Project 1: Setup, Hello Android

Knowledge Goals
Learn how to set up a development environment for Android.
Learn how to create, modify and deploy an Android application.

Estimated Time
3 hours.

Prerequisites
A unix or Windows workstation with Java, Ant, a console application and text editor installed.

Resources
APT Appendix A and chapter 1.
Android Developer Hello World Tutorial: http://developer.android.com/resources/tutorials/hello-world.html
BCG chapter 7.

Deliverables
Push the HelloAndroid project and a README to github.
Submit your github url via the form referenced below.

Assessment
Accomplishing the deliverables demonstrates the student's ability to follow documentation instructions, set up a development environment and the ability to create, add, and commit files and changes to a git repository; and to push the repository to a remote on github. The instructor will specifically evaluate repository organization, commit message quality, implementation completion and style.

Instructions
Preliminaries
You are 100% responsible for your development environment. I am not able to provide you personal technical support, and, for this class, I do not care if you have machine issues: that is entirely your business. Our labs are equipped with functioning development environments. I recommend you use the lab machines if running the emulator on your personal workstation is too slow.

You should have the JDK, Ant, a good text editor, and a good console application at your disposal on your workstation before proceeding. Paths to these tools should all be declared in your PATH environment variable. Until this is done, do not proceed with this project.

Read each step entirely before taking action for that step.

Install and Configure git
Visit http://github.com and obtain an account. Next, follow the instructions here: http://help.github.com/set-up-git-redirect . These instructions guide you through installing git, configuring git, and adding your public key to your github profile. You know that everything is correct when you can open a terminal, type `ssh -T git@github.com` and see `Hi` *`username`*`!`.

Install the Android SDK Starter Package
Visit http://developer.android.com/sdk/index.html and download and install the appropriate package for

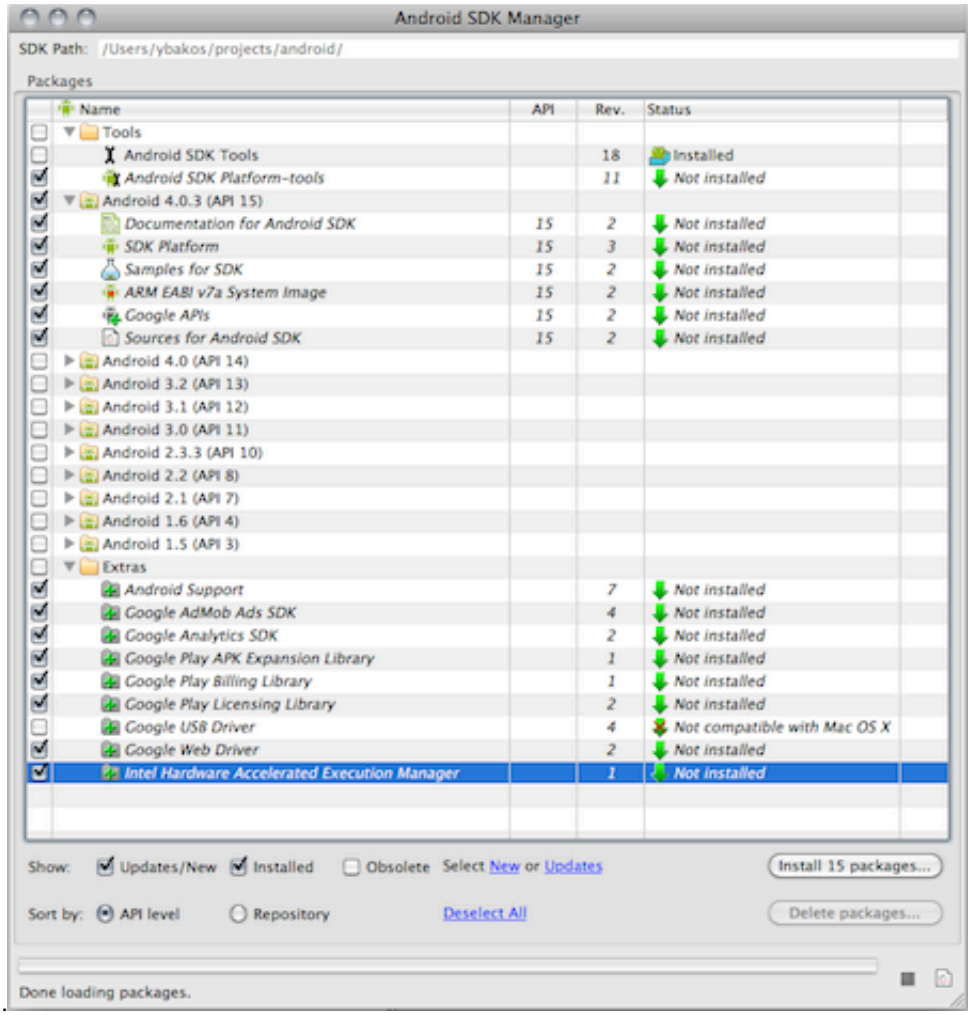
your OS. If you are on Windows, use the installer. If you are on posix, place the sdk somewhere sane (eg `~/projects/android`). If you already have the Android SDK installed, use the Android SDK and AVD Manager to update your system rather than downloading a new package.

Download and Install Eclipse
Visit http://www.eclipse.org/downloads/ and install the *Eclipse IDE for Java Developers*.

Install the Eclipse ADT Plugin
Follow the instructions in the tutorial or here: http://developer.android.com/sdk/eclipse-adt.html#installing . Just check the box next to "Development Tools" and proceed. You are downloading and installing some extensions for Eclipse that facilitate the management of Android projects. Eclipse may prompt you to restart. That is fine, restart Eclipse.

Configure the ADT Plugin
Follow the instructions in the APT Appendix A or here: http://developer.android.com/sdk/eclipse-adt.html#configuring . You are telling the ADT plugin where the SDK is. Do not tell Eclipse to install an SDK, you have already done this. Specify the path to the android SDK on your machine. Do not worry about the warnings about not having a Platform Tool installed. That is next.

Install the Appropriate Android Platform SDK
Start the Android SDK Manager (Eclipse might do this for you after you complete the previous step). Wait for the tools list to be updated before proceeding. Check the *Android SDK Platform-tools*, *Android 4.0.x*, and all the *Extras*, if you want.



Once you click install, go for a run or go eat something and come back in an hour.

Set Up the Emulator
Start the AVD Manager and continue following the instructions in APT Appendix A. Name your AVD *csci498*, specify a target of *Android 4.0.x*, **enable the snapshot feature**, and don't change the other options.
The AVD you just created should now appear in the AVD Manager list. Select it, and click the "Start..." button. Be sure "Launch from Snapshot" and "Save to Snapshot" are checked, then click the *Launch* button. Go do forty pushups or have a pint of root beer and then come back. Verify that your emulator runs as expected -- you should at least be able to see the home screen of the device. Now close the emulator window and the AVD Manager window.

Install the Eclipse eGit Plugin
Use Eclipse's plugin manager to do this (*Help -> Install New Software...*). See this video http://www.youtube.com/watch?v=I7fbCE5nWPU as a guide. Restart Eclipse when prompted, and then move on to creating your first Android project.

Set Up Your Device (Optional)
See the APT Appendix A section titled *Set Up the Device.*

Create & Run Your First Android Project
Follow APT Chapter 1 for creating the "Now" project. But, call it *HelloAndroid*. Use a sane package namespace (eg, `csci498.hello`).
Start the emulator, and then run your project. You should witness that your app was deployed to the emulator and is running. If not, seek assistance before continuing.

Place Your Project Under Version Control
Right-click your project, select *Team -> Share Project*. Select *Git*. Click your HelloAndroid project in the list, and then click *Create*. For the name, type `csci498`. Then click the *Finish* button.
Eclipse will create a directory called *csci498* (which we will call your "repo root") and will have moved your *HelloAndroid* project into your repo root.

Open a terminal window, navigate to your repo root and create a file called `.gitignore` whose contents should look like this: https://github.com/github/gitignore/blob/master/Android.gitignore . If you do not correctly create this file, you will receive a 0 for this project. Add and commit the `.gitignore` file:

```
git add .gitignore
git commit -m "Adding ignore."
```

You've just informed git about files and directories in your repository that it should ignore.

Next, using the console and/or your editor of choice, create a file called `README.md` in your repo root, whose contents look like this:

```
# CSCI498A Mobile Device Programming w/ Android
## Fall 2012
### John Denver

This repository contains my project work for CSCI498 at Mines.
```

Add and commit this `README.md` file to your repo.

```
git add README.md
git commit -m "Adding readme."
```

Next, add and commit the `HelloAndroid` project directory to your repo.

```
git add HelloAndroid
git commit -m "Adding HelloAndroid project files."
```

Create a Remote on Github and Push Your Work
Visit github.com and create a new repository. Call it `csci498android`. On the next screen pay very special attention to the section titled *Existing GIt Repo?.* From the console, assuming you are already in your repo root, execute the commands that github tells you to. *Be precise.* When github tells you to `git push -u origin master`, *then type that exact friggin thing and nothing else*. If you ignore the `-u`, then I will ignore you when you ask me why you can't automatically merge to master when pulling. You will think I am mean and you will cry.

```
git remote add origin git@github.com:YOU/csci498android.git
git push -u origin master
```

Make a Change, Run It, Commit
Now follow the "Building the User Interface" steps as described in google's tutorial here (stop at "Adding a

Button": http://developer.android.com/training/basics/firstapp/building-ui.html

However, change the displayed string to use your name, (eg, `"Hello, Ernie!"`). Run your project and confirm that your app runs correctly on the emulator. (Do you see your name?)

Now, add and commit this change. But this time, use Eclipse. Right click your project and select *Team -> Commit*. Make sure all relevant files are checked in the window that appears (eg, `BlankActivity.java`). Enter a meaningful commit message and click the *Commit* button.

Read the section titled "R Class" in the sidebar of that google tutorial, and take a look at this archived doc: http://web.archive.org/web/20101206060405/http://developer.android.com/resources/tutorials/hello-world.html

Next, introduce a bug into your project by following "Debug Your Project" here: http://web.archive.org/web/20101206060405/http://developer.android.com/resources/tutorials/hello-world.html.
Witness the error dialog that the emulator displays. Complete the debugging portion of the tutorial (setting a breakpointer, re-running and witnessing the Debug Perspective). To stop debugging, click the root node of the application in the debug perspective and then click the red stop button. When you are done, click the "Java" perspective button in the upper-right corner of the Eclipse window.

Now, quickly revert your buggy change and restore `BlankActivity.java` to its last committed state. Right-click `BlankActivity.java` and select *Replace With -> File in Git Index*. Witness that the code displayed is now identical to the previous commit.

Let's Practice This Again
Jump to BCG chapter 7, "Rewriting Your First Project." Follow the instructions, taking the extra care to make the names of things fit your HelloAndroid application. Witness your application's new behavior in the emulator. When you are finished, commit your work and *be sure to write a meaningful commit message*.

Push Your Work and Send Me Your Repository URL
You're almost done. Your local repository has changes that you need to publish to the remote on github. To do so, simply push.

```
git push
```

Visit the Web page for your remote repository on github to verify that your work has been pushed.
If it looks like this: https://github.com/ybakos/csci498android/tree/v0.0.2 then you're in great shape.

Fill out this form: https://docs.google.com/spreadsheet/viewform?formkey=dGNDYXlzc2VQTTI2VWo5VElWeUEtMWc6MQ .
Be absolutely certain that you are entering the *repository url* and not the Web address of your repository's main page.

I Want to Work on Two Workstations, How Do I Do This?

First, make sure that your 2nd workstation's public key has been added to your github profile. Do not proceed unless you can `ssh -T git@github.com` and see a successful "hello!" message.
Visit github and copy your *repo url*.
In eclipse, select *File -> Import -> Git -> Projects from Git -> URI*.
Paste your repo url in the *Location->URI:* field and click *Next*.
Enter your ssh key passphrase if prompted, and select the *master* branch.
Now specify where you'd like your repository to live. It's really up to you.
Select *Import Existing Projects* and click *Next*.
Be sure the HelloAndroid project is checked and click *Next*.