Special Interest Group on Innovative Software 2012-2013

Workshop on Android app development

Session 2: User Interface and Navigation

Contents

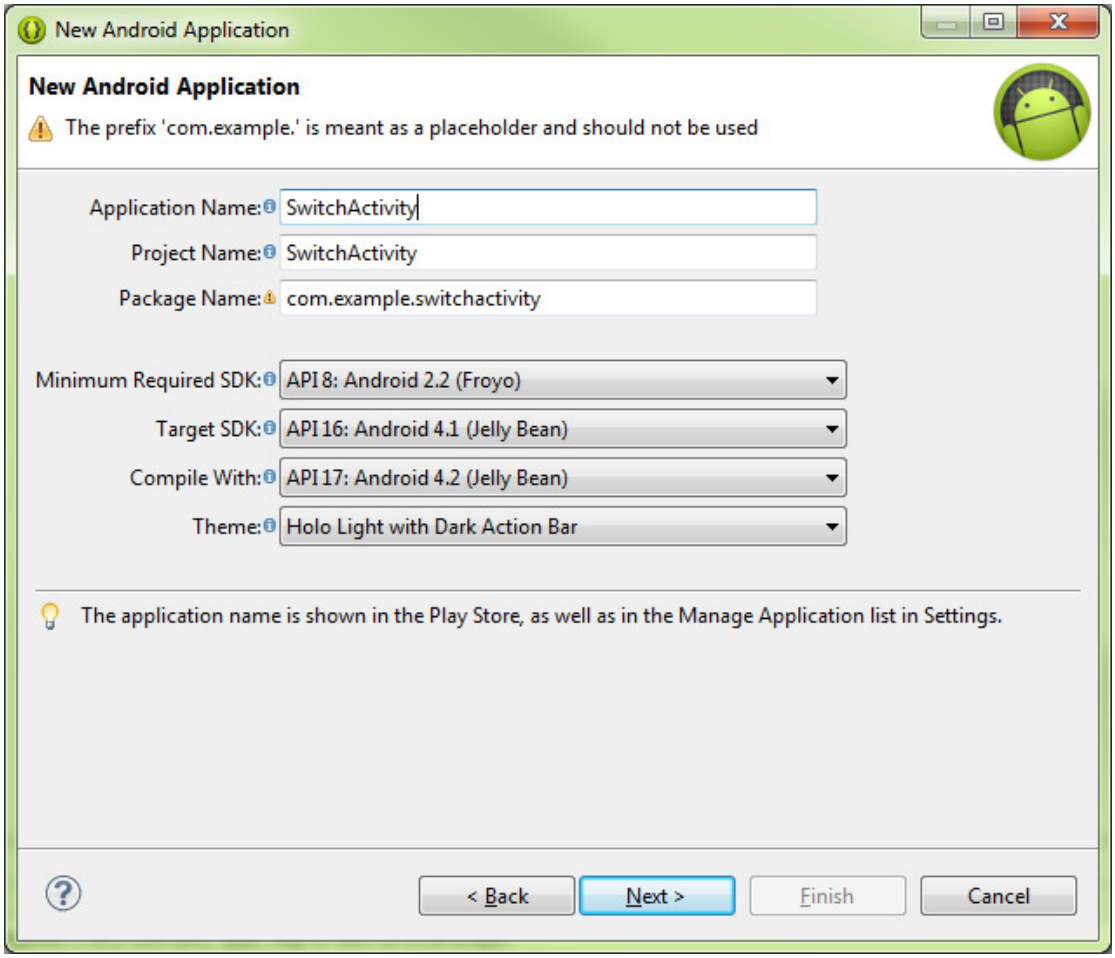
Prepared by Mr. Felix Tam, Committee Member of the Special Interest Group (SIG) on Innovative Software 2012-2013, Department of Computer Science, Hong Kong Baptist University.

1 Switch between activities

1.1 Switch to another page by starting another activity

Every single page in an app is an activity. If we want to navigate to another page, we have to start another activity. In this example, we will learn how to start an activity, how to send the data to another activity and how to get back the result from an activity.

1. Create a new project called **SwitchActivity** with **Create activity** option checked in **Configure Project** step**.** Click next in the rest of steps and refer to the screenshot below for the information at the first step.

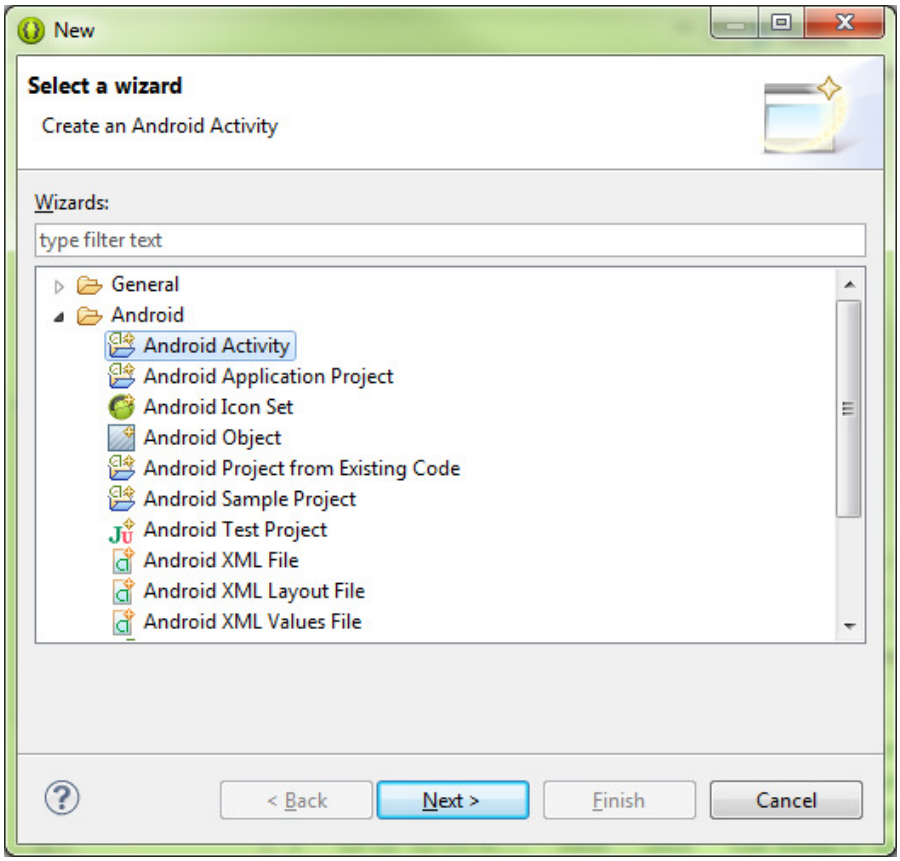


2. Replace the content of **activity_main.xml** with the following markup:

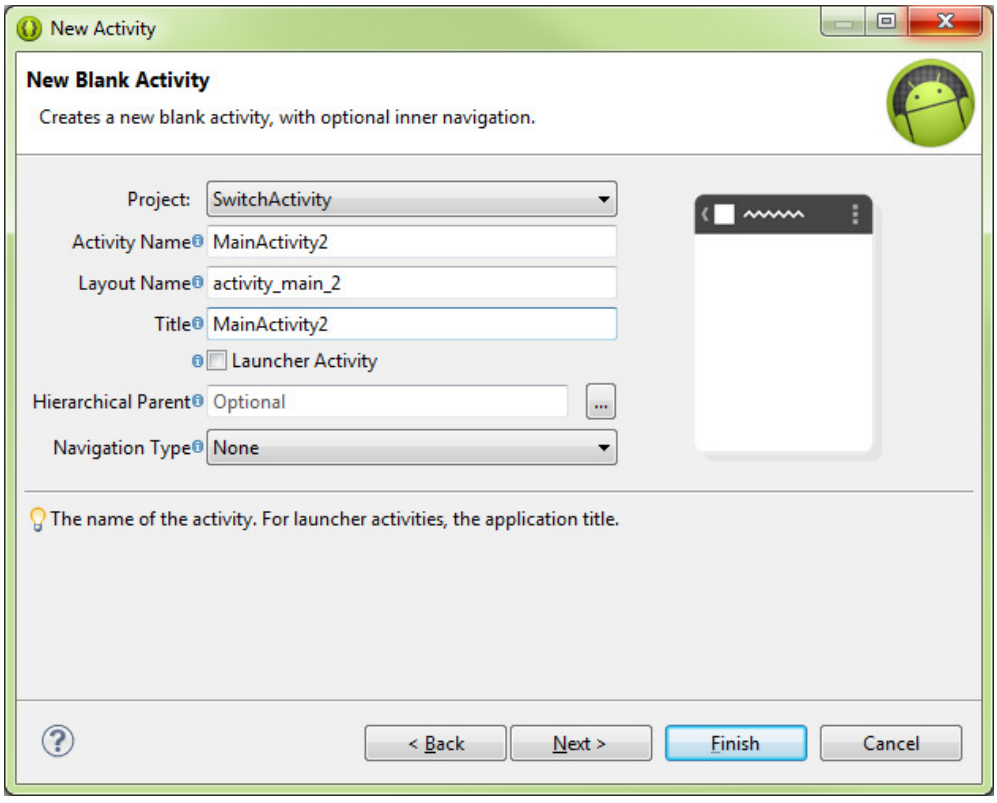
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

```
        android:orientation="vertical">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Input from Activity 2:"
            android:textSize="20sp" />

        <TextView
            android:id="@+id/resultFromAct2"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="5dp"/>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:layout_marginTop="30dp">

            <EditText
                android:id="@+id/inputAct1"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_weight="1"/>

            <Button
                android:id="@+id/go"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_weight="0"
                android:text="Go"/>
        </LinearLayout>
    </LinearLayout>
```

3. Right click **src/com.example.switchactivity,** select **New > Other > Android > Android Activity** to create the second activity in the same project.



4. Select **BlankActivity,** type **MainActivity2** in **Activity Name** and uncheck **Launcher Activity.** Click **Finish** to create the second activity.



5. Replace the content of **activity_main_2.xml** with the following markups:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Input from Activity 1:"
        android:textSize="20sp" />

    <TextView
        android:id="@+id/resultFromAct1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="5dp"/>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_marginTop="30dp">

        <EditText
            android:id="@+id/inputAct2"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"/>

        <Button
            android:id="@+id/back"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="0"
            android:text="Back"/>
```

```
        </LinearLayout>
    </LinearLayout>
```

6. Replace the contents of **MainActivity.java** with the following codes:

```
package com.example.switchactivity;
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends Activity {

    TextView resultFromAct2;
    EditText inputAct1;
    Button go;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        resultFromAct2 = (TextView)findViewById(R.id.resultFromAct2);
        inputAct1 = (EditText)findViewById(R.id.inputAct1);
        go = (Button)findViewById(R.id.go);
        go.setOnClickListener(goBtnListener);
    }

    private OnClickListener goBtnListener = new OnClickListener(){

        @Override
        public void onClick(View arg0) {
            Bundle bundle = new Bundle();
            bundle.putString("myText", inputAct1.getText().toString());
```

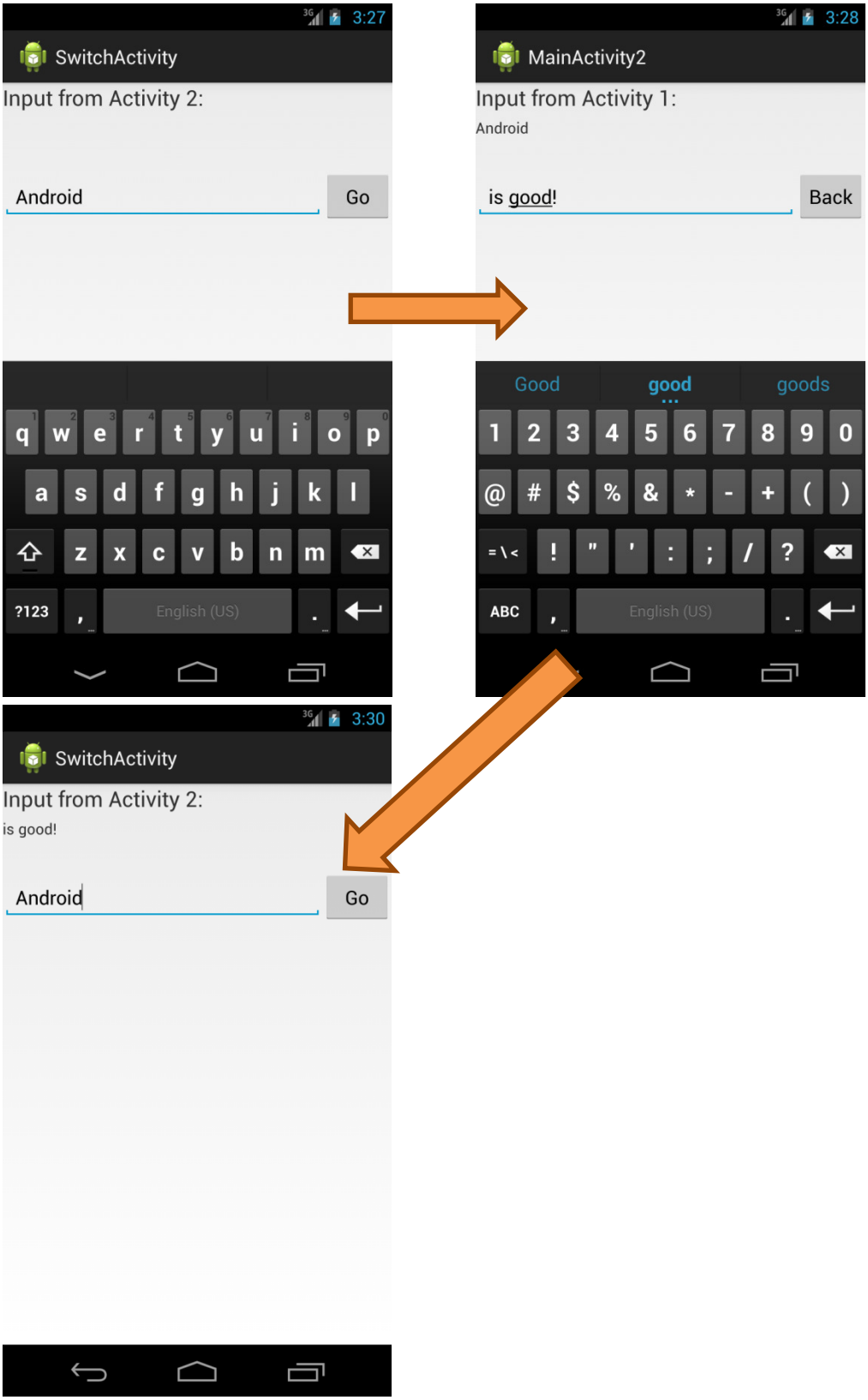
```
                startActivityForResult(new Intent(MainActivity.this,
MainActivity2.class).putExtras(bundle), 0);
            }

        };

        @Override
        protected void onActivityResult(int requestCode, int resultCode, Intent data) {
            super.onActivityResult(requestCode, resultCode, data);
            if(requestCode == 0 && resultCode == RESULT_OK){
                Bundle bundle = data.getExtras();
                if(bundle!=null){
                    Log.v("resultFromAct2", bundle.getString("resultFromAct2"));
                    resultFromAct2.setText(bundle.getString("resultFromAct2"));
                }
            }
        }

}
```

7. Replace the contents of **MainActivity2.java** with the following codes:

```
package com.example.switchactivity;
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity2 extends Activity {

    TextView resultFromAct1;
    EditText inputAct2;
    Button back;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main_2);
        resultFromAct1 = (TextView)findViewById(R.id.resultFromAct1);
        inputAct2 = (EditText)findViewById(R.id.inputAct2);
        back = (Button)findViewById(R.id.back);
        back.setOnClickListener(backBtnListener);

        Bundle b = getIntent().getExtras();
        resultFromAct1.setText(b.getString("myText"));
    }

    private OnClickListener backBtnListener = new OnClickListener(){

        @Override
        public void onClick(View arg0) {
            Bundle bundle = new Bundle();
            bundle.putString("resultFromAct2", inputAct2.getText().toString());
            Intent intent = new Intent();
            intent.putExtras(bundle);
            setResult(RESULT_OK, intent);
            finish();
        }
    };
}
```

8. Run the app in the emulator, type something in the first activity and you will get back what you typed in second activity after you tapped on the Go button and vice versa.



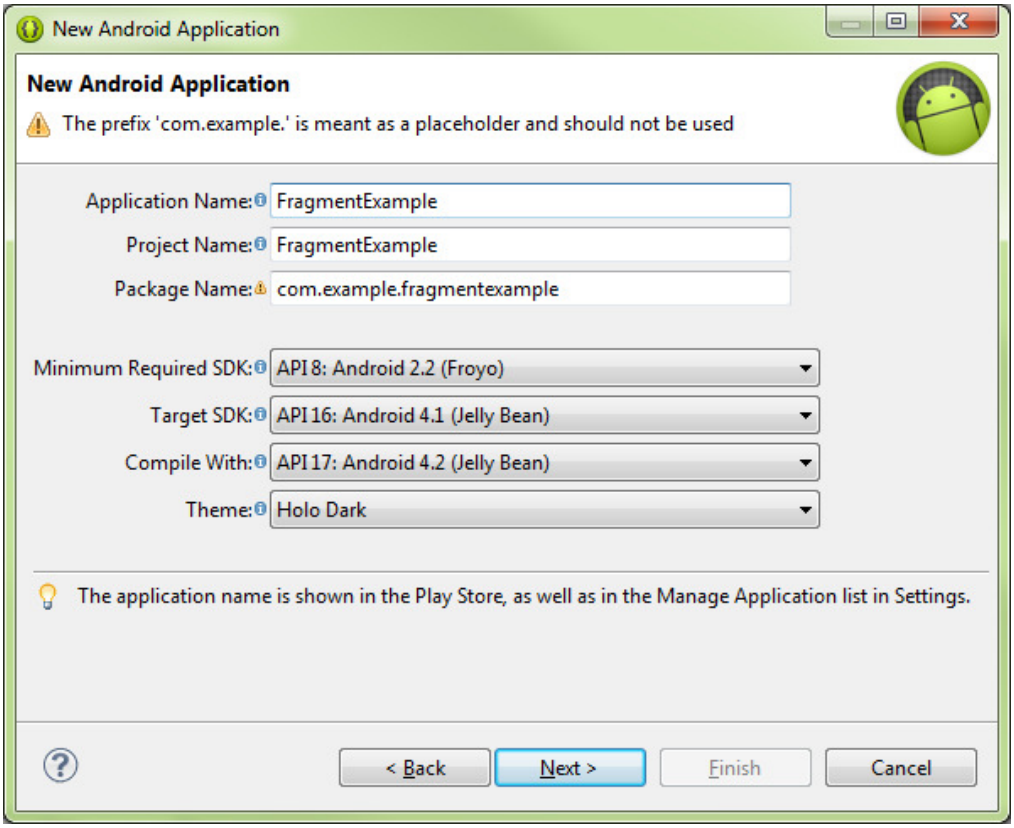
The complete project (sample codes) is available at **SwitchActivity.zip**

2 Understand fragment

Fragment has been introduced since Android 3.0 (API Level 11) or using the **Android Support Library** to provide limited support with **ActionBar**. Fragment is a portion of an activity which just likes a sub-activity. It is useful when you plan to support both Phone UI and Tablet UI, or create reusable UI components. Modern Android apps make use of fragments and **ActionBar** to create uniformed user navigation and experience. Good examples include **GMail**, **Maps** from Google prior to Android 2.

2.1 Define XML layout for main activity and fragments

1. Start a new **Android Application Project** in Eclipse and refer to the screenshot below for application details. Again we will also create a new **BlankActivity** with default activity name and layout.



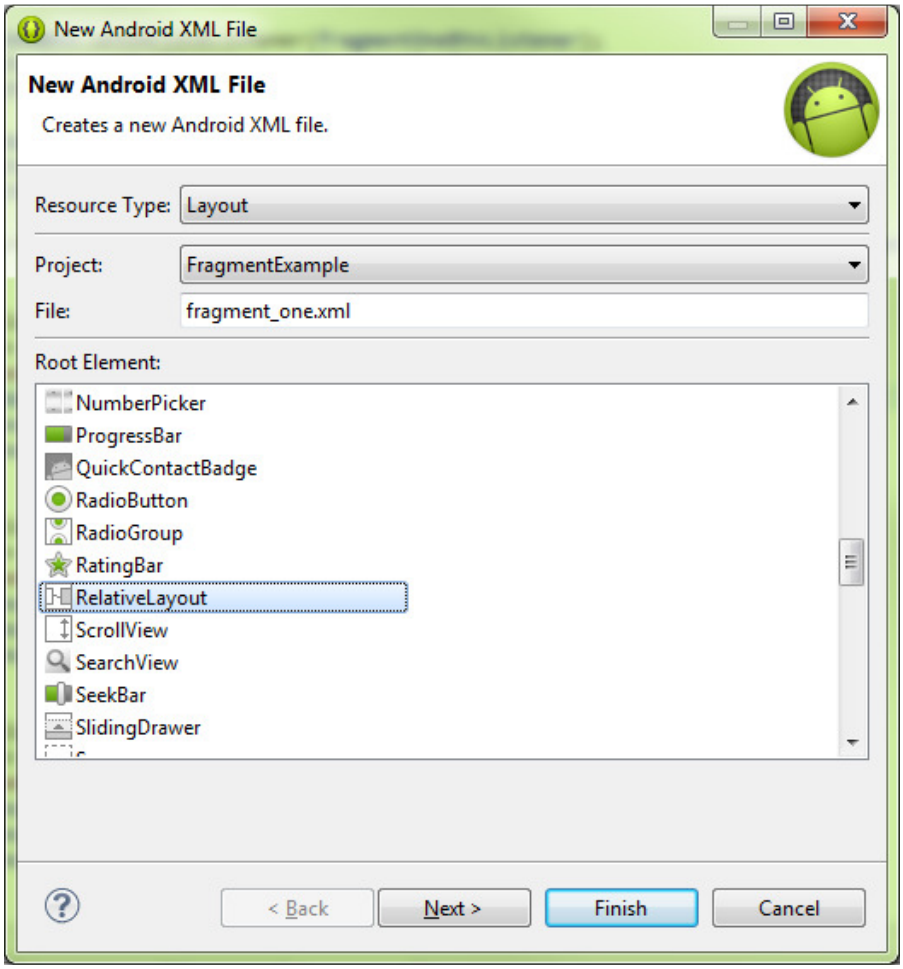
2. Replace the contents with the following markups in **activity_main.xml**. We will add three buttons at the top and a **FrameLayout** to the **MainActivity**.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
```

```
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical" >

        <LinearLayout
         android:layout_width="match_parent"
         android:layout_height="wrap_content"
         android:gravity="center_horizontal"
         android:orientation="horizontal" >
         <Button
               android:id="@+id/fragButton1"
               android:layout_width="wrap_content"
               android:layout_height="wrap_content"
               android:text="Frag. 1" />

         <Button
               android:id="@+id/fragButton2"
               android:layout_width="wrap_content"
               android:layout_height="wrap_content"
               android:text="Frag. 2" />

         <Button
               android:id="@+id/fragButton3"
               android:layout_width="wrap_content"
               android:layout_height="wrap_content"
               android:text="Frag. 3" />
    </LinearLayout>

        <FrameLayout
             android:layout_width="match_parent"
             android:layout_height="match_parent"
             android:id="@+id/fragment_container" >
        </FrameLayout>

    </LinearLayout>
```

3. We also need to create the layout of each fragment, namely **fragment_one.xml**, **fragment_two.xml** and **fragment_three.xml**. To create an xml layout, simply right

click on the layout folder (in Package Explorer of Eclipse) and select **New > Android XML File**.



4. Use the follow markups for the three XML documents respectively:

fagment_one.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:text="This is fragment 1"
        android:textSize="20sp" />
    <RatingBar
```

```
        android:id="@+id/ratingBar1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textView1"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="19dp" />
    <CheckBox
        android:id="@+id/checkBox1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/ratingBar1"
        android:layout_below="@+id/ratingBar1"
        android:text="CheckBox" />
</RelativeLayout>
```

fagment_two.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:text="This is fragment 2"
        android:textSize="20sp" />

    <RatingBar
        android:id="@+id/ratingBar1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textView1"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="19dp" />

    <RadioButton
```

```
            android:id="@+id/radioButton1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignLeft="@+id/ratingBar1"
            android:layout_below="@+id/ratingBar1"
            android:text="RadioButton" />
        <RadioButton
            android:id="@+id/radioButton2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignLeft="@+id/radioButton1"
            android:layout_below="@+id/radioButton1"
            android:text="RadioButton" />
    </RelativeLayout>
```

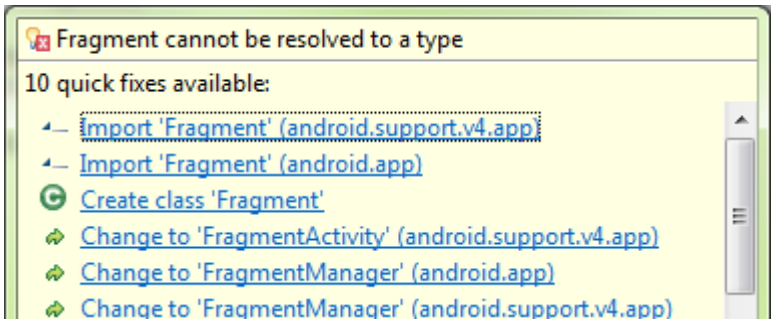
fagment_three.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:text="This is fragment 3"
        android:textSize="20sp" />
    <RatingBar
        android:id="@+id/ratingBar1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textView1"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="19dp" />
    <SeekBar
        android:id="@+id/seekBar1"
        android:layout_width="match_parent"
```

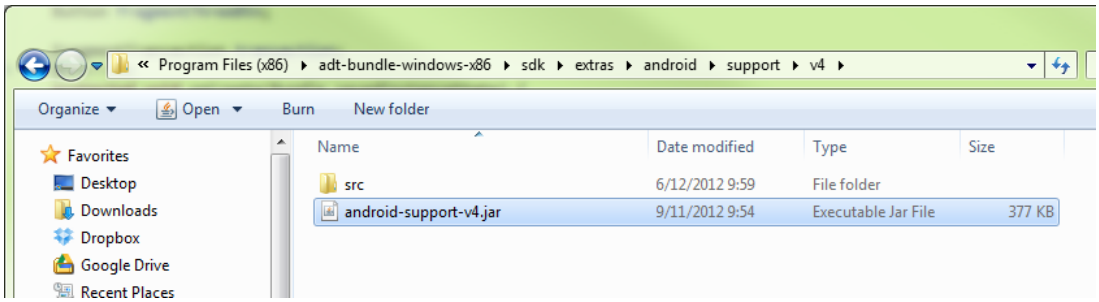
```
        android:layout_height="wrap_content"
        android:layout_below="@+id/ratingBar1"
        android:layout_centerHorizontal="true" />
    <SeekBar
        android:id="@+id/seekBar2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/seekBar1" />
    <SeekBar
        android:id="@+id/SeekBar01"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/seekBar2" />
</RelativeLayout>
```

2.2 Write codes for main activity and fragments

Note: Import **android.support.v4.app** when prompt in order to provide backward compatibility from Android 1.6 to Android 2.x

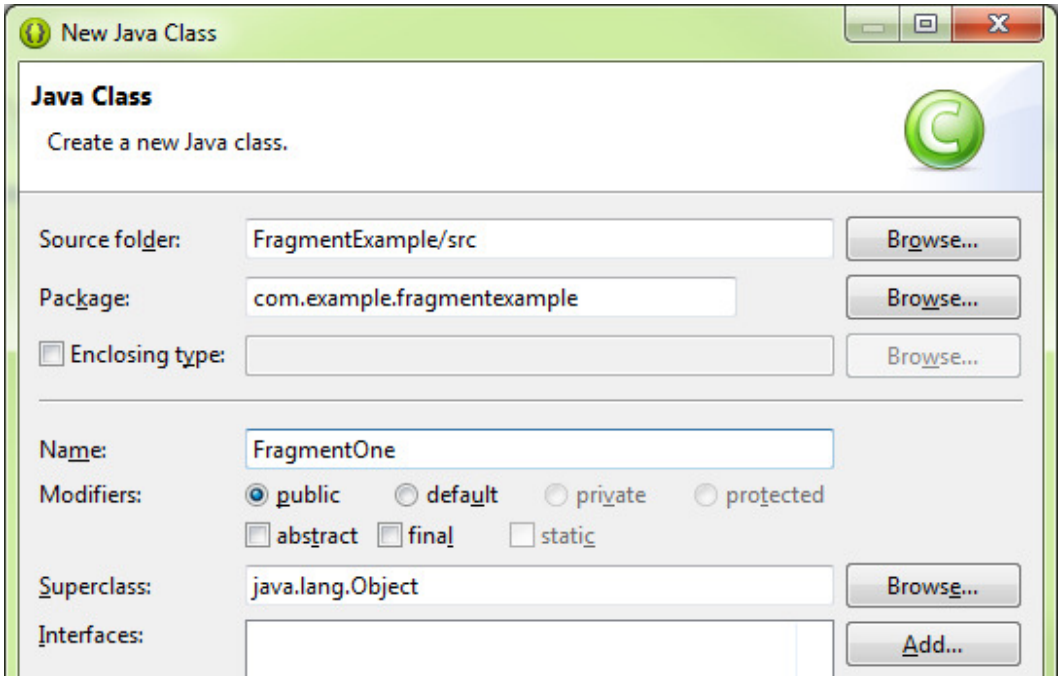


1. If the **libs** folder does not exist in the project root, simply create it and copy the **android-support-v4.jar** from **<your_sdk_folder>/sdk/extras/android/support/v4/ android-support-v4.jar**.



2. Right click the **android-support-v4.jar** in Package Explorer and then select **Build Path > Add to Build Path**.

3. We will create three fragment java classes and associate them with the XML layout files respectively. To create a blank java class file, right click **src/com.example.fragmentexample** and select **New > Class.** Enter **FragmentOne** for the name of the first fragment class.



4. Similar to the way of creating an activity, we need to extend the **Fragment** class if we want to create a fragment. To define the layout of a fragment, use **LayoutInflater** to load the XML layout in **onCreateView()**.

   Replace the content of the first fragment class with the following codes:

```
package com.example.fragmentexample;
import android.os.Bundle;

import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class FragmentOne extends Fragment{
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
        // Inflate the layout for this fragment
```

```
            return inflater.inflate(R.layout.fragment_one, container, false);
        }
    }
```

5. To create the second and the third fragment classes, select FragmentOne.java in the **Package Explorer**, copy and paste it as **FragmentTwo.java** and **FragmentThree.java** respectively.

   Note: Replace the resource (**R.layout.fragment_one**) with **FragmentTwo** and **R.layout.fragment_two respectivelty** for **FragmentTwo.java**. Perform the same steps with **FragmentThree** and **R.layout.fragment_three** for **FragmentThree.java**.

6. To perform a **transaction** such as switch between different fragments in the main activity, we will use the **FragmentManager** to create a **FragmentTransaction**. The **FragmentTransaction** allows us to add, remove, replace or even add the fragment to the back history stack. The **FrameLayout** defined in **activity_main.xml** is used to act as fragment container to hold the fragments.

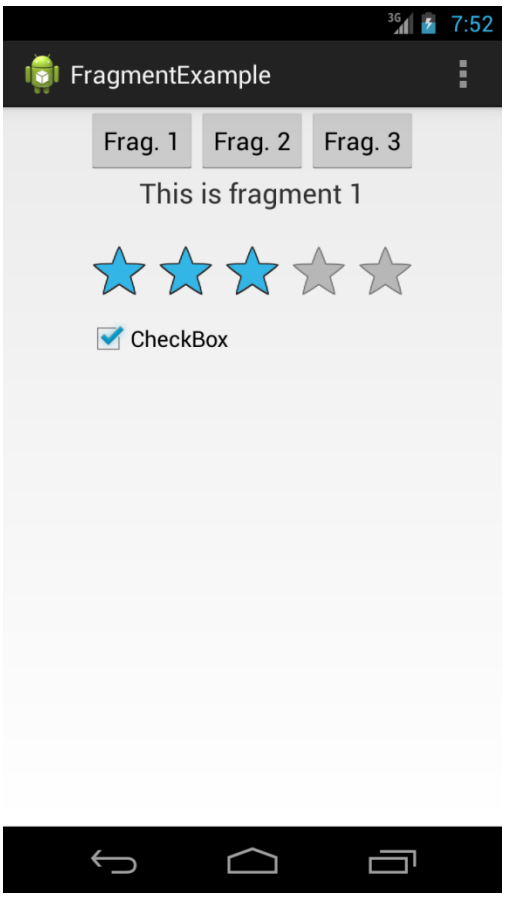
   Replace the content of **MainActivity.java** with the following code:

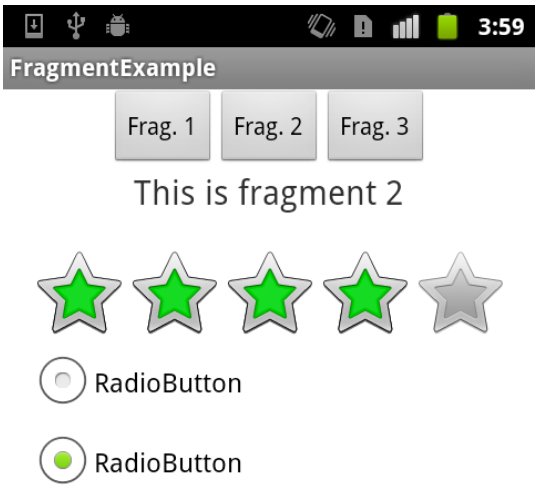
```
package com.example.fragmentexample;
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;
import android.support.v4.app.FragmentTransaction;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class MainActivity extends FragmentActivity {

    FragmentOne fragmentOne;
    FragmentTwo fragmentTwo;
    FragmentThree fragmentThree;
    Button fragmentOneBtn;
    Button fragmentTwoBtn;
    Button fragmentThreeBtn;
```

```
        FragmentTransaction transaction;
        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);
            fragmentOneBtn = (Button)findViewById(R.id.fragButton1);
            fragmentTwoBtn = (Button)findViewById(R.id.fragButton2);
            fragmentThreeBtn = (Button)findViewById(R.id.fragButton3);

            fragmentOneBtn.setOnClickListener(fragmentOneBtnListener);
            fragmentTwoBtn.setOnClickListener(fragmentTwoBtnListener);
            fragmentThreeBtn.setOnClickListener(fragmentThreeBtnListener);

            fragmentOne = new FragmentOne();
            fragmentTwo = new FragmentTwo();
            fragmentThree = new FragmentThree();


   getSupportFragmentManager().beginTransaction().add(R.id.fragment_container,
fragmentOne).commit();
        }

        private OnClickListener fragmentOneBtnListener = new OnClickListener(){
            @Override
            public void onClick(View arg0) {
                transaction = getSupportFragmentManager().beginTransaction();
                transaction.replace(R.id.fragment_container, fragmentOne);
                transaction.commit();
            }
        };
        private OnClickListener fragmentTwoBtnListener = new OnClickListener(){
            @Override
            public void onClick(View arg0) {
                transaction = getSupportFragmentManager().beginTransaction();
                transaction.replace(R.id.fragment_container, fragmentTwo);
                transaction.commit();
            }
        };
```

```
    private OnClickListener fragmentThreeBtnListener = new OnClickListener(){
        @Override
        public void onClick(View arg0) {
            transaction = getSupportFragmentManager().beginTransaction();
            transaction.replace(R.id.fragment_container, fragmentThree);
            transaction.commit();
        }
    };
}
```

7. Now try to run the app and you will able to switch the fragments when you click on the buttons. It also works on Android 1.6 to Android 2.x devices since we are utilizing the android-support-v4 library in this example.





The screeshot on the left is taken on Android 4.x while the right one is taken on Android 2.x.

The complete project (sample codes) is available at **FragmentExample.zip**

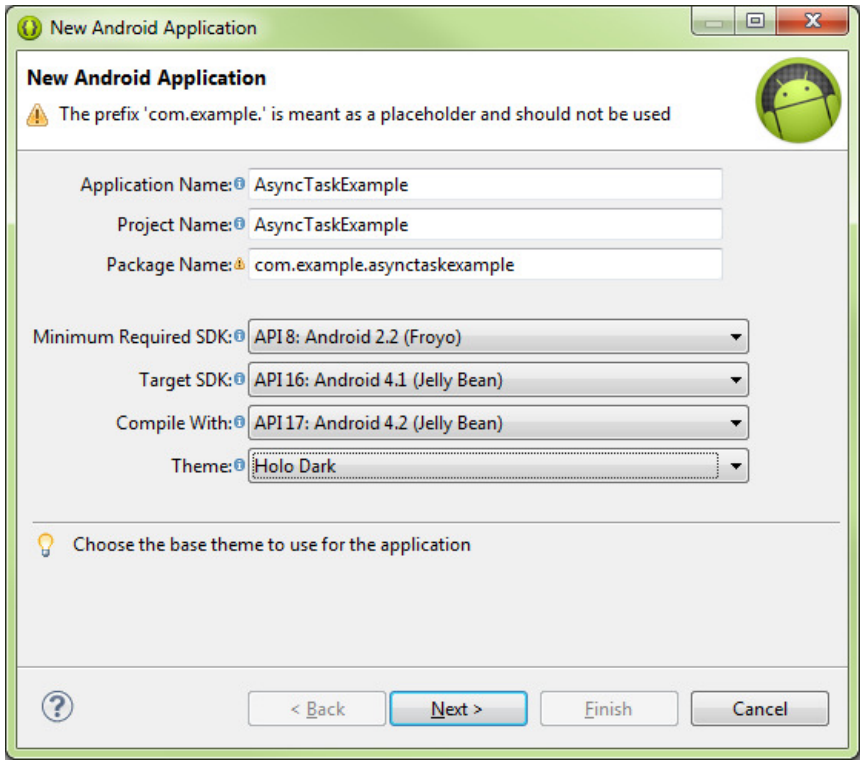
3 Handle long-running task

If you have certain time consuming tasks or there are some operations need to be executed in background, you must implement the block of codes in a separated thread respectively. Since the main thread of an app in Android will also act as a UI thread, it is very important to keep this thread to response users action immediately, otherwise they will receive an **ANR** (Application not responding) dialog. In this workshop, we will use **AsyncTask** to achieve multi-threading.

When to use AsyncTask and Service?
To make it simple, use **Service** if the task doesn't require interacting with the main thread (e.g. UI elements), and the task needs to be executed in background (e.g. Listen to an incoming message) even the app is not brought to the front.
On the other hand, use **AsyncTask** for those tasks which will block the main thread. However, the limitation of **AsyncTask** is that it can be created and executed from the main thread only.

3.1 Use AsyncTask to load an image from Internet

1. Start a new **Android Application Project** in Eclipse and refer to the screenshot below for application details. Again we will also create a new **BlankActivity** with default activity name, layout and navigation type.



2. Replace the contents in **activity_main.xml** with the following markups. We will add an **ImageView** and a **Button** to the screen.

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

    <ImageView
        android:id="@+id/myImage"
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:scaleType="fitStart"/>

    <Button
     android:id="@+id/loadImage"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Load image"/>

     <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"/>

</LinearLayout>
```

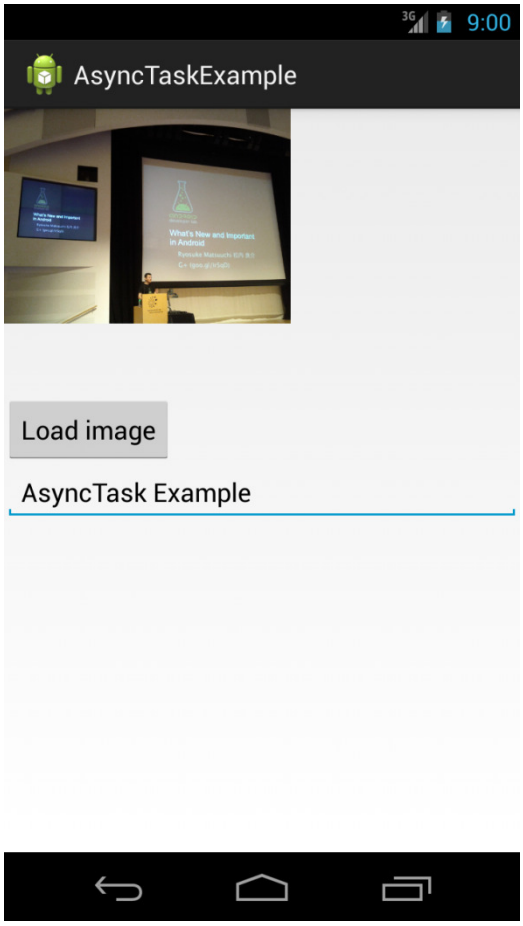
3. Replace the contents in **src/MainActivity.java** with the following code:

```
package com.example.asynctaskexample;
import java.io.InputStream;
import java.net.URL;
import java.net.URLConnection;

import android.app.Activity;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.AsyncTask;
```

```
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.ImageView;

public class MainActivity extends Activity {

 ImageView iv;
 Button loadBtn;

 @Override
 protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_main);
       iv = (ImageView)findViewById(R.id.myImage);
       loadBtn = (Button)findViewById(R.id.loadImage);
       loadBtn.setOnClickListener(loadBtnListener);
 }

 private OnClickListener loadBtnListener = new OnClickListener(){
       @Override
       public void onClick(View arg0) {
             new DownloadImageTask().execute();
       }
      };


 private class DownloadImageTask extends AsyncTask<Integer, Void, Bitmap> {
       @Override
           protected Bitmap doInBackground(Integer... params) {
            try{
                  String imgLink =
"http://www.comp.hkbu.edu.hk/~sigis/android/public/picture1.jpg";
                  URL url = new URL(imgLink);
                  URLConnection conn = url.openConnection();
                        conn.connect();
                        InputStream is = conn.getInputStream();
```

```
                    BitmapFactory.Options options = new
BitmapFactory.Options();
                    options.inSampleSize = 4;
                    Bitmap bm = BitmapFactory.decodeStream(is, null, options);
                    is.close();
                    return bm;
                } catch (Exception e) {
                    return null;
                }
            }

            @Override
            protected void onPostExecute(Bitmap result) {
              super.onPostExecute(result);
              if(result!=null){
                    iv.setImageBitmap(result);
              }
            }
        }
}
```

4. Since we will get the image through the Internet, we have to declare additional permission in AndroidManifest.xml. Go to the **AndroidManifest.xml** and add **<uses-permission android:name="android.permission.INTERNET"/>** above the **<application>** tag.
5. Run the app and click on the **Load image** button. After a while, you will have the image loaded in the **ImageView**. You can continue with your typing in the text field while the image is still loading.



6. To see the difference between with and without using the AsyncTask, you can copy the code from **doInBackground** to and add it to the **OnClickListener** inside **MainActivity.java.** Next, add **Thread.sleep(10000)** before loading the image to onClick() and doInBackground() respectively . You will find that you won't be able to type anything in the text field while the image is loading and the ANR dialog will be popped out soon.

   You may refer to the code below of loadBtnListener for not using AsyncTask:

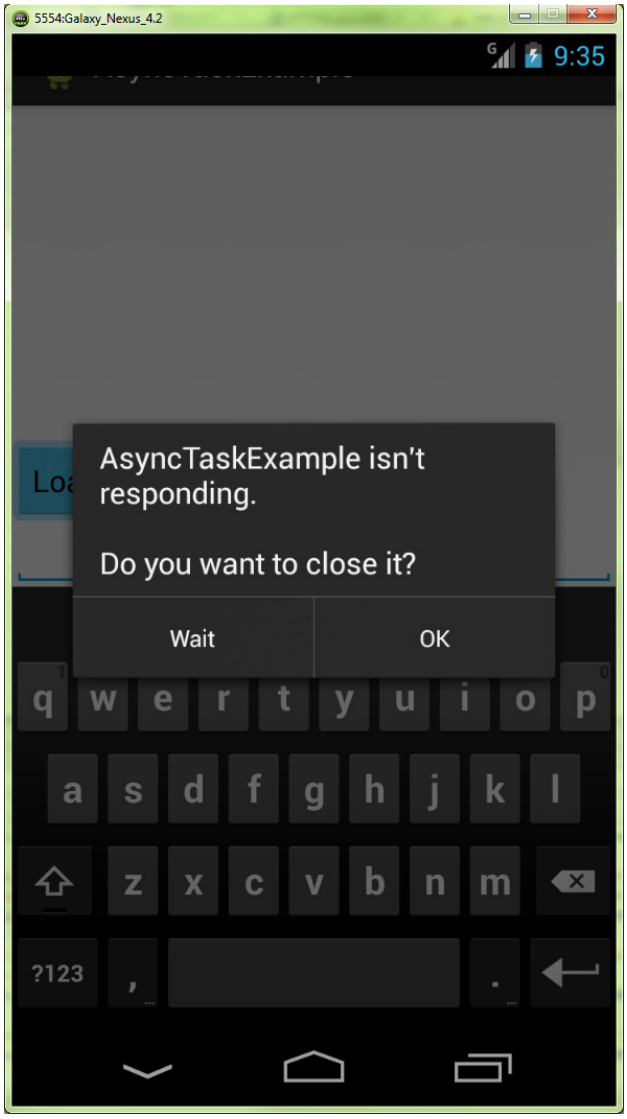
```
private OnClickListener loadBtnListener = new OnClickListener(){
        @Override
        public void onClick(View arg0) {
            //new DownloadImageTask().execute();
            try{
                Thread.sleep(10000);
                String imgLink =
"http://www.comp.hkbu.edu.hk/~sigis/android/public/picture1.jpg";
```

```
                URL url = new URL(imgLink);
                URLConnection conn = url.openConnection();
                    conn.connect();
                    InputStream is = conn.getInputStream();
                    BitmapFactory.Options options = new
BitmapFactory.Options();
                    options.inSampleSize = 4;
                    Bitmap bm = BitmapFactory.decodeStream(is, null, options);
                    is.close();
                    iv.setImageBitmap(bm);
                } catch (Exception e) {

                }
        }
      };
```

Note: Starting from Android 3.0, the system does not allow any network operation on the main thread. In order to simulate the **Activity Not Responding** situation, you also need to add the following code in the onCreate() otherwise android.os.NetworkOnMainThreadException will be thrown.

```
StrictMode.ThreadPolicy policy =
        new StrictMode.ThreadPolicy.Builder().permitAll().build();
StrictMode.setThreadPolicy(policy);
```

And then add **@SuppressLint("NewApi")** above **@Override** of the **onCreate()** since we set the minSdkVersion to 8 in AndroidManifest.xml


5554:Galaxy_Nexus_4.2
9:35
AsyncTaskExample isn't responding.
Do you want to close it?
Wait
OK

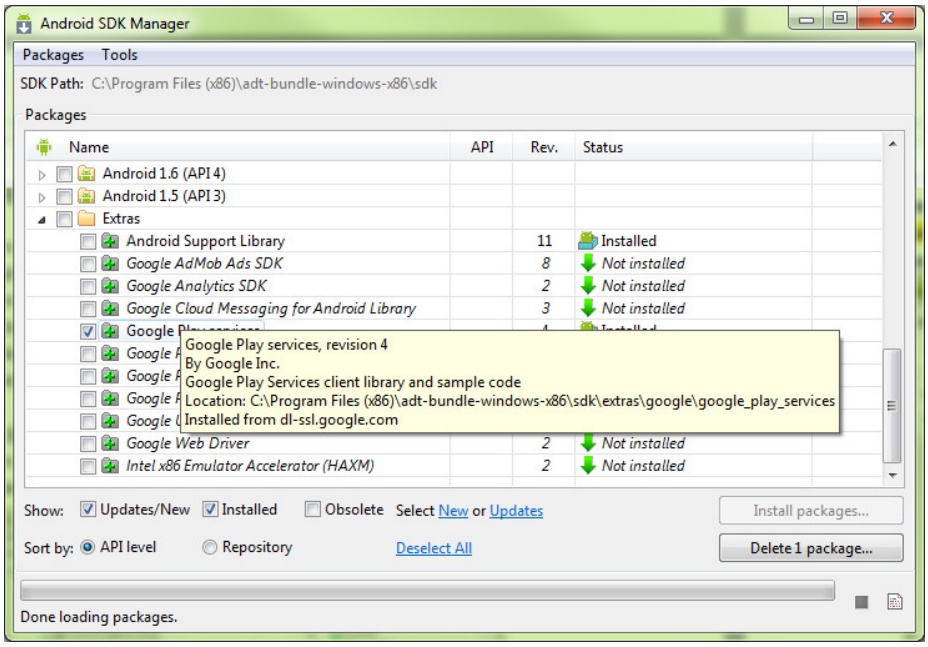
4 Embed Google Maps (API v2) in your app

There are several prerequisites for embedding Google Maps into any apps using Google Maps API v2:

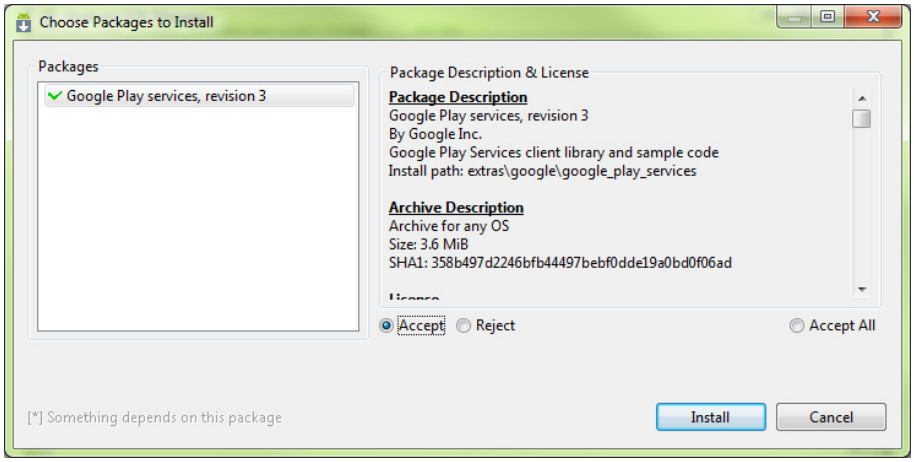
1. Download and configure Google Play services SDK
2. Get two Maps API keys for debug build and release build respectively
3. Modify settings in AndroidManiFest.xml
4. Add the Maps (MapFragment) to the activity

4.1 Download and configure Google Play services SDK

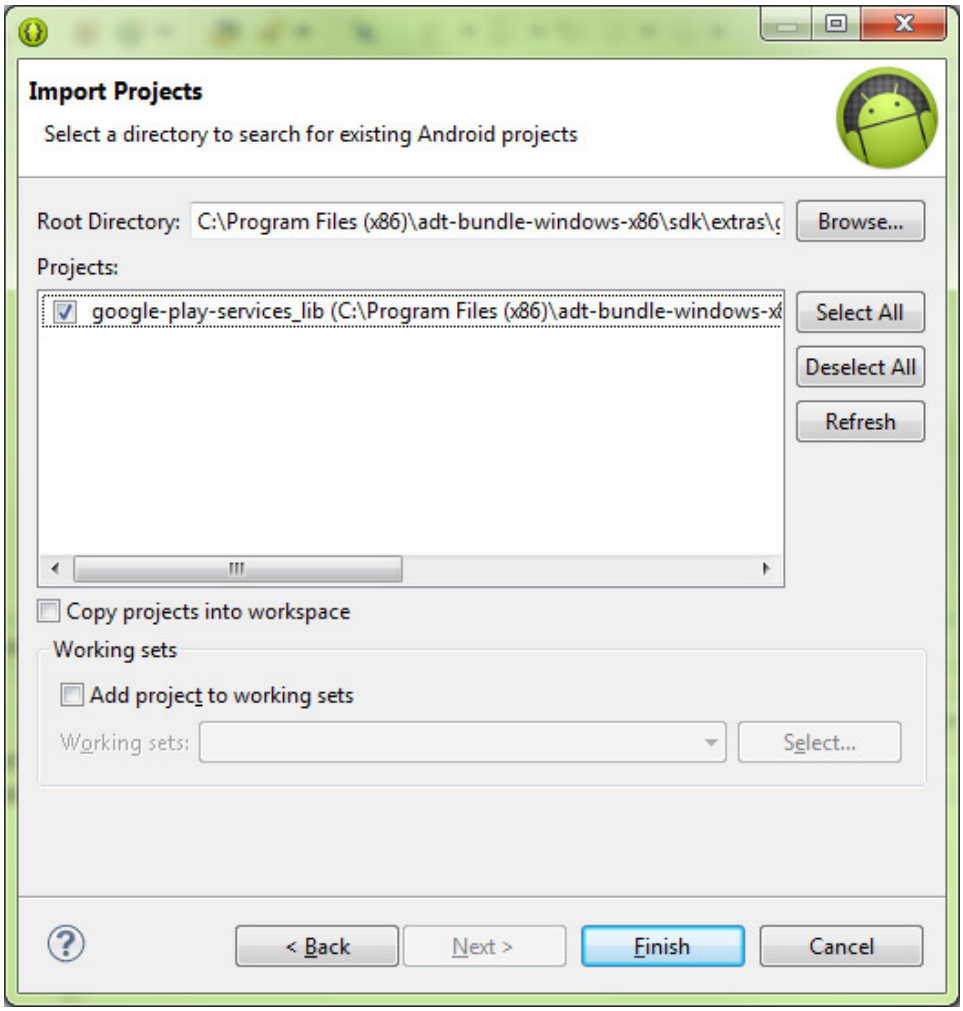
1. In Eclipse, select **Window > Android SDK Manager**.
2. Select **Google Play services** and then click **Install**.



3. Accept the license and click **Install** again.



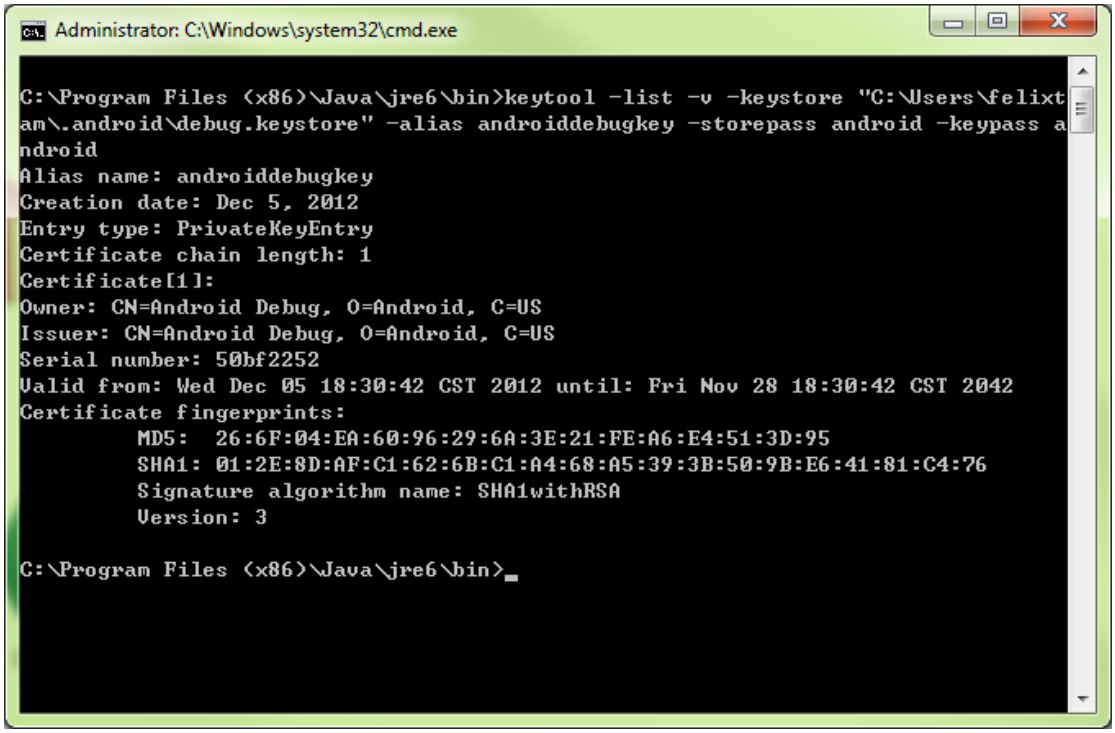
4. In Eclipse, select **File > Import > Android > Existing Android Code Into Workspace**. Click **Browse** and navigate to select **\adt-bundle-windows-x86\sdk\extras\google\google_play_services\libproject\google-play-services_lib** for the **Root Directory**.



5. Select **google-play-services_lib** under **Projects** and click **Finish** to import the library project.

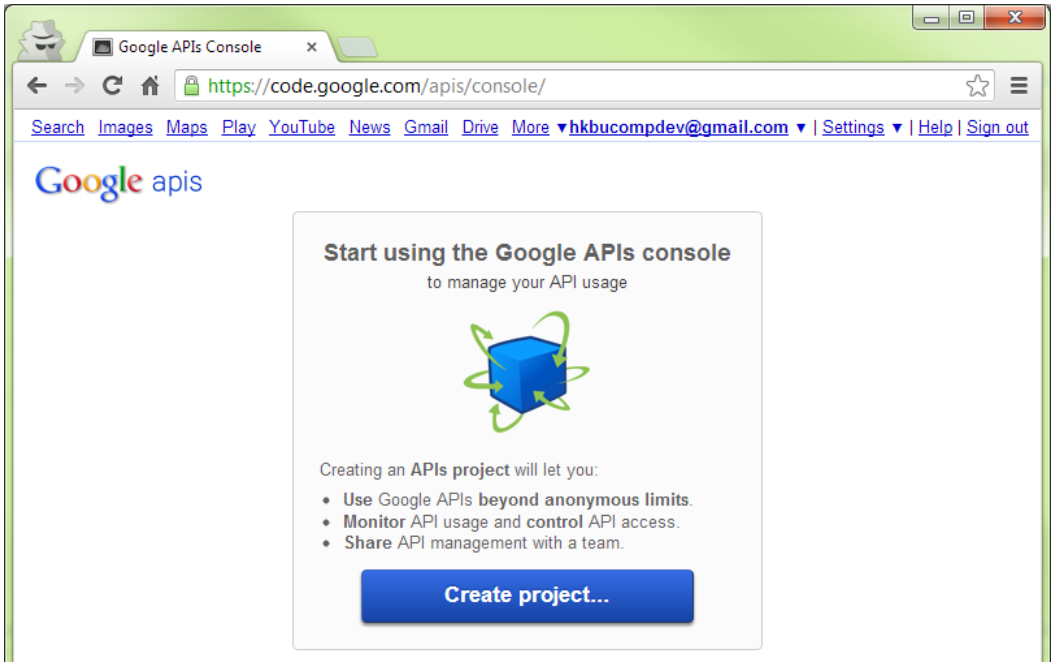
4.2 Obtain Google Maps API key

1. Locate the keytool.exe in bin folder of Java JDK or JRE. E.g. C:\Program Files\Java\jre6\bin. Press **Shift+Right Click** in the bin folder and choose **Open command window here**.
2. Locate the **debug.keystore** (Debug keystore) in **C:\Users\<user_name>\.android\.** Type **keytool -list -v -keystore "C:\Users\<user_name>\.android\debug.keystore" -alias androiddebugkey -storepass android -keypass android**.

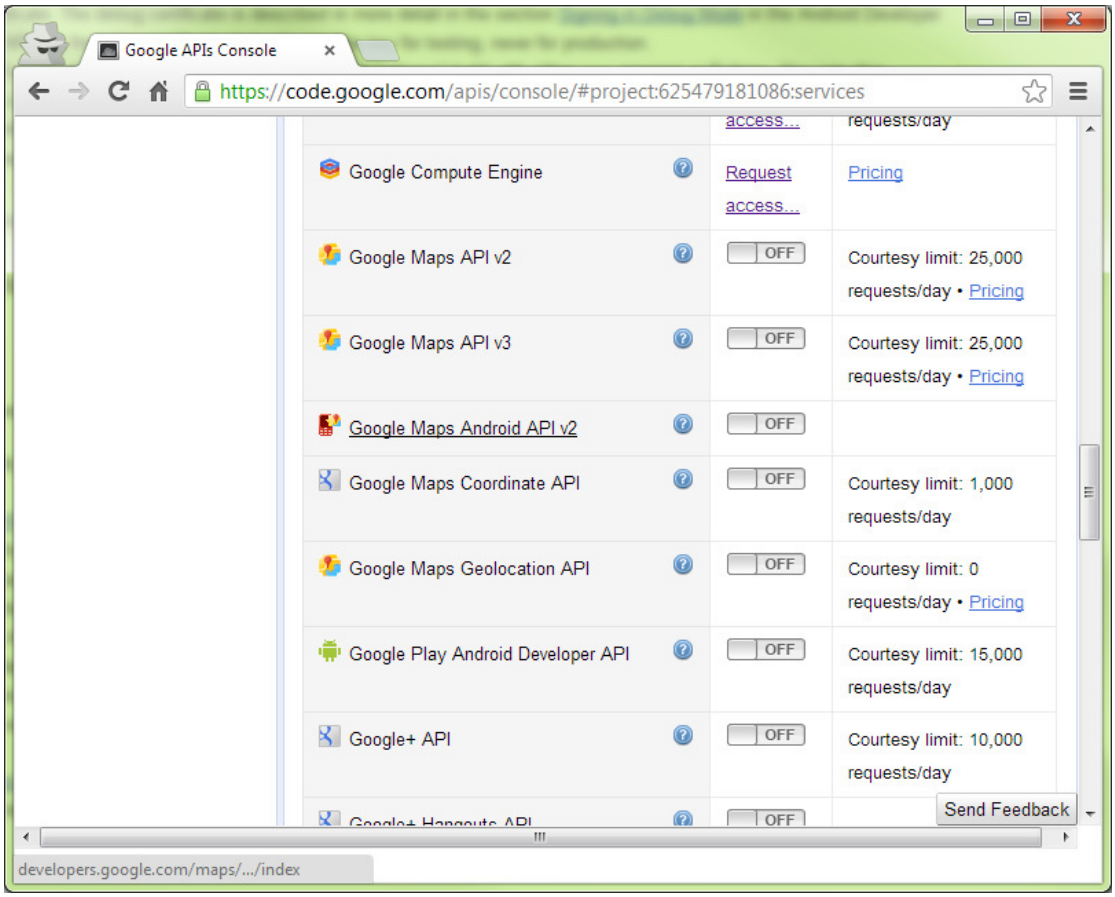


Note: If you are using windows XP, replace **C:\Users\<user_name>** with **C:\Documents and Settings\<user_name>** of the path.

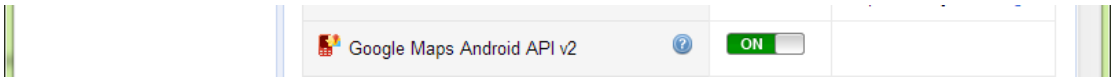
3. Copy the 20 two-digit hexadecimal numbers SHA-1 fingerprint. This fingerprint will be used to obtain the corresponding Google Maps API key. (You can copy the numbers by **Right click > Highlight the number > Right click**)
4. Start your browser and go to https://code.google.com/apis/console/. Login with your Google account and click the **Create project** button.



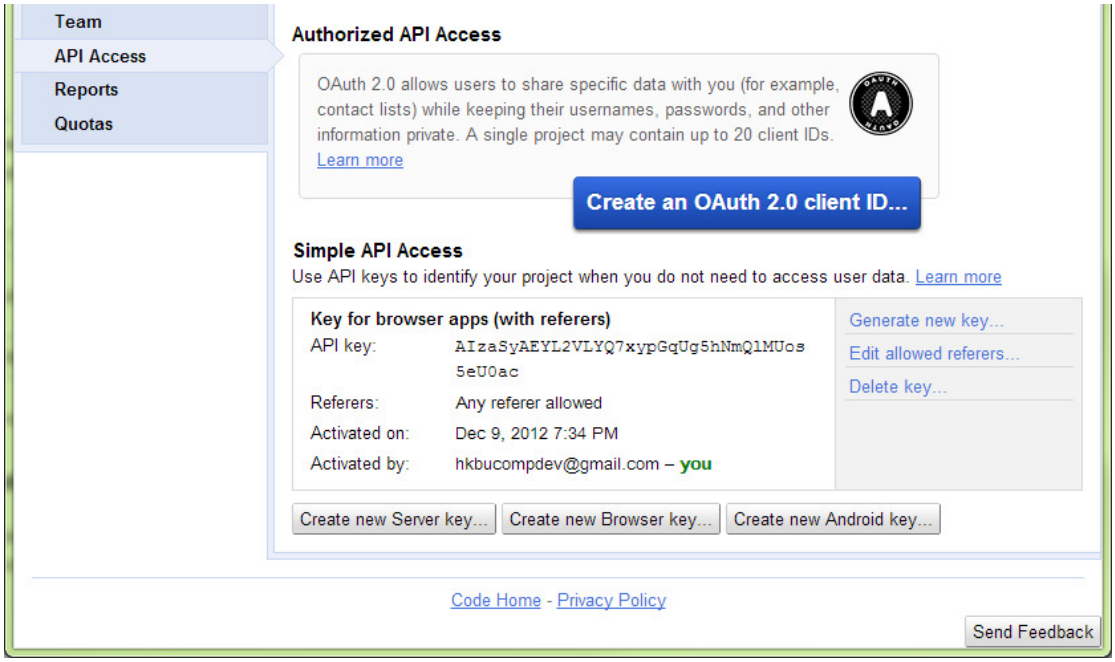
5. Scroll down and find out **Google Maps Android API v2.** Activate it and accept the license agreement.



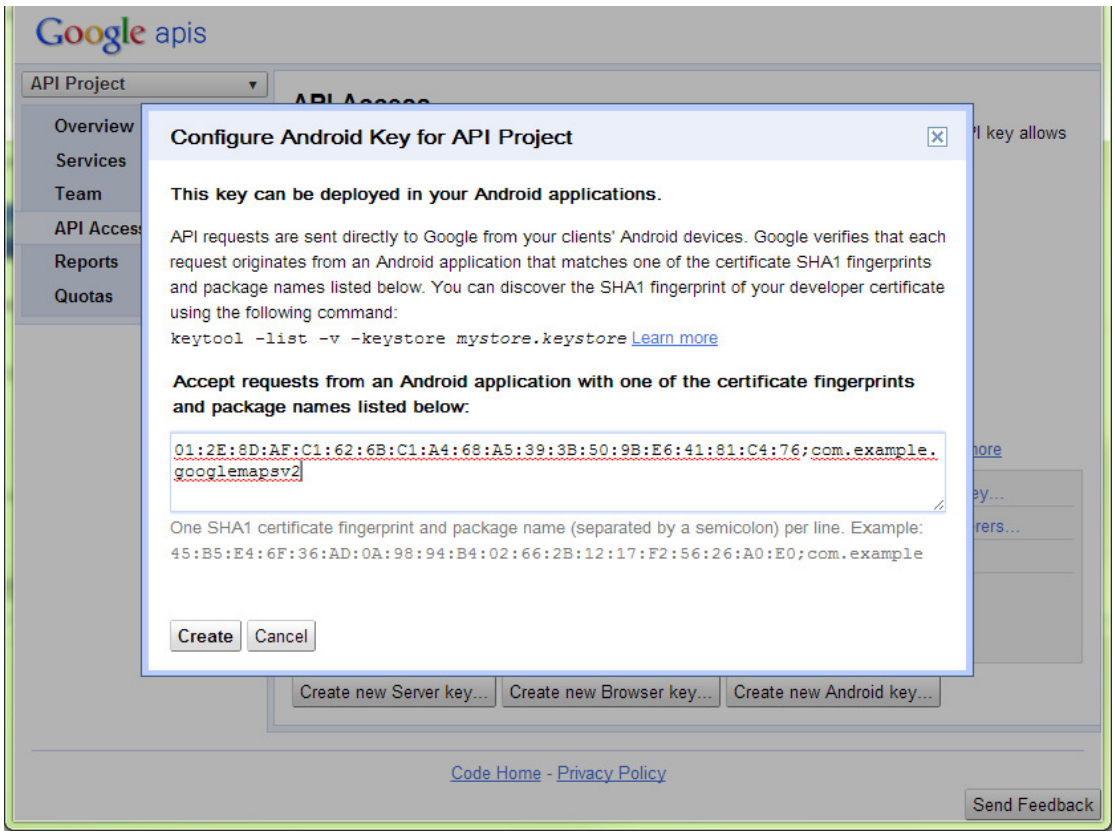
6. Make sure the status of the API is **On**.



7. Click on **API Access** in the left panel and select **Create new Android key**.

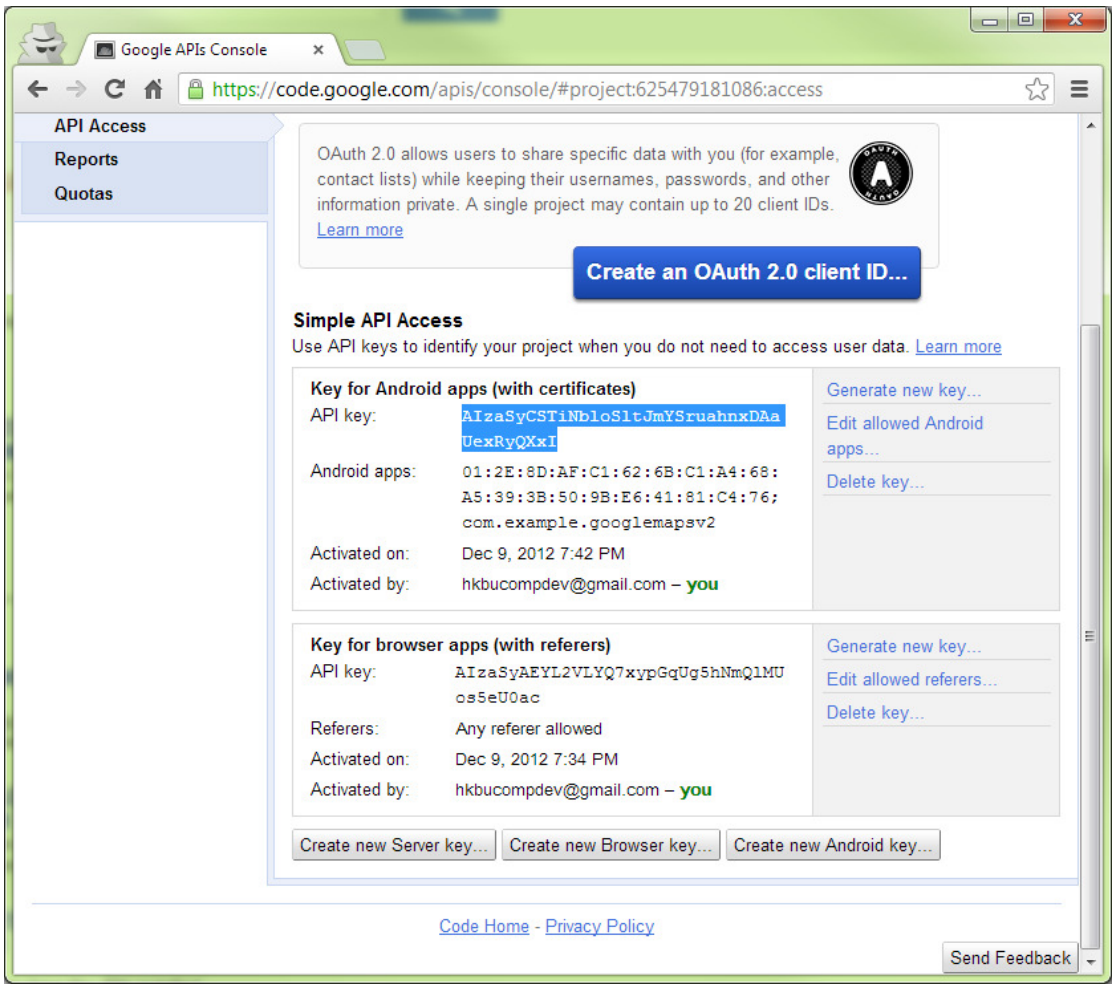


8. Paste the SHA-1 fingerprint and add the package name of the app separated by a semi-colon. In our example, the package name will be **com.example.googlemapsv2** which will be used in the next section.



9. Click the create button and you will get the API key of the debug certificate under

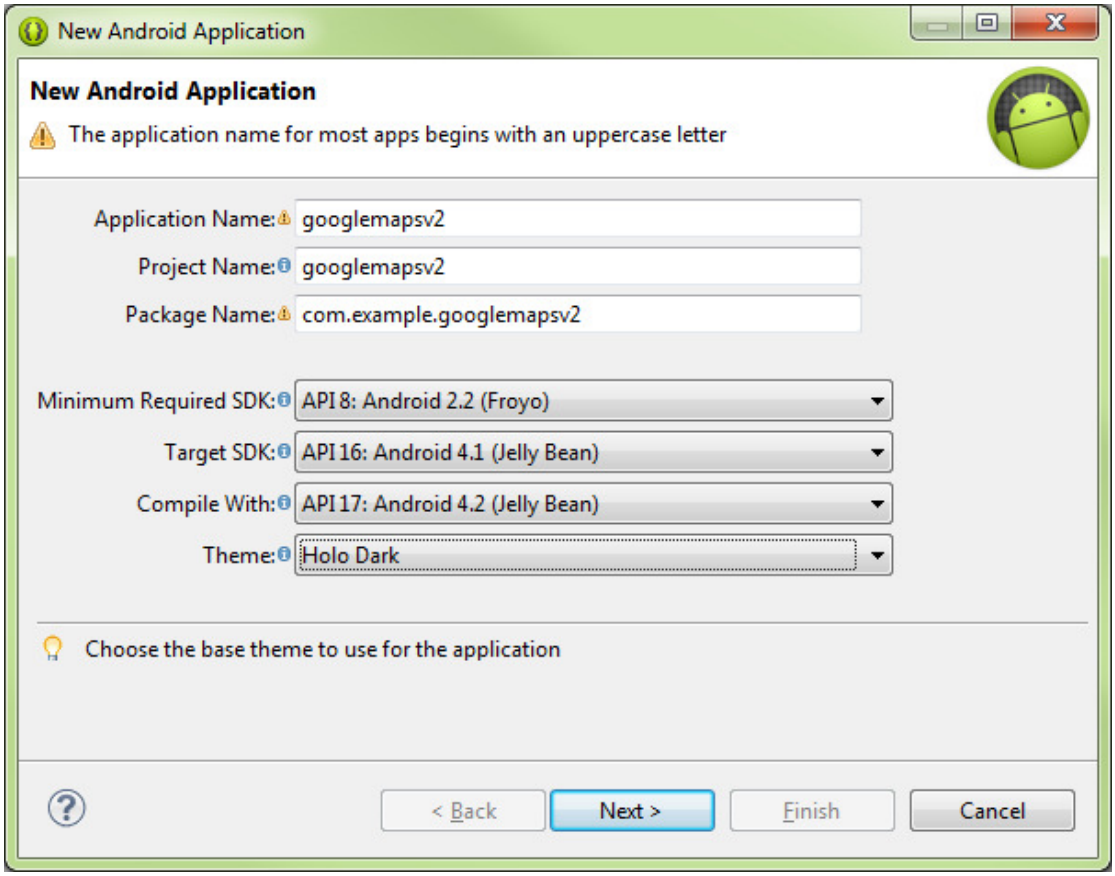
Key for Android apps (with certificates).



10. For the API key of the release certificate, simply repeat the steps but replace the location of the debug certificate with the release certificate in **step 2**.

4.3 Embed Google Maps in your app

1. Start a new **Android Application Project** in Eclipse and refer to the screenshot below for application detail. Again we will also create a new **BlankActivity** with default activity name, layout and navigation type.

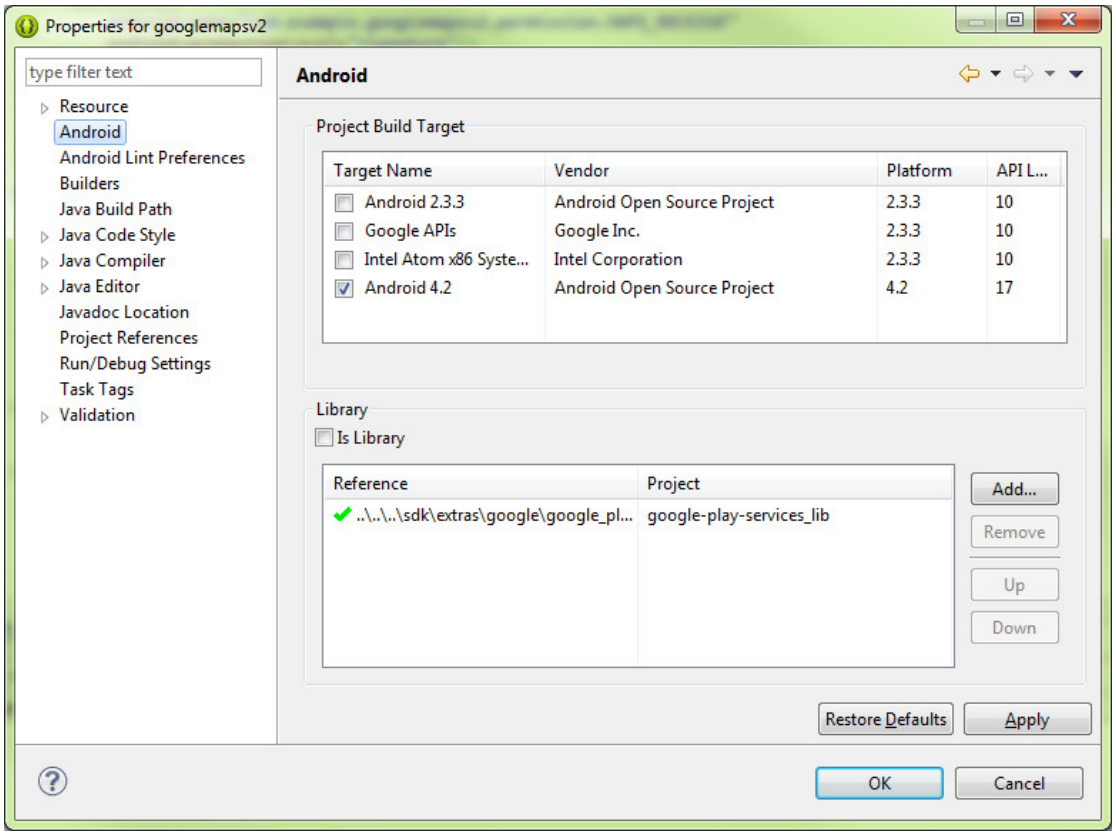


2. Open AndroidManifest.xml and add the following markups:
   - Add **<meta-data android:name="com.google.android.maps.v2.API_KEY" android:value="your_api_key"/>** as a child of **<application>** element. Replace the API key with the one you generated in step 9 of section 4.2.
   - Add **<permission android:name="com.example.googlemapsv2.permission.MAPS_RECEIVE" android:protectionLevel="signature"/> <uses-permission android:name="com.example.googlemapsv2.permission.MAPS_RECEIVE" />** above the **<application>** tag. Remember to replace the **com.example.googlemapsv2** with your own package name of your app.
   - Add **<uses-permission android:name="android.permission.INTERNET"/> <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/> <uses-permission android:name= "com.google.android.providers.gsf.permission.READ_GSERVICES"/>** above the **<application>** tag. This will ask the permission from users to

download the map tiles through the Internet, allow the API to access Google web-based services and cache the map tiles in the external storage respectively.

- Add
  <uses-feature
  android:glEsVersion="0x00020000"
  android:required="true"/> above the **<application>** tag. This is used to declare the hardware or software feature which is required of a device.

3. In Package Explorer, right click the project **(googlemapsv2)** and select **Properties > Android > Add**. Select **google-play-services_lib** and press **OK**. This will include the Google Play Service library when compiling the app.



4. Replace the content of **activity_main.xml** with the following markups:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity" >
```

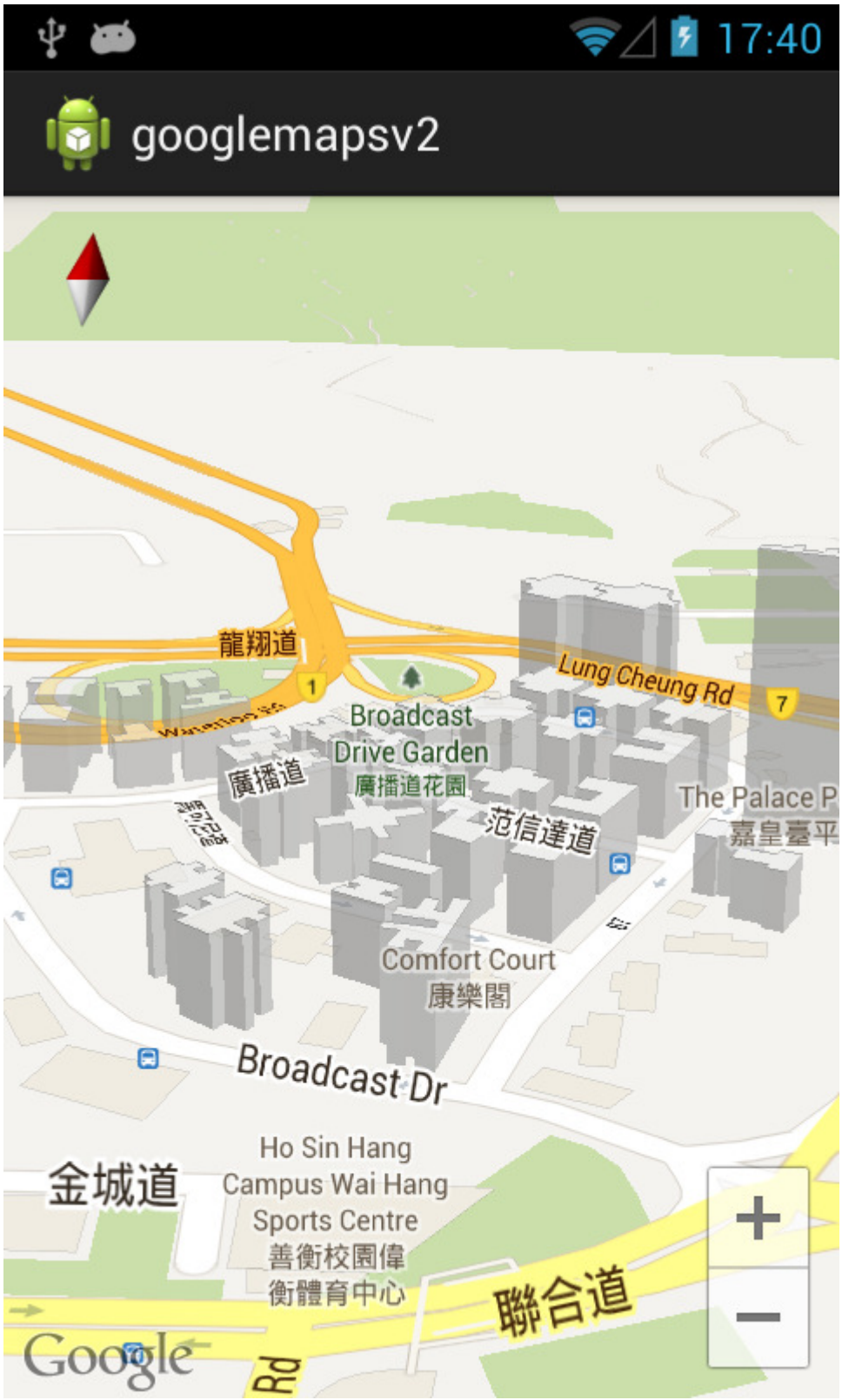
```
    <fragment xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/map"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        class="com.google.android.gms.maps.SupportMapFragment"/>

</RelativeLayout>
```

5. In package explorer, right click **libs/android-support-v4.jar** and choose **Build Path** > **Add to the Build Path**.

6. Replace the contents in MainActivity.java with the following code:

```
package com.example.googlemapsv2;
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;

public class MainActivity extends FragmentActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

7. Please note that you have to test the app in the real device instead of emulator. Emulator does not include Google Play Service (Even you pushed Google Play services APK to the emulator). Remember to activate your internet connection. You will soon see the map and able to use two fingers to change the bearing and angle of the view.



The complete project (sample codes) is available at **googlemapsv2.zip**

5 Reference and learning resources

- Official Android developer website:
  http://developer.android.com/index.html
- Many common questions raised by other developers previously:
  http://stackoverflow.com/
- Many Android tutorials with complete source code:
  http://www.anddev.org/
- More details on implementing Action Bar
  http://developer.android.com/guide/topics/ui/actionbar.html
- Special Interest Group on Innovative Software homepage:
  http://www.comp.hkbu.edu.hk/~sigis/