Accelerating Information Technology Innovation
http://aiti.mit.edu

Lecture 4: Event Handling and Multi-page Applications

AITI Nigeria Summer 2012
University of Lagos.

Agenda

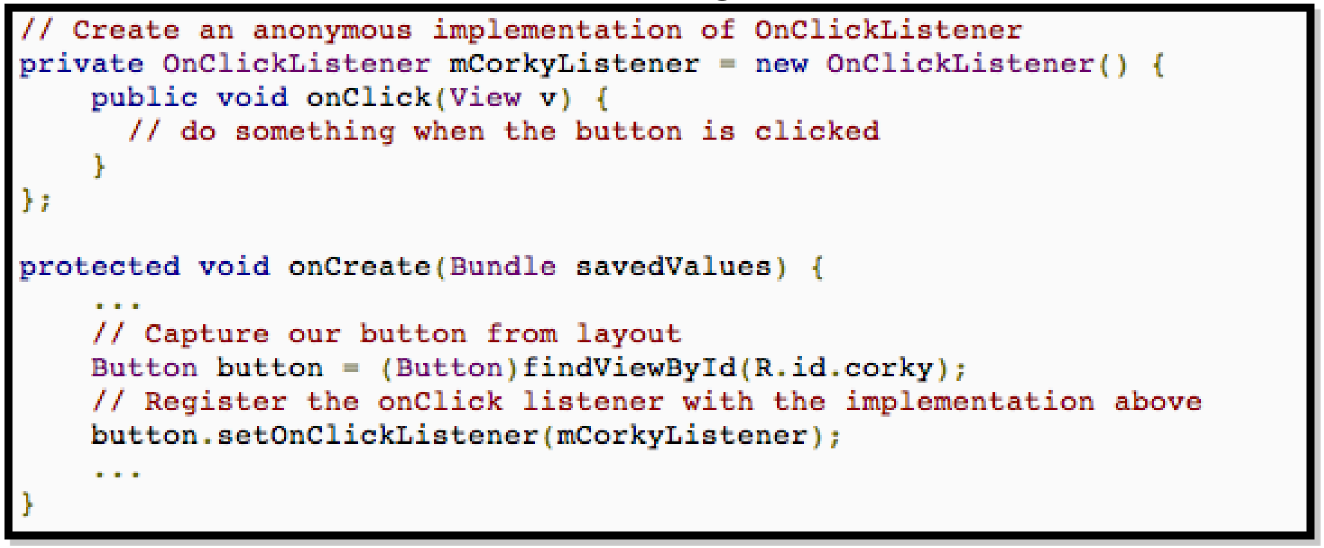
- EventListeners and callback methods
- Switching Activities
- Passing data between Activities

Handling events

- Listen to events using callback methods:
  - onClick()
  - onLongClick()
  - onFocusChange()
  - onKey()
  - onTouch()
  - onCreateContextMenu()

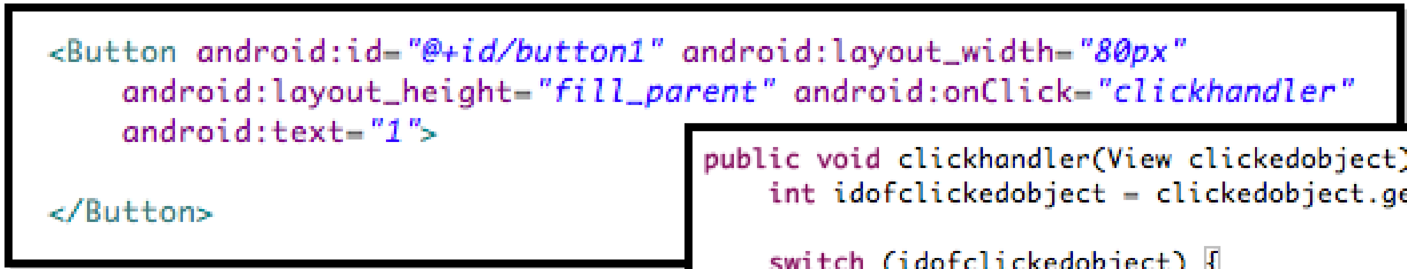
Example: Event-handling with Buttons

Method 1: Define call-backs using code

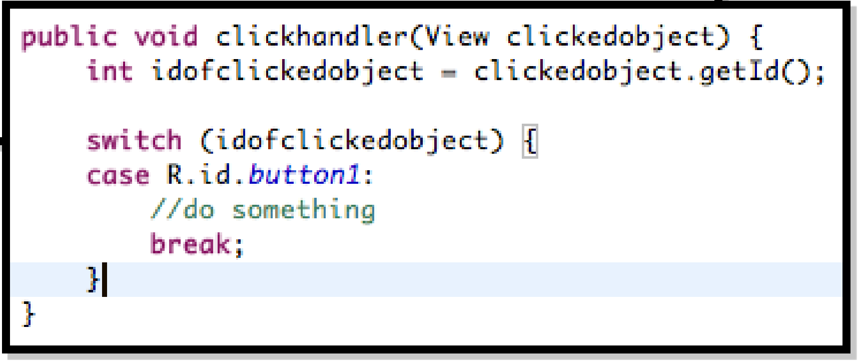


```
// Create an anonymous implementation of OnClickListener
private OnClickListener mCorkyListener = new OnClickListener() {
    public void onClick(View v) {
      // do something when the button is clicked
    }
};

protected void onCreate(Bundle savedValues) {
    ...
    // Capture our button from layout
    Button button = (Button)findViewById(R.id.corky);
    // Register the onClick listener with the implementation above
    button.setOnClickListener(mCorkyListener);
    ...
}
```

Method 2: Define call-backs in Layout XML files



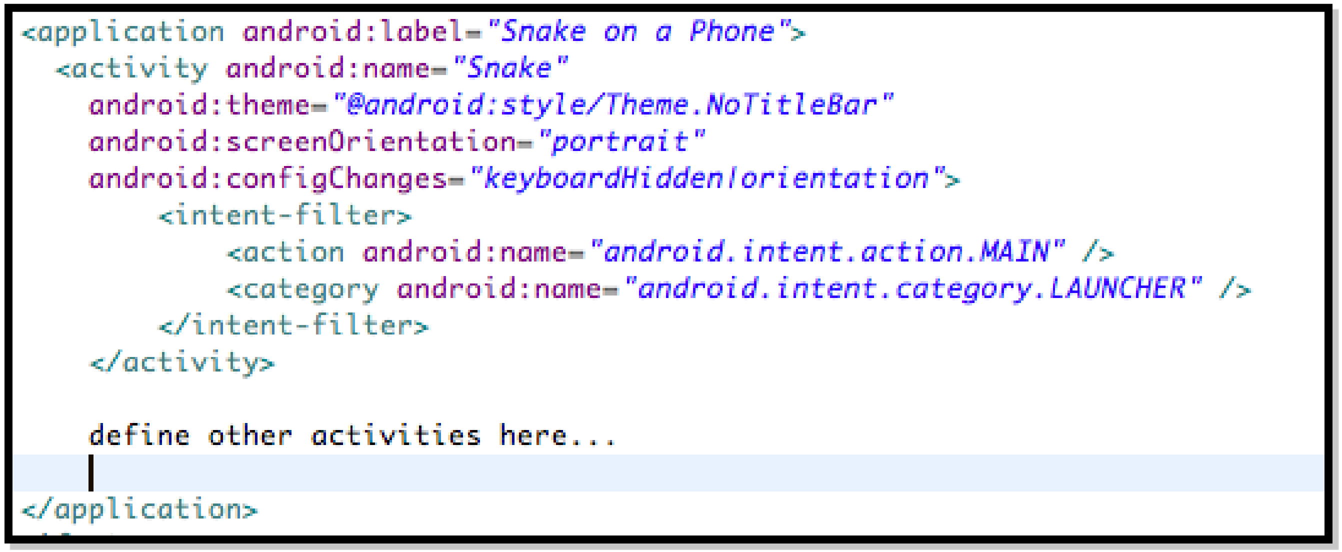
```
<Button android:id="@+id/button1" android:layout_width="80px"
    android:layout_height="fill_parent" android:onClick="clickhandler"
    android:text="1">

</Button>
```



```
public void clickhandler(View clickedobject) {
    int idofclickedobject = clickedobject.getId();

    switch (idofclickedobject) {
    case R.id.button1:
        //do something
        break;
    }
}
```

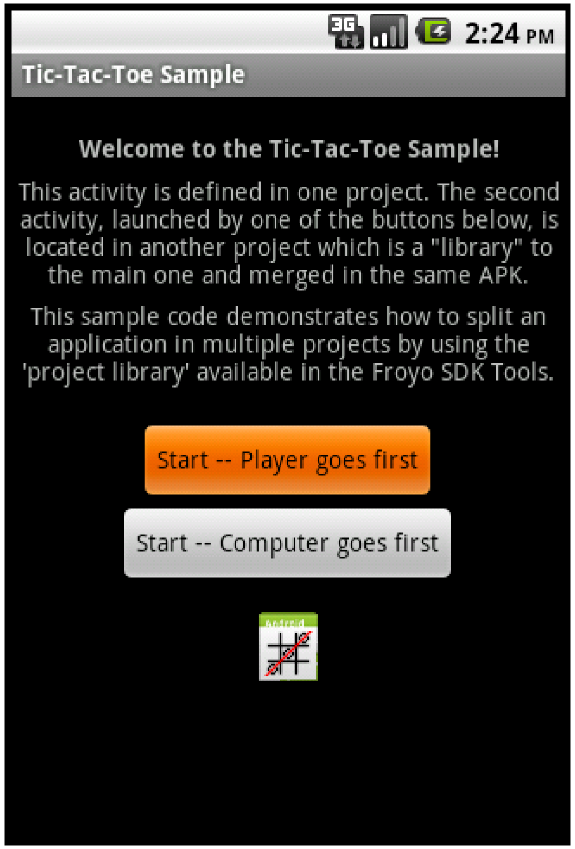
Multiple Activities

- An android application consists of multiple Activity objects
- Each Activity is like one “page” of the app
- Only one activity can be the *main* activity



```
<application android:label="Snake on a Phone">
  <activity android:name="Snake"
    android:theme="@android:style/Theme.NoTitleBar"
    android:screenOrientation="portrait"
    android:configChanges="keyboardHidden|orientation">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>

    define other activities here...

</application>
```

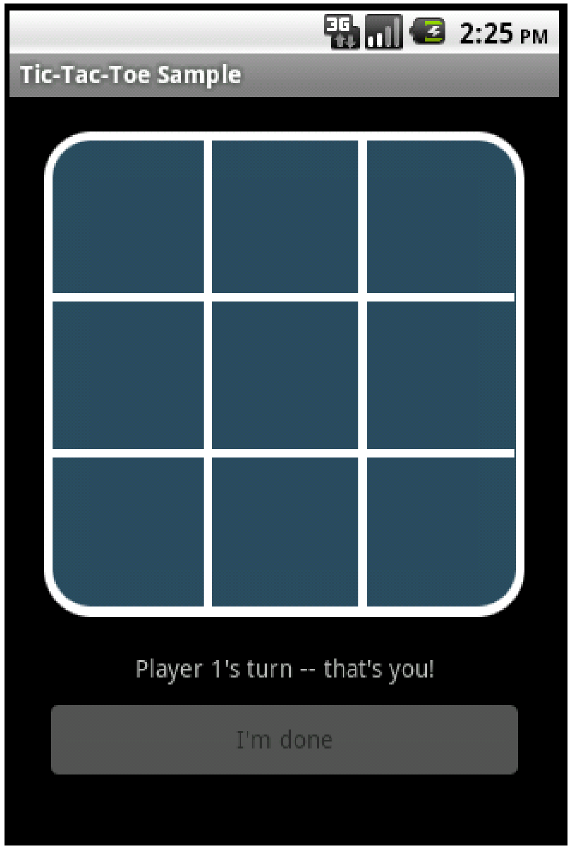
Multiple Activities, example:


Tic-Tac-Toe Sample

Welcome to the Tic-Tac-Toe Sample!

This activity is defined in one project. The second activity, launched by one of the buttons below, is located in another project which is a "library" to the main one and merged in the same APK.

This sample code demonstrates how to split an application in multiple projects by using the 'project library' available in the Froyo SDK Tools.

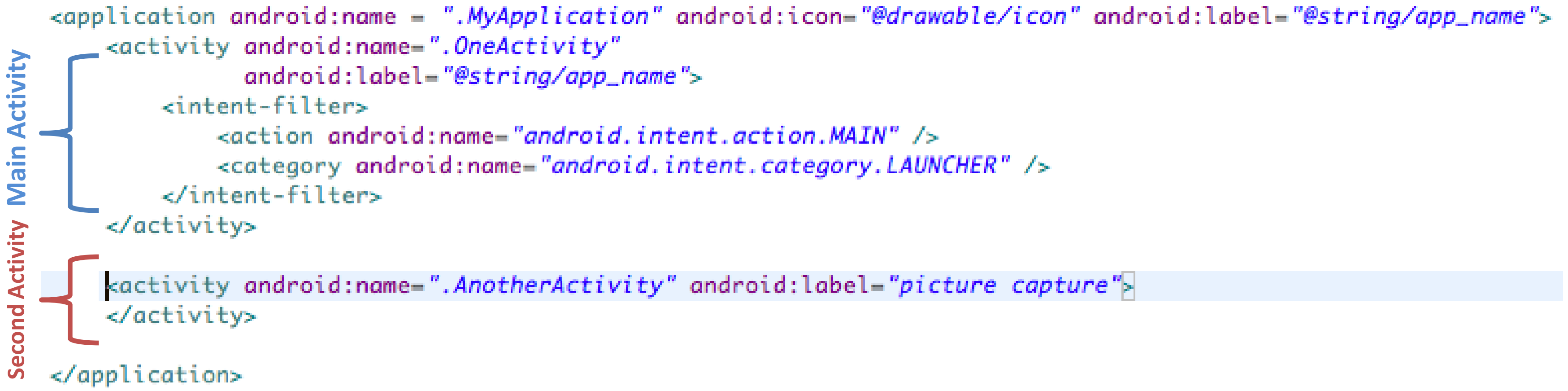


Main Activity (first thing you see when App starts)

Second Activity (clicking on a button on the Main Activity brings user to this one)

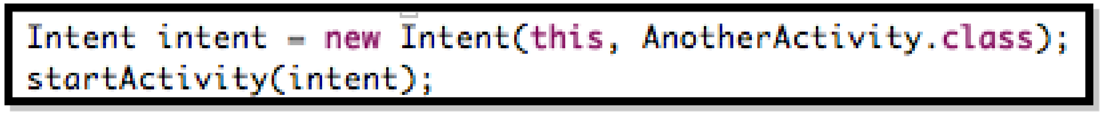
Switching between Activities

Step 1: Define all Activities in your App in the AndroidManifest.xml file



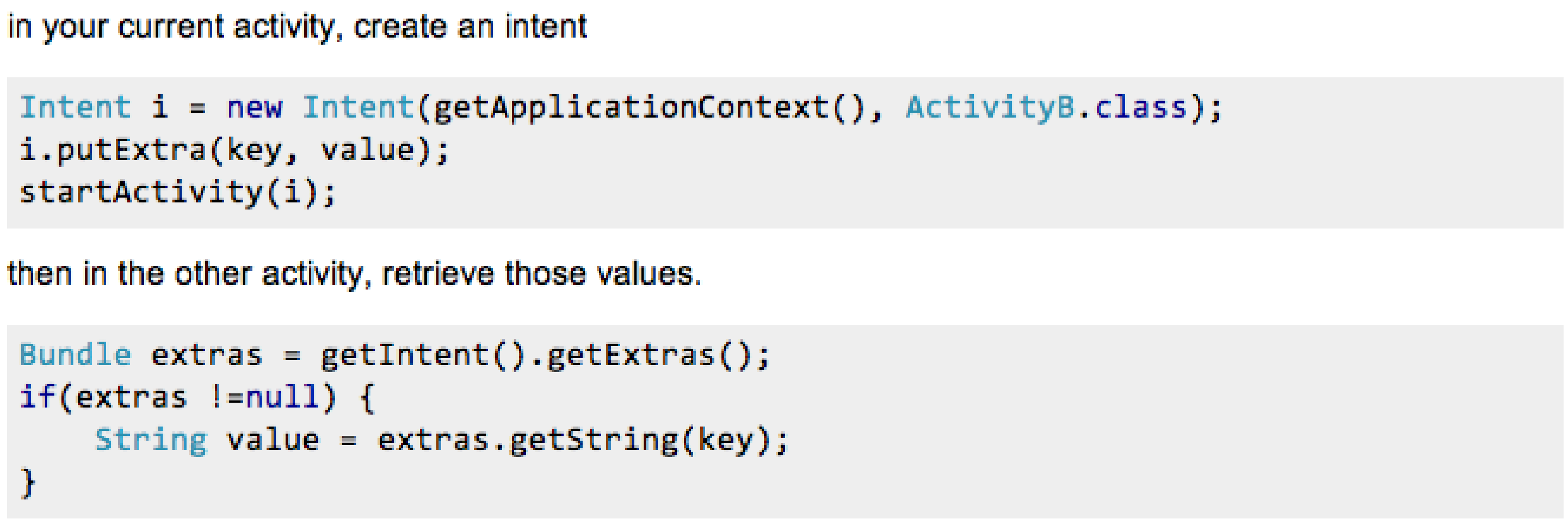
```xml
<application android:name = ".MyApplication" android:icon="@drawable/icon" android:label="@string/app_name">
    <activity android:name=".OneActivity"
              android:label="@string/app_name">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>

    <activity android:name=".AnotherActivity" android:label="picture capture">
    </activity>

</application>
```

Step 2: Switch from Main Activity to the activity defined in **AnotherActivity.class**, using Intent objects.



```java
Intent intent = new Intent(this, AnotherActivity.class);
startActivity(intent);
```

Passing data between Activities



in your current activity, create an intent

```
Intent i = new Intent(getApplicationContext(), ActivityB.class);
i.putExtra(key, value);
startActivity(i);
```

then in the other activity, retrieve those values.

```
Bundle extras = getIntent().getExtras();
if(extras !=null) {
    String value = extras.getString(key);
}
```

Note: you can use the putExtra method to add data in key value pairs to the Intent. The key must be a String object but the value can be any of the following: integer, integer[], float, float[], double, double[], String, String[], etc…

Then, you fetch that data in the second activity using the .getExtras().getString(key) approach.

Other ways to exchange data between Activities

- Intent approach is best for *primitive* data types that don't need to last forever (i.e. they are *not persistent*)
- For *primitive* types that need to last forever (i.e. *persistent* objects), use Preferences
- For *non-primitive* types that are *not persistent*:
  - Public Static Fields
  - Maintain global application state in the Application class (all Activity objects have access to this).
- For non-primitive types that are *persistent*:
  - Use ContentProvider, SQL Database on the phone, Files, etc.