Demo

We'll write an application based on our running example.

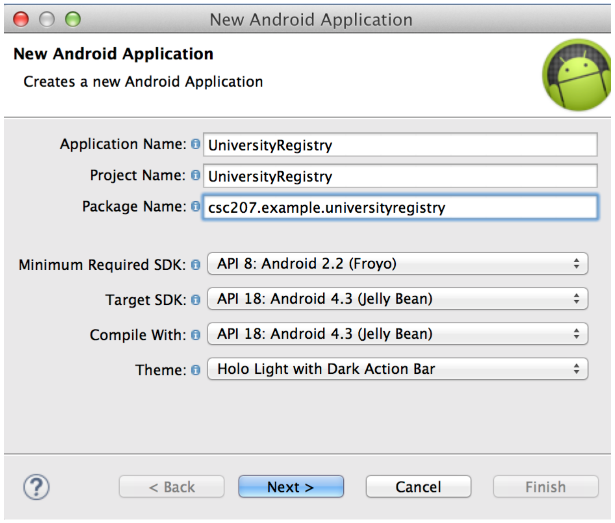
A Registry application that collects information about the population and stores the data in a file.

The remaining slides summarize the demo.

(See also the detailed slides for creating projects and adding `Activitys` to project on CDF and on your laptops.)

New Android Project

These are the settings you must use for your project.



New Android Application

New Android Application

Creates a new Android Application

Application Name: UniversityRegistry

Project Name: UniversityRegistry

Package Name: csc207.example.universityregistry

Minimum Required SDK: API 8: Android 2.2 (Froyo)

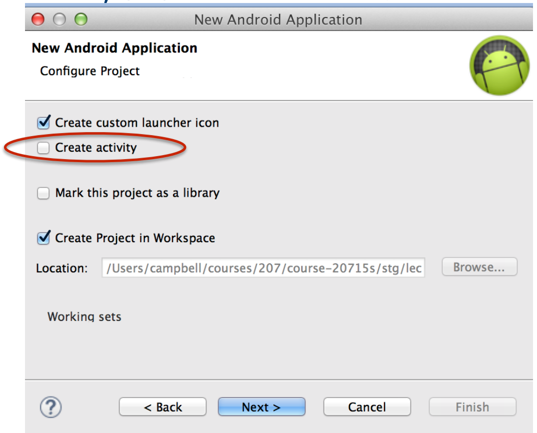
Target SDK: API 18: Android 4.3 (Jelly Bean)

Compile With: API 18: Android 4.3 (Jelly Bean)

Theme: Holo Light with Dark Action Bar

First Activity

If working on your own laptop, unselect the "Create activity" option, click Next, and then click Finish:



New Android Application

New Android Application

Configure Project

☑ Create custom launcher icon

□ Create activity

□ Mark this project as a library

☑ Create Project in Workspace

Location: /Users/campbell/courses/207/course-20715s/stg/lec

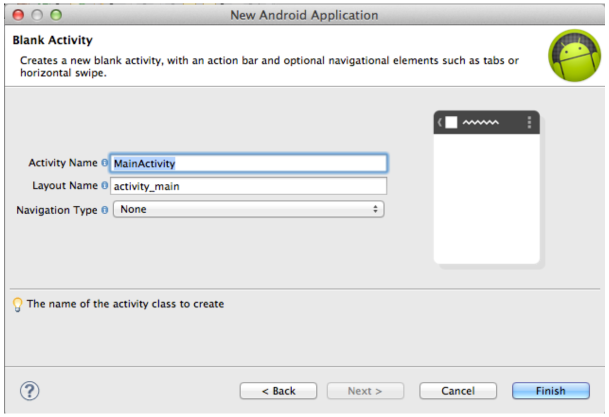
Working sets

The main `Activity`

An app is made up of multiple **`Activity`**s.

In this example, an **`Activity`** corresponds to a single screen with a user interface.

The first **`Activity`** is typically called the *main activity*.



New Android Application

Blank Activity

Creates a new blank activity, with an action bar and optional navigational elements such as tabs or horizontal swipe.

Activity Name: MainActivity

Layout Name: activity_main

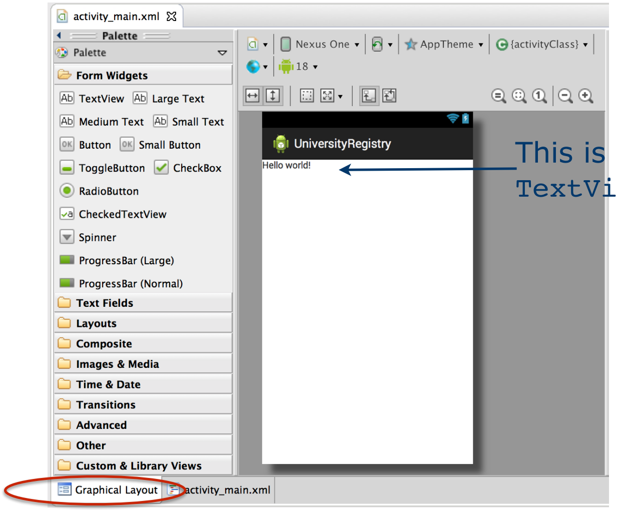
Navigation Type: None

The name of the activity class to create

On your laptop, use Empty activities.

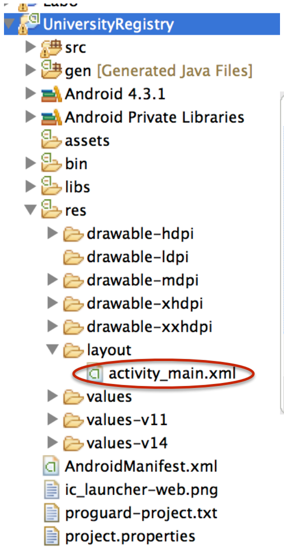
On CDF, use Blank activities.

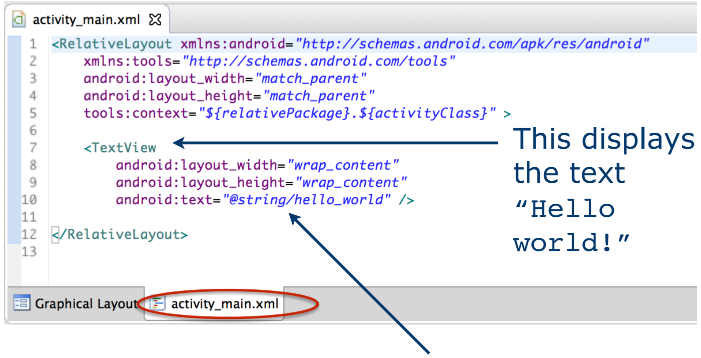
WYSIWYG view of main `Activity`



This is a `TextView`.

XML View of main `Activity`



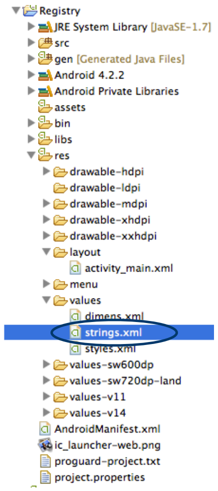


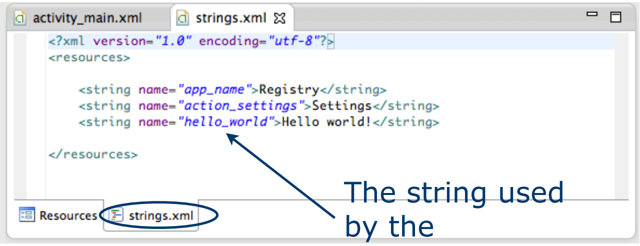
```xml
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="${relativePackage}.${activityClass}" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

</RelativeLayout>
```

This displays the text "`Hello world!`"

`@string/hello_world` is set to "`Hello world!`" in file `strings.xml`

`strings.xml`





```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">Registry</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>

</resources>
```

The string used by the `TextView`.

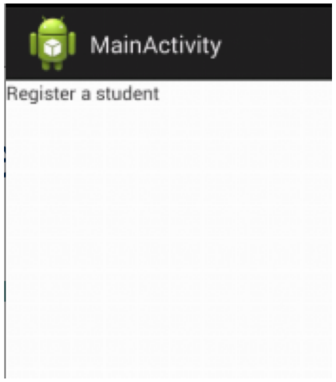
Changing the text displayed by `TextView`

We'll add a new string to `string.xml`:

```xml
<string name="register_student">Register a student</string>
```

We'll change the variable used by the `TextView` in `activity_main.xml`:

```xml
<TextView
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="@string/register_student" />
```



Changing the layout

Two types of layouts are:

- `LinearLayout`: components are arranged horizontally or vertically.

- `RelativeLayout`: components are arranged relative to something else (e.g., at the left edge of a component).

We'll edit `activity_main.xml` to use a `LinearLayout`:

```
<LinearLayout 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"
    android:orientation="vertical"
    tools:context=".MainActivity" >   ... </LinearLayout>
```

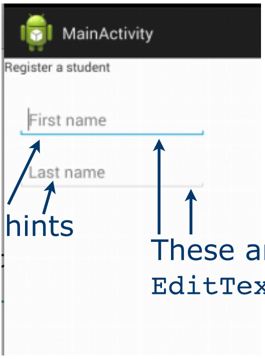
Adding two `EditText` fields

In activity_main.xml, below the `TextView`:

```
    <EditText android:id="@+id/first_name_field"
        …
        android:hint=“@string/first_name_text" />

    <EditText android:id="@+id/last_name_field"
        …
        android:hint=“@string/last_name_text" />
```

In strings.xml:

```
    …
    <string name="first_name">First name</string>
    <string name="last_name">Last name</string>
    …
```



hints

These are `EditText`s.

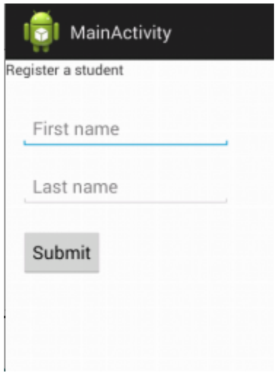
Adding a `Button`

In activity_main.xml, below the 2nd `EditText`:

```
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/submit_text"
        android:onClick="registerStudent"   />
```

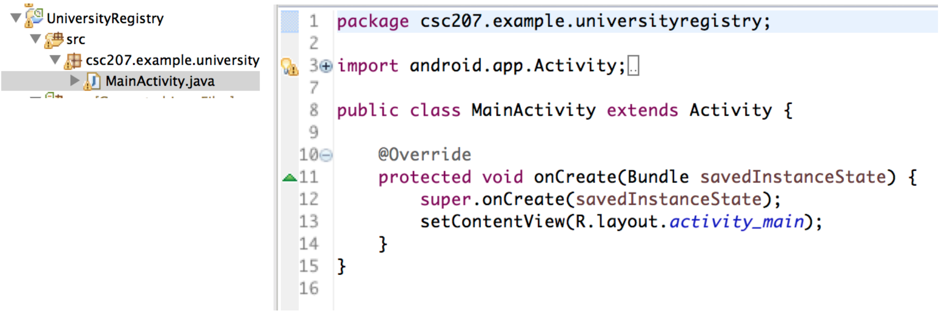
In strings.xml, add a new string for the button text:

```
    <string name=“submit_text”>Submit</string>
```



But nothing happens when we click the button! We need to implement method `registerStudent`.

`MainActivity.java`



```
package csc207.example.universityregistry;

import android.app.Activity;

public class MainActivity extends Activity {

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

Implementing registerStudent

When the button is clicked, we specified that the method registerStudent will be called:

```
// Note: an onClick method must return void and have one View parameter.
public void registerPerson(View view) {
   // Specifies the next Activity to move to: DisplayActivity.
   Intent intent = new Intent(this, DisplayActivity.class);

   // Gets the first name from the 1st EditText field.
   EditText firstNameText = (EditText) findViewById(R.id.first_name_field);
   String firstName = firstNameText.getText().toString();

   // Gets the last name from the 2nd EditText field.
   EditText lastNameText = (EditText) findViewById(R.id.last_name_field);
   String lastName = lastNameText.getText().toString();
   String[] name = {firstName, lastName};

   // See sample code; TODO: add EditTexts for DOB.
   Student student = new Student(…);

   // Passes the Student object to DisplayActivity.
   intent.putExtra("studentKey", student);
   startActivity(intent);     // Starts DisplayActivity.
}
```

Adding our Student class

File -> New -> Class

Add our Student class to the package.

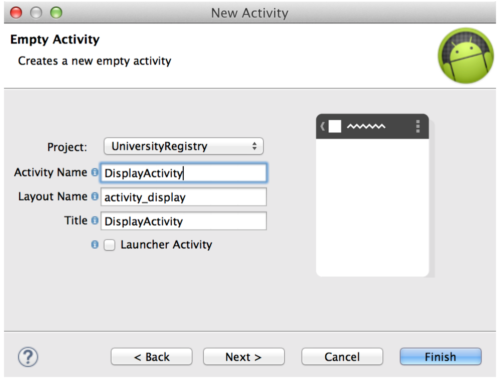
The Student class must implement Serializable (but we already have that!):

```
public class Student implements Serializable {

   // This UID was generated by Eclipse.
   private static final long serialVersionUID = 7865834197781540789L;

  ...

}
```

To pass complex objects from one Activity to another, the data must be serialized. Serializing data translates it into a storable form that can be converted back to original form.

Adding Another Activity

File -> New -> Other -> Android -> Android Activity



Empty Activity

Creates a new empty activity

Project: UniversityRegistry

Activity Name: DisplayActivity

Layout Name: activity_display

Title: DisplayActivity

□ Launcher Activity

This generates activity_display.xml and DisplayActivity.java.

Edit DisplayActivity.java

The onCreate method is called when the Activity is created. Get the Student object passed from MainActivity and display the name of the Student:

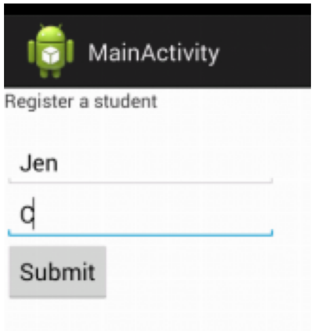
```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_display);

   // Gets the Intent passed from MainActivity
   Intent intent = getIntent();

   // Uses key "studentKey" to get Student object.
   Student student = (Student) intent.getSerializableExtra("studentKey");

   // Sets TextView to the Person's name.
   TextView newlyRegistered = (TextView) findViewById(R.id.registered_field);
   newlyRegistered.setText("Registered: " + student.toString());
}
```

Part of activity_display.xml:

```
  <TextView android:id="@+id/registered_field"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/newly_registered"/>
```

Transition between activities

`MainActivity`

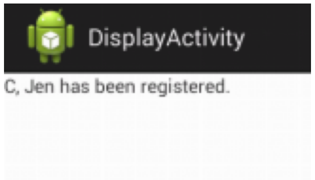


Register a student

Jen

C

`DisplayActivity`



C, Jen has been registered.

Complex objects pass via the `Intent` from one `Activity` to another must be serialized.
Multiple objects can be passed by giving each a different key.