cisc3650, spring 2012, lab III.1 / prof sklar
Introduction to Android

overview

This exercise will comprise part of the homework assignment for Unit III. The complete assignment will be given in class next week, and will be due 2 weeks later.

In this lab, you will write your first Android application and run it on a simulator. This is an exercise and does not need to be submitted.

You will find it helpful to install Android on the machine you use at home for doing your homework. You can find the Android Software Development Kit (SDK) available online for free here:

`http://developer.android.com/sdk/index.html`

We will be using Android integrated with the Eclipse Integrated Development Environment (IDE). You can find Eclipse available online for free here:

`http://www.eclipse.org/downloads/`

This lab was created by J. Pablo Munoz, our friendly graduate teaching assistant!

1 Creating a new project in Eclipse

As above, we are going to use the Eclipse IDE for programming Android. Eclipse conveniently provides us with all the necessary tools to develop Android applications. In order to run Eclipse, double-click on the Eclipse Icon: on the $Desktop$.

Once you have opened Eclipse, follow these steps to create a new Android Project:

1. In the menu bar go to **File** → **New** → **Android Project**. If Android Project is not on the list, select **Other**, and then choose **Android Project**. A new window will pop-up asking for some information about your project.
2. Give your project a name following this pattern: `<YourName>FirstProject`.
3. Double check that the "Create new project in workspace" radio button is selected, and that "Use default location" is checked. Click **Next**.
4. A list of all the available targets will be displayed. Check $Android2.1$. In this step, we are selecting the minimum SDK for our application. That is, we are telling Eclipse what is the minimum version of Android that our application can run on. Most of the Android devices run on Android 2.1 or 2.2. It is important to keep this in mind if you want to develop an application that can be distributed widely.
5. The next window asks for information about your project. Leave the name of the application the same as your project. Write "yourname.bc.edu" for the **package name**. Make sure that "Create Activity" is checked, and leave the default name given to your Activity (e.g. `<yourname>FirstProjectActivity`). We will learn more about Activities soon. For now, keep in mind that an Activity is one of the most important components in an Android application.
6. Click "Finish". Eclipse will generate a few files for you. Next we are going to explore some of these files.

2 Running your project

Eclipse provides us with a set of Android Virtual Devices (AVDs) in which we can test our code. For this lab, an AVD has been already created. To give you an idea of how it works, let's run our empty application by following these steps:

1. Open your main Activity (`<YourName>FirstProjectActivity.java`) using the project explorer. Your main Activity should be under the *src* folder, and inside the subfolder with *yourName.bc.edu*.

2. If this is the first time that you run your application, there are a few ways in which you can ask Eclipse to display the window to configure how your application will run in the AVD.
   (A) Click on the white arrow inside a green circle icon, and select "Run configurations";
   (B) go to the menu bar and go to **Run** → **Run configurations**;
   (C) right-click over your main Activity in the project explorer window, and select **Run as** → **Run configurations**.
   A new window will pop-up.

3. In the *Run configurations* window, double-click on *Android Application* to create another configuration. Give your configuration a name, and then check that your project is selected in the project tab. Otherwise, click on *Browse* and select your project.

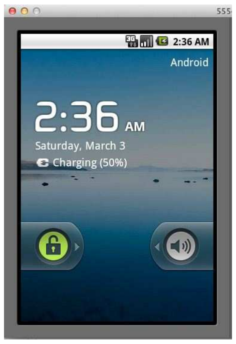
4. In the Android tab, and under *Launch Activity*, check that *Launch Default Activity* is selected.

5. Select the *Target* tab in the *Run configurations* window and select an AVD. You should have one AVD ready to be used. Check that *Automatic* is checked under the "Deployment Target Selection Mode" text.

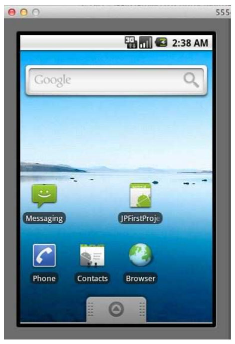
6. Click on *Apply* and then *Run*. If everything is correct, a new window with the Android emulator will appear. Give the emulator a few seconds to load the Android OS and install your application on it. Please be patient. It can take almost a minute for the emulator to be ready.
   Note that once the emulator has started up the first time, you do not need to close it. You can edit your program, re-compile and re-run it without re-starting the emulator.

3 The Android Virtual Device (AVD) running your app

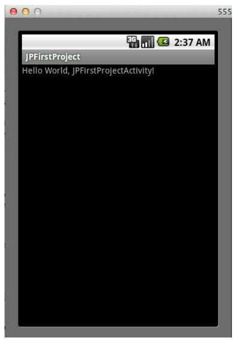
The steps below illustrate running your application on the simulator:



(1)



(2)



JPFirstProject

Hello World, JPFirstProjectActivity!

(3)

4 Improving the appearance of your app

As you can see in the AVD, your application is really simple. It only displays the name of the app, and the infamous "Hello World" message common to any programmer exploring a new programming environment. Our next goal is to change this message for our custom text.

1. Using our project explorer, let's take a look at the `string.xml` file located in the `res/values/string.xml` subfolder of our project.
2. You might observe that in the lower part of this file there are two tabs: *resources* and *xml* (`string.xml`). Eclipse provides us with a great interface to manage xml files. The resource tab is opened by default. If this is not the case, choose the *resource* tab.
3. Click on the "hello" resource, which is of type String. On the right side, the name and the value of the resource will be displayed. Change the value for the message that you would like to be displayed at the top of your application.

The resources, like the one you just modified, can be displayed using *layouts*. Let's take a look at the basic layout that Eclipse has created for us.

1. Open the `main.xml` file located in the `res/layout/` subfolder. This file was also created by Eclipse as a basic initial structure for our application. Similar to when we opened the `string.xml` file, Eclipse gives us to alternative views for this file: a plain xml view, and *Graphical Layout* view.
2. Observing the xml file, you can see the pattern that your application is using to display elements on the screen. For now, we have a *LinearLayout* xml tag, with several attributes, and inside it, there is a *TextView* xml tag. In this last tag we can observe that our resource *@string/hello* is being used. You will need to add an *id* to this text view, so later we can reference it from our Java code in your Activity. Go ahead and add `android:id="@`id/message+ as the first line inside the *TextView* xml tag. You should get something like this:

```
...
<TextView
   android:id="@+id/message"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="@string/hello" />
...
```

3. Let's go back to the *Graphical Layout* tab to see how easy is to add graphical components to our application. To go back to the *Graphical Layout*, click on the tab at the bottom of the opened `main.xml` file.
4. Once we are in *Graphical Layout* view, we can start adding other elements by dragging them from the *Palette* located on the left side of the window. You are going to add some elements to your application using the *Graphical layout* interface. Later you can see how Eclipse helps you create the xml code for these elements. Try adding a button by dragging it from the *Palette* and dropping it on the screen below the message text. In the next section we will give this button some functionality.
5. Once the new button is on the screen, right click on top of it, and select *Edit ID*. A small window will pop up. Write in the textfield provided the new id for your button. For now, give your button the ID: *yourButton*.

5 Changing the code in your Activity to change the message displayed on the screen

This part of the lab will give you an idea of how the elements interact in the Android OS graphical environment, and how we program them using the Java programming language. In a previous lab, you created an HTML form that helped the user to put together different types of information into a text area. Now, you are going to do something similar with your Android app. The goal is to change the text that is displayed by a *TextView* element in your application.

Open the main Activity of your application (`<yourname>FirstProjectActivity.java`). At this point, it is important to have a better understanding of what Activities are in the context of the Android OS. As we mentioned at the beginning of this lab, Activities are some of the most important components of an Android application. Our *FirstProjectActivity* class is extended from the *Activity* class, which has some methods that, among other things, let us indicate how the activity should react when changing to a different state. Activities have a life cycle, that is, they can be in a particular state at a moment in time. For now, your only concern will be the state in which the activity is created.

onCreate() is the method that is called the moment that your application is run by the user. We can use this method to indicate what we want to happen when the application is created. Eclipse has already created a basic `onCreate()` method which for now contains the following lines:

```
super.onCreate( savedInstanceState );
setContentView( R.layout.main );
```

The second line uses the `main.xml` file that we manipulated before in order to display some widgets on the screen. Let's add some code to the *onCreate*() method.

1. Add a *TextView* member to your Activity class and call it *message*:

   ```
   ...
   public class <yourName>FirstProjectActivity extends Activity {
      TextView message;
   ...
   ```

2. Add the following lines inside the *onCreate*() method after the line where you are setting the content view of your application (`setContentView(R.layout.main)`);

   ```
   ...
   message = (TextView)findViewById( R.id.message );
   Button yourButton = (Button)findViewById( R.id.yourButton );
   yourButton.setOnClickListener( new OnClickListener() {
      @Override
         public void onClick( View arg0 ) {
            message.setText("Your new message");
         }
   });
   ...
   ```

3. . Run your application. If you don't have any errors, you will be able to change the message of your *TextView* by clicking the button.

In future labs, you will continue adding more functionality to your application.

Challenge exercises

After you have completed the above exercise, you should spend some time playing with Android in order to get to know it better. For information, you can follow any of the tutorials listed below and/or look things up in the reference guide. You could also get a book. The one I've been using is called *Learning Android*, by Marko Gargenta (O'Reilly publishing, (c) 2011).

Here are a few small extensions to the exercise above that you might try:

- Change the message (label) on the button when you click on it.
- Change the background color on the button when you click on it.
- Change the background color of the Android window when you click on the button.
- Create a second button. When you click on the second button, change the content of the message in the TextView to a different message (different from the message that appears when you click on the first button).

References

There are MANY online references and tutorials for Android. The "official" ones are on the Android developer site:

`http://developer.android.com/`

which is where you can DOWNLOAD the SDK (as above) and find many resources.

The REFERENCE GUIDE is here:

`http://developer.android.com/reference/packages.html`

The TUTORIALS are here:

`http://developer.android.com/resources/browser.html?tag=tutorial`