COSC2543 Mobile Application development

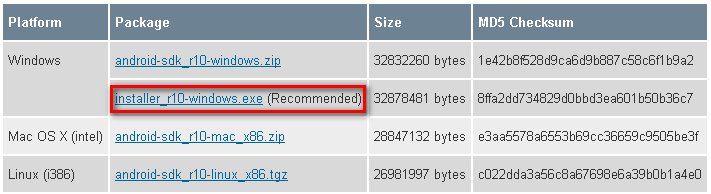
# TUTORIAL WEEK 8B: INTRODUCTION TO ANDROID

**Objectives:**

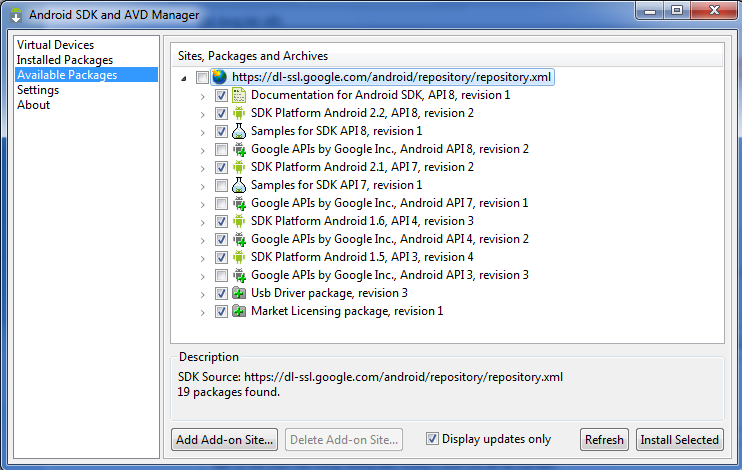
* Set up Android Development Environment
* Familiar with simple Activity
* Know where android app stores resources

**Task 1: Set up environment**

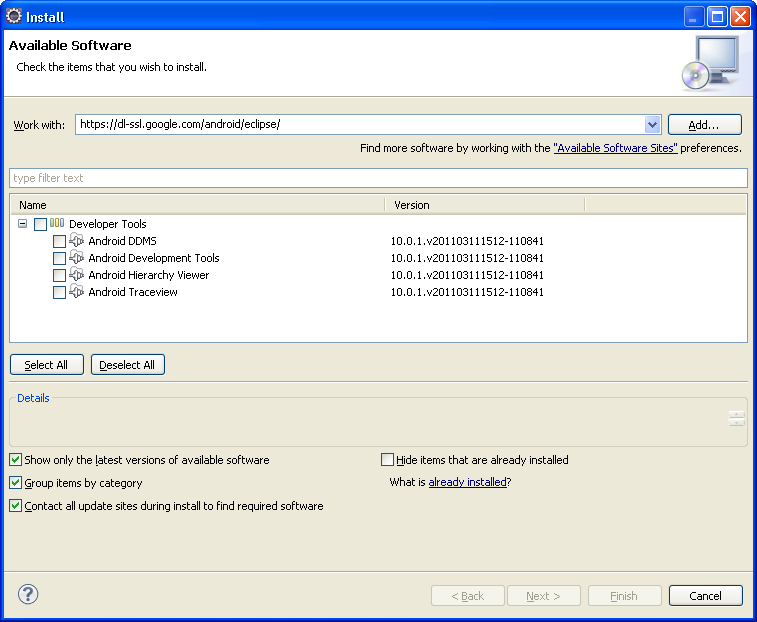
* Visit <http://developer.android.com/sdk/index.html>
* Download your SDK version

****

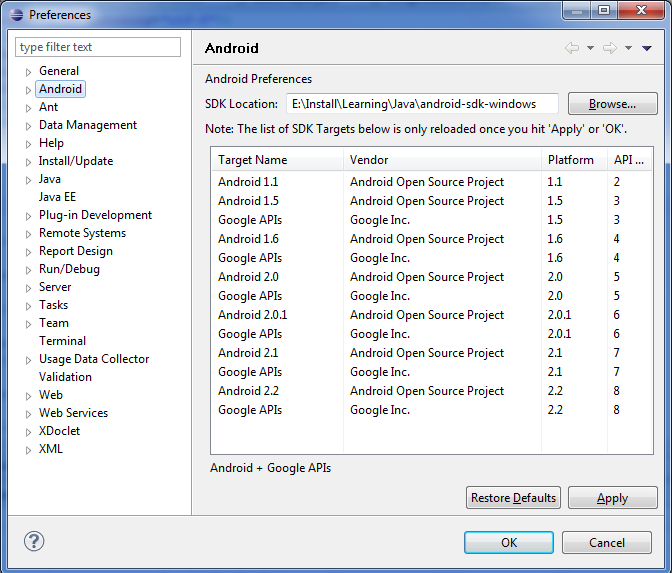
* Install and run SDK
* Install platforms that you want to work on

****

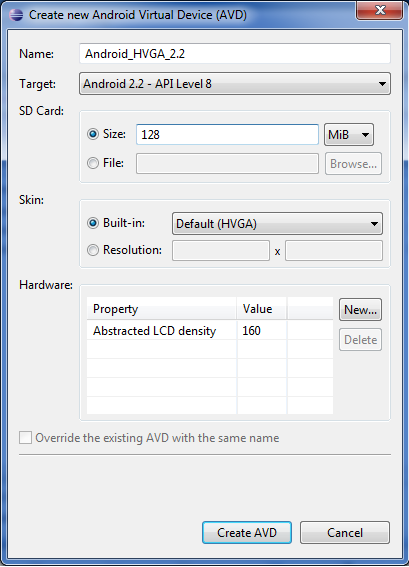
* Install Android plug-in for Eclipse
* Run Eclipse
* Navigate to Help -> Install new software
* Enter either links below:
  + <https://dl-ssl.google.com/android/eclipse/>
  + <http://dl-ssl.google.com/android/eclipse/>
* Install All Developer’ Tools



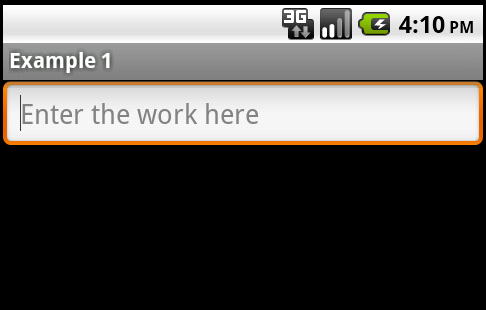
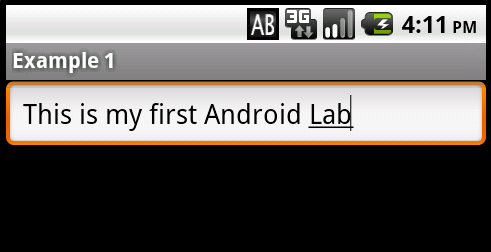
* Configure SDK in Eclipse
* Go to Preference 🡪 Android
* Browse the location where you installed your SDK

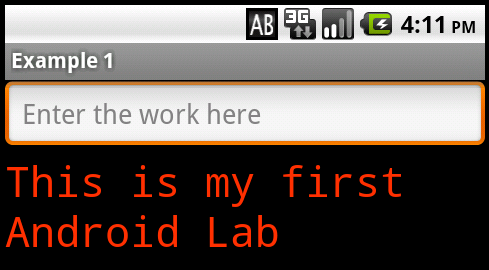


* Create Android Virtual Device (Emulator)
* Run Android SDK Manager
* Under Virtual Device, create new device

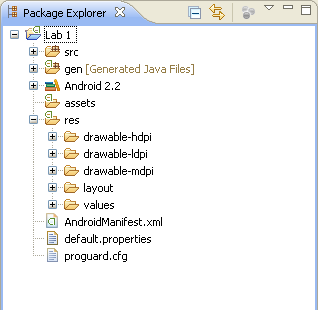


**Task 2: Simple example with activity**



1. Creating a project using Eclipse
2. File -> New -> Android Project. If you're new Android programming, perhaps the first Android Project will not show up, then down to the bottom and select “Other” in Android 🡪 Android Project.
3. Enter the information for project
4. In the Package Explorer on the left side, go to the res folder, you will see three subdirectories



* drawable: the directory containing the images & resources for interface
* layout: xml file to contain the interface design.
* values​​: it contains the value used in the application that you define, such as lines of characters (string), the color (color), the themes ...

1. In Layout folder, select main.xml  type the following code replace entire content  (Eclipse has the drag and drop support for the xml but I do not use)

<?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"

>

<EditText

android:id="@+id/edit\_text"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:hint="@string/edit\_hint"

/>

<TextView

android:id="@+id/text\_view"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:textColor="@color/text\_color"

android:textSize="28px"

android:typeface="monospace"

/>

</LinearLayout>

In the XML we declare a LinearLayout with two children of its components is an EditText (used to type string) with a TextView (display string).

Linear layouts are declared with a keyword to indicate orientation of the two-dimensional arrangement is the vertical component. With layout\_width, layout\_height, you can set it to "fill\_parent" or"wrap\_content" to indicate that this component will have a width (length) covered its parent width (length) or just its content

In the EditText and the TextView you can see the tag id, tag id allows declaration of the component to retrieve the code.  
Also in the EditText, we see "@ string/edit\_hint" indicates that we will get the message string named edit\_hint in strings.xml. Similarly for color

1. In the res folder 🡪 values and select the strings.xml, add the definition for edit\_hint as follows:

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

<resources>

<string name="hello">Hello World, Example!</string>

<string name="app\_name">Example 1</string>

<string name="edit\_hint">Enter the work here</string>

</resources>

1. In values folder, create colors.xml file (right-click the folder, select New -> Android XML file, and note the "s", not the color.xml). Add the following :

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

<resources>

<color name="text\_color">#ff3300</color>

</resources>

1. So we have finished the XML interface, it’s time to write code to handle events for the components. Under src fodler (source code of the project) 🡪 lab1.android🡪 Example.java, type the following code:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38 | package lab1.android;  import android.app.Activity;  import android.os.Bundle;  import android.view.KeyEvent;  import android.view.View;  import android.view.View.OnKeyListener;  import android.widget.EditText;  import android.widget.TextView;    public class Example extends Activity {      @Override      public void onCreate(Bundle savedInstanceState) {          super.onCreate(savedInstanceState);            //Get the interface from main.xml          setContentView(R.layout.main);            //Retrieve components in main.xml via id          final EditText edit = (EditText) findViewById(R.id.edit\_text);          final TextView text = (TextView) findViewById(R.id.text\_view);            //Handle event when center button is pressed          edit.setOnKeyListener(new OnKeyListener() {              @Override              public boolean onKey(View v, int keyCode, KeyEvent event) {                  if (event.getAction() == KeyEvent.ACTION\_DOWN                          && keyCode == KeyEvent.KEYCODE\_DPAD\_CENTER) {                      text.setText(edit.getText().toString());                      edit.setText("");                      return true;                  }                  else {                      return false;                  }              }          });      }  } |

**Task 3: Research Layouts and practice with event handling in Android**

Research different layouts in Android and produce the following UI (3 buttons) by XML.

* Initially, all buttons do not have labels.
* When users click on buttons, its labels will be the assigned from 1 to 3. The order of the labels is the order of the click events.

