Android SDK and Eclipse IDE

Installing Android and Eclipse

You need to install the Android SDK; you should also install the Eclipse IDE. Instructions are found at http://developer.android.com/sdk/installing.html

Android SDK+Eclipse is installed in computer lab 7017.

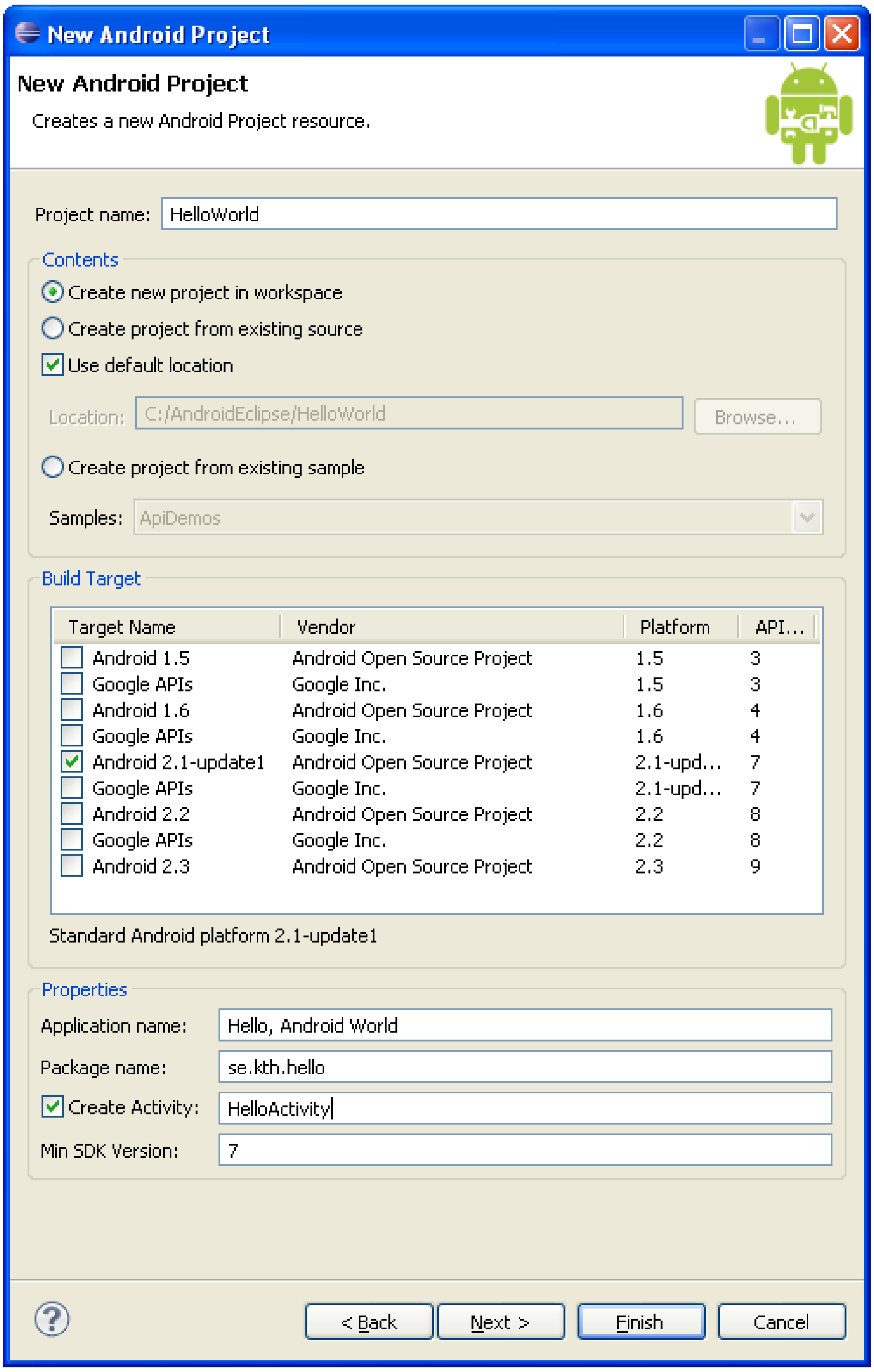
Set the path to the Android SDK

The first time you run Eclipse you should check that the path to the Android SDK is set.
From the menu, select Window/Preferences and click on Android. Browse to the Android SDK location.

Hello World

Select File/New/Android Project from the menu.

Fill in the Project Name, chose a Build Target (at present time, 2.1 is recommended). Then add the Application Name, Package Name (at least one sub package is required), the Activity Name and the Minimum SDK version (make sure this version matches the Build Target).
An example is provided below.



New Android Project

Creates a new Android Project resource.

Project name: HelloWorld

Contents

● Create new project in workspace

○ Create project from existing source

✓ Use default location

Location: C:/AndroidEclipse/HelloWorld   Browse...

○ Create project from existing sample

Samples: ApiDemos

Build Target

Target Name	Vendor	Platform	API...
☐ Android 1.5	Android Open Source Project	1.5	3
☐ Google APIs	Google Inc.	1.5	3
☐ Android 1.6	Android Open Source Project	1.6	4
☐ Google APIs	Google Inc.	1.6	4
✓ Android 2.1-update1	Android Open Source Project	2.1-upd...	7
☐ Google APIs	Google Inc.	2.1-upd...	7
☐ Android 2.2	Android Open Source Project	2.2	8
☐ Google APIs	Google Inc.	2.2	8
☐ Android 2.3	Android Open Source Project	2.3	9

Standard Android platform 2.1-update1

Properties

Application name: Hello, Android World

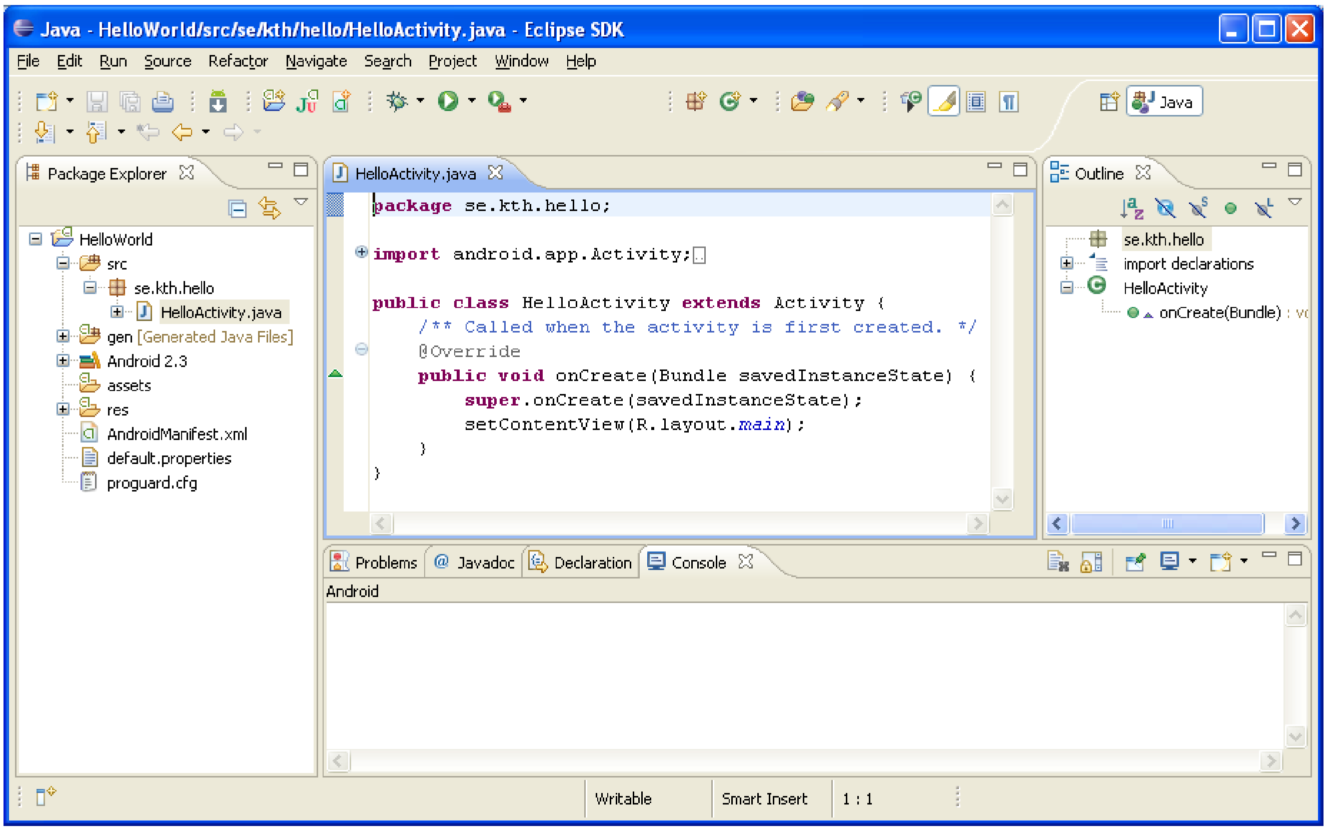
Package name: se.kth.hello

✓ Create Activity: HelloActivity

Min SDK Version: 7

Select Finish.

Your project, including a class for your main activity and some resources, is created. The source code for this class is located in the in the Package Explorer (left window), src/se.kth.hello/HelloActivity.java.



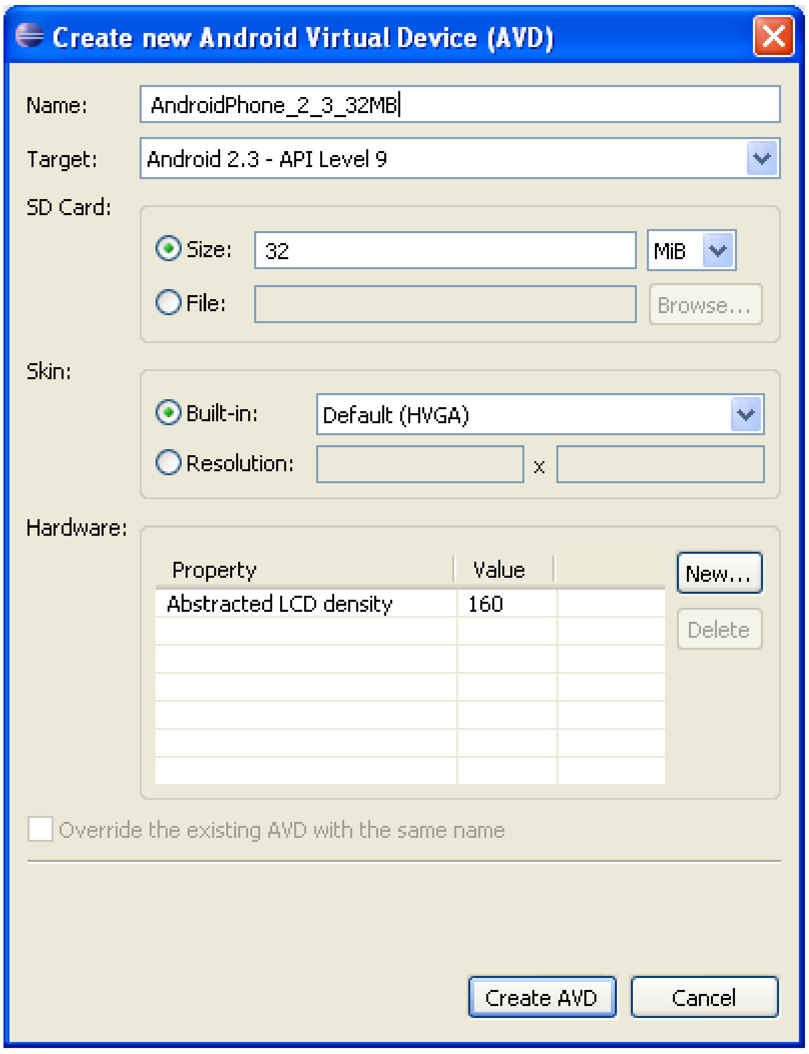
```
package se.kth.hello;

import android.app.Activity;

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

Create an Android Virtual Device, AVD

To (test) run your application, you need an Android Virtual Device, that is an emulator emulating a smart phone on your computer.

Select from the menu, Window/Android SDK and AVD Manager. Select Virtual Devices and click on New, then fill in a Name for your AVD, pick a Target (must match the Build Target) and size of the SD-card (memory card). See the example below.

If you want to add properties, such as an accelerometer, to your virtual device, select Hardware/New.



Name: AndroidPhone_2_3_32MB
Target: Android 2.3 - API Level 9
SD Card:
(●) Size: 32 MiB
( ) File:
Skin:
(●) Built-in: Default (HVGA)
( ) Resolution: x
Hardware:

Property	Value
Abstracted LCD density	160

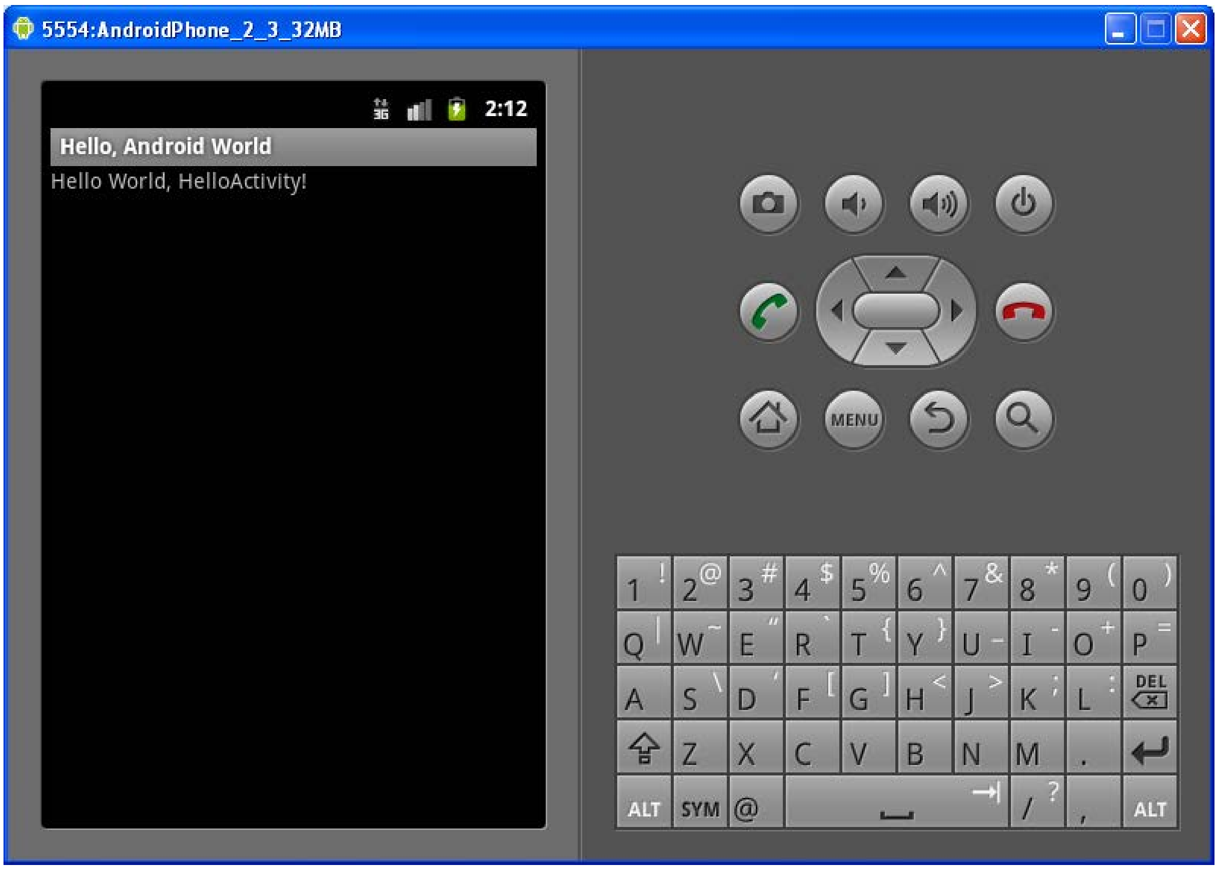
☐ Override the existing AVD with the same name

Click the Create AVD button.

For testing purpose, you may create several AVDs, with different configurations.

Run your application

Right-click on the icon for your project in the left window, then select Run As…/Android Application. If there are no errors, your Virtual Device is launched. This will take some time, be patient. You should then see a device as below.



Hello World, HelloActivity!

If you click on the "back button" on the virtual device, you will see the main menu of the phone. To restart your application, click on the menu button, or on Launcher Icon on the device screen, and select your application.

A small change in the layout

The components in an Android User Interface are usually defined in an XML-file.
The line `setContentView(R.layout.main)` in the HelloActivity class refers to a layout defined in the in the file res/layout/main.xml (select the main.xml tab at the bottom to see the xml code).
The line `android:text="@string/hello"` tell us the message is defined in another xml file, res/values/strings.xml. Open this file and change the value of the string labeled hello.

Run the new version of your application

In Eclipse, select Run As…/Android Application to upload the new version of your application to your virtual device.
There's no need to restart the virtual device between different runs; if you do so you once again have to wait for a looooong time.

Problems and some "magic"

If you encounter problems with your application although the Java and XML-code seems correct (e.g. the R-file is marked missing) the actions listed below might be of use.

- Select Project/Clean, and then Project/Build project.
- Right-click the project header and select Android Tools/Fix Project Properties.

Managing the Android Manifest file

When e.g. adding Activities or permissions to your application, you have to add xml-code to the manifest file. The easiest way to do this is by using the Manifest Editor. Double-click on the AndroidManifest.xml file to open the editor.

Add permissions

Select the Permissions tab in the editor.
To add the uses-permission for internet access, click the Add button and in the pop up, then double-click the Uses Permission alternative. You now have a drop down menu at the right side, select from this menu "android.permission.INTERNET".
Save the file.
View the xml-code generated by selecting the AndroidManifest.xml tab.

Register a new Activity

First you have to create a new Activity class in your project. Right-click the project header and select New/Class. Make sure you select the proper package and set your class to extend Activity.

To register a new Activity in your application, select the Application tab. Under Application Nodes, click the Add button and, in the pop-up, double-click Activity. You get a new view where you can browse to, or input, the name of your Activity (add a period at the beginning of the class name).
Save the file.

Debugging and Logcat

There's no console on a smart phone, print-outs with System.out won't produce any result (not even in the Eclipse IDE).

To produce print-outs for debugging purposes you might use the LogCat tool and the class android.util.Log.
In Eclipse, you open a Logcat tab for monitoring print-outs by selecting Window/Show view/Other.../LogCat.
To generate print-outs from your application code, write code like:
Log.i("SomeTag", "in method foo, value of x = " + x); // i - information, v - verbose, ...
A short introduction to Logcat:
http://www.droidnova.com/debugging-in-android-using-eclipse,541.html

A more general description of debugging devices for Android:
http://developer.android.com/guide/developing/debugging/index.html

Export/Import Eclipse projects

Export

Select from the menu: File/Export.../Archive file...
In the wizard, select the desired project. Browse to the destination folder and enter a name for the archive. Click OK.

Import

This instruction assumes the project is exported as an archive file.

Select from the menu: File/Import.../General/*Existing projects into Workspace*...
In the wizard, select the radio button "Select *archive* file", and browse to the desired archive file (zip or tar). Click OK.

More on Android

The best site for information on Android application development is
http://developer.android.com/index.html

A more elaborate Hello World tutorial:
http://developer.android.com/guide/tutorials/hello-world.html
UI tutorials, "Hello View":
http://developer.android.com/resources/tutorials/views/index.html
Notepad tutorial (step 1 – 3):
http://developer.android.com/resources/tutorials/notepad/index.html

Developers Guide, lots of information and examples on most basic components in an Android application:
http://developer.android.com/guide/index.html

The Android API documentation, information on all classes the API:
http://developer.android.com/reference/packages.html

Problem with Code Completion freezing Eclipse

Unfortunately there's a bug in the Eclipse plugin related to Code Completion (Code Completion suggests methods and types when you write code); for certain classes in the API, Eclipse IDE freezes for some time.

One workaround is to turn off Code Completion; Window/Preferences/Java/Editor/Content Assist/Advanced. Because Code Completion is helpful, a better way, until this bug is fixed, is to download the API sources locally, as described below.
http://android.opensourceror.org/2010/01/18/android-source/, scroll down to "What you came here for". We will mostly use version 2.1, API-level 7 (Eclair).

Appendix A; problem when entering text in the AVD

If, when entering text in the AVD, you get Chinese(?) characters instead of Latin characters, long-click on the text component. A context menu appears, select Input Method, and then the proper input method. Or, better, change the settings for language and keyboard on your AVD (it's done the same way as on a real device).

Appendix B; installing an application on a device

To install your application on an actual Android smartphone, you must first create an application package (an APK file). This instruction assumes the device has the Astro File Manager installed; download it from http://www.androidfreeware.net/download-astro-file-manager.html .

Create an APK file

Right-click on the project header and select Android Tools > Export *signed* application package. In the wizard, select Project and create a new keystore. Add the details on the key and create it. Provide a name for your APK file, with the suffix .apk.

Download and install

Connect the device to your computer through a USB cable, then install (mount) it (you can see how to do this on the device). After this you can find the device on your computer as a separate drive (in Windows, open My Computer). Copy the APK-file to this unit.

Uninstall the device and remove the USB cable. Start the Astro File Manager and browse to your application. Select install.
Newer versions of the Android OS require a permission to download applications from other sources than the Android Market. The setting is found on your device via Settings/Applications/Unknown sources[1].

Appendix C; Emulating network connections in the AVD

This works fine in most cases.

A problem arises when connecting to your AVD from another application, e.g. when you want to connect a socket from outside the AVD to a server socket on your AVD.
Let's say an application A, running on your computer hosting the AVD tries to connect a socket to a server socket in an (Android) application B running in the AVD. B is listening to port 6000 on the virtual device (this device actually has its own IP-address, 10.0.2.15, but your application on the host, A, doesn't know of this address). A connects its socket to 127.0.0.1 and some local port, let's say 5000. To make this work we have to redirect localhost:5000 to the AVD's port 6000.
First, determine the console port number for the emulator instance running (shown in the AVD's upper left corner, probably 5554). Next, use telnet to connect to the console of the emulator instance, specifying its console port number, as follows:

```
"telnet localhost 5554"
```

Then run

[1] Be careful not to download applications from entirely unknown, or untrusted, sources.

“`redir add tcp:5000:6000`” (syntax: redir add <protocol>:<host port>:<emulator port>)
Socket connections to host-port 5000 will now be redirected to the emulators port 6000.

More on emulator and networking:
http://developer.android.com/guide/developing/devices/emulator.html#emulatornetworking