CIT 4404 Mobile App Development

Topic3: Basic User-Interface Android Components

Dr. Fullgence Mwakondo
Institute of Computing and Informatics
Technical University of Mombasa
mwakondo@tum.ac.ke

Basic User-Interface Android Components

a. Introduction to Activities and Lifecycle
b. Activity Code Format
c. Activity Layout
d. Hiding Activity Title
e. Using Manifest file to apply styles and themes
f. Displaying Dialog Window
g. Displaying Progress Dialog
h. Introduction to intents and Lifecycle
i. Intent Code Format
j. Linking Activities using Intents
k. Returning Results from Activity using Intent
l. Passing Data to Activity using Intent
m. Introduction to Fragments and Lifecycle
n. Fragment Code Layout
o. Adding Fragments to Activity
p. Interaction between Fragmnts
q. Displaying notifications

Activities

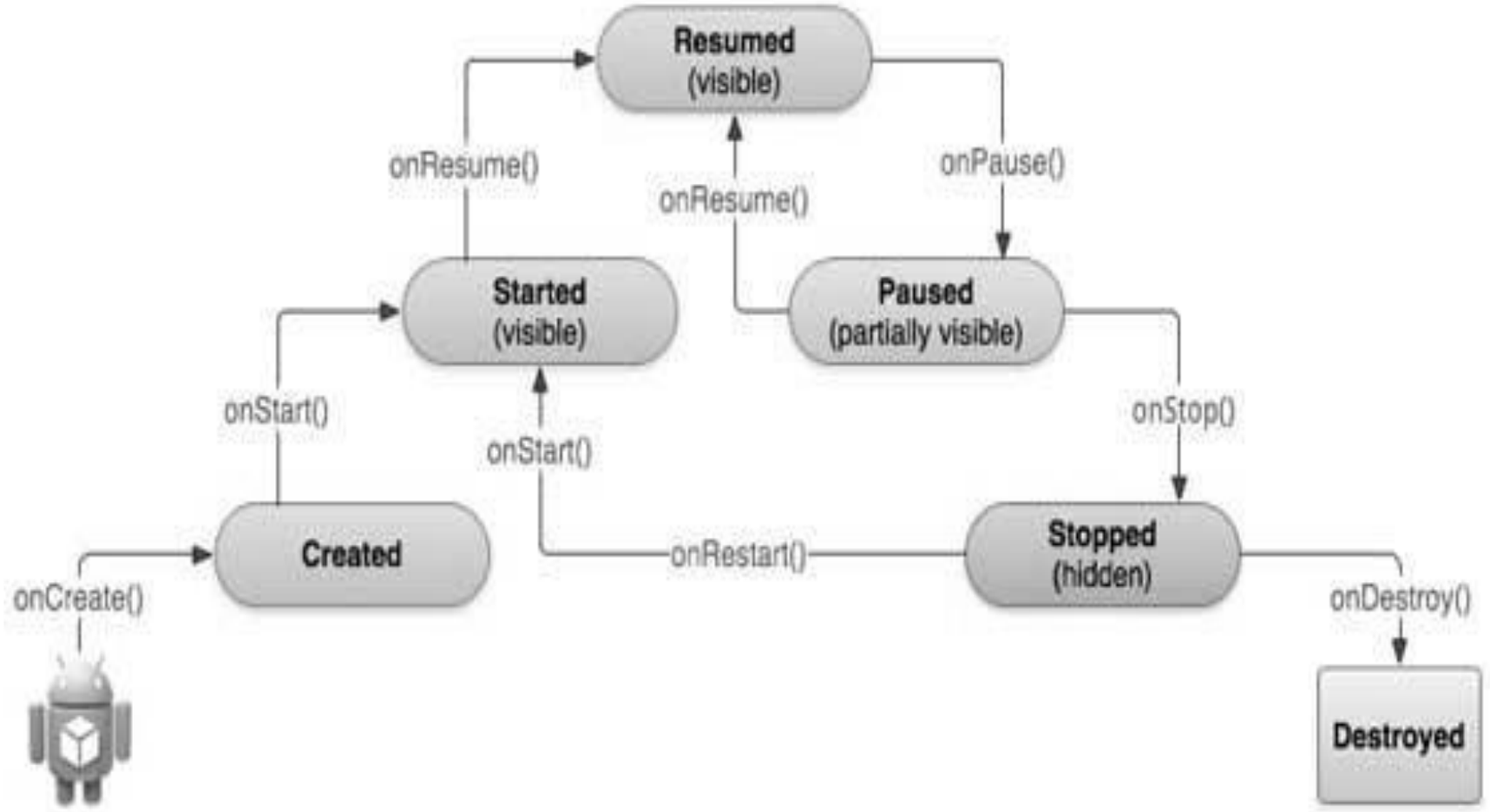
- Typically, applications have one or more activities
- The main purpose of an activity is to interact with the user
- An activity's life cycle: from the moment an activity appears on the screen to the moment it is hidden, it goes through a number of stages

Different types of Android Activity

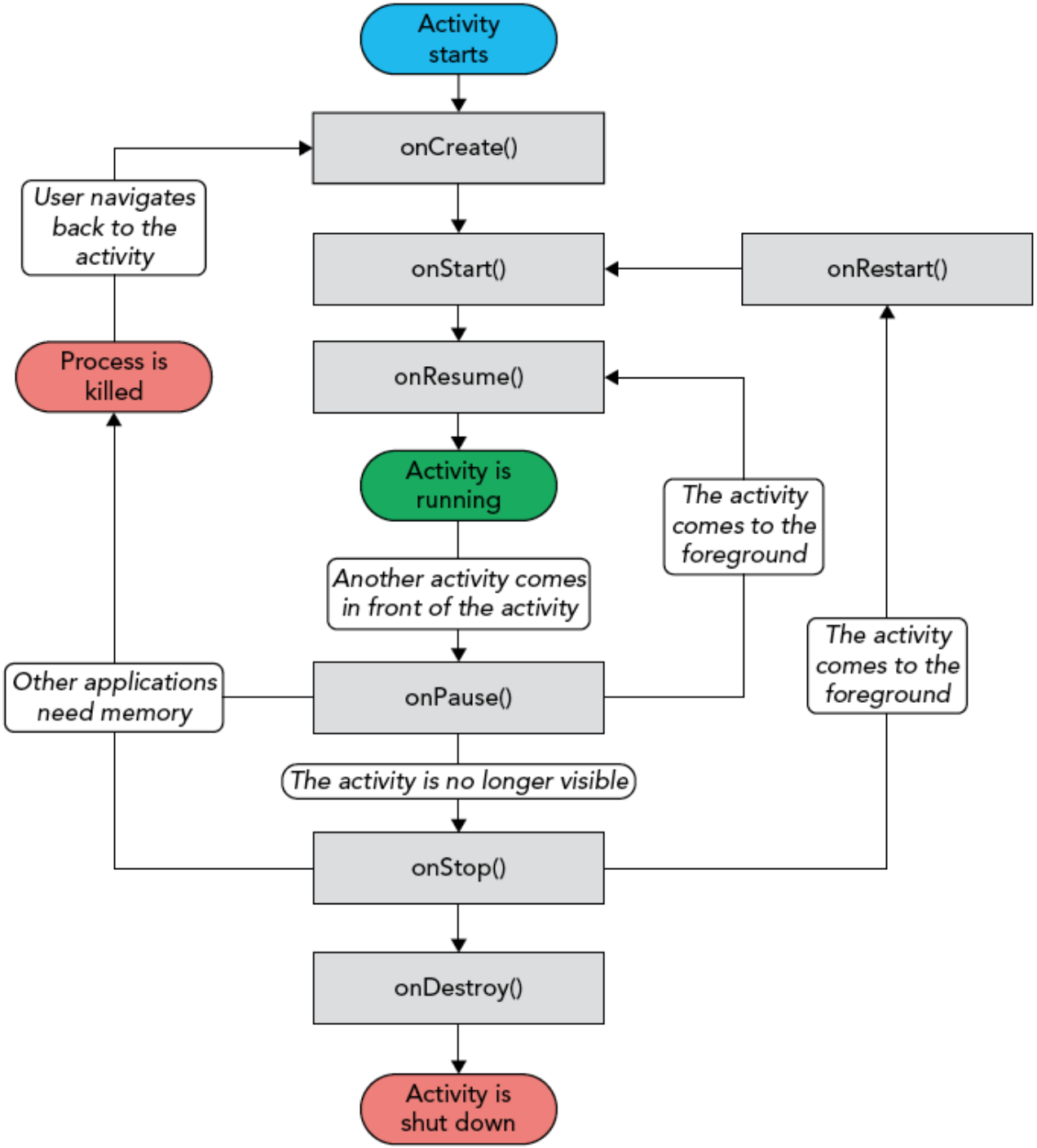
Different types of android activity available that can be used depending upon the requirements of the application.

(1) Basic Activity
(2) Bottom Navigation Activity
(3) Empty Activity
(4) FullScreen Activity
(5) Login Activity
(6) Master / Detail Flow
(7) Navigation Drawer Activity
(8) Scrolling Activity
(9) Settings Activity
(10) Tabbed Activity

Activity States



Activity Life Cycle



Activity Life Cycle

- onCreate()—Called when the activity is first created
- onStart()—Called when the activity becomes visible to the user
- onResume()—Called when the activity starts interacting with the user
- onPause()—Called when the current activity is being paused and the previous activity is being resumed
- onStop()—Called when the activity is no longer visible to the user
- onDestroy()—Called before the activity is destroyed by the system (either manually or by the system to conserve memory)
- onRestart()—Called when the activity has been stopped and is restarting again

```
package name;
import statements;
public class NamedActivity extends  Activity
{
    Declaration statement;
    @Override
    protected void onCreate(Bundle savedInstanceState)
   {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.LayoutName);
        statements;
   }
   public void onStart()
   {
       super.onStart(); statements;
    }
   public void onRestart()
   {
       super.onRestart(); statements;
   }
   public void onResume()
   {
       super.onResume(); statements;
}
   public void onPause()
   {
       super.onPause(); statements;
    }
   public void onStop()
   {
       super.onStop(); statements;
    }
   public void onDestroy()
   {
        super.onDestroy(); statements;
   }
}
```

Observe Activity Life Cycle

- Using Android Studio, create a new Android project and name it Activity101
- In the Activity101 Activity.java file, add the following highlighted statements
  - Throughout this example, be sure to change all references to "com.jfdimarzio" to whatever package name your project is using

```
package com.jfdimarzio.activity101;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
public class MainActivity extends AppCompatActivity
{
    String tag = "Lifecycle Step";
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.d(tag, "In the onCreate() event");
    }
    public void onStart()
    {
        super.onStart();
        Log.d(tag, "In the onStart() event");
    }
    public void onRestart()
    {
        super.onRestart();
        Log.d(tag, "In the onRestart() event");
    }
```

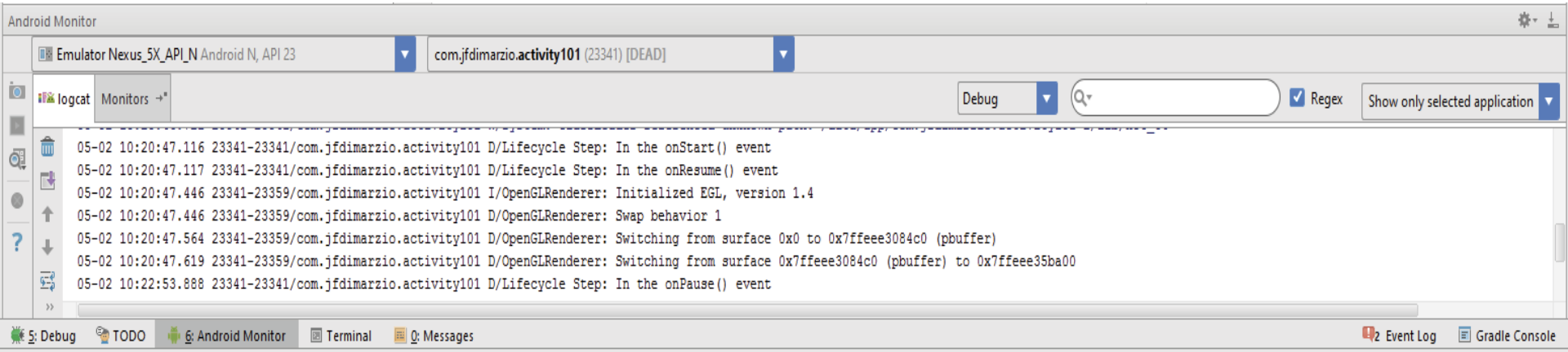
Observe Activity Life Cycle

- Run => Debug, or Shift + F9 to debug the app

```
    public void onResume()
    {
        super.onResume();
        Log.d(tag, "In the onResume() event");
    }
    public void onPause()
    {
        super.onPause();
        Log.d(tag, "In the onPause() event");
     }
    public void onStop()
    {
        super.onStop();
        Log.d(tag, "In the onStop() event");
     }
    public void onDestroy()
    {
         super.onDestroy();
         Log.d(tag, "In the onDestroy() event");
    }
}
```

Observe Activity Life Cycle

- When the activity is first loaded, you should see something very similar to the following in the logcat console



- If you click the Back button on the Android emulator, you will see:

```
11-16 06:29:26.665: D/Lifecycle Step(559): In the onPause() event
11-16 06:29:28.465: D/Lifecycle Step(559): In the onStop() event
11-16 06:29:28.465: D/Lifecycle Step(559): In the onDestroy() event
```

Observe Activity Life Cycle

- Click the Home button, click the Overview icon, select the Activity101 app, you will see:

```
11-16 06:31:08.905: D/Lifecycle Step(559): In the onCreate() event
11-16 06:31:08.905: D/Lifecycle Step(559): In the onStart() event
11-16 06:31:08.925: D/Lifecycle Step(559): In the onResume() event
```

- Click the Home button and then click the Phone button on the Android emulator so that the activity is pushed to the background

```
11-16 06:32:00.585: D/Lifecycle Step(559): In the onPause() event
11-16 06:32:05.015: D/Lifecycle Step(559): In the onStop() event
```

- Exit the phone dialer by clicking the Back button, the activity is now visible again:

```
11-16 06:32:50.515: D/Lifecycle(559): In the onRestart() event
11-16 06:32:50.515: D/Lifecycle(559): In the onStart() event
11-16 06:32:50.515: D/Lifecycle(559): In the onResume() event
```

Observe Activity Life Cycle: Summary

- Use the onCreate() method to create and instantiate the objects that you will be using in your application
- Use the onResume() method to start any services or code that needs to run while your activity is in the foreground
- Use the onPause() method to stop any services or code that does not need to run when your activity is not in the foreground
- Use the onDestroy() method to free up resources before your activity is destroyed

Intent

- An intent is the “glue” that enables activities from different applications to work together seamlessly, ensuring that tasks can be performed as though they all belong to one single application

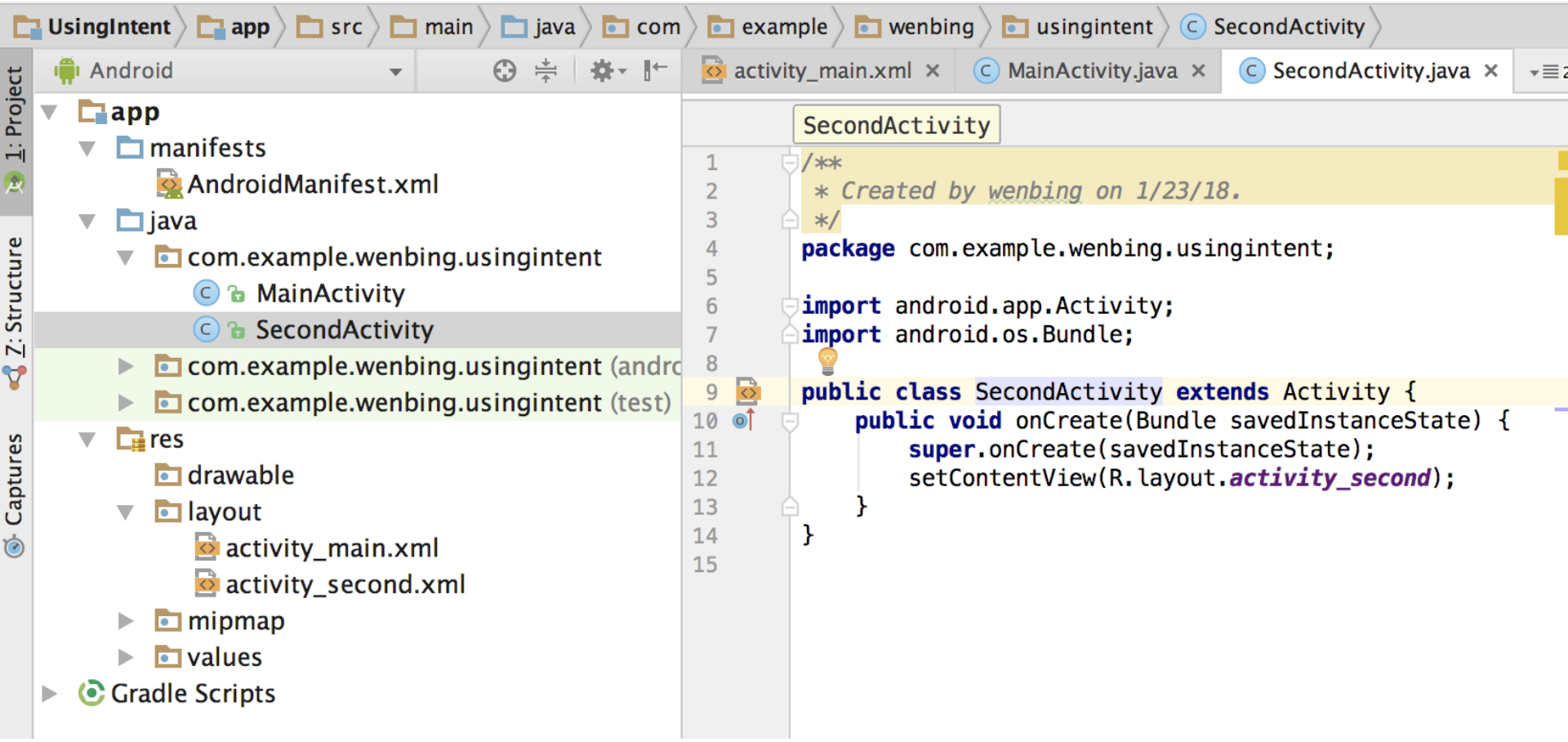
Intents

- When your application has more than one activity, you often need to navigate from one to another. In Android, you navigate between activities through what is known as an intent

Linking Activities with Intents: Example

- Using Android Studio, create a new Android project with an empty Activity named MainActivity; name the project **UsingIntent**
- Right-click your package name under the java folder in the Project Files windows and select New ➪ Java Class
- Name the new class SecondActivity and click OK
- Add the bolded statements from the following code to the AndroidManifest.xml file

Linking Activities with Intents: Example



Linking Activities with Intents: Example

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.jfdimarzio.usingintent">
   <application
       android:allowBackup="true”
       ...
       <activity android:name=".MainActivity">
          .....
       </activity>
       <activity android:name=".SecondActivity" >
          <intent-filter >
             <action android:name="com.username.usingintent.SecondActivity" />
             <category android:name="android.intent.category.DEFAULT" />
          </intent-filter>
        </activity>
    </application>
</manifest>
```

Linking Activities with Intents: Example

- Make a copy of the activity_main.xml file (in the res/layout folder) by right-clicking it and selecting Copy. Then right-click the res/layout folder and select Paste. Name the file **activity_second.xml**
- Modify the activity_second.xml file as follows:

```
<?xml version="1.0" encoding="utf-8"?>
.....
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.jfdimarzio.usingintent.SecondActivity">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="This is the Second Activity!" />
</RelativeLayout>
```

Linking Activities with Intents: Example

- In the SecondActivity.java file, add the bolded statements from the following code:

```
import android.app.Activity;
import android.os.Bundle;

public class SecondActivity extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);
    }
}
```

Linking Activities with Intents: Example

- Add the bolded lines in the following code to the activity_main.xml file:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android=http://schemas.android.com/apk/res/android
    .....
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Main Activity!"
        android:id="@+id/textView" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Display second activity"
        android:onClick="onClick"
        android:id="@+id/button"
        android:layout_below="@+id/textView"
        android:layout_alignParentStart="true"
        android:layout_marginTop="56dp" />
</RelativeLayout>
```

Linking Activities with Intents: Example

- Modify the MainActivity.java file as shown in the bolded lines in the following code:

```
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.view.View;

public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void onClick(View view) {
        startActivity(new Intent("com.username.usingintent.SecondActivity"));
    }
}
```

Linking Activities with Intents: Example

- Press Shift+F9 to debug the application on the Android emulator
- When the first activity is loaded, click the button and the second activity also loads

Homework #1

- Add a UI control on the screen of the second activity so that you can go back to the first activity (i.e., the main activity). In addition, on the main activity, display an iteration count on the number of times the main activity is displayed.

Intents

- Returning data from an activity
- Passing data to an activity

Returning data from an activity

- Modify the UsingIntent app to see how this is done
- Replace secondactivity.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout android:orientation="vertical"
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.username.usingintent.SecondActivity">
```

Returning data from an activity

- Replace secondactivity.xml:

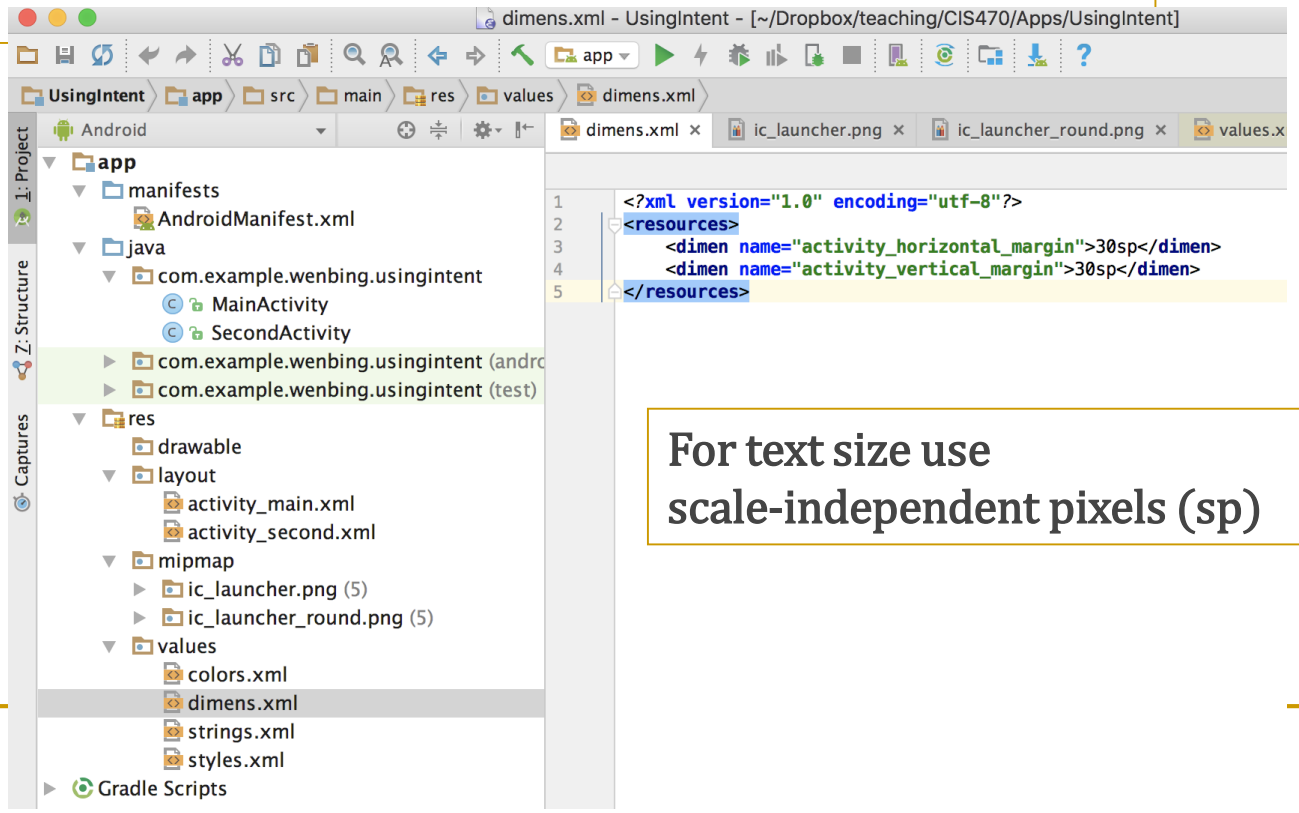
```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="This is the Second Activity!"
    android:id="@+id/textView2" />
<TextView
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Please enter your name"
   android:id="@+id/textView3" />
```

```
   <EditText
      android:layout_width="match_parent”
      android:layout_height="wrap_content"
      android:id="@+id/txtUsername" />
   <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="OK"
      android:onClick="onClick"
      android:id="@+id/button2" />
</LinearLayout>
```

Returning data from an activity

- Create a dimen.xml file to define dimen values:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <dimen name="activity_horizontal_margin">30sp</dimen>
   <dimen name="activity_vertical_margin">30sp</dimen>
</resources>
```



For text size use scale-independent pixels (sp)

Returning data from an activity

- Add the **bolded** statements in the following code to SecondActivity.java:

```
package com.example.wenbing.usingintent;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.content.Intent;
import android.net.Uri;

public class SecondActivity extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);
    }
```

Returning data from an activity

- Add the **bolded** statements in the following code to SecondActivity.java:

```
public class SecondActivity extends Activity {
    .....
    public void onClick(View view) {
        Intent data = new Intent();
//---get the EditText view---
        EditText txt_username = (EditText)findViewById(R.id.txtUsername);
//---set the data to pass back---
        data.setData(Uri.parse( txt_username.getText().toString()));
        setResult(RESULT_OK, data);
//---closes the activity---
        finish();
    }
}
```

For an activity to return a value to the calling activity, you use an Intent object to send data back via the setData() method

The setResult() method sets a result code (either RESULT_OK or RESULT_CANCELLED) and the data (an Intent object) to be returned back to the calling activity

Returning data from an activity

- Add (or replace with) the bolded statements in the following code to the MainActivity.java file:

```
import android.view.View;
import android.widget.Toast;
public class MainActivity extends Activity {
    int request_Code = 1;
    .....
    public void onClick(View view) {
        startActivityForResult(new Intent("com.username.usingintent.SecondActivity"),request_Code);
    }
    public void onActivityResult(int requestCode, int resultCode, Intent data)
    {
        if (requestCode == request_Code) {
            if (resultCode == RESULT_OK) {
                Toast.makeText(this,data.getData().toString(),
                    Toast.LENGTH_SHORT).show();
            }
        }
    }
}
```

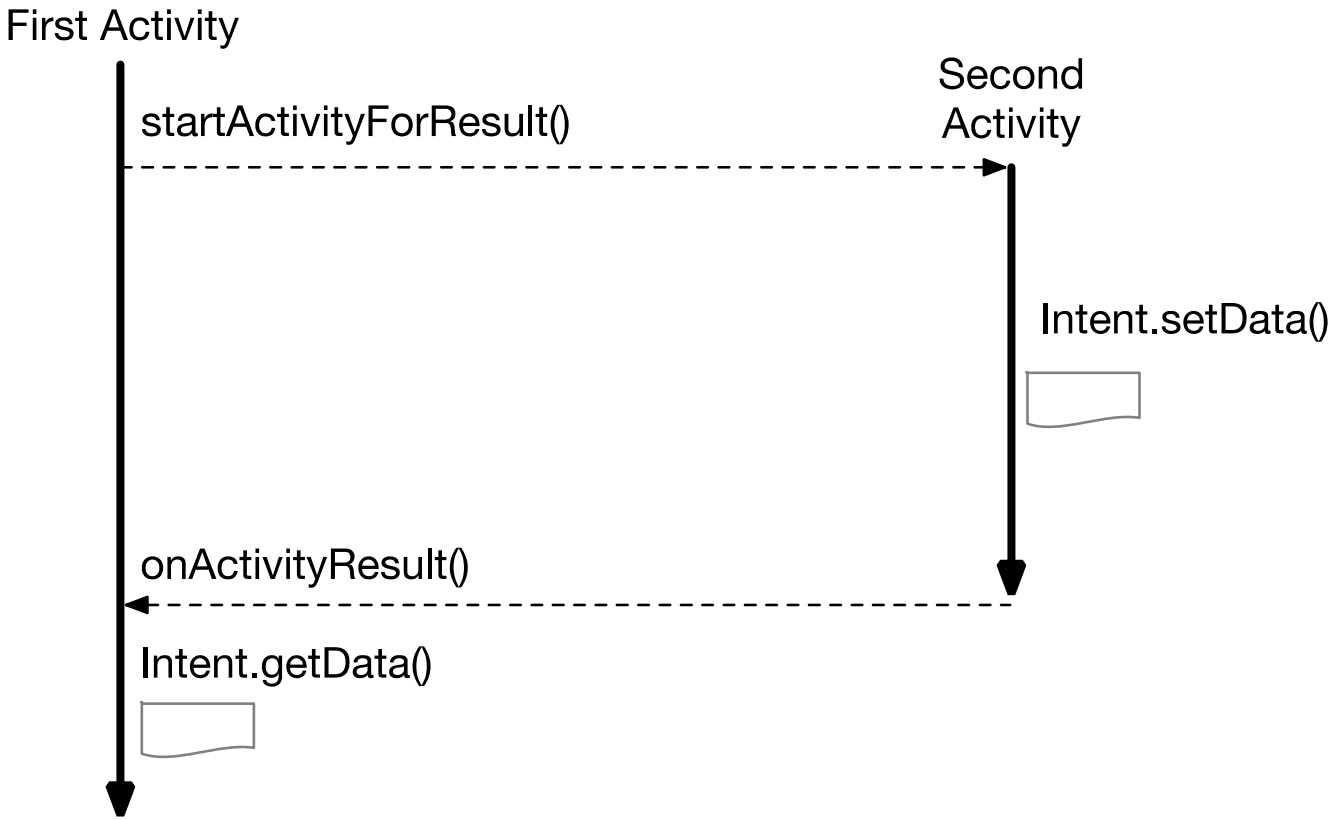
The request code is an integer value that identifies an activity you are calling (cannot be -1)

When an activity returns, you need this request code to determine which activity is actually returned

A toast is a view containing a quick little message for the user

To retrieve the data set using the setData() method, use the getData() method

Returning data from an activity



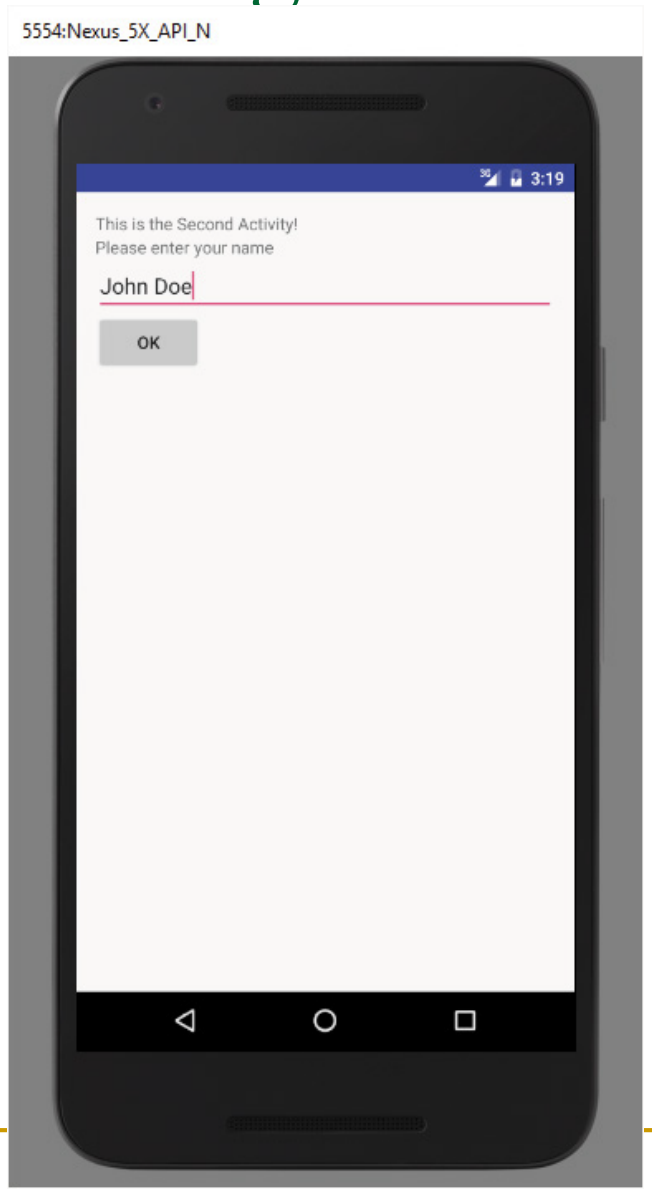
Toast

public class Toast extends Object
java.lang.Object
↳android.widget.Toast

- A toast is a view containing a quick little message for the user
- When the view is shown to the user, appears as a floating view over the application
- The idea is to be as unobtrusive as possible, while still showing the user the information you want them to see Two examples are the volume control, and the brief message saying that your settings have been saved

int	LENGTH_LONG Show the view or text notification for a long period of time.
int	LENGTH_SHORT Show the view or text notification for a short period of time.

Returning data from an activity



Passing data to an activity

- Create a new project and name it **PassingData**
- Add/replace with the bolded statements in the following code to the activity_main.xml file:

Create a dimen.xml file to define dimen values

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout android:orientation="vertical"
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   tools:context="com.username.passingdata.MainActivity">
   <Button
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="Click to go to Second Activity"
       android:id="@+id/button"
       android:onClick="onClick"/>
</LinearLayout>
```

Replace old code with this!

Passing data to an activity

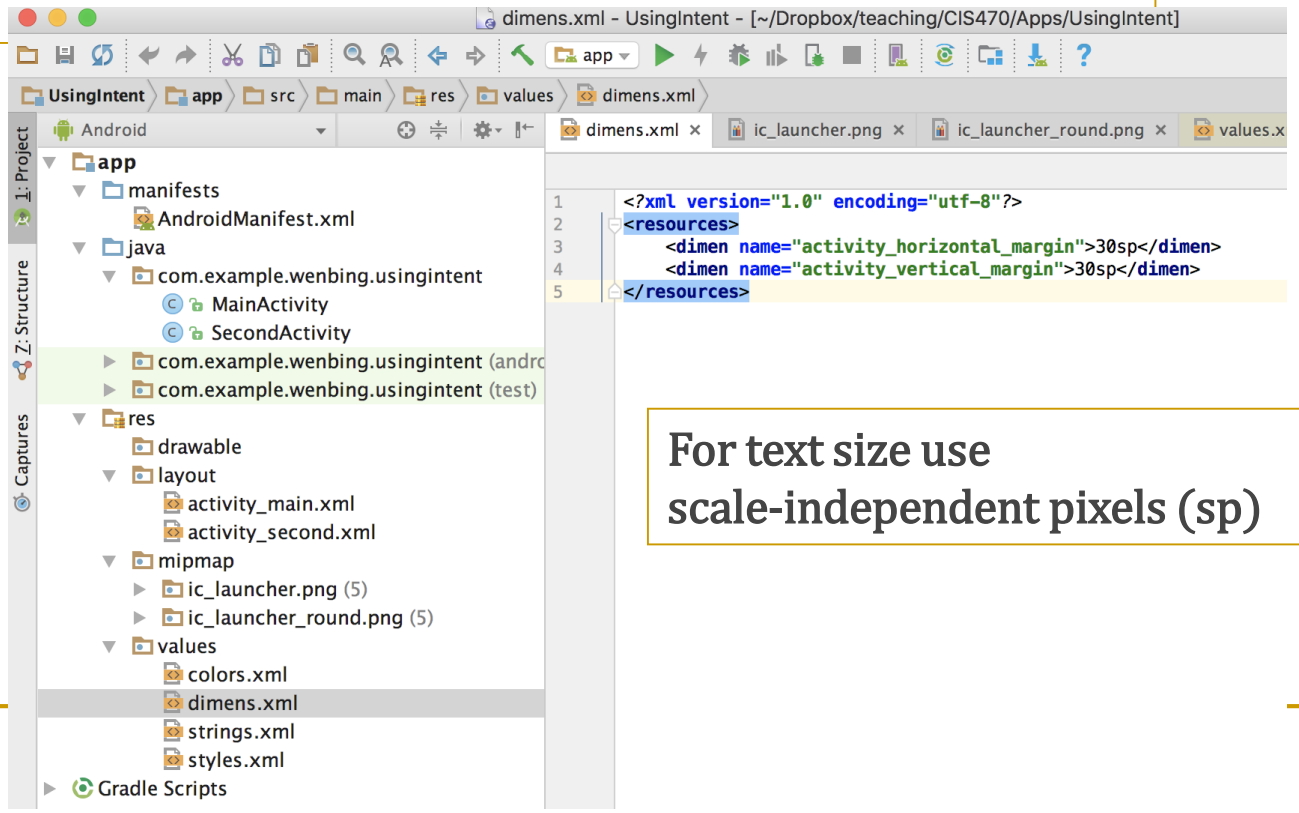
- Add a new XML file to the res/layout folder and name it activity_second.xml. Populate it as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout android:orientation="vertical"
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.username.passingdata.MainActivity">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Welcome to the Second Activity"
        android:id="@+id/textView" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Click to go to Main Activity"
        android:id="@+id/button"
        android:onClick="onClick"/>
</LinearLayout>
```

Passing data to an activity

- Create a dimen.xml file to define dimen values:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <dimen name="activity_horizontal_margin">30sp</dimen>
   <dimen name="activity_vertical_margin">30sp</dimen>
</resources>
```



For text size use scale-independent pixels (sp)

Passing data to an activity

- Add a new Class file to the package and name it SecondActivity. Populate the SecondActivity.java file as follows:

```
package com.username.passingdata;
import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;
```

```
public class SecondActvity extends Activity {
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_second);
   //---get the data passed in using getStringExtra()---
      Toast.makeText(this,getIntent().getStringExtra("str1"),
           Toast.LENGTH_SHORT).show();
   //---get the data passed in using getIntExtra()---
      Toast.makeText(this,Integer.toString(
            getIntent().getIntExtra("age1", 0)),
            Toast.LENGTH_SHORT).show();
   //---get the Bundle object passed in---
       Bundle bundle = getIntent().getExtras();
   //---get the data using the getString()---
       Toast.makeText(this, bundle.getString("str2"),
             Toast.LENGTH_SHORT).show();
   //---get the data using the getInt() method---
       Toast.makeText(this,Integer.toString(bundle.getInt("age2")),
             Toast.LENGTH_SHORT).show();
}
```

To obtain the data sent using the Intent object, you first obtain the Intent object using the getIntent() method. Then, call its getStringExtra() method to get the string value set using The putExtra() method

use getInt() method to retrieve an integer value

Passing data to an activity

- Add a new Class file to the package and name it SecondActivity. Populate the SecondActivity.java file as follows:

```
    public void onClick(View view) {
        //---use an Intent object to return data---
        Intent i = new Intent();
        //---use the putExtra() method to return some value---
        i.putExtra("age3", 45);
        //---use the setData() method to return some value---
        i.setData(Uri.parse("Something passed back to main activity"));
        //---set the result with OK and the Intent object---
        setResult(RESULT_OK, i);
        //---destroy the current activity---
        finish();
    }
}
```

Passing data to an activity

- Add the bolded statements from the following code to the AndroidManifest.xml file:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.username.passingdata">
   <application
   ...
      <activity android:name=".SecondActvity">
         <intent-filter>
            <action android:name="com.username.passingdata.SecondActivity" />
            <category android:name="android.intent.category.DEFAULT" />
         </intent-filter>
       </activity>
    </application>
</manifest>
```

Passing data to an activity

- Add the bolded statements from the following code to the MainActivity.java file:

```
import android.content.Intent;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
```

Passing data to an activity

- Add the bolded statements from the following code to the MainActivity.java file:

```
public void onClick(View view) {
        Intent i = new
            Intent("com.username.passingdata.SecondActivity");
        //---use putExtra() to add new name/value pairs---
        i.putExtra("str1", "This is a string");
        i.putExtra("age1", 25);
        //---use a Bundle object to add new name/values pairs---
        Bundle extras = new Bundle();
        extras.putString("str2", "This is another string");
        extras.putInt("age2", 35);
        //---attach the Bundle object to the Intent object---
        i.putExtras(extras);
        //---start the activity to get a result back---
        startActivityForResult(i, 1);
}
```

putExtra() method of an Intent object to add a name/value pair

You can also create a Bundle object and then attach it using the putExtras()method Bundle object contains a set of name/value pairs.

Passing data to an activity

- Add the bolded statements from the following code to the MainActivity.java file:

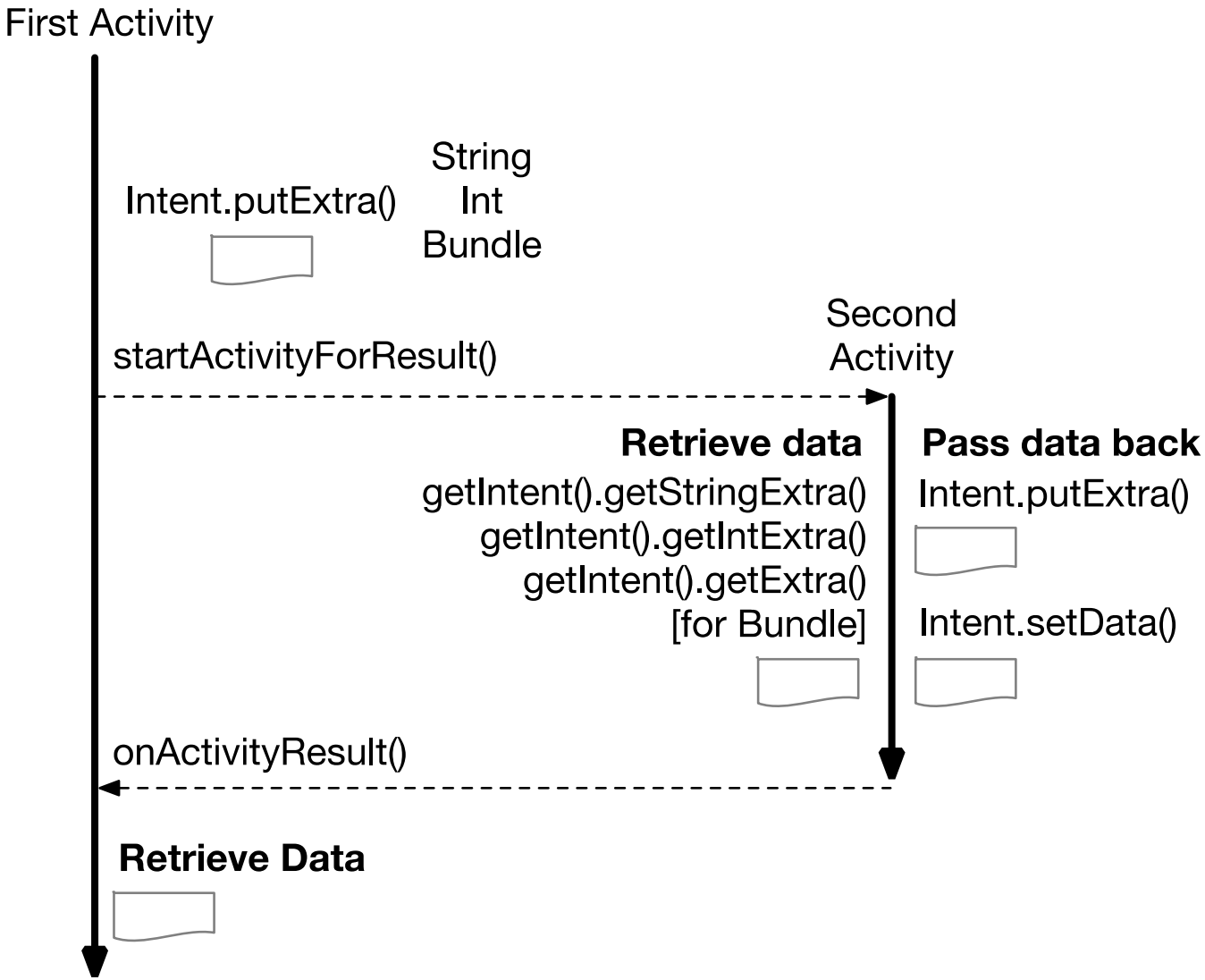
```
    public void onActivityResult(int requestCode, int resultCode, Intent data)
    {
        //---check if the request code is 1---
        if (requestCode == 1) {
            //---if the result is OK---
            if (resultCode == RESULT_OK) {
                //---get the result using getIntExtra()---
                Toast.makeText(this, Integer.toString(
                    data.getIntExtra("age3", 0)), Toast.LENGTH_SHORT).show();
                //---get the result using getData()---
                Toast.makeText(this, data.getData().toString(),
                    Toast.LENGTH_SHORT).show();
            }
        }
    }
}
```

For the integer value, use the getIntExtra() method

To retrieve the Bundle object, use the getExtras() method

To get the individual name/value pairs, use the appropriate method. For the string value, use the getString() method

Passing data to an activity



Homework #2

- Develop an contact management app (without persistency for now)
  - The main activity would start with no contact, but it has a button where the user could use to add a new contact
  - On clicking the button in the main activity, it goes to the second activity, where the user could add the contact information such as name, phone number, email, etc.
  - After entering a new contact, the user could click the submit button in the second activity, which the app will go back to the main activity
  - When there is at least one contact, the main activity could display a list of contacts identified by the full name of each contact
  - In the main activity, a user could click each row of the contact, a third activity would open to display the contact details