Lab #1
Android Application Development
Advanced Hello World App
Due: 2/10/2017

1. Create a new Android Project

1. Open Android Studio
2. Click the menu **File** -> **New** -> **Project**.
3. Name the Application "**Lab1**"
   Fill in the properties:
   - **Application Name** = Lab1
   - **Package Name** = edu.csulb.android.lab1
   - **Create Activity** = Lab1Activity
   - **Minimum SDK** = 15
4. Choose a Blank Activity and name it **"Lab1Activity**"
5. Click "Finish."

2. Run "Hello World" on the Emulator

Before we can run the application, we need to setup an Android Virtual Device (AVD), or emulator, to run it on:

- Select the menu **Tools -> Android** -> "**AVD Manager**."
- Click the **Create** button.
- Give your AVD a name.
- Select the target build that we would like to run the application on.
- Click **Create AVD** and close out the AVD Manager.

We're now ready to run our application.

- Select the menu **Run** -> **Run**

  *Note: The emulator may take a long time to start up. Another way to run your application is to right-click on the project in the Package Explorer, then select **Run As** -> **Android Application**.* You can interact with the emulator using the mouse just like you would with a device. To get started, press the Menu

key to see the home screen.

Congratulations! You've just created and an Android Application.

3. Getting the User's Name

To get the user's name, you will be creating an Activity class, which will allow the user to enter their name into a text field and press a button when finished to proceed to the "Hello" greeting Activity. There are three separate steps to accomplish here. You must first layout your user interface in XML. Then you must create the Activity class to parse the input from the user and initiate the Lab1 Activity. Finally, you will have to reconfigure the application to use your new name retrieval Activity on startup.

4. Create a New Activity Class

- Create a new class that extends android.app.Activity class and implements android.view.View.OnClickListener interface. Implement the OnClickListener interface by creating a method stub with the following signature: public void onClick(View v). We'll fill in this method later.

The new class will look like this:

```
package edu.csulb.android.lab1;

public class GetName {

   }
```

Modify it so it extends Activity & implements android.view.View.OnClickListener: The finished class code looks like this so far:

```
package edu.csulb.android.lab1;

import android.app.Activity;
import android.view.View;

public class GetName extends Activity implements
android.view.View.OnClickListener{

     public void onClick(View arg0) {
          // TODO Auto-generated method stub

     }
}
```

- Declare a data member of type android.widget.EditText like this:

```
android.widget.EditText name;
```

  This will hold a reference to the text field in which the user will enter their name, the same one that you added to the name_getter.xml layout file.

- Add a method with the following signature: public void onCreate(Bundle savedInstanceState.

```
public void onCreate(Bundle savedInstanceState) {
      }
```

  This method will be called when the Activity starts and is where initialization of local and member data will be done. Inside this method perform the following: make a call to super.onCreate(savedInstanceState);

```
public void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          }
```

  This should always be done and is to ensure that any necessary parent class initializations are performed. Also make a call to this.setContentView(R.layout.name_getter);

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

  When you created the XML layout file earlier, the Android Eclipse Plug-in automatically added a static constant to the static R.layout class in the R.java file under the /gen folder. This constant variable has the same name of the file and it's value is used to identify the layout file. This call tells Android to create a screen based off of the layout file.

- Make a call to this.findViewById(R.id.**<EditText id>**) and set your EditText member variable equal to the return value.

  <EditText id> should be replaced with the android:id that was specified in the name_getter.xml layout file for the EditText element. You will have to explicitly cast the return value to the type of EditText as this method only returns objects of type Object. This static constant value was added in the same way as it was done for the R.layout.name_getter value and serves the same purpose.

```
public void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          this.setContentView(R.layout.name_getter);

          name = (EditText) this.findViewById(R.id.editText1);
          }
```

- Make a call to this.findViewById(R.id.**<Button id>**) and set a local android.widget.Button reference equal to the return value.

  <Button id> should be replaced with the android:id that was specified in the name_getter.xml layout file for the Button element.

```
public class GetName extends Activity implements
android.view.View.OnClickListener{

     android.widget.EditText name;
     android.widget.Button button;

     public void onClick(View arg0) {
          // TODO Auto-generated method stub
     }
      public void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          this.setContentView(R.layout.name_getter);

          name = (EditText) this.findViewById(R.id.editText1);
          button = (Button) this.findViewById(R.id.button1);
      }
     }
```

- Make a call to **button**.setOnClickListener(this); **button** should be replaced with the local Button reference you just retrieved.

```
public void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          this.setContentView(R.layout.name_getter);

          name = (EditText) this.findViewById(R.id.editText1);
          button = (Button) this.findViewById(R.id.button1);

          button.setOnClickListener(this);
      }
}
```

- Fill in the onClick method stub:

Retrieve the user entered text from the text field and keep it in a local String variable. Create an android.content.Intent object: new Intent(this, Lab1Activity.class).

We will use the Intent object to start the Lab1Activity greeting activity and pass it information. I will discuss Intents in more depth in later labs. Essentially they are used to interact with other application components.

We are using Intent.putExtra (<key>, <value>) method to pass the user entered name to the Lab1Activity greeting Activity. This method functions like a hashmap, where values can be stored by associating them with a string key and later retrieved with the same key. You can retrieve this hashmap later by calling getExtras(). Make a call to **<intent>**.putExtra(**<key>**, <**value>**);

<intent> should be replaced with the intent object you just created.

<key> should be replaced with a string you will use to access the user's name.

<value> should be replaced with the user's name obtained from the text field. To obtain the text from the EditText object, you must call to **<editText object>.**getText().toString();

The finished code after these steps looks like this:

```
public void onClick(View arg0) {
          Intent myIntent = new Intent(this,Lab1Activity.class);
          myIntent.putExtra("uname",name.getText().toString());
     }
```

- Next, make a call to this.startActivity(**<intent>**);

```
public void onClick(View arg0) {
          Intent myIntent = new Intent(this,Lab1Activity.class);
          myIntent.putExtra("uname",name.getText().toString());
          this.startActivity(myIntent);
     }
```

This command will initiate the switch to the Lab1Activity greeting Activity.

5. Reconfigure the Lab1 Application

The Android Manifest contains information on all of the Application's components, including which component should be used in different scenarios. We need to tell

the application to use the new activity on startup instead of the Lab1 Activity which it was previously using.

- Begin by Double-Clicking the AndroidManifest.xml file to open it.

- Like the Layout file there is also a Manifest Editor. Switch to the XML Editor by clicking the **AndroidManifest.xml** tab along the bottom of the editor.

- Find the first opening **<activity>** tag and change the attribute labeled android:name equal from ".Lab1Activity" to "**<New Activity Class>**" Replace **<New Activity Class>** with the full name of your new Activity Class.

  ```
  android:name=".GetName"
  ```

- Look at the package attribute in the **<manifest>** tag, this declares the top level package for the application. If your activity resides in a sub-package, then you must also include the sub-package in the name of your Activity class.

- The Intent Filter tag you see nested inside the Activity tag is what tells the application that this Activity is the Main, or startup, Activity. Add the following opening and closing Activity tag pair underneath the Activity tag pair you just modified, nested inside the Application tag:

  ```
  <activity android:name="Lab1Activity" ></activity>
  ```

  This declares to the Android device that the application has an Activity component named Lab1Activity. If you don't add this tag, Android will not let you launch this Activity.

  The finished **AndroidManifest.xml** file should look like this:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="edu.csulb.android.lab1"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="14" />

    <application
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name" >
```

```
        <activity
            android:name=".GetName"
            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="Lab1Activity" ></activity>

    </application>

     </manifest>
```

At this point, you should be able to try running the application. The application should start up and display the name retrieval activity you just created. You should be able to enter your name in the text field and hit the button, after which the old Lab1Activity greeting activity should appear.

9. Greeting the User

Now that we've got the name retrieval activity completed, let's update the Lab1Activity greeting Activity to print out the name the user entered. In the **OnCreate** method of **Lab1Activity.java**:

Make a call to this.getIntent.getExtras() to retrieve the hashmap in which you placed the user entered name. This will return an android.os.Bundle object which acts like a hashmap.

You should check to ensure that this value is not null. Retrieve the user entered name from the Bundle object.

Make a call to the bundle's getString(**<key>**) method.

<key> should be replaced with the key String you used to put the user entered name into the hashmap in the name retrieval Activity's onClick method.

After this part the method looks like this:

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

```
        Bundle myInput = this.getIntent().getExtras();
    }
```

- Set a reference to the **TextView** object in your **main.xml** layout file.

```
android:id="@+id/textView2"
```

main.xml now looks like:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/textView2"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello" />

</LinearLayout>
```

Set the text of the TextView object to say Hello **<name>**!
<name> should be replaced with the user entered name that you retrieved from the bundle object.

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

        Bundle myInput = this.getIntent().getExtras();

        TextView t=new TextView(this);
        t=(TextView)findViewById(R.id.textView2);
        t.setText("Hello " + (myInput.getString("uname")));
    }
```

Run your application!

How to turn your work in:

On Android Studio, close the project (File -> Close Project). Zip your application source folder and submit to beachboard dropbox.