G54MDP Lab Exercises
1: Introduction to the SDK

Introduction

The purpose of this exercise is to get you familiar the Android SDK, setting up a virtual handset, creating a project, compiling it with `ant`, and installing the 'app' on your handset. Therefore, this exercise is not assessed. Future exercises will have you build more 'app'-like programs, and will count to the 25% weekly coursework component. As with the lectures, we shall be doing this mainly via the command line — although there is nothing to stop you installing Eclipse and the Android Eclipse plug-in on your home machine.

The Android SDK has already been installed on the School's computers, alongside the Java Development Kit and Apache's ANT which are also used for developing Android apps. If you want to be able to develop your apps on your home computer then you will need to download and install them on your computer. See `http://developer.android.com` for details.

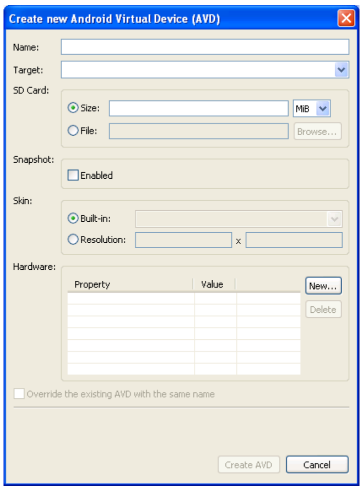
Note: The school machines are currently running an older version of the SDK which I'll endeavour to get upgraded shortly. This means that they do not currently suffer from the bug that means you have to run `ant clean` before rebuilding your project.

1. Creating a virtual Handset

Before we can begin any devlopment, we need a device to test our apps on. If you have an Android phone and are a developing on your own computer then you could use that. However, for the purposes of this course, we shall assume you are using the emulator supplied as part of the SDK. Therefore, the first thing we need to create a virtual device.

From the Start menu, run the **SDK Manager** under **Android SDK Tools**. The first time you run this it will attempt to download some updates from the Google servers, if you are working in the lab then you can safely ignore these and close the dialog box since they have already been installed — if you are working at home you'll want to install them all.

Go to the **Virtual Devices** section of the **SDK Manager** and click the **New** button. A second dialog (pictured) will pop up, asking you for the details of your device. Give it a name you can remember (we'll need it later when we start the emulator), and set the `Target` to be something sensible such as `Android 2.3.3 - API Level 10`. Set the SD Card size to 64MB and leave the skin at the default. Finally, click Create AVD to create the virtual device. This will take a moment, and will pop up an alert to tell you that it is complete.



You can now start the emulator by selecting your new virtual device and clicking **Start**. The emulator should start and boot the phone. This will take some time the first time you run it, but should eventual reach the phone's start screen.

2. Creating your first project

As the examples in lectures have shown, an Android project is constructed from a large number of files. The majority of these are 'boilerplate' and are near-identical from project to project with only the Java and XML files changing to any degree. The android SDK provides a program, `android`, which can (amongst other things) construct the project for us automatically. From this base, we can then add our code to build our app.

To create a project, android is called with the aptly-named argument `create project`. If this is run with no additional arguments then it will print out a help message showing what arguments it requires. Most of the arguments are self-explanatory, the *name* of the project, the *path* to store the files in, the Java package to use and the name of the default Activity. The last is the *Target ID* of the project — in other words, which revision of the API are we aiming this app at. To find out what the different options are, type:

```
android list targets
```

which will list all the available targets, in a form similar to this entry:

```
id: 1 or "android-10"
     Name: Android 2.3.3
     Type: Platform
     API level: 10
     Revision: 2
     Skins: HVGA, QVGA, WQVGA400, WQVGA432, WVGA800 (default), WVGA854
     ABIs : armeabi
```

The exact list depends on the SDK components that have been installed. In our case, we are looking for a target which matches the device we want to develop for, our virtual phone. In this case, we specified that our handset supported API level 2.3, so we want to chose a target that matches — such as the one specified above, id `1`.

Having found our target, we can put it together with the other details to create our project. With your command line pointing at a sensible place, type:

```
android create project --name HelloWorld --path ./HelloWorld \
        --target 1 --package uk.ac.nott.cs.g54mdp.srb.helloworld \
        --activity HelloWorld
```

to create a new project called `HelloWorld` and place the files in a directory also called `HelloWorld`. The package line needs to be unique for each project, so make sure you replace my username with your own. Also, you will need to change the `--target 1` to match the id given in the output from `android list targets`. The program will then create all the necessary initial files for your app.

3. Building the project

To build your app, you need to change into the directory containing the project (if you followed the example above, that directory will be called and`HelloWorld`),

```
ant debug
```

which will build your app in debug mode (this avoids us having to deal with signing the app). If you modify the code for your app, then you can rebuild it by typing the command again.

4. Installing the project on the device

Once you have compiled and built your first app using `ant`, then it is time to install it on the device. This is done using the `adb` command. `adb` performs many functions and so, as with `android`, it takes a sub-command as the first parameter, such as `install` to install an app, or `logcat` to display the device's log file. To install our app, we type the following command:

```
adb install bin/HelloWorld-debug.apk
```

at the command line from the root of project and with the emulator running. If you called the project something else, then alter `HelloWorld` above to match.

Assuming everything works, you should see the command print:

```
826 KB/s (13218 bytes in 0.015s)
        pkg: /data/local/tmp/HelloWorld-debug.apk
Success
```

If it doesn't, then try rerunning the command—experience has shown that it can sometimes hiccup. If you want to reinstall the app after recompiling it then you will need to add the `-r` flag, like so:

```
adb install -r bin/HelloWorld-debug.apk
```

You should now be able to navigate to your app on the (emulated) phone and run it. That's it for this exercise, although if you may want to try some of the samples available here: `http://developer.android.com/resources/browser.html?tag=tutorial`.