COMP4521 EMBEDDED SYSTEMS SOFTWARE

LAB 1: DEVELOPING SIMPLE APPLICATIONS FOR ANDROID

INTRODUCTION

Android is a mobile platform/OS that uses a modified version of the Linux kernel. It was initially developed by Android Inc., a firm later purchased by Google. It allows developers to write managed code in the Java language, controlling the device via Google-developed Java libraries. Today, we will learn how to create, build, run and debug a program in Android. Part of the materials used in this lab, specifically the "Hello, World!" program, is adapted from the official developer website of Android: http://developer.android.com.

The preparatory work for this lab has already been completed by us to save time, including: installing an IDE, typically Eclipse; installing an Android SDK environment. Thus the Eclipse platform we use in this lab has already been equipped with Android plugins.

OBJECTIVES

- Setting up the Development Environment
- Creating a "Hello, Android" Android Application
- Understanding the various parts of an Android Project
- Using the Android Emulator
- Creating a simple User Interface

DOWNLOADING AND INSTALLING ANDROID SDK AND ECLIPSE

If you want to install the Android SDK and Eclipse IDE on your own PC, you can do one of the following two steps:

1. Install an already fully-configured **Windows** version of the Android SDK and Eclipse SDK that is available at the following links:

   http://course.cse.ust.hk/comp4521/labs/android-sdk-windows.zip

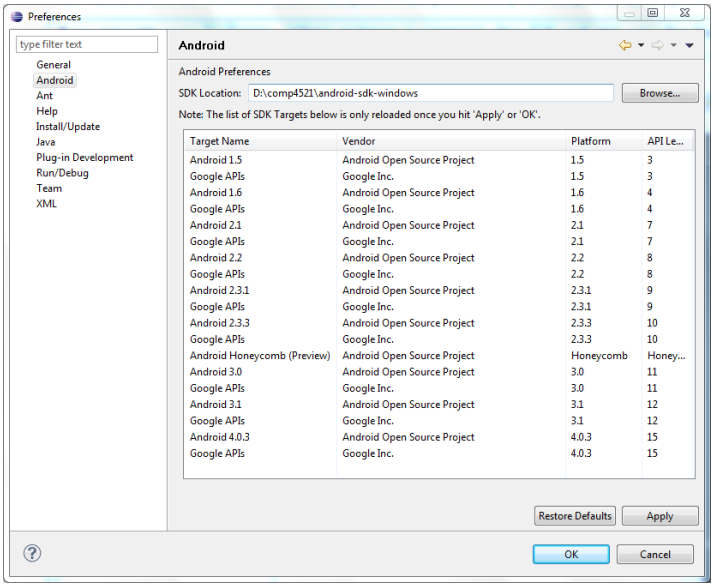
   http://course.cse.ust.hk/comp4521/labs/eclipse.zip

   Download the above two Zip files to your windows D:\ drive and unzip them. Once installed, you can go to the D:\eclipse folder and start *eclipse*.

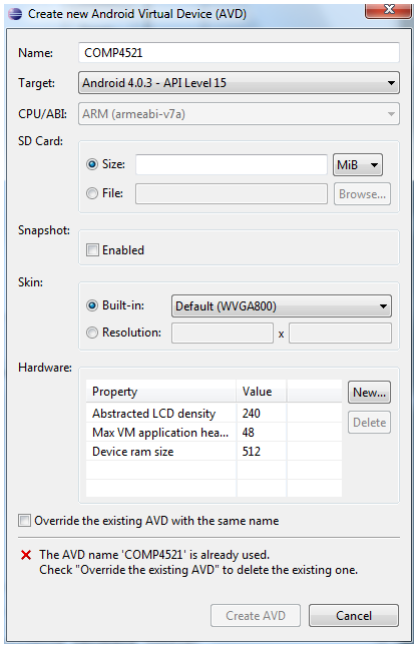
2. Find the detailed instructions on how to install Android SDK and Eclipse IDE at http://developer.android.com/sdk/installing.html. Also if you would like to do the development on a Mac, please go to the link above for the instructions to download and install the SDK and Eclipse IDE.

IMPLEMENT THE "HELLO, ANDROID" APPLICATION

1. Find the Android SDK and Eclipse already installed in D:\comp4521.
2. Locate the SDK of Android for Eclipse. In Eclipse folder, run eclipse.exe, set your workspace to D:\Temp. When eclipse is opened, the IDE will ask you for the location of the Android SKD. Select "**existing SDKs"**, and choose the SDK location (must be D:\comp4521\android-sdk-windows) as where you have put the Android SDK.



3. **Create an Android Virtual Device (AVD):** In Eclipse, select **Window**->**AVD Manager**, and click **New**. Define the name as "COMP4521", and in **Target**, select **Android 4.0.3 – API Level 15** (the version of the Android SDK). Leave other parameters unchanged, and click **Create AVD**. This AVD is the virtual device (emulator) that we use to test our program. An AVD defines the system image and device settings used by the emulator.



4. Create a new Android Project. In Eclipse, click **File**-> **New** -> **Android project**.

Meaning of the different fields:

Project Name

This is the Eclipse Project name — the name of the directory that will contain the project files. Use lab1_<yourID> (i.e., if your ID is John, you may use lab1_John.

Build Target

The version of Android platform you wish your application to run. Since Android applications are forward-compatible, and recall that we have select our AVD version as Android 4.0.3, you may select any Android version that is not higher than 4.0.3.

Application Name

This is the human-readable title for your application — the name that will appear on the Android device. Use **Hello Android**.

Package Name

This is the package namespace (following the same rules as for packages in the Java programming language) that you want all your source code to reside under. This also sets the package name under which the stub Activity will be generated.

Your package name must be unique across all packages installed on the Android system; for this reason, it's important to use a standard domain-style package for your applications. Here we use the "com.example.HelloAndroid " namespace, which is a

namespace reserved for example documentation — when you develop your own applications, you should use a namespace that's appropriate to your organization or entity.

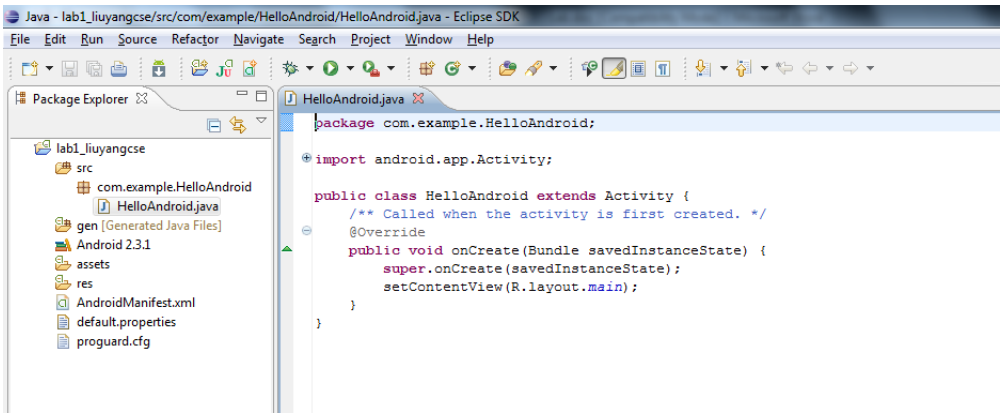
Create Activity

This is the name for the class stub that will be generated by the plugin. This will be a subclass of Android's Activity class. An Activity is simply a class that can run and do work. It can create a UI if it chooses, but it doesn't need to. As the checkbox suggests, this is optional, but an Activity is almost always used as the basis for an application. Here we use **HelloAndroid**.

Min SDK Version

This specifies the minimum API Level on which your application can run. By default this is set to the API Level of the Build Target Platform. As new APIs are added to newer Versions, their API levels increase as well. A Program that uses an API Level of four won't be able to run on a platform that has a lower API Level.

5. Now your Android project is ready. In Package Explorer, click **src**-> **com.example.HelloAndroid**. It should be like this:



Notice that the class is based on the **Activity** class. An Activity is a single application entity that is used to perform actions. An application may have many separate activities, but the user interacts with them one at a time. The **onCreate()** method will be called by the Android system when your Activity starts — it is where you should perform all initialization and UI setup. An activity is not required to have a user interface, but usually will.

6. Construct the UI. Use following code to replace the default code of **onCreate()**.

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    TextView tv = new TextView(this);
    tv.setText("Hello, Android");
    setContentView(tv);
}
```

Since the class TextView is not accepted by default, you should click on it and select Import 'TextView' (android.widget). Or you can also type yourself: import android.widget.TextView.[1]

An Android user interface is composed of hierarchies of objects called Views. A View is a drawable object used as an element in your UI layout, such as a button, image, or (in this case) a text label. Each of these objects is a subclass of the View class and the subclass that handles text is TextView.

In this change, you create a TextView with the class constructor, which accepts an Android Context instance as its parameter. A Context is a handle to the system; it provides services like resolving resources, obtaining access to databases and preferences, and so on. The Activity class inherits from Context, and because your HelloAndroid class is a subclass of Activity, it is also a Context. So, you can pass this as your Context reference to the TextView.

Next, you define the text content with setText().

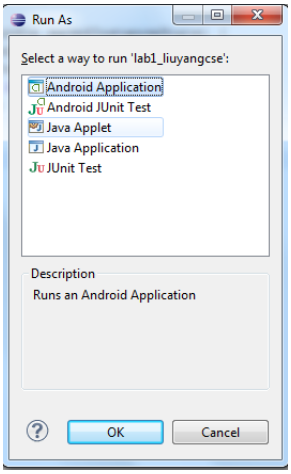
Finally, you pass the TextView to setContentView() in order to display it as the content for the Activity UI. If your Activity doesn't call this method, then no UI is present and the system will display a blank screen.

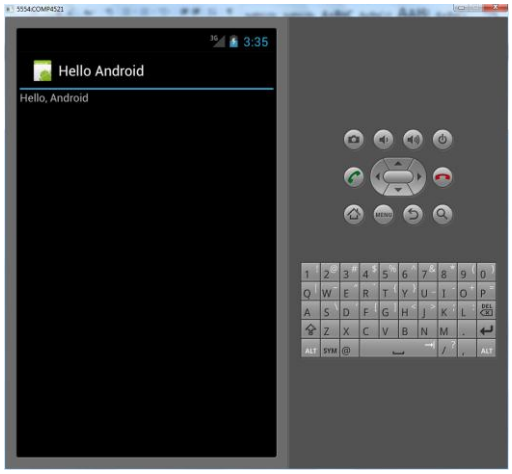
There it is — "Hello, World" in Android! The next step, of course, is to see it running.

7. Run the application. Click **Run**-> **Run**, and select **Android Application**. Eclipse will build the whole project and deploy it to an emulator automatically. You can find your application in **Menu**.

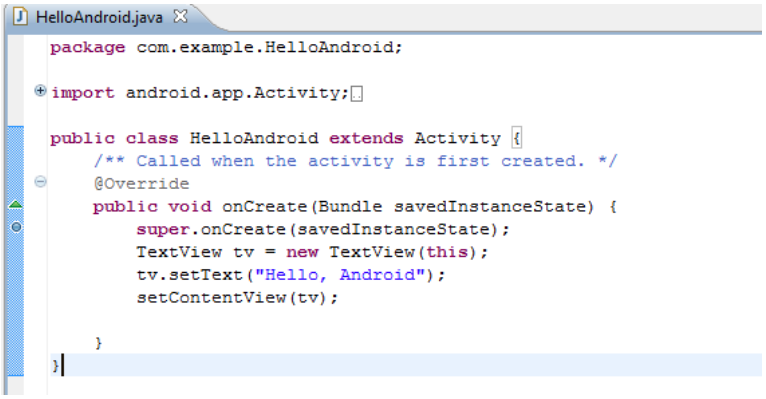
---

[1] *Tip: An easy way to add import packages to your project is to press Ctrl-Shift-O (Cmd-Shift-O, on Mac). This is an Eclipse shortcut that identifies missing packages based on your code and adds them for you.*





8. Debug your project. Put a breakpoint for your application by double-clicking on the marker bar next to the source code line.

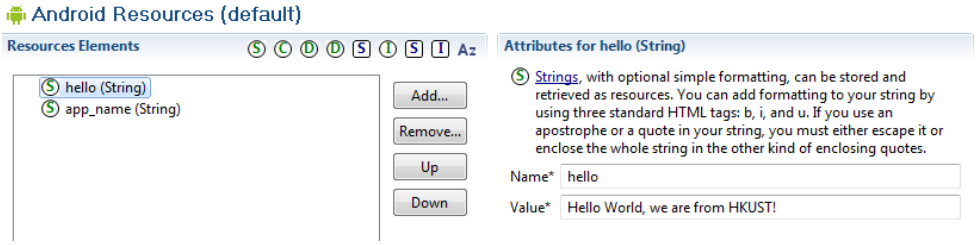


After setting a breakpoint, select Run-> Debug, and Eclipse will restart your emulator. But this time it will suspend when it reaches the breakpoint you set. You can then step through the code in Eclipse's Debug Perspective, just as you would for any other application.

9. Upgrade the UI to an XML Layout. This is an easier way to apply your modification to the UI to different applications. In the Eclipse **Package Explorer**, select **/res/layout/main.xml**. This xml layout file can be used by the application to construct user interfaces. Modify the contents of the file to the following:

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@+id/textview"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:text="@string/hello"/>
```

10. Open **/res/values/strings.xml**, and you can change the values of the two strings: *hello* and *app_name*.



11. Now modify your **HelloWorld.java** file. Replace the content with following code:

```
package com.example.helloandroid;
import android.app.Activity;
import android.os.Bundle;
public class HelloAndroid extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Compared to the previous code, the importing of android.widget.Textview is not needed, and instead of passing setContentView() a View object, you give it a reference to the layout resource. The resource is identified as R.layout.main, which is actually a compiled object representation of the layout defined in **/res/layout/main.xml**.

Now you can run the application again, and see that the title of the application and the text has been changed.

DESIGN A SIMPLE USER INTERFACE LAYOUT

1. Create a new Android Project. In Eclipse, click **File**-> **New** -> **Android project**.

Meaning of different fields:

Project Name

Use **Welcome4521**.

Build Target

Android 4.0.3.

Application Name

Use **Course Info COMP4521**.

Package Name

Here we use the "hkust.comp4521.welcome" namespace.

Create Activity

Here we use **welcome**.

2. Now that your Android project is ready, we will define colors for the application. Find the folder **/res/Values**, right click, select **New-> File**, set the file name as **colors.xml**, and set the content to the following code:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<color name="ustblue">#003366</color>
<color name="ustgold">#996600</color>
</resources>
```

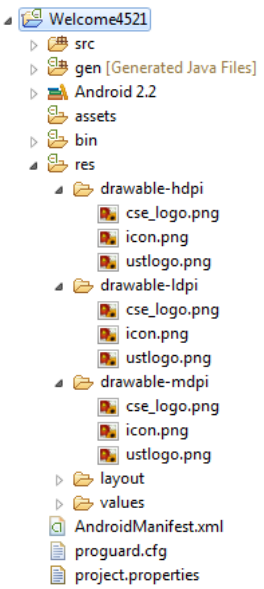
This is how we define new colors that can be used in our applications.

3. Open **strings.xml**, and change its content with following code:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Course Info COMP4521</string>
    <string name="Welcome">Welcome to</string>
    <string name="CourseCode">COMP4521</string>
    <string name="CourseTitle">Embedded Systems Software</string>
    <string name="Semester">Spring 2012</string>
</resources>
```

We are basically defining several new strings that we can use for the text in the user interface that we design next. It is very easy to replace these string values with other values to reconfigure the application user interface without needing to change any code.

4. Import some pictures to use in designing the user interface: There are icons for three different resolutions (high, medium and low): hdpi, mdpi and ldpi. On the lab page you will find three Zip files: ldpi.zip, mdpi.zip and hdpi.zip. Download the three zip files to the respective folders in the project home directory (must be D:\temp\Welcome4521\res) and unzip them. Then, right click on the project name in Eclipse, and select refresh. You can see the icons appear in the project folder.



The Android device will use the appropriate icons that are suitable for the resolution of the device.

5. Configure the user interface layout: In the **/res/layout** folder, rename **main.xml** to **welcome.xml** (just press F2 to rename), and change its content to the following code:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@color/ustblue"
    >
  <LinearLayout
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="0dp"
    android:layout_weight="1">

    <ImageView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:gravity="center_vertical|center_horizontal"
    android:src="@drawable/ustlogo"
    android:maxHeight="10dp"
    android:background="@android:color/white"
    />

    <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/Welcome"
    android:gravity="center_vertical|center_horizontal"
    android:textColor="@android:color/white"
    android:textSize="12pt"
    />

    <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/CourseCode"
    android:gravity="center_vertical|center_horizontal"
    android:textColor="@color/ustgold"
    android:textSize="18pt"
    />

    <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/CourseTitle"
    android:gravity="center_vertical|center_horizontal"
    android:textColor="@color/ustgold"
    android:textSize="14pt"
    />

    <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/Semester"
    android:gravity="center_vertical|center_horizontal"
    android:textColor="#aaaaaa"
    android:textSize="16pt"
    />
```

```
    </LinearLayout>

    <ImageView
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:gravity="center_vertical|center_horizontal"
     android:src="@drawable/cse_logo"
     android:maxHeight="10dp"
     android:background="@android:color/white"
     />
</LinearLayout>
```

Now if you preview the graphical layout of **welcome.xml**, it should look like this:

What we have done by modifying the welcome.xml file is to define the new user interface. We use the TextView layout and set the text of the textview layout to the appropriate string value. Note how the string values are referenced. We use ImageView to include the pictures.

6. Then change the content of **welcome.java** with following code:

```
package hkust.comp4521.welcome;

import hkust.comp355.welcome.R;
import android.app.Activity;
import android.os.Bundle;

public class welcome extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.welcome);
    }
}
```

7. After that, you can run the application. The application will run and generate the following screen: