Mobile Programming

Dr. Nader Mahmoud

Lecturer at Computer Science department

Course Content

- **Chapter 1: Getting Started with Android Programming**
- **Chapter 2: Using Android Studio for Android Development**
- **Chapter 3: Activities, Fragments, and Intents**
- **Chapter 4: Getting to know the Android User Interface**
- **Chapter 5: Designing Your User Interface with Views**
- **Chapter 6: Displaying Pictures and Menus with Views**
- **Chapter 7: Data Persistence**
- **Chapter 8: Content Providers**
- **Chapter 9: Messaging**
- **Chapter 10: Location-Based Services**
- **Chapter 11: Networking**
- **Chapter 12: Developing Android Services**

Agenda

Chapter 3 - Activities, Fragments, and Intents

- **Understanding Activities**
  - Applying Styles and Themes to an Activity
  - Hiding the Activity Title
  - Displaying a Progress Dialog
- **Linking Activities Using Intents**
  - Returning Results from an Intent
  - Passing Data Using an Intent Object
- **Fragments**
  - Adding Fragments Dynamically
  - Life Cycle of a Fragment
  - Interactions Between Fragments
  - Understanding the Intent Object
  - Using Intent Filters
- **Displaying Notifications**

Introduction

- An Android application can have one or more activities. The main purpose of an activity is to interact with the user.

- From the moment an activity appears on the screen to the moment it is hidden, it goes through a number of stages. These stages are known as an activity's life cycle.

- Understanding the life cycle of an activity is very important to ensuring that your application works correctly.
  - Example] when you close an activity, you may need to take a confirmation from user.

Introduction

- Android also supports fragments, a feature that was introduced for tablets in Android 3.0 and for phones in Android 4.0. Think of fragments as "miniature" (مصغر) activities that can be grouped to form an activity.

- Another unique concept in Android is intent. An intent is basically the "glue" (صمغ) 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.

Understanding Activities

- To create an activity, you create a Java class that extends the Activity base class (AppCompatActivity):

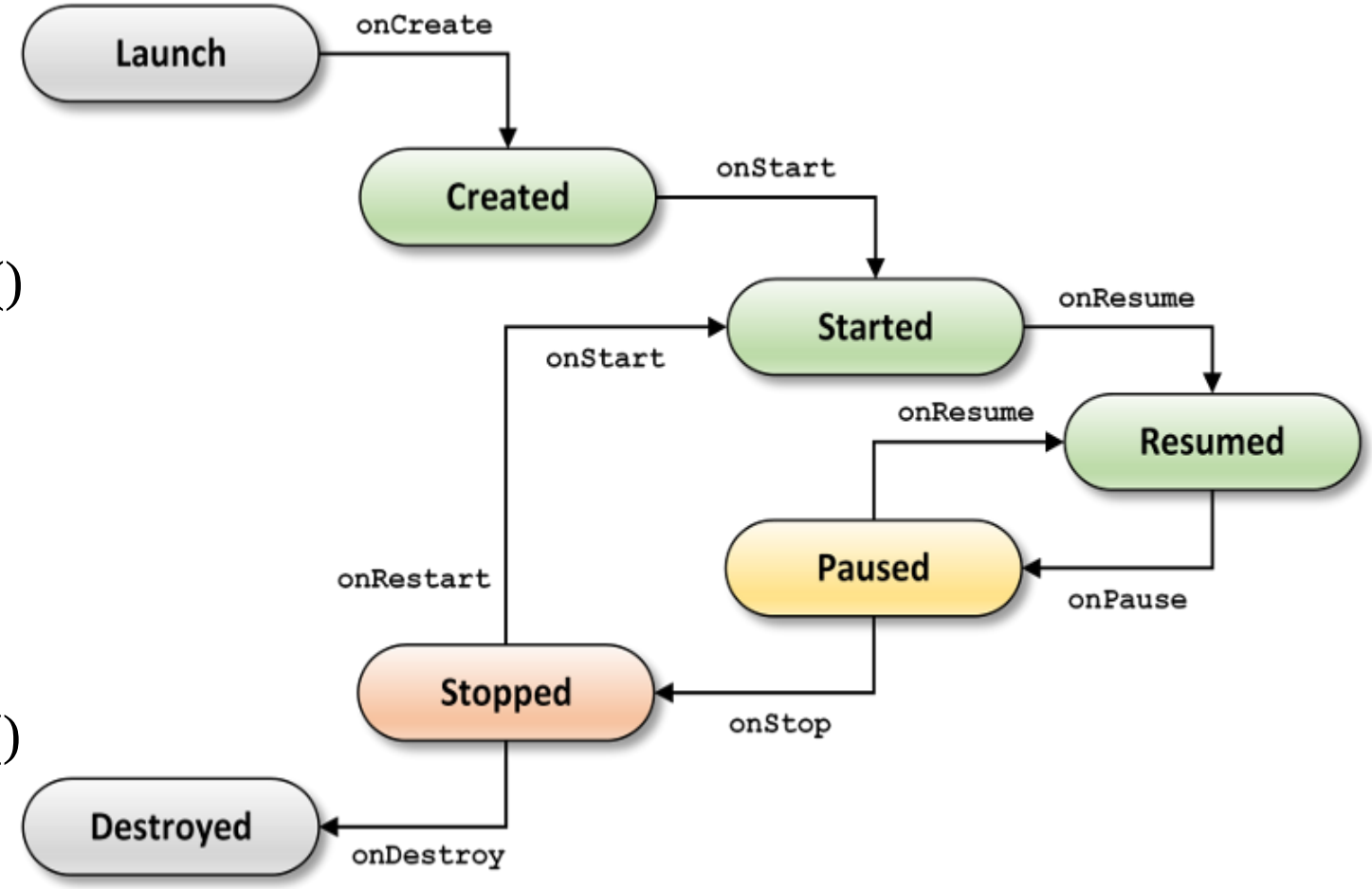
```
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
public class MainActivity extends AppCompatActivity {
        @Override
        protected void onCreate(Bundle savedInstanceState) {
           super.onCreate(savedInstanceState);
           setContentView(R.layout.activity_main);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

- Every activity must be declared in AndroidManifest.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android
package="com.example.myapplication">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name“
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Understanding Activities

- The Activity base class defines a series of events that govern the life cycle of an activity:
  - onCreate()
  - onStart()
  - onResume()
  - onRestart()
  - onPause()
  - onStop()
  - onDestroy()



Understanding Activities

- The meaning of each event in Activity class:
  - **onCreate()**: Called when the activity is first created, by default this event is usually included in any project
  - **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

Example

- This example will show a message that demonstrate each activity event.

- It will use Log class to display messages in Logcat.

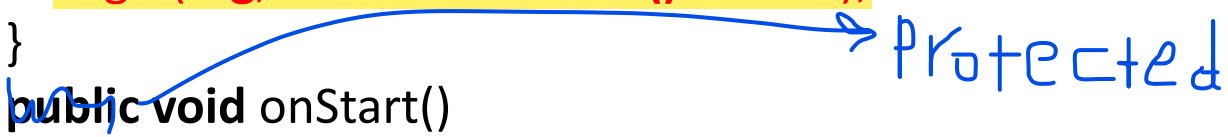
- This example contains three files:
  - AndroidManifest.xml → Default
  - MainActivity.java → Modified
  - activity_main.xml → Default

Start Logcat from View->Tools Window-> Logcat

Log.e(String, String) (error)
Log.w(String, String) (warning)
Log.i(String, String) (information)
Log.d(String, String) (debug)
Log.v(String, String) (verbose)

Example

```
import androidx.appcompat.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");
    }
```



Example

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

Example

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

Applying Styles and Themes to an Activity

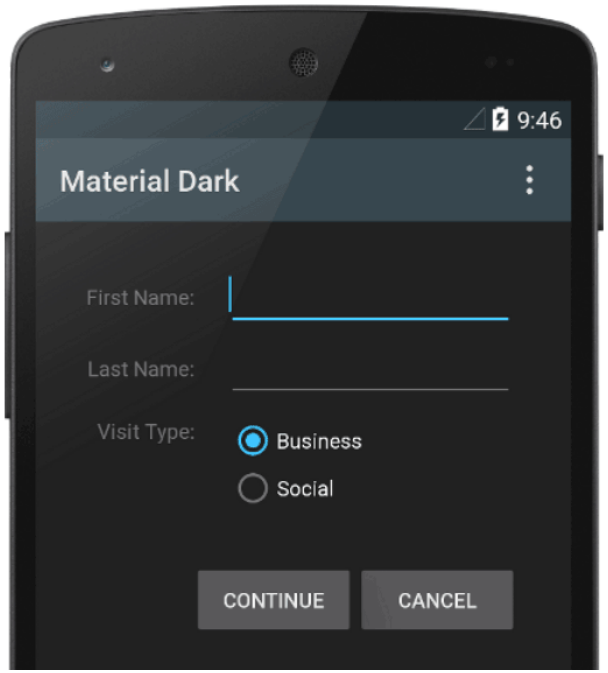
- An activity is themed to the default Android theme. However, there are other themes such as Material (available from android V7).

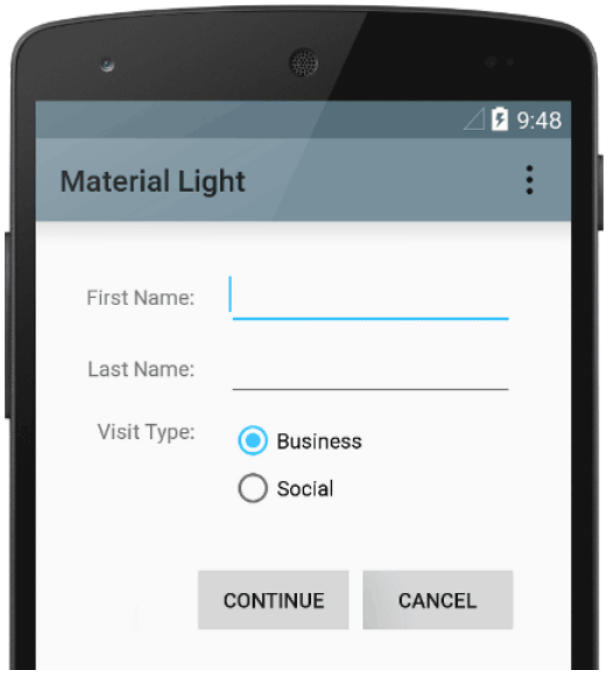
- The Material theme has a much more modern and clean look to it. There are two versions of the Material theme available : Material Light and Material Dark.

- Either of these themes can be applied from the AndroidManifest.xml. To apply one of the Material themes to an activity, simply modify the <Application> element in the AndroidManifest.xml file by changing the default android:theme attribute.

Dark Material Theme VS Light Material Theme

android:theme="@android:style/Theme.Material"

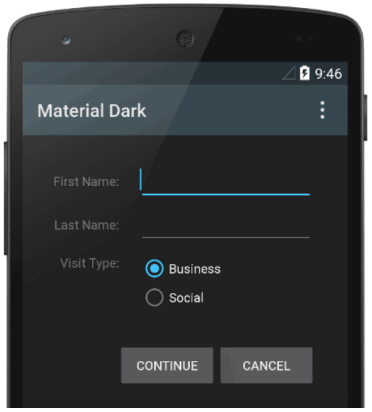




android:theme="@android:style/Theme.Material.Light"

Applying Dark Material Theme

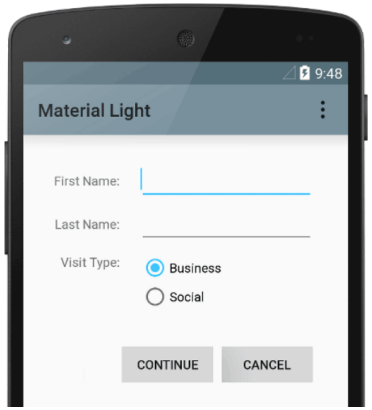
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"  package="com.example.myapplication">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@android:style/Theme.Material">
            <activity android:name=".MainActivity">
                <intent-filter>
                    <action android:name="android.intent.action.MAIN" />
                    <category android:name="android.intent.category.LAUNCHER" />
                </intent-filter>
            </activity>
    </application>
</manifest>
```



`@style/Theme.AppCompat`

Applying Light Material Theme

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"  package="com.example.myapplication">
    <application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:supportsRtl="true"
      android:theme="@android:style/Theme.Material.Light">
          <activity android:name=".MainActivity">
              <intent-filter>
                  <action android:name="android.intent.action.MAIN" />
                  <category android:name="android.intent.category.LAUNCHER" />
              </intent-filter>
          </activity>
    </application>
</manifest>
```



Hiding the Activity Title

- To hide the title of an activity, use the requestWindowFeature() method and pass it the Window.FEATURE_NO_TITLE constant:

```
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Window;
public class MainActivity extends AppCompatActivity {
@Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
    }
}
```

Hiding the Activity Title

- Now you need to change the theme in the AndroidManifest.xml to a theme that has no title bar.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
package="com.example.myapplication">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@android:style/Theme.NoTitleBar">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

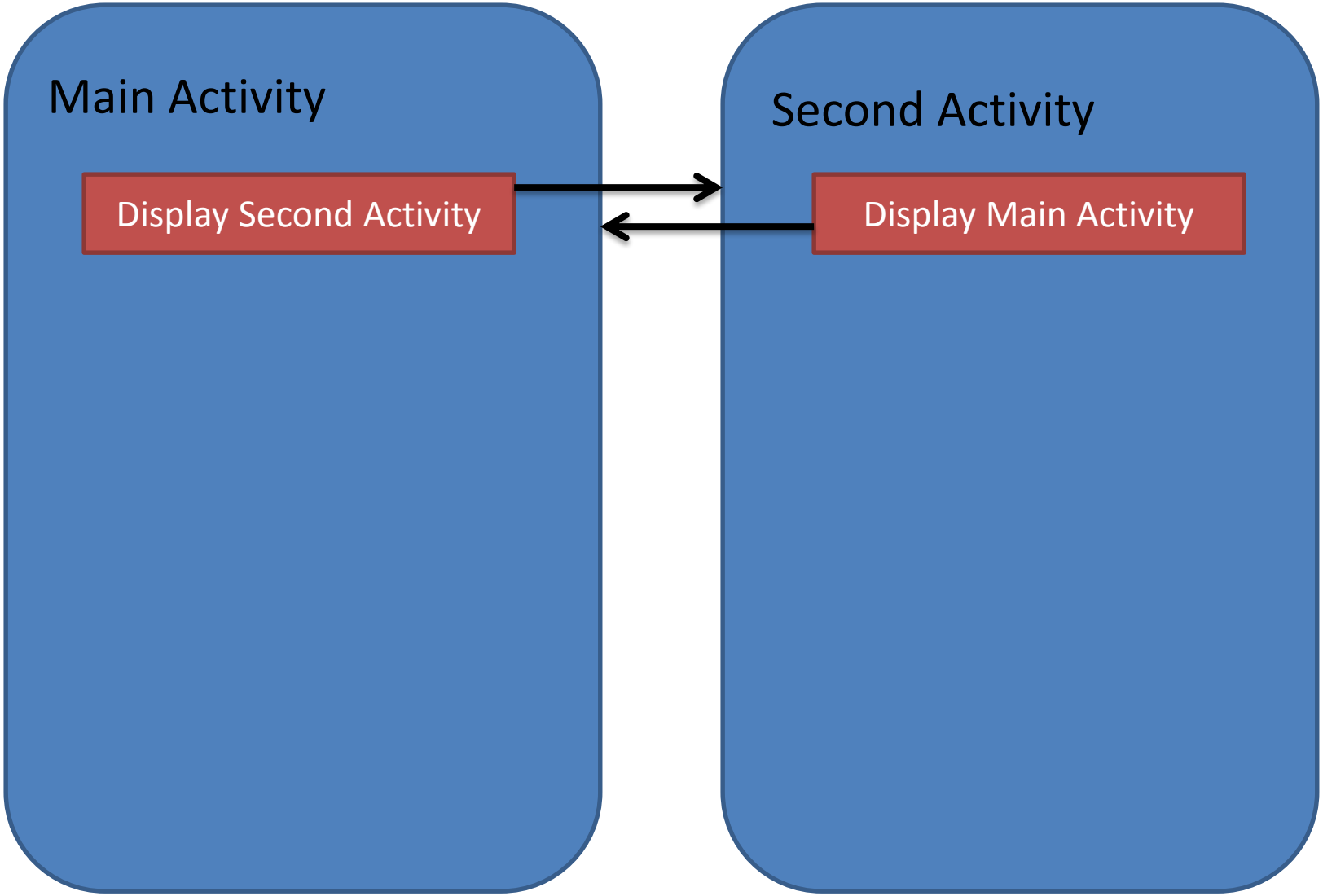
Linking Activities using Intents

- An Android application can contain one or more activities. 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.

- The best way to understand this very important but somewhat abstract concept is to experience it firsthand and see what it helps you achieve.

Linking Activities using Intents



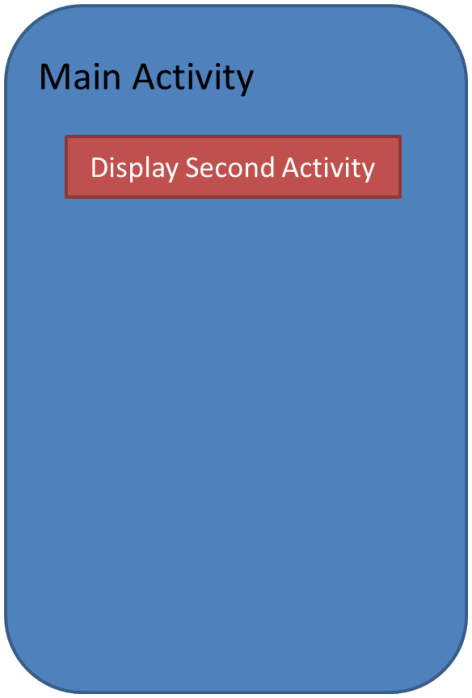
AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package=“fci.thirdyear.test">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
```

AndroidManifest.xml

```
        <activity android:name=".SecondActivity" >
            <intent-filter >
                <action android:name="fci.thirdyear.test.SecondActivity"/>
                <category android:name="android.intent.category.DEFAULT"/>
            </intent-filter>
        </activity>
    </application>
</manifest>
```

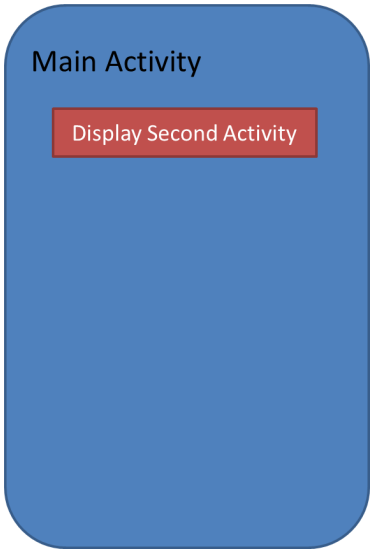
MainActivity.java




```
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
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("fci.thirdyear.test.SecondActivity"));
        }
}
```

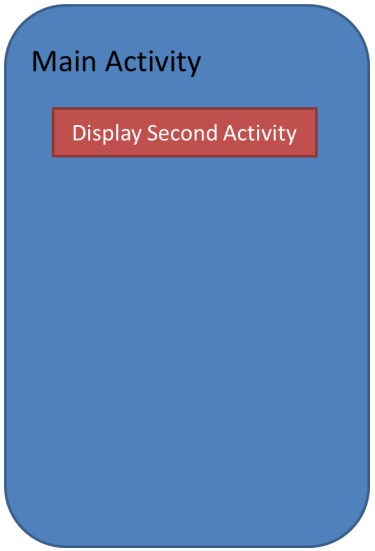
activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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=“fci.thirdyear.test.MainActivity">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Main Activity"
        android:id="@+id/textView" />
```

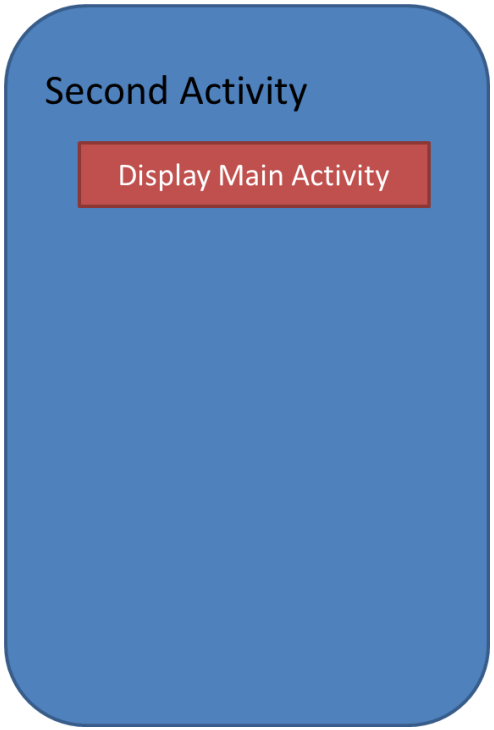


activity_main.xml

```
    <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>
```



SecondActivity.java

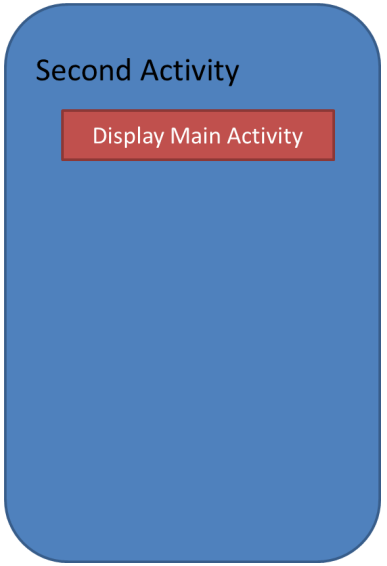



```
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
public class SecondActivity extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);
    }
    public void onClick(View view) {
        startActivity(new Intent("fci.thirdyear.test.MainActivity"));
    }
}
```

```
startActivity(new Intent(this, SecondActivity.class));
```

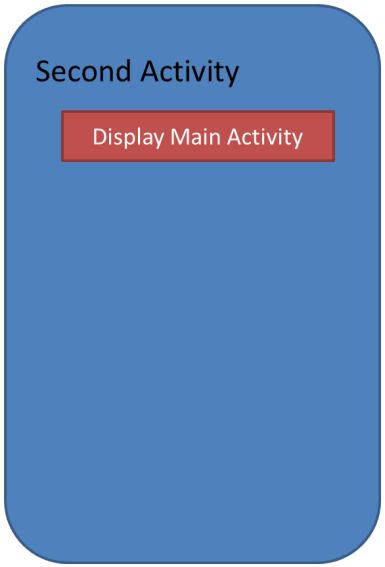
activity_second.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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=“fci.thirdyear.test.SecondActivity">
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Second Activity" />
```



activity_second.xml

```
<Button
    android:id="@+id/button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textView1"
    android:layout_alignParentStart="true"
    android:layout_alignParentLeft="true"
    android:layout_marginTop="56dp"
    android:onClick="onClick"
    android:text="Display Main Activity" />
</RelativeLayout>
```



Linking Activities using Intents

- An activity is made up of a UI component (for example, activity_main.xml) and a class component (for example, MainActivity.java).
- If you want to add another activity to a project, you need to create these two components.
- Specifically, you need to add the following to the AndroidManifest.xml file:

```
<activity android:name=".SecondActivity" >
    <intent-filter >
        <action android:name="com.example.myapplication.SecondActivity" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

Linking Activities using Intents

- When you add a new activity to the application, be sure to note the following:
  - The name (class) of the new activity is SecondActivity.
  - The intent filter name for the new activity is **<Your Package Name>.SecondActivity**. Other activities that want to call this activity invoke it via this name. Ideally, you should use the reverse domain name of your company as the intent filter name to reduce the chances of another application having the same intent filter name.
  - The category for the intent filter is **android.intent.category.DEFAULT**. You need to add this to the intent filter so that this activity can be started by another activity using the **startActivity()** method.

Linking Activities using Intents

- When the Display Second Activity button is clicked, you use the startActivity() method to display SecondActivity by creating an instance of the Intent class and passing it the intent filter name of SecondActivity:

```
public void onClick(View view) {
    startActivity(new Intent("fci.thirdyear.test.SecondActivity"));
}
```

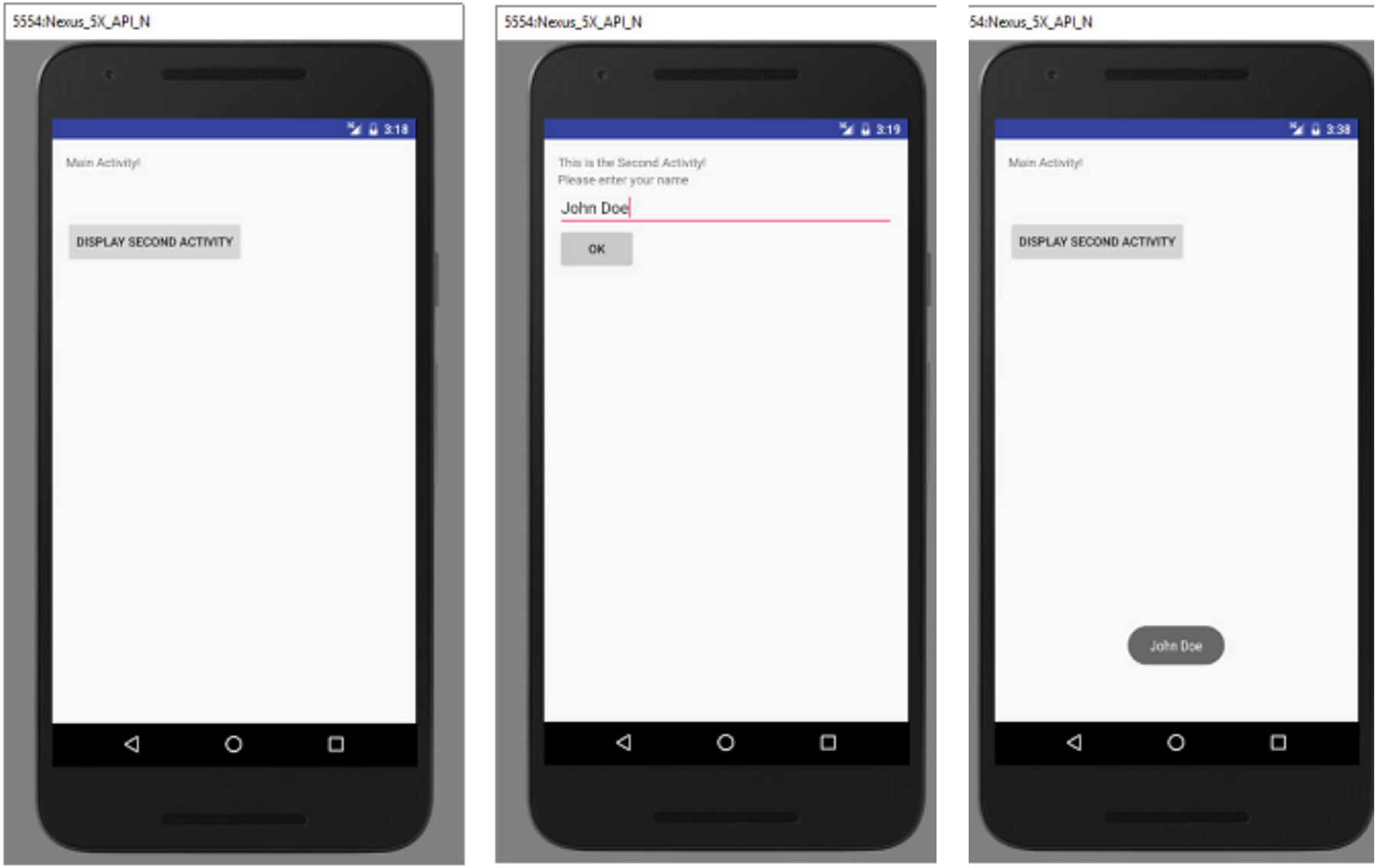
- Activities in Android can be invoked by any application running on the device.
- If the activity you want to invoke is defined within the same project, you can rewrite the preceding statement like this:

```
startActivity(new Intent(this, SecondActivity.class));
```

Returning Results from an Intent

- The **startActivity()** method invokes another activity but does not return a result to the current activity.
- For example, you might have an activity that prompts the user for username and password.
- The information entered by the user in that activity needs to be passed back to the calling activity for further processing.
- If you need to pass data back from an activity, you should instead use the **startActivityForResult()** method.

Returning Results from an Intent



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.myapplication">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".SecondActivity" >
            <intent-filter >
                <action android:name="fci.thirdyear.test.SecondActivity" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

AndroidManifest.xml

MainActivity.java

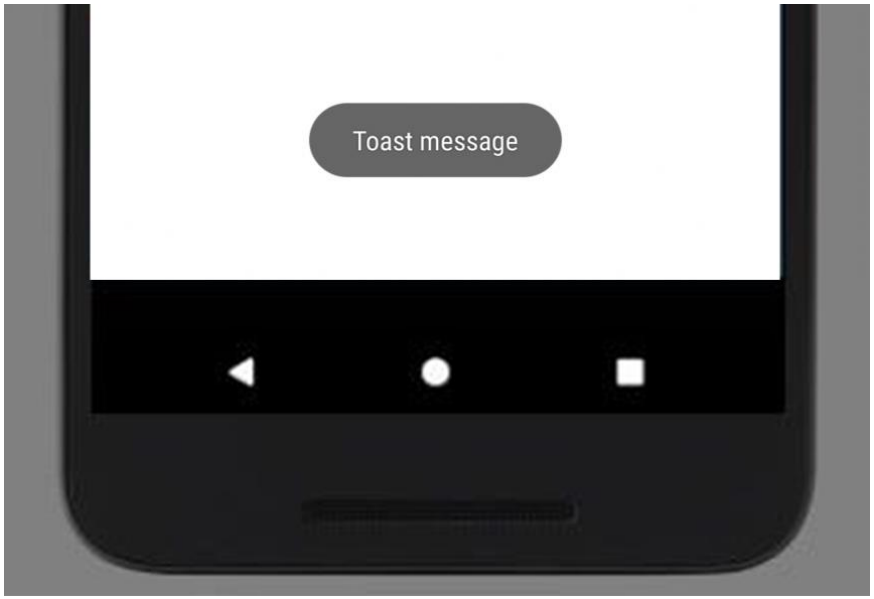
```
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;
public class MainActivity extends Activity {
    int request_Code = 1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void onClick(View view) {
        startActivityForResult(new
        Intent("com.example.myapplication.SecondActivity"),request_Code);
    }
```

The request code is simply an integer value that identifies an activity you are calling

you might be calling multiple activities at the same time

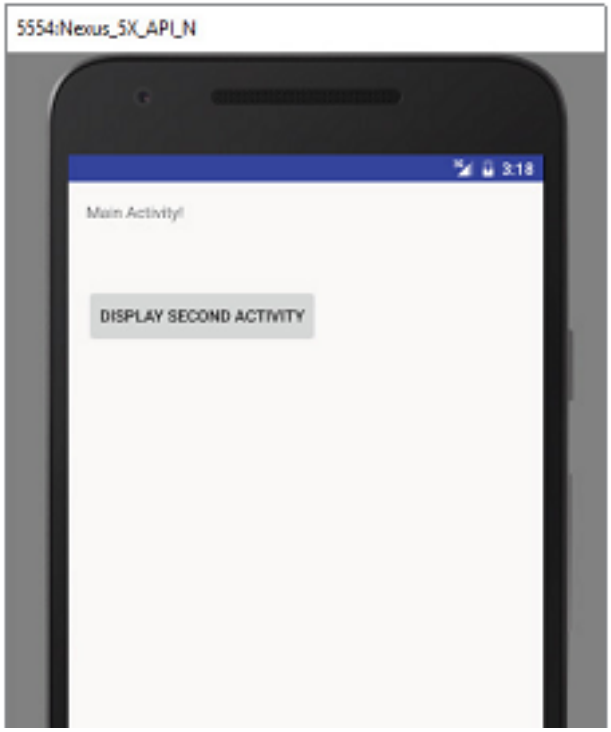
MainActivity.java

```
    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();
            }
        }
    }
}
```



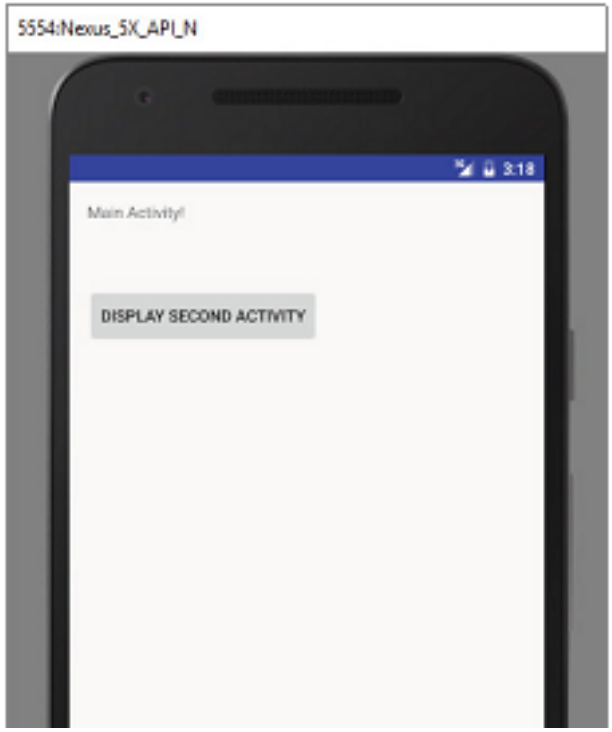
activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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"
    tools:context=“fci.thirdyear.test.MainActivity">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Main Activity!"
        android:id="@+id/textView" />
```



activity_main.xml

```
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textView"
        android:layout_alignParentStart="true"
        android:layout_marginTop="56dp"
        android:onClick="onClick"
        android:text="Display second activity"
        android:layout_alignParentLeft="true" />
</RelativeLayout>
```



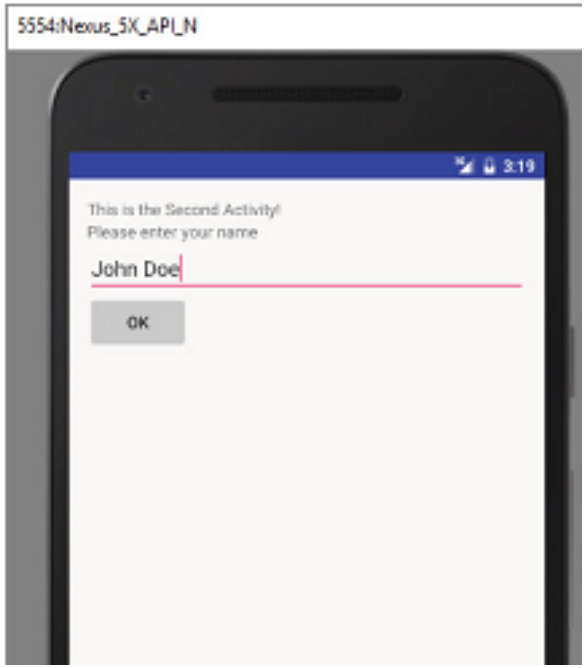
SecondActivity.java

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

public class SecondActivity extends Activity {

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

SecondActivity.java

```
    public void onClick(View view) {
        Intent data = new Intent();
        EditText txt_username = (EditText)findViewById(R.id.txtUsername);
        data.setData(Uri.parse(txt_username.getText().toString()));
        setResult(RESULT_OK, data);
        finish();
    }
}
```

closes the activity
and returns control to the calling activity.



activity_second.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"
    tools:context="“fci.thirdyear.test.SecondActivity">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="This is the Second Activity!"
        android:id="@+id/textView2" />
```

activity_second.xml

```
<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 Results from an Intent

- To call an activity and wait for a result to be returned from it, you need to use the **startActivityForResult()** method, like this:

startActivityForResult(new Intent (“fci.thirdyear.test.SecondActivity"),request_Code);

- The request code is simply an integer value that identifies an activity you are calling. This is needed because when an activity returns a value, you must have a way to identify it.
- note If the request code is set to –1, then no result is returned.

Returning Results from an Intent

- In order for an activity to return a value to the calling activity, you use an **Intent** object to send data back via the **setData()** method:

```
Intent data = new Intent();
EditText txt_username = (EditText) findViewById(R.id.txt_username);
data.setData(Uri.parse(txt_username.getText().toString()));
setResult(RESULT_OK, data);
finish();
```

- 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.
- The finish() method closes the activity and returns control to the calling activity.

Returning Results from an Intent

- In the calling activity, you need to implement the onActivityResult() method, which is called whenever an activity returns:

```
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 returned result is passed in via the data argument; and you obtain its details through the getData() method.

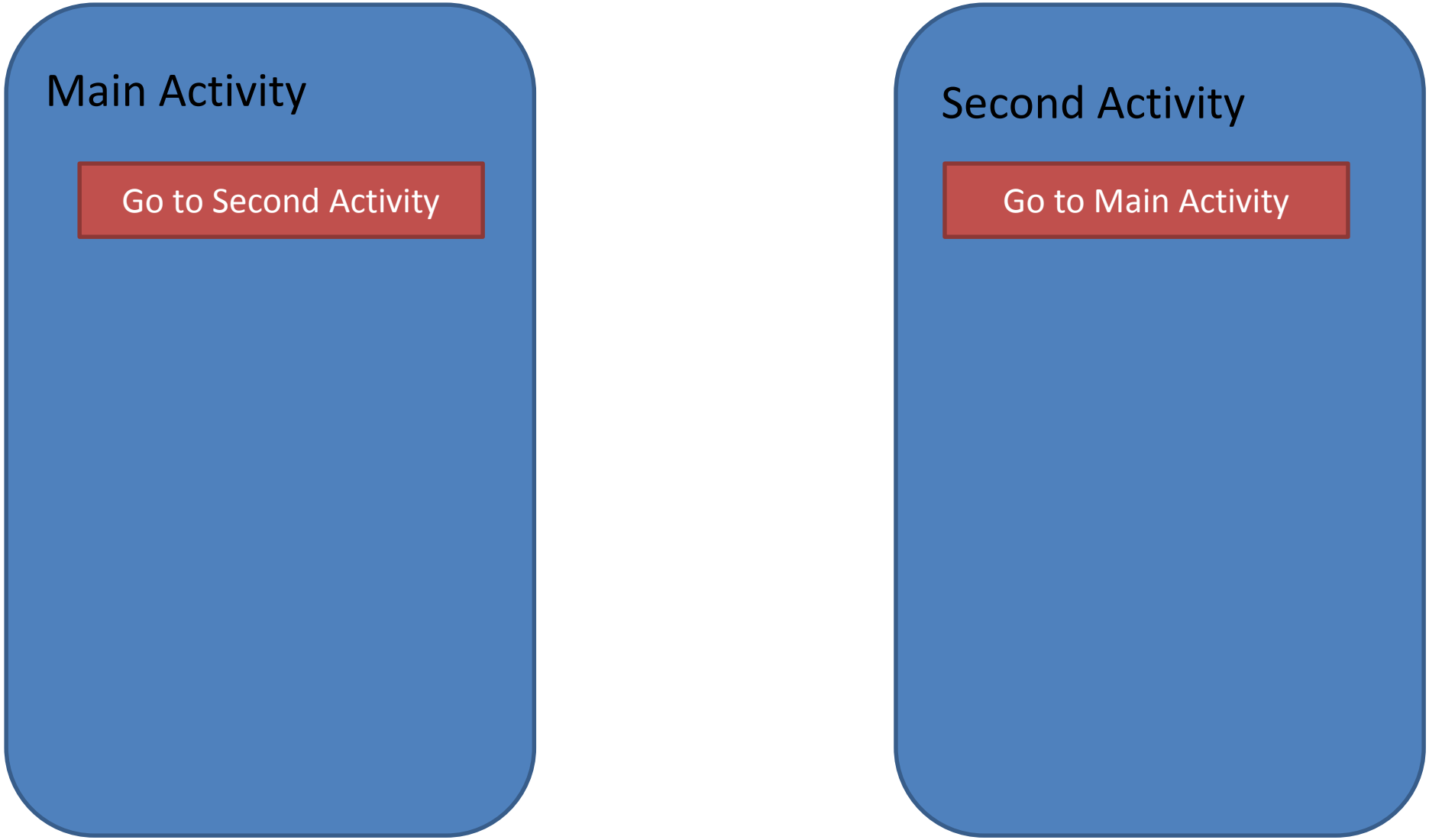
Passing Data using an Intent Object

- Besides returning data from an activity, it is also common to pass data to an activity.

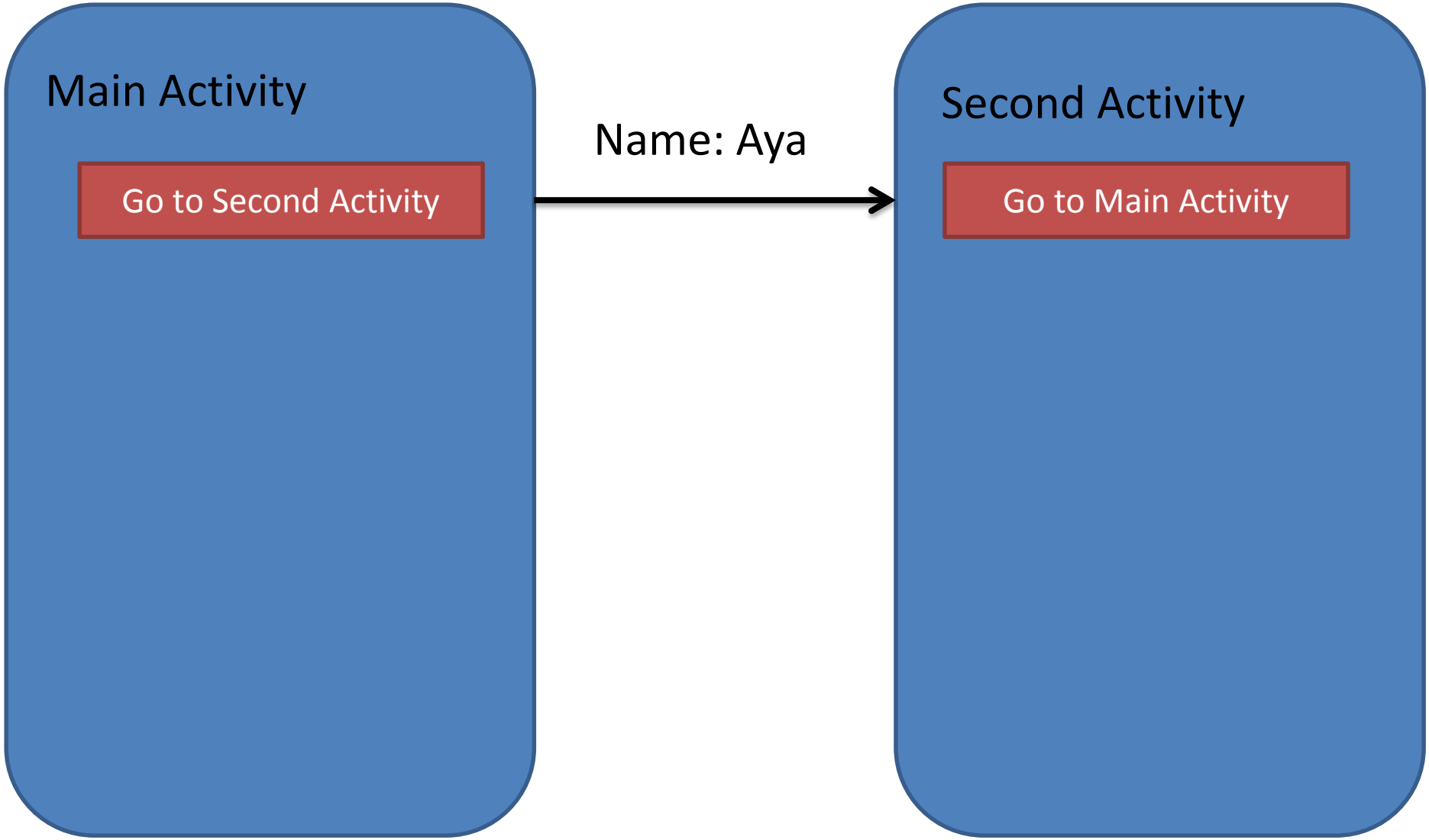
- For example, in the previous example, you might want to set some default text in the EditText view before the activity is displayed.

- In this case, you can use the Intent object to pass the data to the target activity.

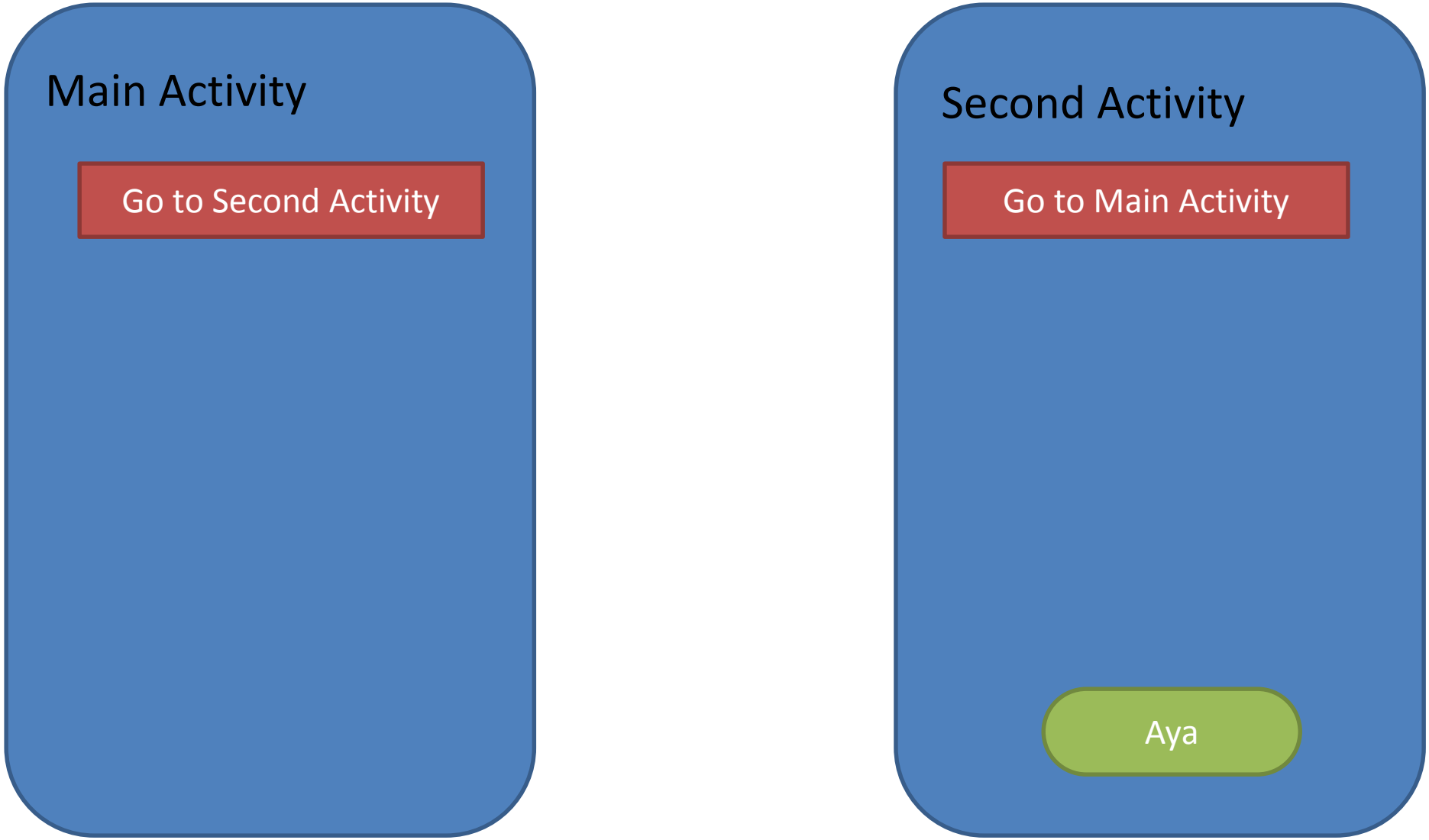
Passing Data using an Intent Object



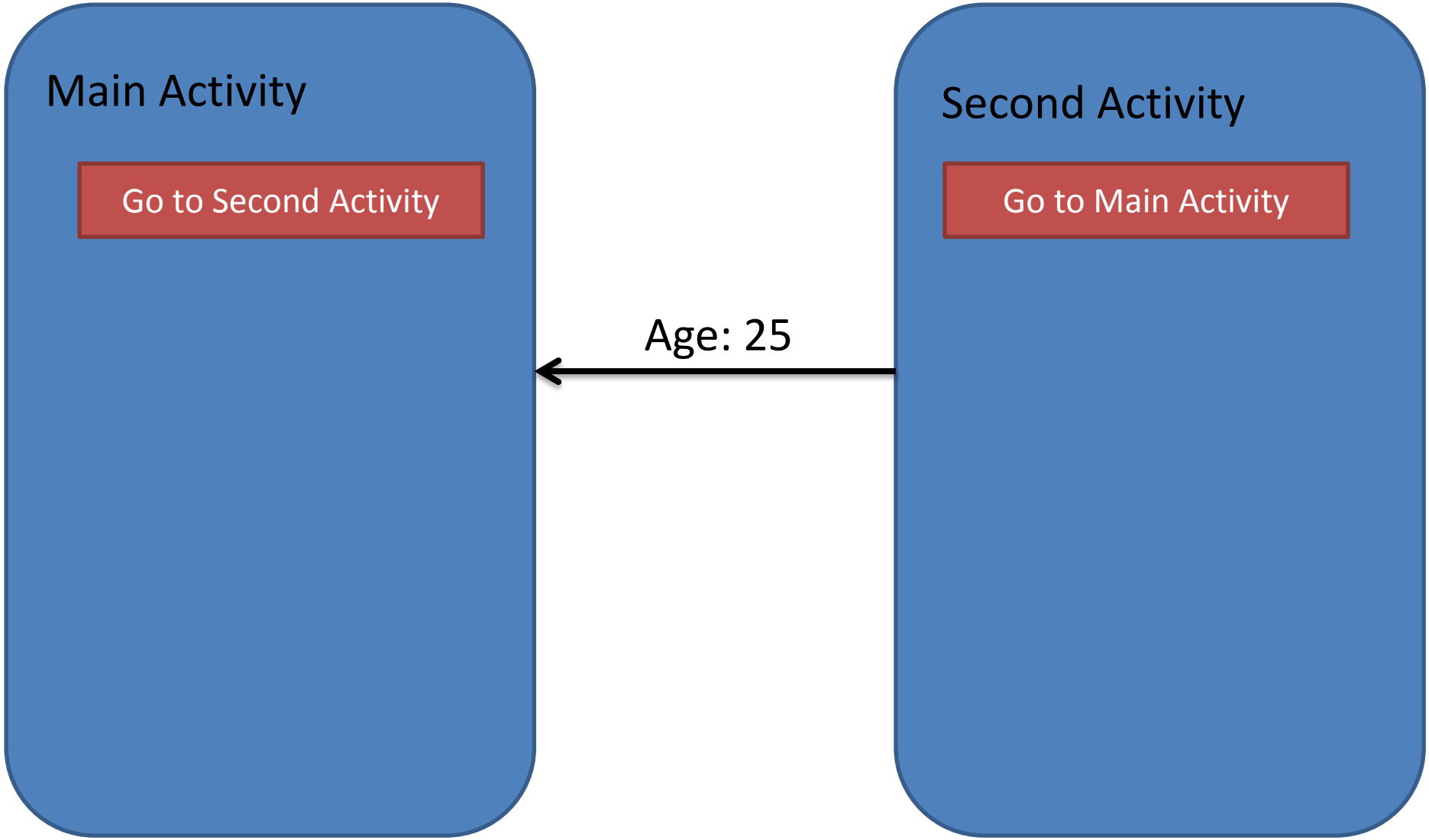
Passing Data using an Intent Object



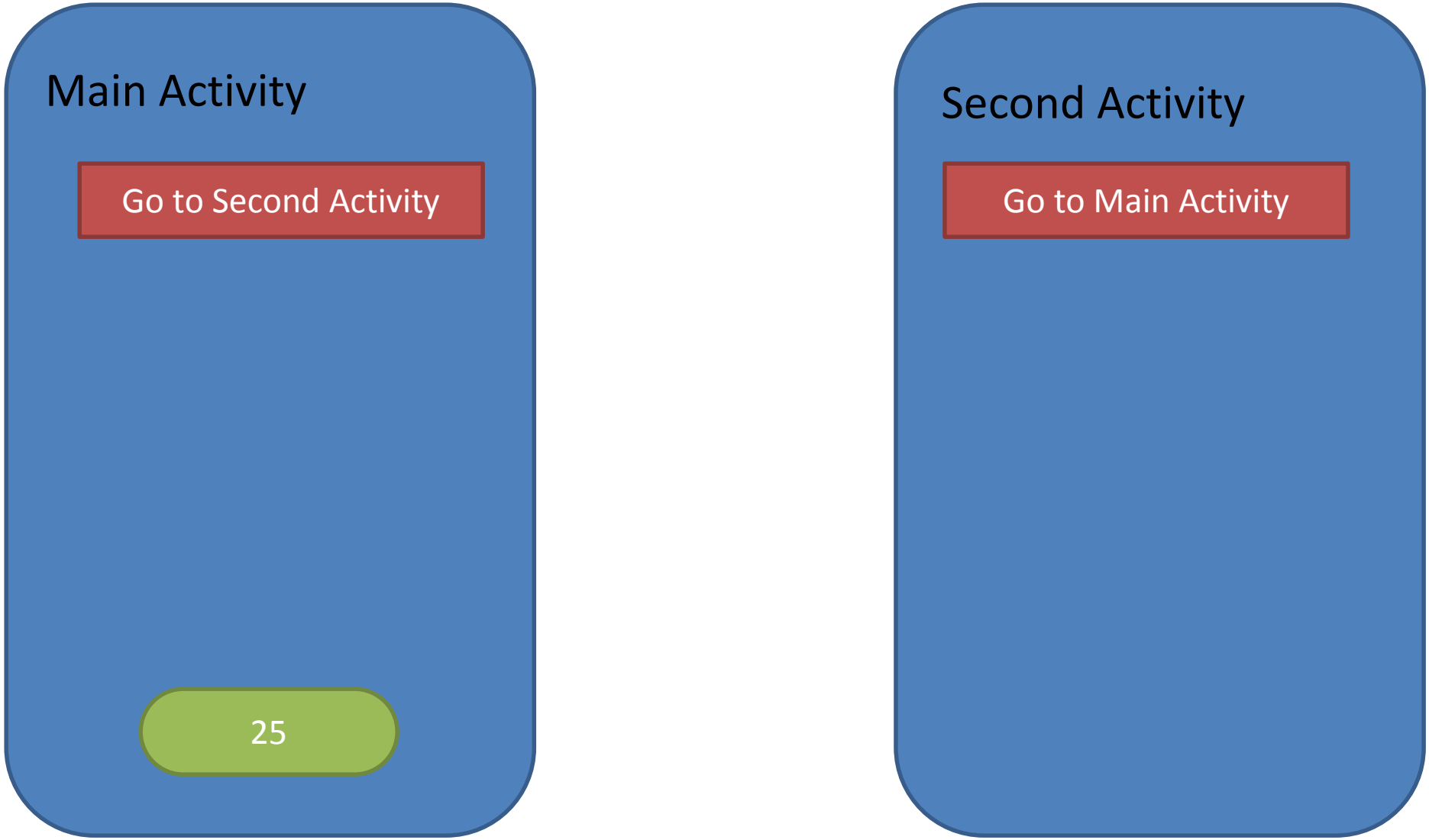
Passing Data using an Intent Object



Passing Data using an Intent Object



Passing Data using an Intent Object

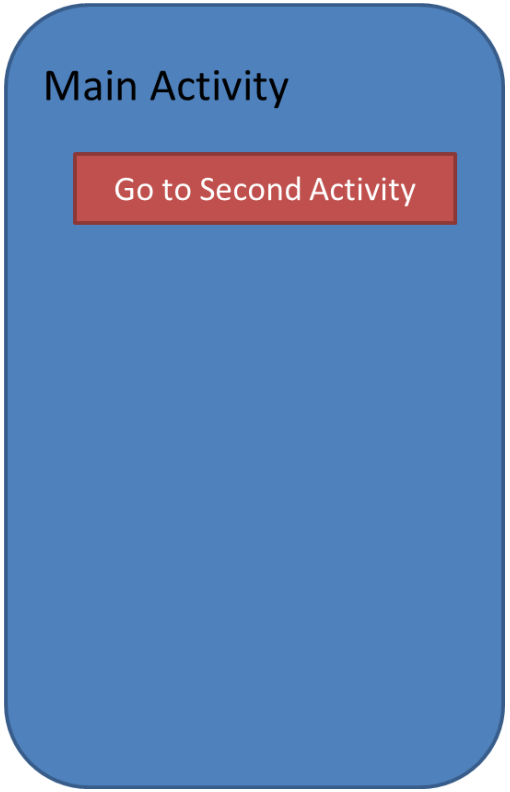


AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="com.example.app">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".SecondActvity" >
            <intent-filter >
                <action android:name="fci.thirdyear.test.SecondActivity" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
    </activity>
    </application>
</manifest>
```

activity_main.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"
        tools:context="fci.thirdyear.test.MainActivity">
        <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Main Activity"
                android:id="@+id/textView1" />
        <Button
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Go to Second Activity"
                android:id="@+id/button"
                android:onClick="onClick"/>
</LinearLayout>
```

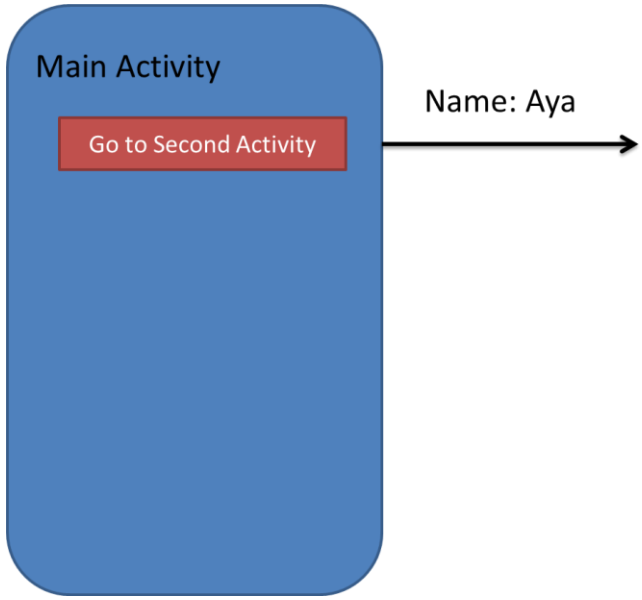


MainActivity.java

```
import android.content.Intent;
import android.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);
    }
```

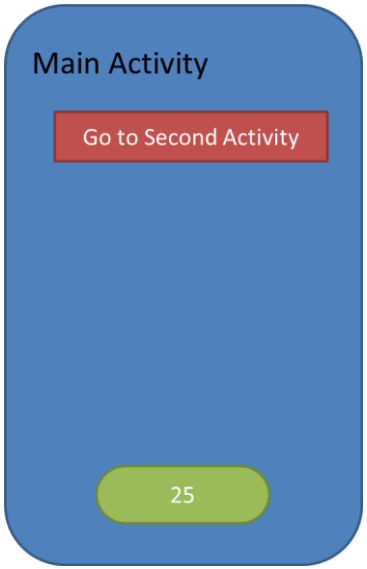
MainActivity.java

```
public void onClick(View view) {
    Intent i = new Intent("fci.thirdyear.test.SecondActivity");
    //---use putExtra() to add new name/value pairs---
    i.putExtra("name", "Aya");
    //---use a Bundle object to add new name/values  pairs---
    // Bundle extras = new Bundle();
    // extras.putString("name", "Aya");
    // i.putExtras(extras);
    //---start the activity to get a result back---
    startActivityForResult(i, 1);
}
```



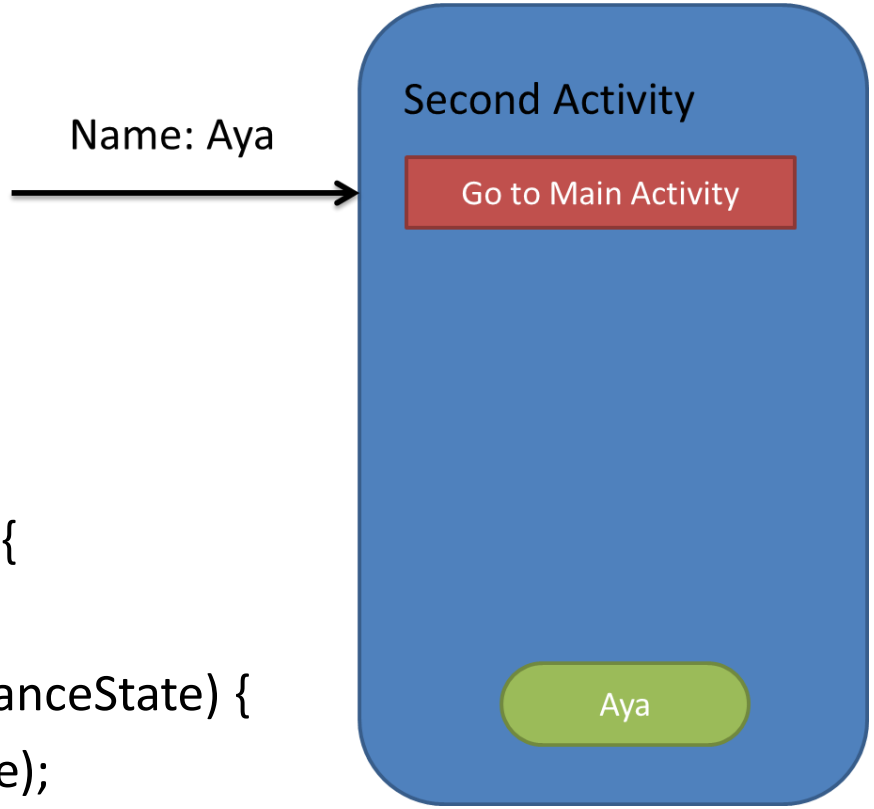
MainActivity.java

```
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("age", 0)),
            Toast.LENGTH_SHORT).show();
            //---get the result using getData()---
            // Toast.makeText(this,
             //    data.getData().toString(),
            // Toast.LENGTH_SHORT).show();
        }
    }
}
}
```



SecondActivity.java

```
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("name"),
        Toast.LENGTH_SHORT).show();
```



SecondActivity.java

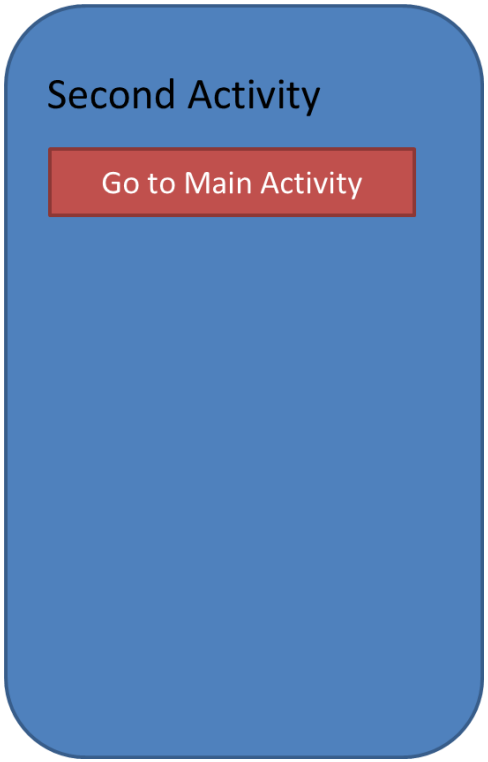
```
        //---get the Bundle object passed in---
        // Bundle bundle = getIntent().getExtras();
        //---get the data using the getString()---
        //Toast.makeText(this, bundle.getString(“name"), Toast.LENGTH_SHORT)
        //.show();
    }
    public void onClick(View view) {
        //---use an Intent object to return data---
        Intent i = new Intent();
        //---use the putExtra() method to return some value---
        i.putExtra("age", 25);
        setResult(RESULT_OK, i);
        finish();
    }
}
```

activity_second.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= “fci.thirdyear.test.SecondActivity">
```

activity_second.xml

```
<TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Second Activity"
        android:id="@+id/textView" />
<Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Go to Main Activity"
        android:id="@+id/button"
        android:onClick="onClick"/>
</LinearLayout>
```



Passing Data using an Intent Object

- While this application is not visually exciting, it does illustrate some important ways to pass data between activities.
- First, you can use the putExtra() method of an Intent object to add a name/value pair:

```
//---use putExtra() to add new name/value pairs---
i.putExtra("name", "Aya");
```

- The preceding statements add name/value pairs of type string to the Intent object.

Passing Data using an Intent Object

- Besides using the putExtra() method, you can also create a Bundle object and then attach it using the putExtras() method. Think of a Bundle object as a dictionary object—it contains a set of name/value pairs.
- The following statements create a Bundle object and then add two name/value pairs to it. The Bundle object is then attached to the Intent object:

```
//---use a Bundle object to add new name/values pairs---
Bundle extras = new Bundle();
extras.putString("Name", "Aya");
i.putExtras(extras);
```

Passing Data using an Intent Object

- 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:

```
//---get the data passed in using getStringExtra()---
Toast.makeText(this,getIntent().getStringExtra("Name"),
Toast.LENGTH_SHORT).show();
```

- In this case, you have to call the appropriate method to extract the name/value pair based on the type of data set.

Passing Data using an Intent Object

- To retrieve the Bundleobject, use the getExtras()method:

```
//---get the Bundle object passed in---
Bundle bundle = getIntent().getExtras();
```

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

```
//---get the data using the getString()---
Toast.makeText(this, bundle.getString("name"), Toast.LENGTH_
SHORT).show();
```

End of Lecture