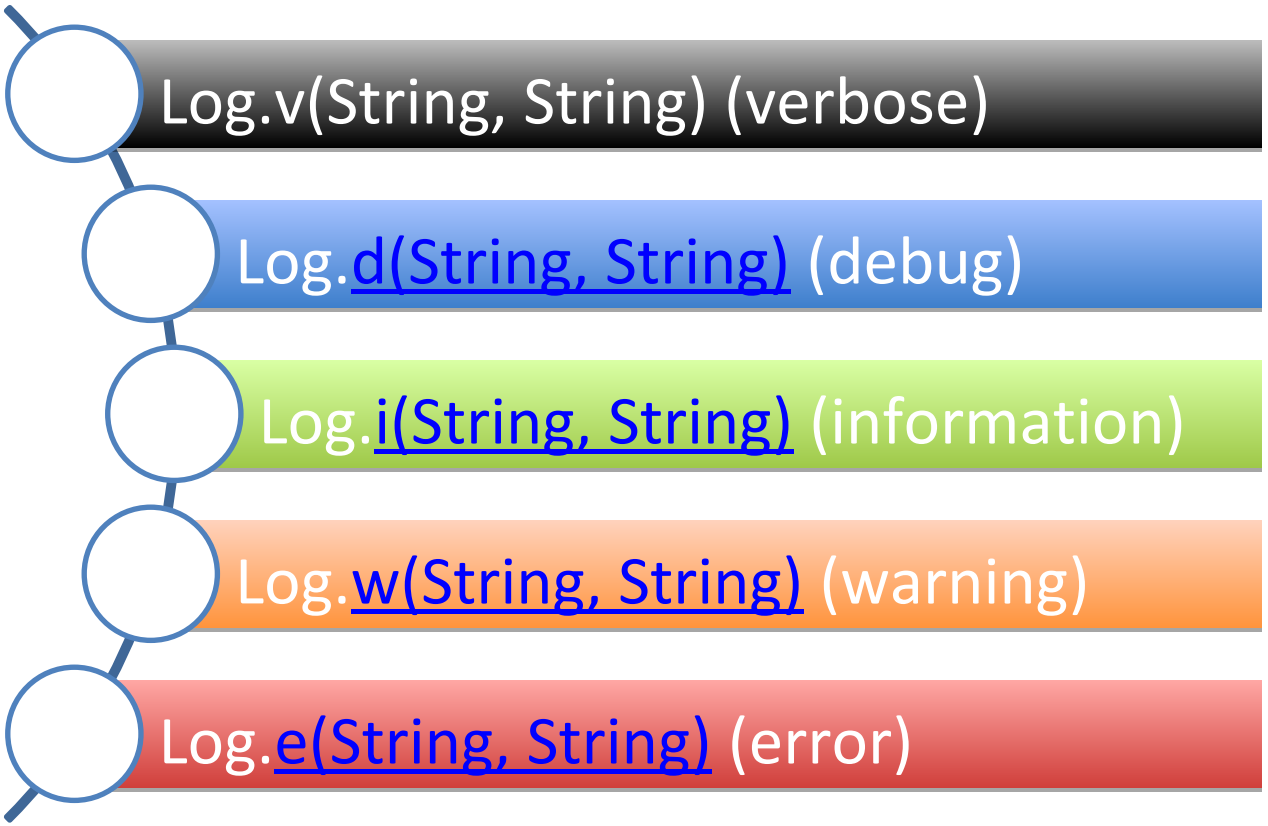
Explain what happens when this code is executed.

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

  Intent intent = getIntent();
  String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
  TextView textView = new TextView(this);
  textView.setTextSize(40);
  textView.setText(message);

  ViewGroup layout = (ViewGroup) findViewById(R.id.activity_display_message);
  layout.addView(textView);
}
```

Using the Log class to track the order of execution

Log Class

Options



http://developer.android.com/tools/debugging/debugging-log.html

Log Class

Example

The order in terms of verbosity, from least to most is ERROR, WARN, INFO, DEBUG, VERBOSE.

```
private static final String TAG = "MyActivity";
```

```
Log.v(TAG, "index=" + i);
```

http://developer.android.com/tools/debugging/debugging-log.html

Log Class

Logcat

Activity 5

Logcat is a command-line tool that dumps a log of system messages, including stack traces when the device throws an error and messages that you have written from your app with the Log class.

What should you do to run logcat as an adb command or directly in a shell prompt of your emulator or connected device?

Logcat output is available from within inside android studio. However, to have more control, you may want to run logcat in another adb shell. You can find your device or emulator id using adb and then create a shell connection to the device or emulator:

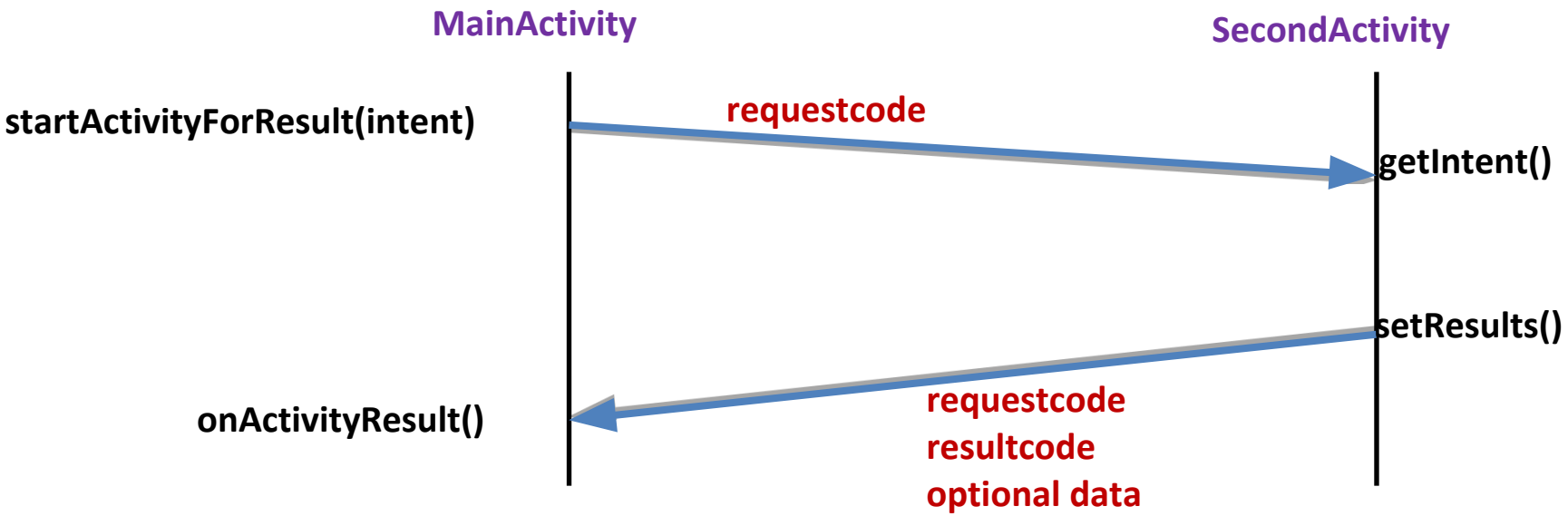
C:\Android\sdk\platform-tools>adb devices
$ adb shell <device>
# logcat

Managing Multiple Activities

Managing Multiple Activities

Getting a Result from an Activity

- Starting another activity doesn't have to be one-way.
- You can also start another activity and receive a result back.
- To receive a result, call **startActivityForResult()** (instead of **startActivity()**).



https://developer.android.com/training/basics/intents/result.html

Managing Multiple Activities

Start the Activity

Starting an activity that allows the user to pick a contact

```
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);
}
```

https://developer.android.com/training/basics/intents/result.html

Managing Multiple Activities

Receive the Result

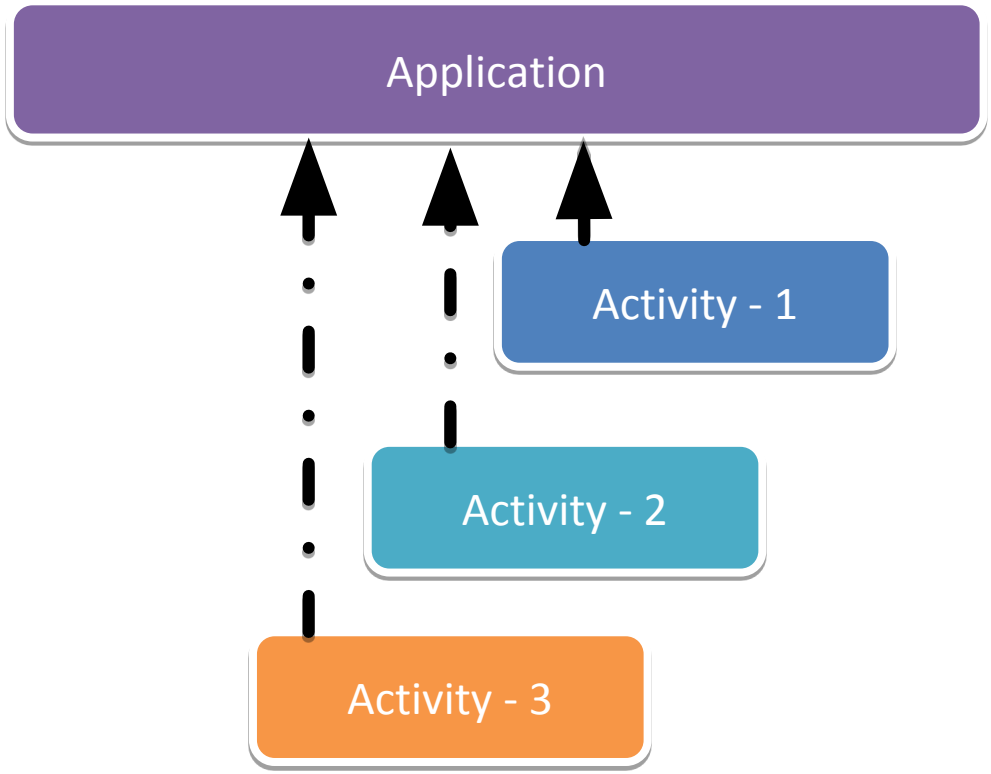
```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    // Check which request we're responding to
    if (requestCode == PICK_CONTACT_REQUEST) {
        // Make sure the request was successful
        if (resultCode == RESULT_OK) {
            // The user picked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

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

https://developer.android.com/training/basics/intents/result.html

Managing Multiple Activities

Activity lifecycle

- Remember that Every Activity has a Life Cycle
- Remember that Only one activity can run in the foreground at one time. The rest are paused or stopped



Managing Multiple Activities

Main Activity

- when we have an application with multiple activities, we need to define the “main” or entry-point activity.

- In the Android framework, the manifest file provides the information about all of the activities that make up an application as well as define the activity that will serve as the entry point.

The main activity for your app must be declared in the manifest with

- an **<intent-filter>** that includes the **MAIN** action and **LAUNCHER** category

```
<activity
        android:name="tru.comp2160.lab1.HelloWorldActivity"
        android:label="@string/app_name" >
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
```

Study Code

Reading and understanding professionally written code is a great way to learn good programming practices

https://developer.android.com/samples/

Android Studio, **select File > Import Sample**

Log Class

Managing Activities

Activity 6

When the system calls onPause() for your activity, it technically means your activity is still partially visible, but most often is an indication that the user is leaving the activity and it will soon enter the Stopped state.
How should a developer use the onPause() callback?

You should usually use the onPause() callback to:
Check if the activity is visible; if it is not, stop animations or other ongoing actions that could consume CPU. Remember, beginning with Android 7.0, a paused app might be running in multi-window mode. In this case, you would not want to stop animations or video playback.
Commit unsaved changes, but only if users expect such changes to be permanently saved when they leave (such as a draft email).
Release system resources, such as broadcast receivers, handles to sensors (like GPS), or any resources that may affect battery life while your activity is paused and the user does not need them.

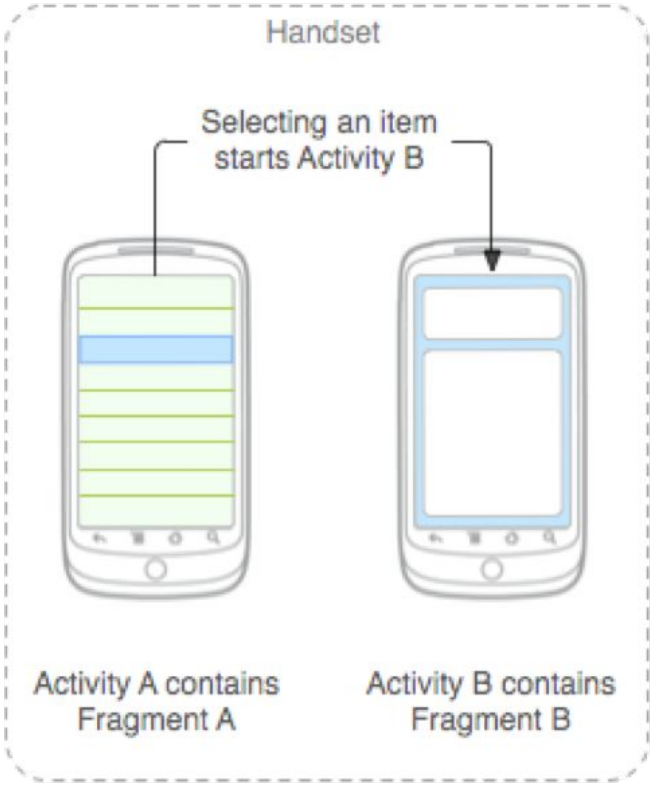
The life cycles of a Fragment

The life cycles of a Fragment

Introduction

You can think of a fragment as a modular section of an activity, which has its own lifecycle, receives its own input events, and which you can add or remove while the activity is running

You can combine multiple fragments in a single activity to build a multi-pane UI and reuse a fragment in multiple activities.

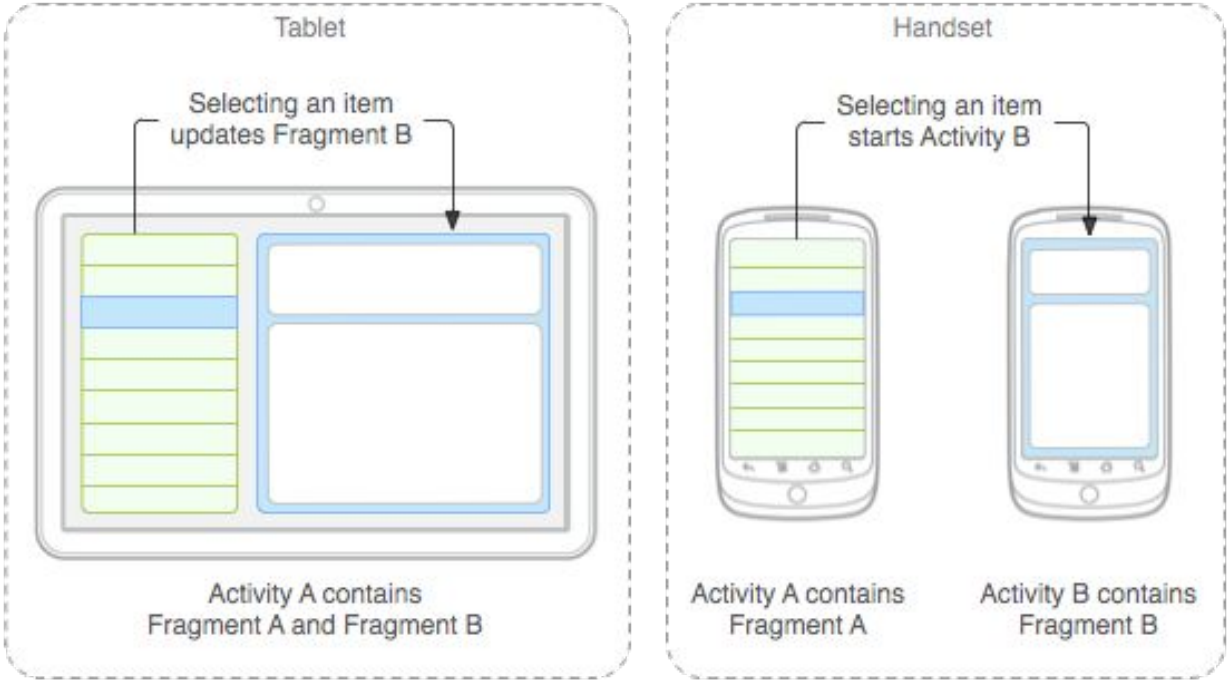


Reference: https://developer.android.com/guide/components/fragments.html

The life cycles of a Fragment

Introduction

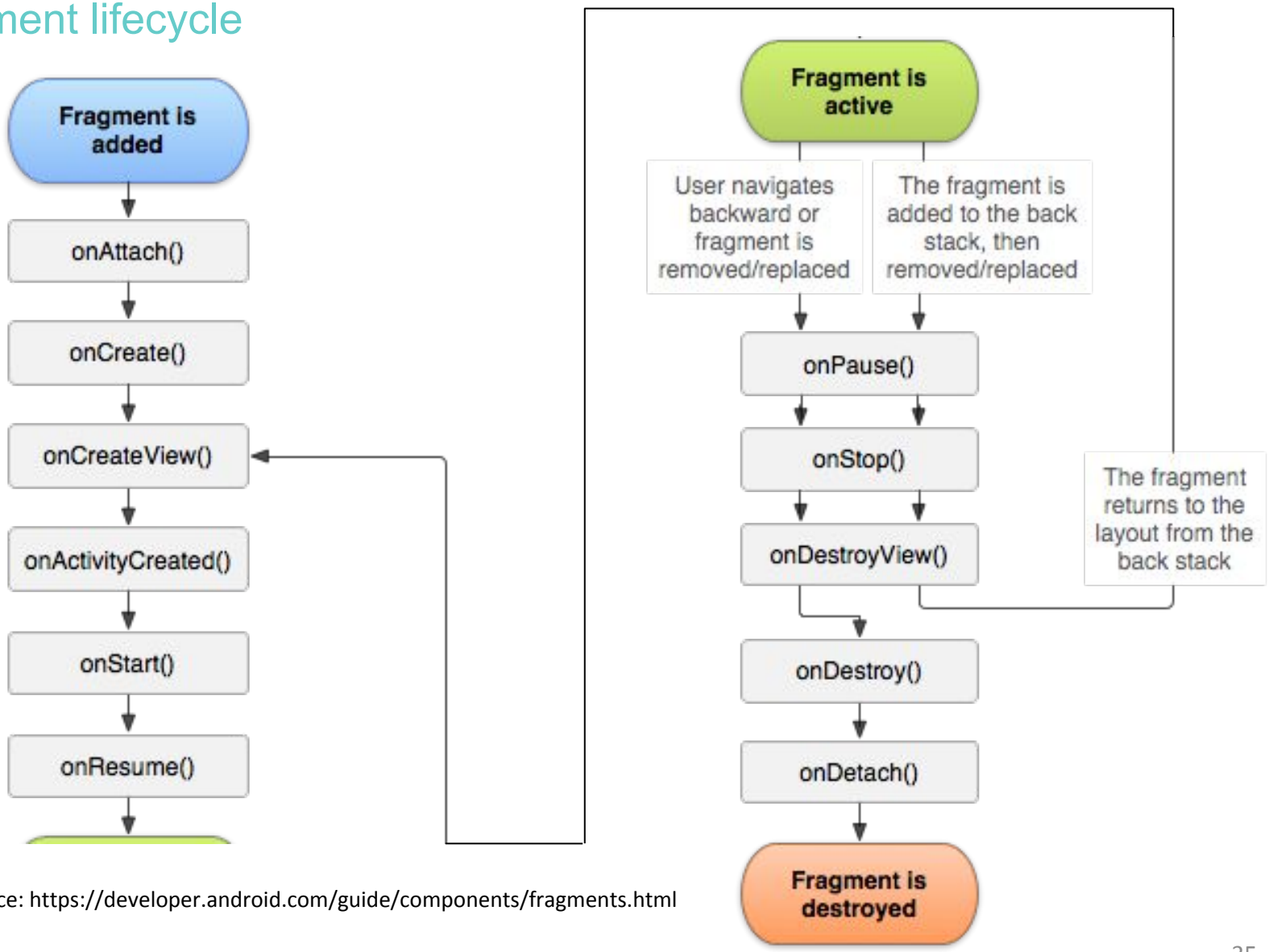
- You should design each fragment as a modular and reusable activity component.
- This is especially important because a modular fragment allows you to change your fragment combinations for different screen sizes.



Reference: https://developer.android.com/guide/components/fragments.html

The life cycles of a Fragment

Fragment lifecycle



Reference: https://developer.android.com/guide/components/fragments.html

The life cycles of a Fragment

Fragment Common SubClasses

- DialogFragment
  - Displays a floating dialog.
  - Using this class to create a dialog is a good alternative to using the dialog helper methods in the Activity class

- ListFragment
  - Displays a list of items that are managed by an adapter
  - Provides several methods for managing a list view

- PreferenceFragment
  - Displays a hierarchy of Preference objects as a list
  - Useful when creating a "settings" activity for your application.
  -

Reference: https://developer.android.com/guide/components/fragments.html

The life cycles of a Fragment

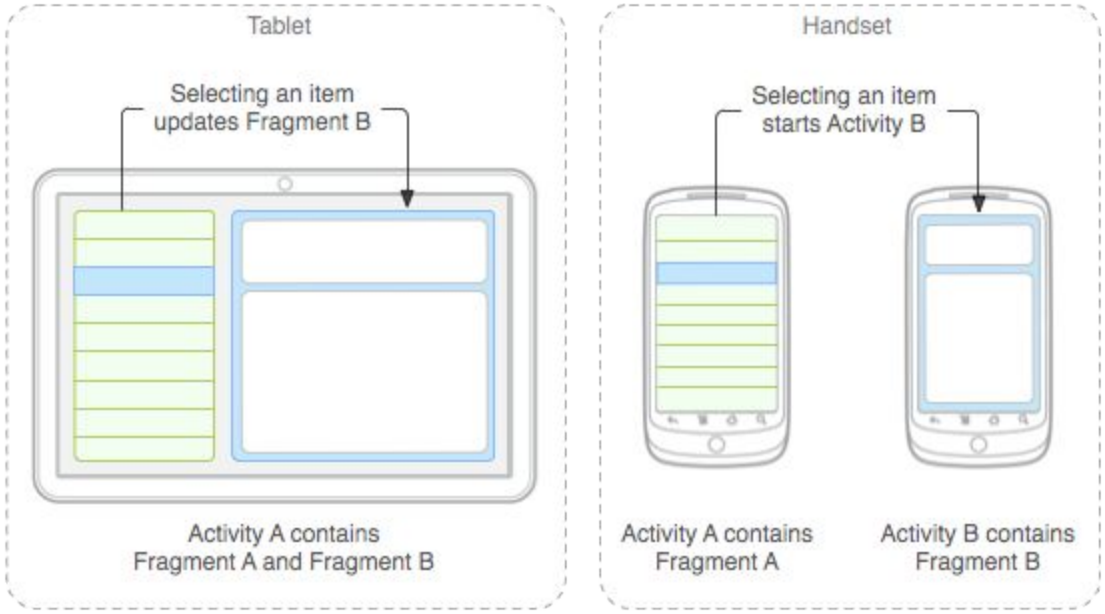
Creating Fragments

Activity 7

Most applications should implement at least three methods for every fragment. What are these three methods?

onCreate()
onCreateView()
onPause()

Managing Fragments


Tablet
Selecting an item
updates Fragment B
Activity A contains
Fragment A and Fragment B
Handset
Selecting an item
starts Activity B
Activity A contains
Fragment A
Activity B contains
Fragment B

Managing Fragments

Creating Fragments

There are two ways you can add a fragment to the activity layout:

1. Declare the fragment inside the activity's layout file.
2. Programmatically add the fragment to an existing ViewGroup.

Reference: https://developer.android.com/guide/components/fragments.html

Managing Fragments

Creating Fragments - Static

Declare the fragment inside the activity's layout file

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="horizontal"
   android:layout_width="match_parent"
   android:layout_height="match_parent">
   <fragment android:name="com.example.news.ArticleListFragment"
         android:id="@+id/list"
         android:layout_weight="1"
         android:layout_width="0dp"
         android:layout_height="match_parent" />
   <fragment android:name="com.example.news.ArticleReaderFragment"
         android:id="@+id/viewer"
         android:layout_weight="2"
         android:layout_width="0dp"
         android:layout_height="match_parent" />
</LinearLayout>
```

Reference: https://developer.android.com/guide/components/fragments.html

Managing Fragments

Creating Fragments - Dynamic

Programmatically add the fragment to an existing ViewGroup

```
FragmentManager fragmentManager = getFragmentManager();
FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();

ExampleFragment fragment = new ExampleFragment();
fragmentTransaction.add(R.id.fragment_container, fragment);
fragmentTransaction.commit();
```

Reference: https://developer.android.com/guide/components/fragments.html

Managing Fragments

FragmentManager

To manage the fragments in your activity, you need to use FragmentManager.

To get it, call getFragmentManager() from your activity.

Some things that you can do with FragmentManager include:

- Get fragments that exist in the activity
- Pop fragments off the back stack
- Register a listener for changes to the back stack

Reference: https://developer.android.com/guide/components/fragments.htm

Managing Fragments

Adding a fragment without a UI

Activity 8

Is it possible to use a fragment to provide a background behavior for an activity without presenting additional UI?

Yes. You can use a fragment to provide a background behavior for the activity without presenting additional UI.

Communicating with Activities

Communicating with Activities

Access an Activity instance/Call methods in Fragments

A fragment can access an Activity instance with getActivity() and easily perform tasks such as find a view in the activity layout:

```
View listView = getActivity().findViewById(R.id.list);
```

An activity can call methods in a fragment by acquiring a reference to the Fragment from FragmentManager, using findFragmentById()or findFragmentByTag().

```
ExampleFragment fragment = (ExampleFragment)
getFragmentManager().findFragmentById(R.id.example_fragment);
```

Reference: https://developer.android.com/guide/components/fragments.html

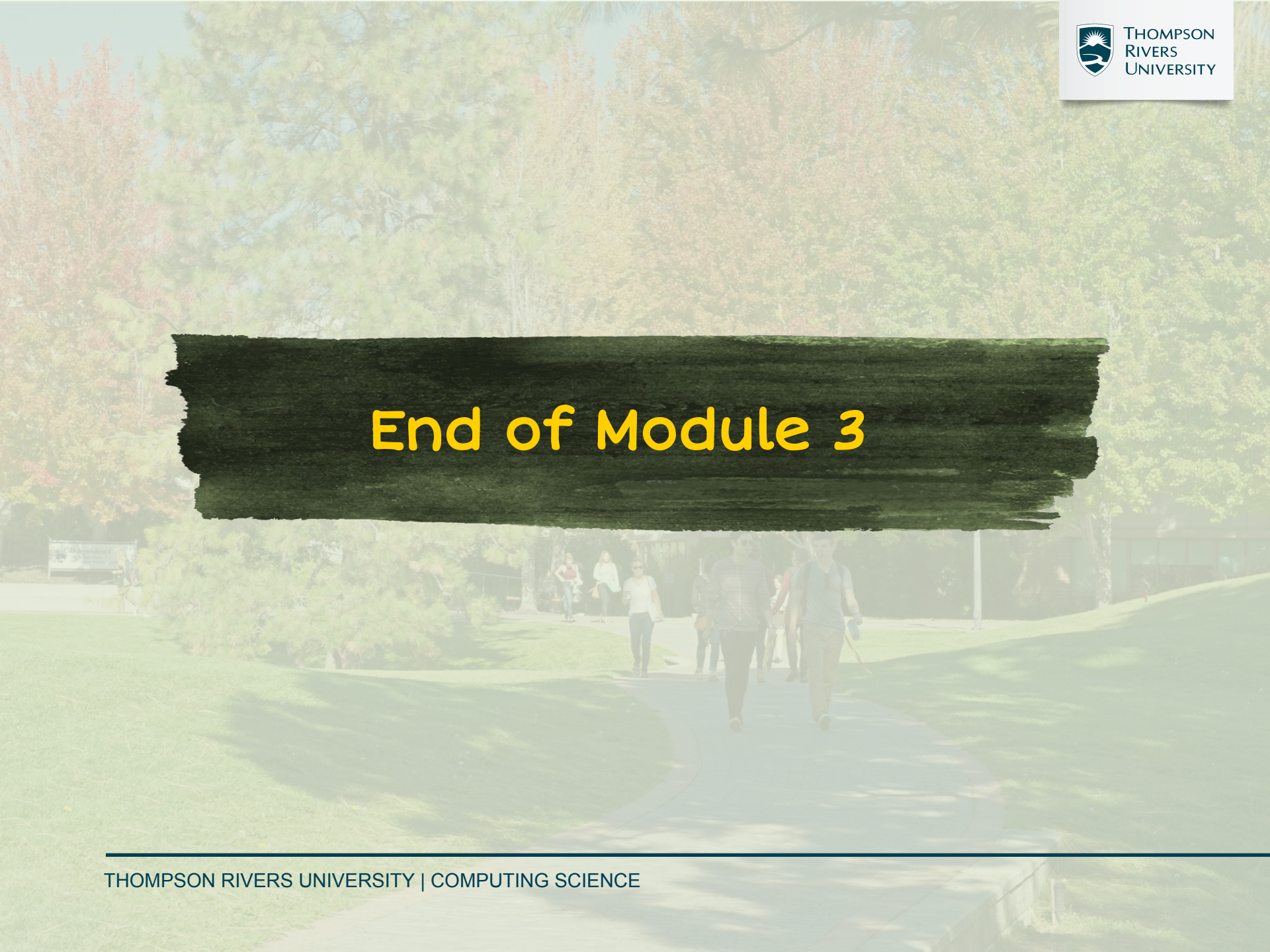
Communicating with Activities

Fragment Independence

Activity 9

Although a Fragment is implemented as an object that's independent from an Activity and can be used inside multiple activities, a given instance of a fragment is directly tied to the activity that contains it. **True or False?**

True


Thompson
Rivers
University
End of Module 3
THOMPSON RIVERS UNIVERSITY | COMPUTING SCIENCE