

Android Development Tutorial

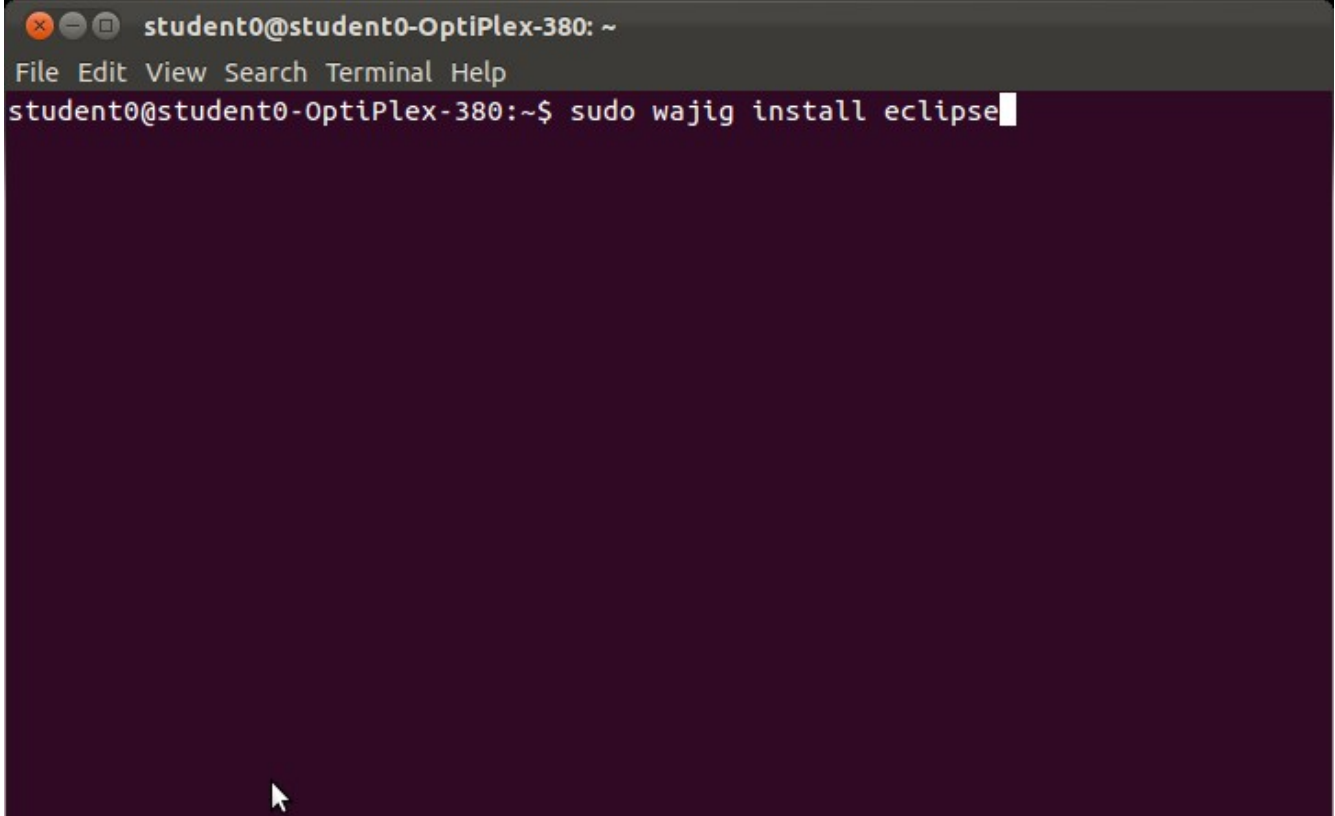
1. Install Eclipse

For install eclipse need start Terminal and type:

```
$ sudo wajig install eclipse
```

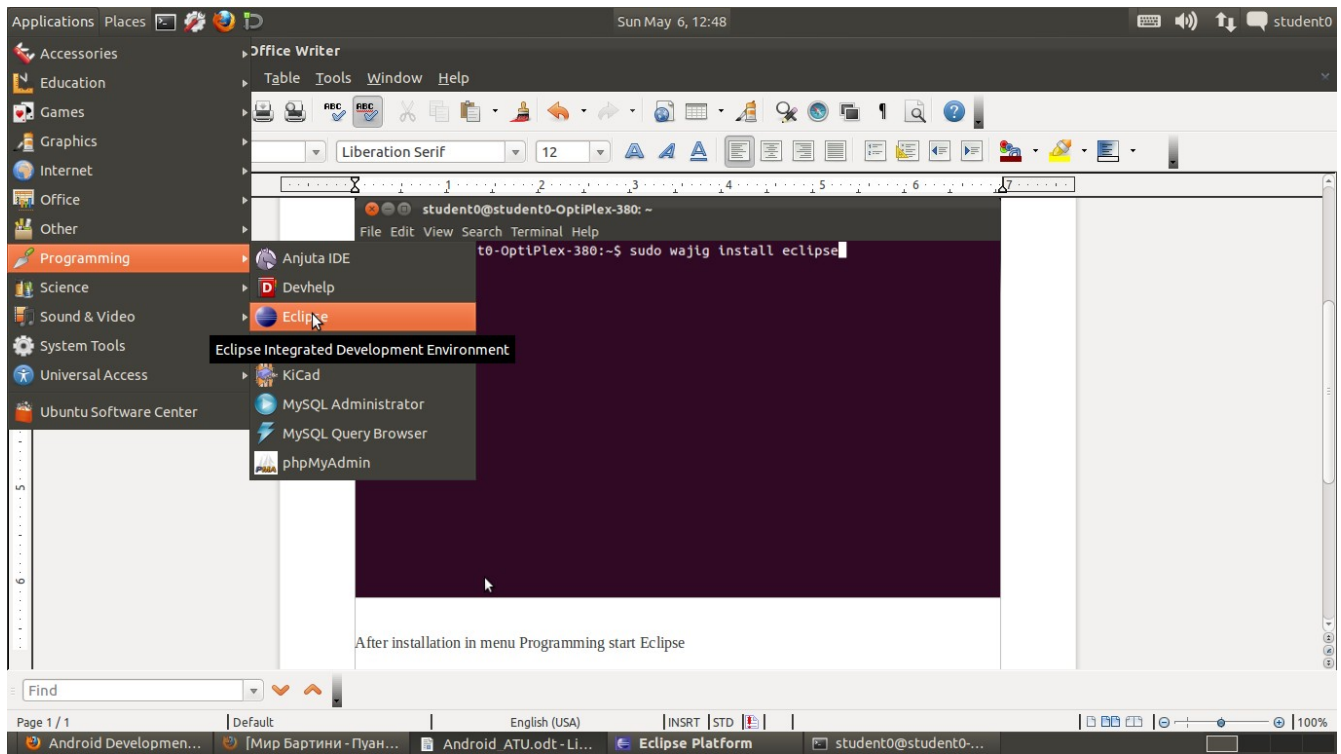
or for root Terminal

```
# wajig install eclipse
```

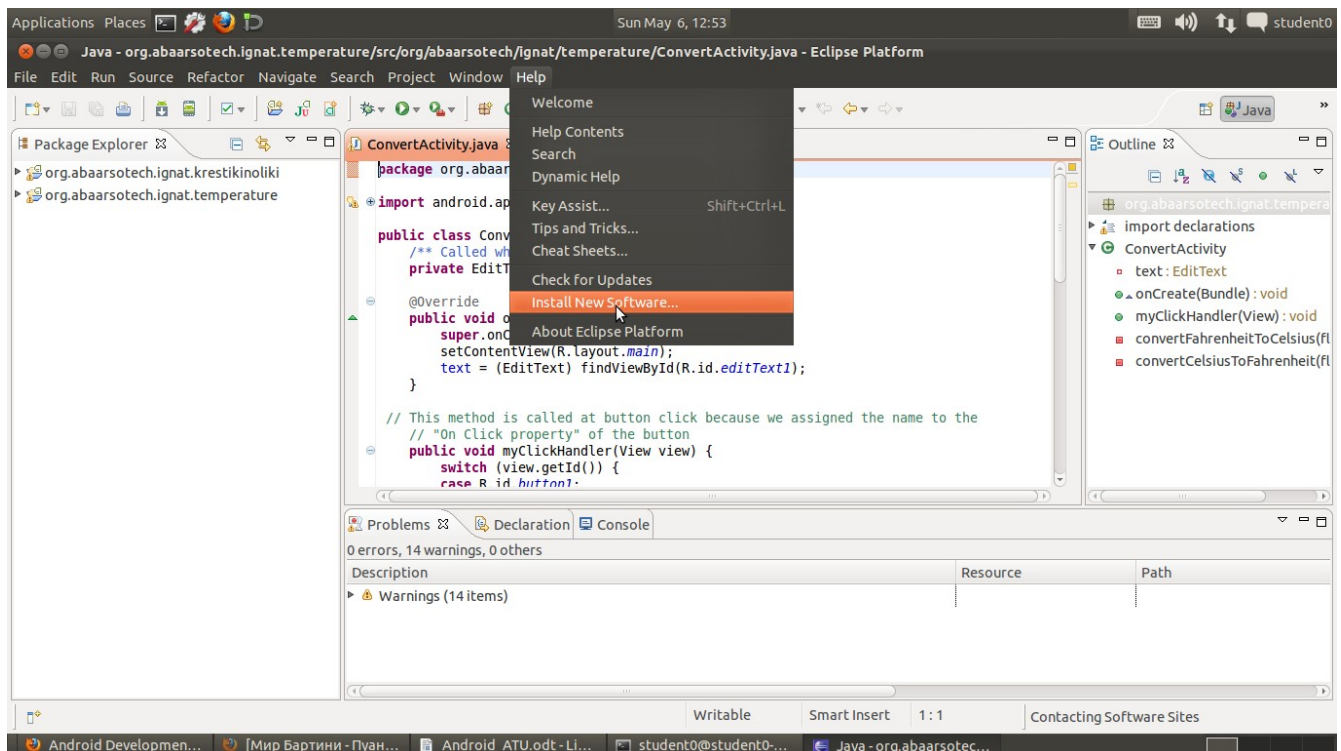
A screenshot of a terminal window. The title bar shows 'student0@student0-OptiPlex-380: ~'. The menu bar includes 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal text shows the command 'student0@student0-OptiPlex-380:~\$ sudo wajig install eclipse' with a cursor at the end of the line. The terminal background is dark purple.

```
student0@student0-OptiPlex-380: ~  
File Edit View Search Terminal Help  
student0@student0-OptiPlex-380:~$ sudo wajig install eclipse
```

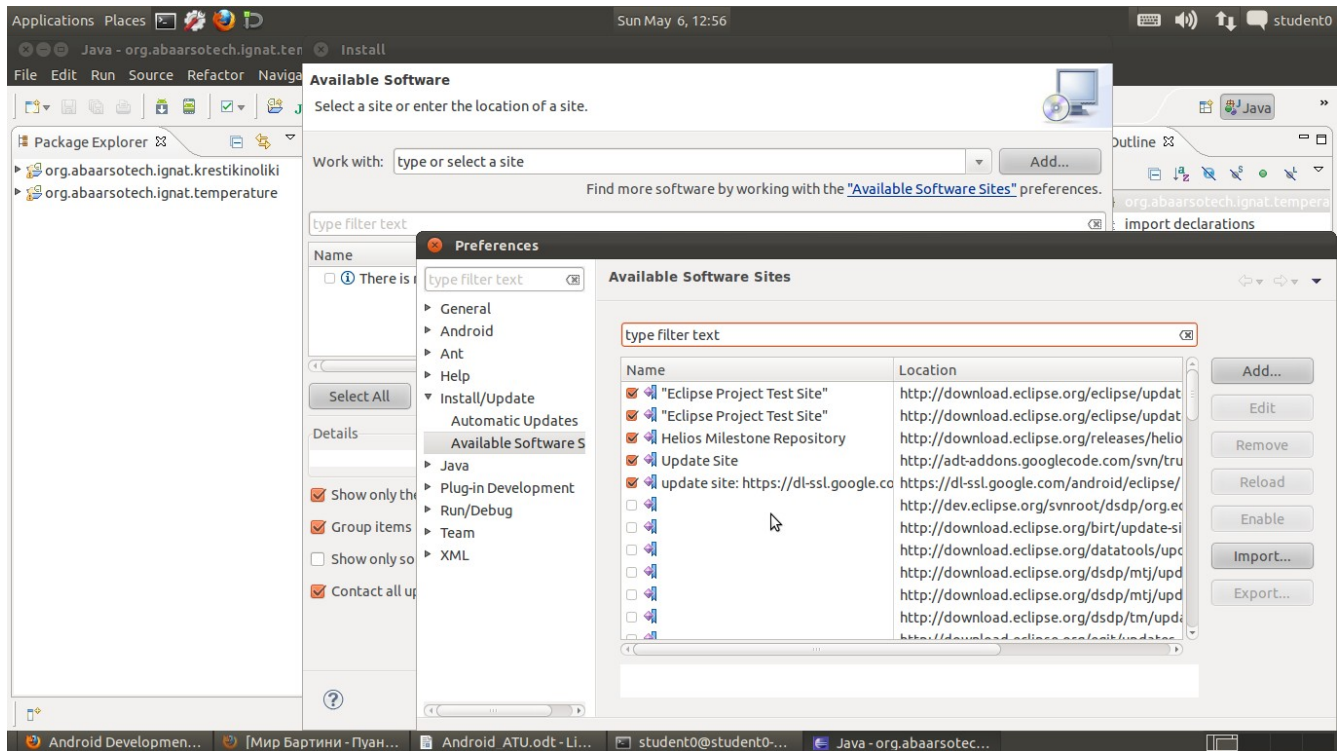
After installation in menu Programming start Eclipse



When Eclipse started make update in menu Help → Install new software

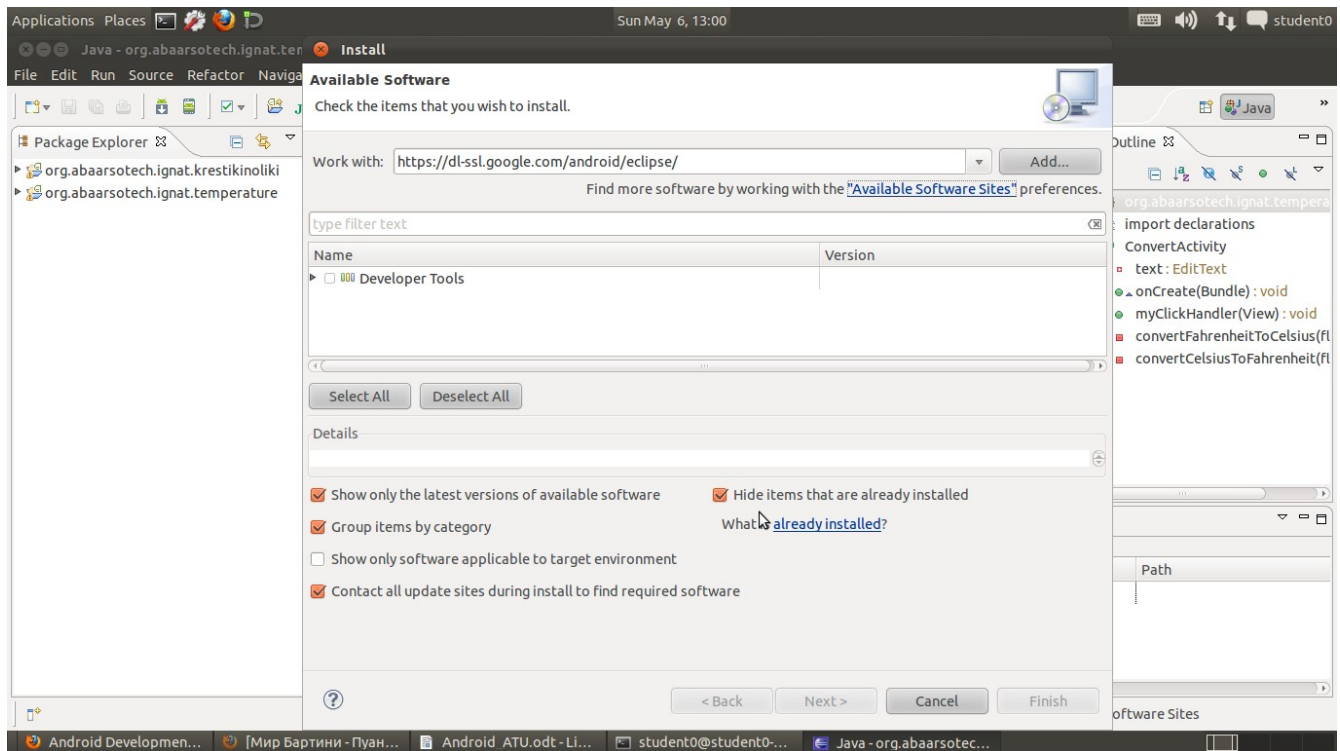


Use Available Software Sites for mark all location <http://download.eclipse.org>

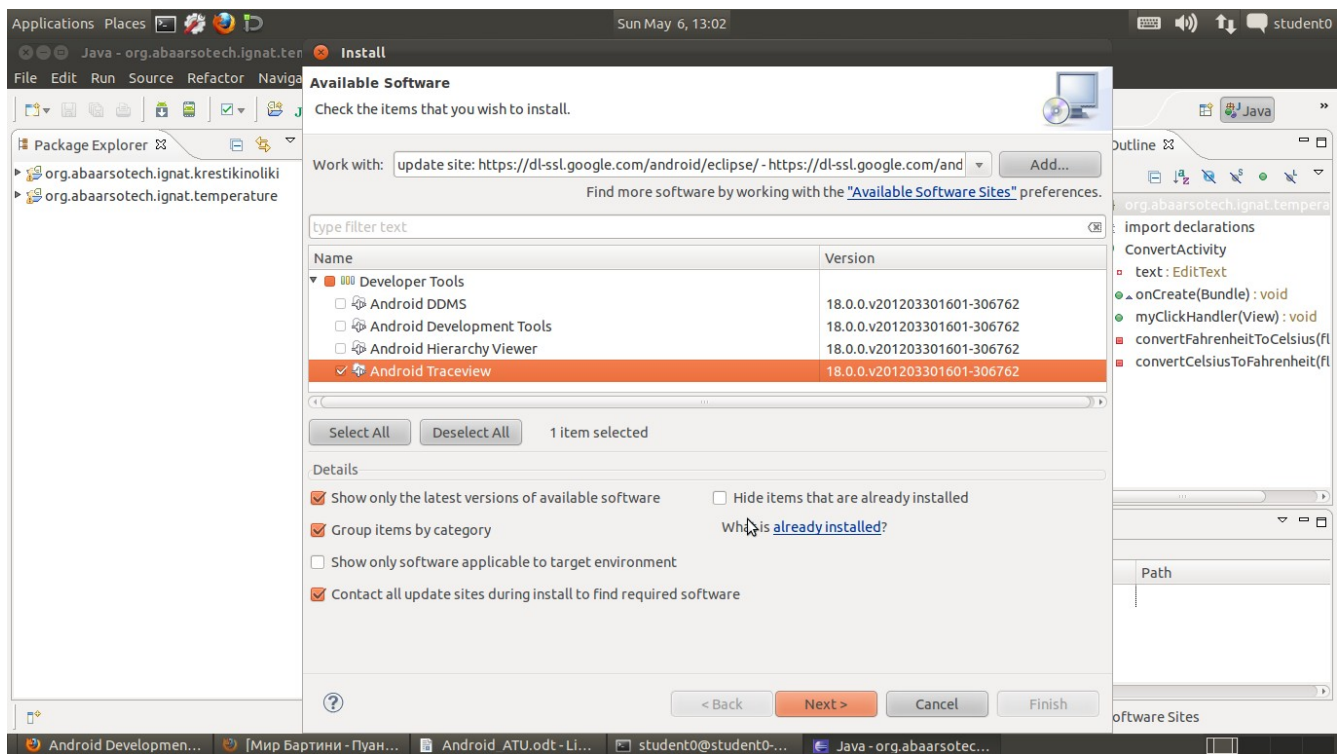


2. Install ADT Plug-ins and Android SDK

Use the Eclipse update manager to install all available components for the Android Development Tools (ADT) from the URL .



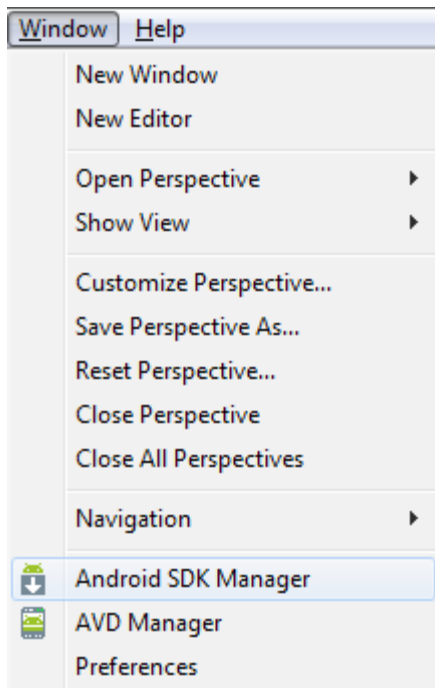
Per time need install just one package (internet not faster for select all packages)



After installing full DevelopmentTools you ready for start make Android programm.

3. Install a specific Android version

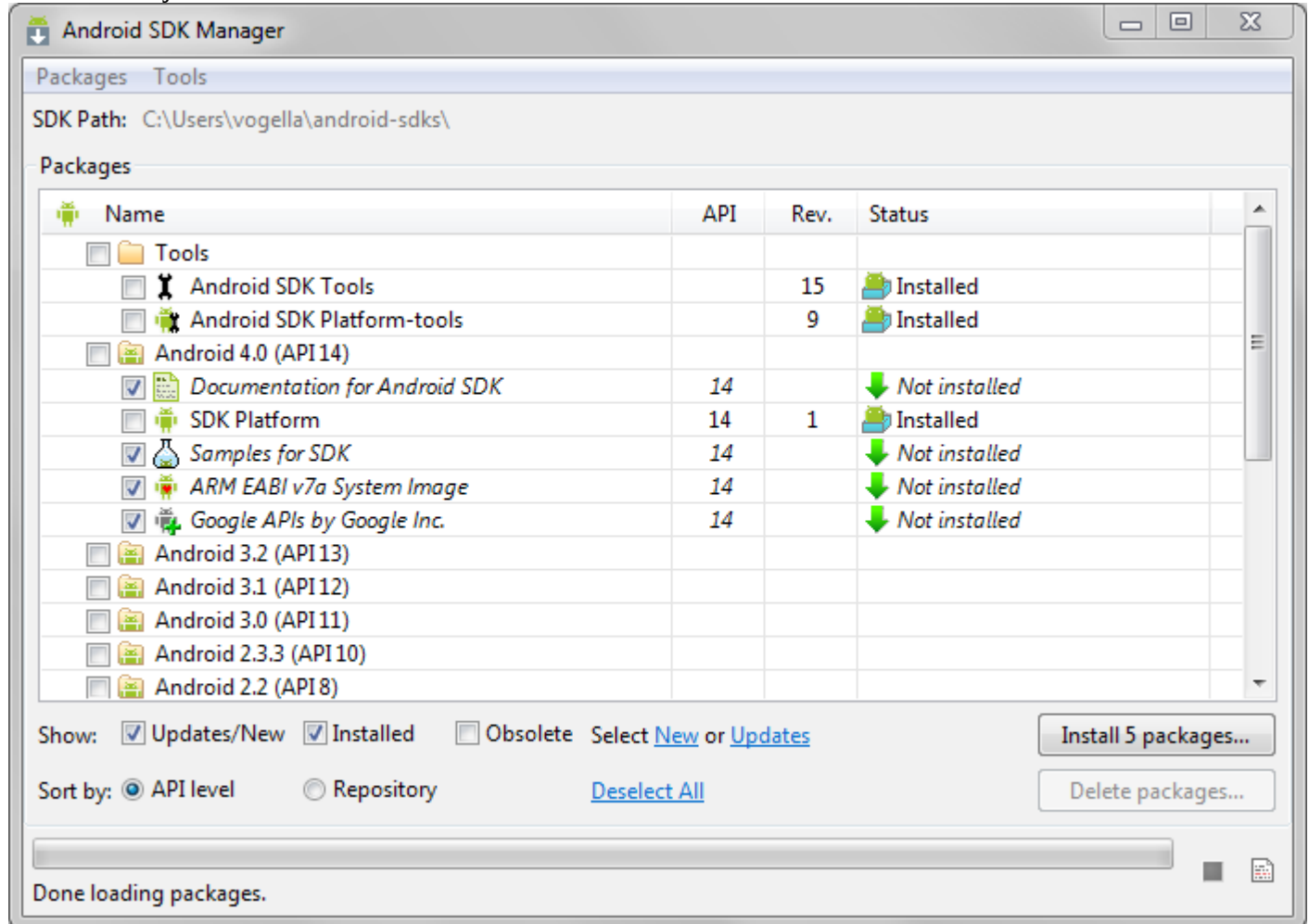
The Android SDK Manager allows you to install specific versions of Android. Select Window → Android SDK Manager from the Eclipse menu.



The dialog allows you to install new packages and also allows you to delete them.

Select "Available packages" and open the "Third Party Add-ons". Select the Google API 15 (Android 4.0.3) version of the SDK and press "Install".

Better select just one



package per time. Press the "Install" button and confirm the license for all packages. After the installation completes, restart Eclipse.

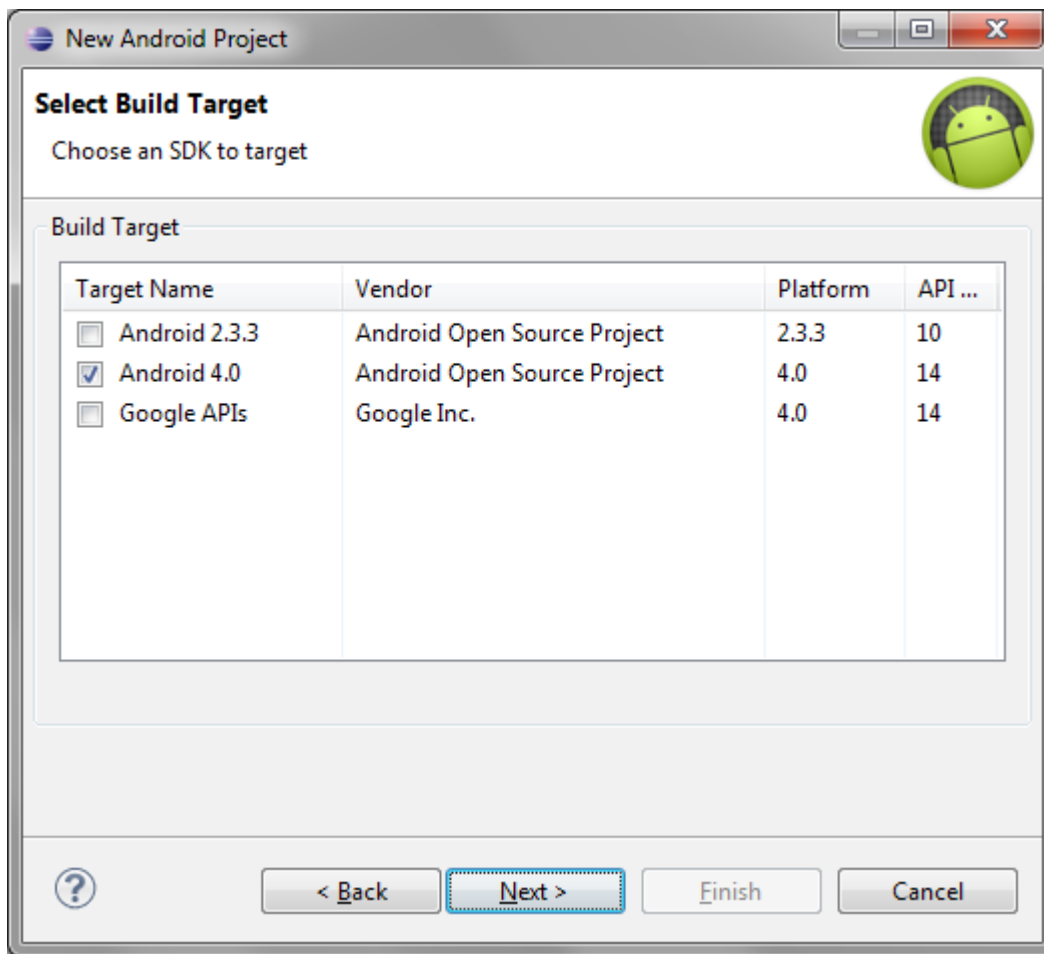
Sources of next information:

<http://www.vogella.com/articles/Android/article.html>

4. Create Project

Select File → New → Other → Android → Android Project and create the Android project "org.abaarsotech.android.ignat99.temperature". Enter the following.

Press "Finish". This should create the following directory structure.



4.1. Modifying resources

As described in the Android Development Tools (ADT) chapter, ADT provides specialized editors for resources files, e.g. layout files. These editors allow to switch between the XML representation of the file and a richer user interface via tabs on the bottom of the editor.

The following description uses the rich user interface to build layout files. For validation purposes, the resulting XML is also included in the description.

4.2. Create attributes

Android allows you to create static attributes, e.g. Strings or colors. These attributes can for example be used in your XML layout files or referred to via Java source code.

Select the file "res/values/string.xml" and press the "Add" button. Select "Color" and enter "myColor" as the name and "#3399CC" as the value.

Add the following "String" attributes. String attributes allow the developer to translate the application at a later point.

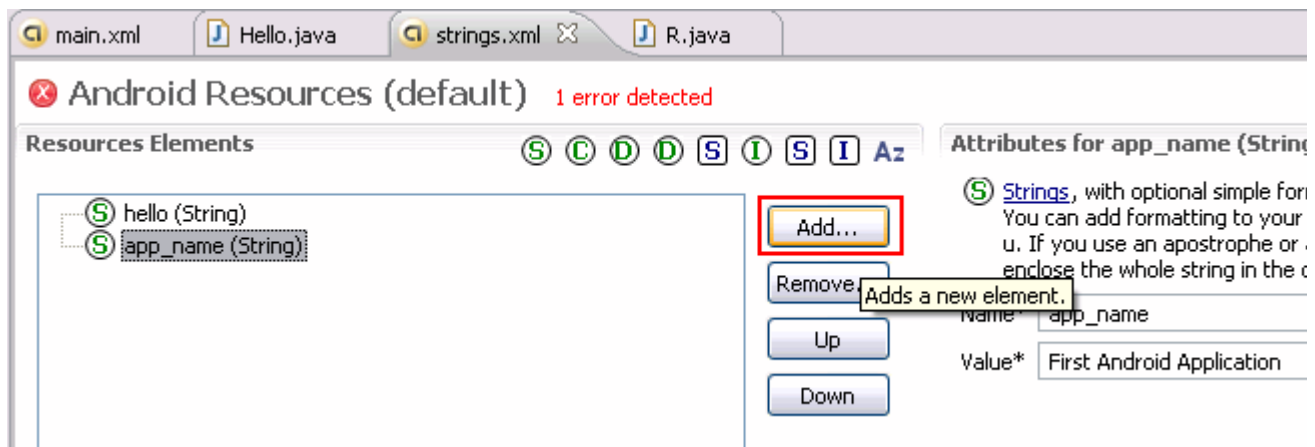


Table 1. String Attributes

Name	Value
celsius	to Celsius
fahrenheit	to Fahrenheit
calc	Calculate

Switch to the XML representation and validate that the values are correct.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, Convert!</string>
    <string name="app_name">Temperature Converter</string>
    <color name="myColor">#3399CC</color>
    <string name="myClickHandler">myClickHandler</string>
    <string name="celsius">to Celsius</string>
    <string name="fahrenheit">to Fahrenheit</string>
    <string name="calc">Calculate</string>
</resources>
```

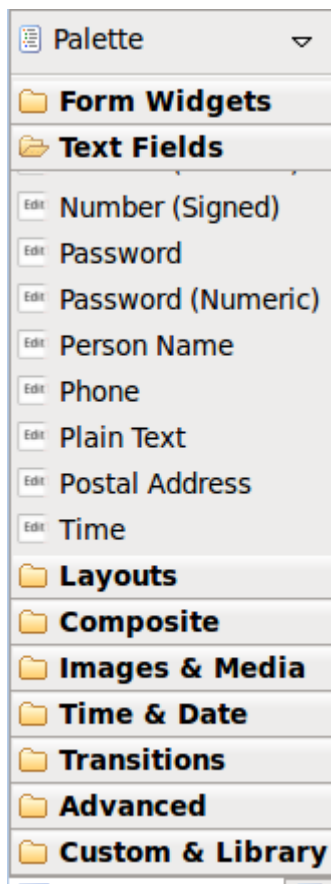
4.4. Add Views

Select "res/layout/main.xml" and open the Android editor via a double-click. This editor allows you to create the layout via drag and drop or via the XML source code. You can switch between both representations via the tabs at the bottom of the editor. For changing the position and grouping elements you can use the Eclipse "Outline" view.

The following shows a screenshot of the "Palette" view from which you can drag and drop new user interface components into your layout. Please note that the "Palette" view changes frequently so your view might be a bit different.

You will now create your new layout.

Right-click on the existing text object "Hello World, Hello!" in the layout. Select "Delete" from the popup menu to remove the text object. Then, from the "Palette" view, select Text Fields and locate



"Plain Text". Drag this onto the layout to create a text input field. All object types in the section "Text Fields" derive from the class "EditText", they just specify via an additional attribute which text type can be used.

Afterwards select the Palette section "Form Widgets" and drag a "RadioGroup" object onto the layout. The number of radio buttons added to the radio button group depends on your version of Eclipse. Make sure there are two radio buttons by deleting or adding radio buttons to the group.

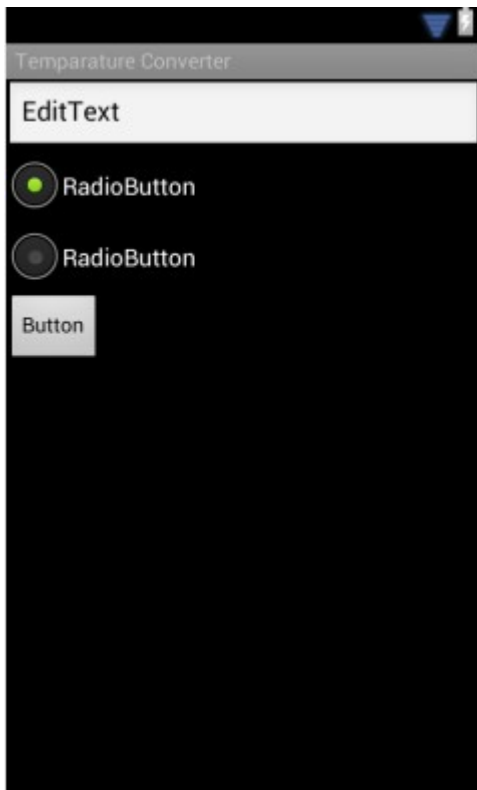
From the Palette section Form Widgets, drag a Button object onto the layout.

The result should look like the following.

Switch to "main.xml" and verify that your XML looks like the following.

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

    <EditText
        android:id="@+id/editText1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="EditText" >
    </EditText>
```



```
<RadioGroup
    android:id="@+id/radioGroup1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >

    <RadioButton
        android:id="@+id/radio0"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:checked="true"
        android:text="RadioButton" >
    </RadioButton>

    <RadioButton
        android:id="@+id/radio1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="RadioButton" >
    </RadioButton>
</RadioGroup>

<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Button" >
</Button>

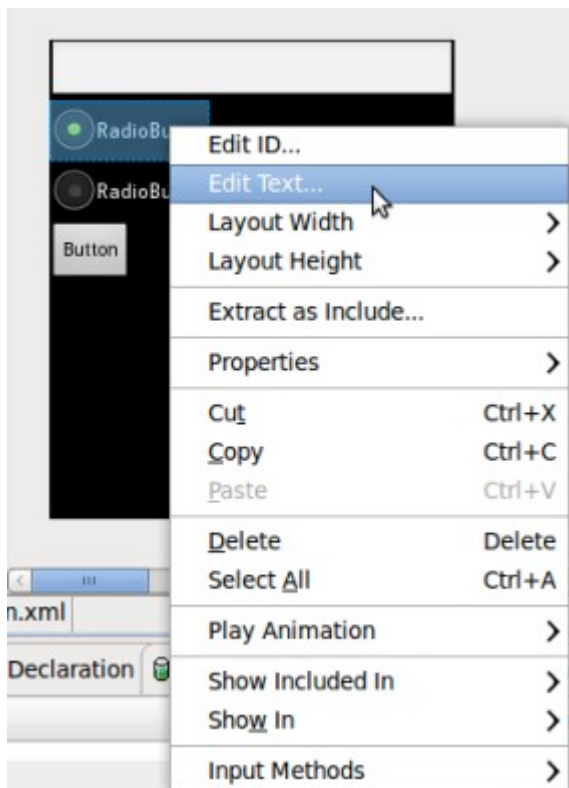
</LinearLayout>
```

4.5. Edit View properties

If you select a user interface component (an instance of `View`), you can change its properties via the Eclipse "Properties" view. Most of the properties can be changed via the right mouse menu. You can also edit properties of fields directly in XML. Changing properties in the XML file is much faster, if you know what you want to change. But the right mouse functionality is nice, if you are searching for a certain property.

Open your file "main.xml". The `EditText` control shows currently a default text. We want to delete this initial text in the XML code. Switch to the XML tab called "main.xml" and delete the `android:text="EditText"` property from the `EditText` part. Switch back to the "Graphical Layout" tab and check that the text is removed.

Use the right mouse click on the first radio button to assign the "celsius" String attribute to its "text" property. Assign the "fahrenheit" string attribute to the second radio button.



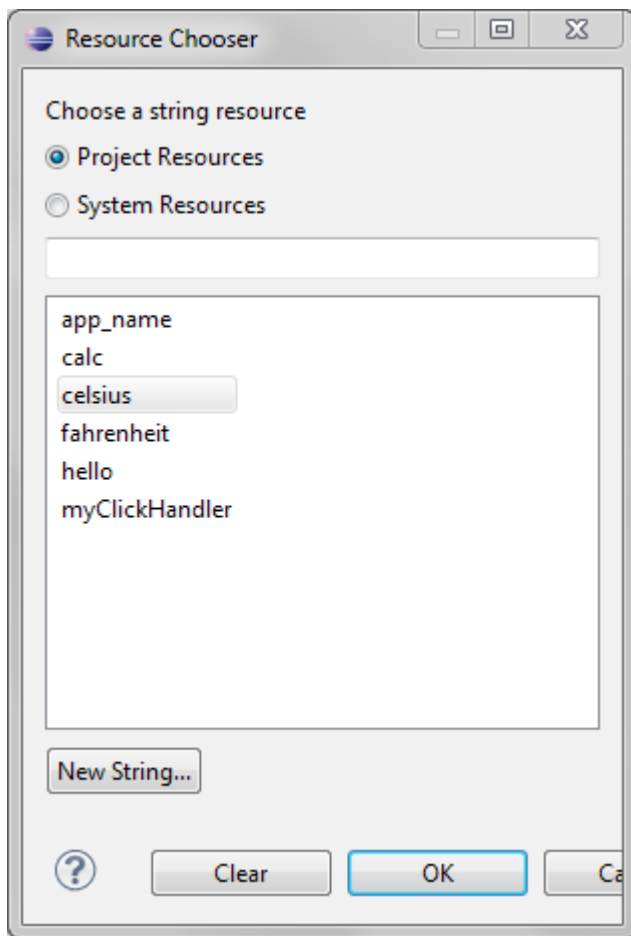
From now on, I assume you are able to use the properties menu on user interface components. You can always either edit the XML file or modify the properties via right mouse click.

Set the property "Checked" to true for the first `RadioButton`.

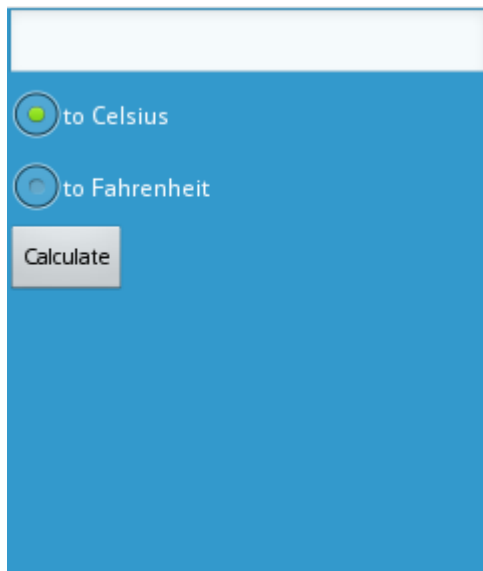
Assign "calc" to the text property of your button and assign "myClickHandler" to the "onClick" property.

Set the "Input type" property to "numberSigned" and "numberDecimal" on your `EditText`.

All your user interface components are contained in a `LinearLayout`. We want to assign a background color to this `LinearLayout`. Right-click on an empty space in Graphical Layout mode,



then select Other Properties → All by Name → Background. Select “Color” and then select "myColor" in the list which is displayed.



Switch to the "main.xml" tab and verify that the XML is correct.

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

```

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@color/myColor"
    android:orientation="vertical" >

    <EditText
        android:id="@+id/editText1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="numberDecimal|numberSigned" >
    </EditText>

    <RadioGroup
        android:id="@+id/radioGroup1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

        <RadioButton
            android:id="@+id/radio0"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:checked="true"
            android:text="@string/celsius" >
        </RadioButton>

        <RadioButton
            android:id="@+id/radio1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/fahrenheit" >
        </RadioButton>
    </RadioGroup>

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="myClickHandler"
        android:text="@string/calc" >
    </Button>

</LinearLayout>

```

4.6. Change the Activity source code

During the generation of your new Android project you specified that an Activity called ConvertActivity should be created. The project wizard created the corresponding Java class.

Change your code in ConvertActivity.java to the following. Note that the myClickHandler will be called based on the onClick property of your button.

```

package de.vogella.android.temperature;

import android.app.Activity;

```

```

import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.Toast;

public class ConvertActivity extends Activity {
    private EditText text;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        text = (EditText) findViewById(R.id.editText1);
    }

    // This method is called at button click because we assigned the name to
the
    // "On Click property" of the button
    public void myClickHandler(View view) {
        switch (view.getId()) {
            case R.id.button1:
                RadioButton celsiusButton = (RadioButton)
findViewById(R.id.radio0);
                RadioButton fahrenheitButton = (RadioButton)
findViewById(R.id.radio1);
                if (text.getText().length() == 0) {
                    Toast.makeText(this, "Please enter a valid
number",
                                Toast.LENGTH_LONG).show();
                    return;
                }

                float inputValue =
Float.parseFloat(text.getText().toString());
                if (celsiusButton.isChecked()) {
                    text.setText(String
                                .valueOf(convertFahrenheitToCelsiu
s(inputValue)));
                    celsiusButton.setChecked(false);
                    fahrenheitButton.setChecked(true);
                } else {
                    text.setText(String
                                .valueOf(convertCelsiusToFahrenhei
t(inputValue)));
                    fahrenheitButton.setChecked(false);
                    celsiusButton.setChecked(true);
                }
                break;
            }
        }

    }

    // Converts to celsius
    private float convertFahrenheitToCelsius(float fahrenheit) {
        return ((fahrenheit - 32) * 5 / 9);
    }
}

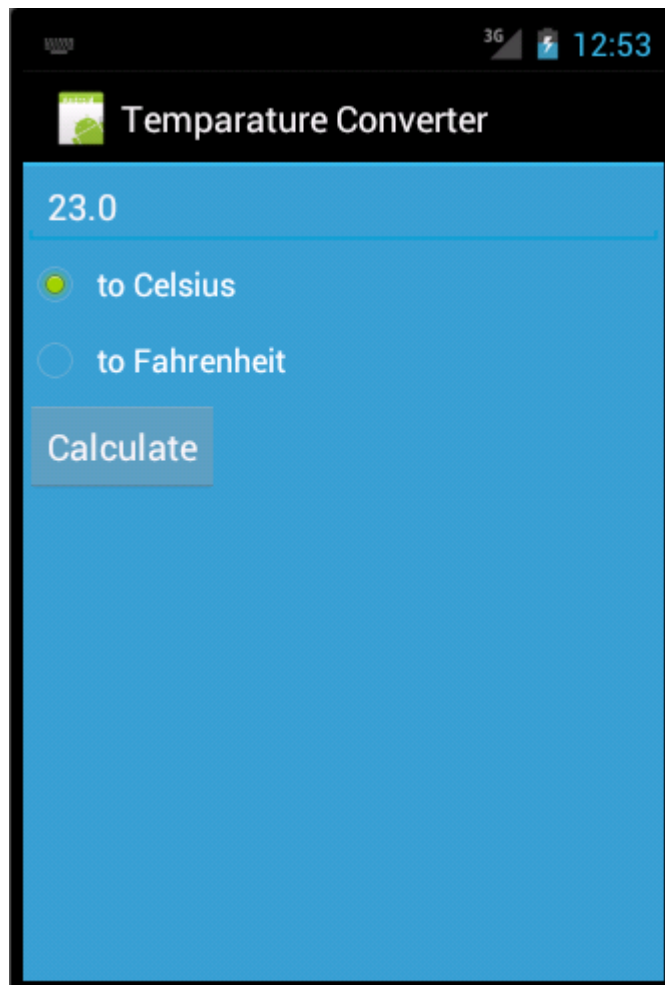
```

```
    // Converts to fahrenheit  
    private float convertCelsiusToFahrenheit(float celsius) {  
        return ((celsius * 9) / 5) + 32;  
    }  
}
```

4.6. Start Project

To start the Android Application, select your project, right click on it, and select Run-As → Android Application. If an emulator is not yet running, it will be started. Be patient, the emulator starts up very slowly.

You should get the following result.



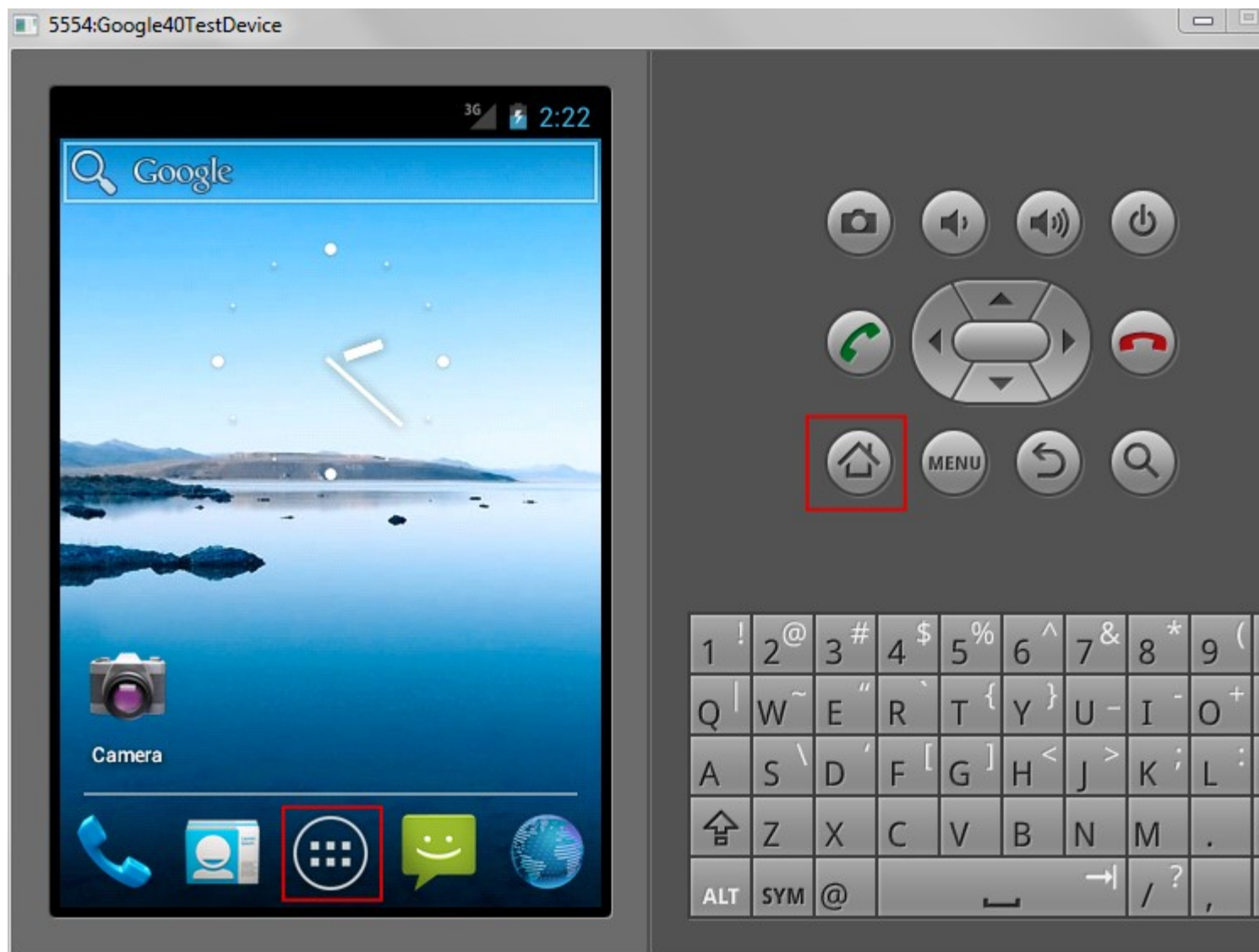
Type in a number, select your conversion and press the button. The result should be displayed and the other option should get selected.

5. Starting an installed application

After you run your application on the virtual device, you can start it again on the device. If you press

the "Home" button you can select your application.

.



3G 2:25

Apps

Widgets



Navigation



People



Phone



Places



Search



Settings



Social App



Speech Reco



Widget Preview