ECE155: Engineering Design with Embedded Systems Winter 2013

Lab 0 (Setting up your Development Environment) — Week 1

Prepared by Kirill Morozov *version 1.2*

1 Objectives

In this lab, you'll familiarize yourself with Eclipse and the Android APIs. In particular, you will:

- Meet up with your lab partners.
- Install software: Eclipse, Subclipse Subversion Client, the Android Eclipse Plug-in, and the Android SDK.
- Run a "Hello World" application on your Android phone.
- Check your code into source control.

2 Lab Groups

Most lab groups this term will have 2 members. One group may have 3 members. We have assigned you a lab partner randomly, and a TA will read out the lab group memberships.

We will provide Nexus One Android phones for the lab sessions. To work on this lab outside of lab hours, you may either check out the Android phones on reserve in the DC library or use your own Android phone. If you use your own Android phone, you are responsible for the consequences. Please return any phones that you may have checked out at the end of each lab session.

Word to the Wise. Chat with your lab partner; you may want to exchange e-mail addresses, phone numbers, or IM info to contact each other to prepare for subsequent labs. Here are a few questions for discussion:

- Have you ever worked with the Android API?
- Do you consider yourself a strong programmer?
- Do you prefer to lead or do you prefer to follow someone else's lead?
- Do you consider yourself to be a procrastinator?
- Do you consider yourself to be a perfectionist?
- Do you have an Android phone?
- Are you usually on-time for labs?

You'll need to assess your skill levels to find a mutually agreeable choice of task assignments. Project coordination is hard; for lab 1, coordinating will probably be harder than the lab itself.

3 Installing Software

There are four pieces of software you need before you can run your code on the Android phones: the Java Development Toolkit, the Android SDK, the Eclipse IDE, and the Android Eclipse Plug-in. You also want to install Subclipse, which lets you commit code to your Subversion repository for grading. Since some of the installation processes are a little involved, we will go through them here[1].

Java Development Toolkit The Java Development Toolkit (JDK) provides tools for compiling Java code. You can download a JDK from `http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html`. On that page, in the grey box labelled "Java SE Development Kit 7u10" select the radio button **Accept License Agreement**. Then select the link that corresponds to the operating system you have. Note: "Windows x86" means 32 bit Windows while "Windows x64" means 64 bit Windows.

1. Run the executable you have just downloaded.
2. In the installer, continue clicking **Next ¿** until it starts installing.
3. It is possible that the installer will also decide to install the Java Runtime Environment (JRE), which runs Java programs. This will launch a separate installer. Click **Next ¿** in this installer until it installs the JRE.

Android SDK. You will need to install the Android Software Development Kit (SDK)[2]. You can download the SDK from `http://developer.android.com/sdk/index.html`

If you download the ADT bundle, you should get both Eclipse and the SDK. In that case, you don't need to get the Eclipse IDE or the Android SDK, but you still need Subclipse.

Part of the SDK installation will involve choosing a platform version. The Nexus One phones run Android 2.3.6 (API 10). You can download as many API versions as you want, but they will take a while to download.

Eclipse IDE. You can download the latest version of Eclipse from here:

`http://eclipse.org/downloads/packages/eclipse-ide-java-developers/junosr1`

A word on terminology To make it easier to talk about various parts of the Eclipse UI, we will define some terms:

- **View** In Eclipse, a view is an informational sub-window which displays data in some way. For example, the panel containing a tree of your projects and files is a view.

[1]For those of you using Ubuntu, you might try `http://www.omgubuntu.co.uk/2012/07/android-sdk-installer-for-linux-debianubuntu`. No guarantees.

[2]If you're running a 64-bit Linux system, you may encounter infelicities. Patrick Lam may be able to help you resolve them.

- **Editor** An editor is the area of the screen where you edit code or other resource files.
- **Perspective** A perspective is an arrangement of views, editors, and menus. Different perspectives will have different options in different places in different drop-down menus. You can change perspectives by selecting the tabs at the top right of the window.

Subclipse. We'll be handling submissions through the departmental SVN repository, so you'll want to install the Subclipse plugin for Eclipse[3]

1. In Eclipse, from the menu-bar, go to: **Help > Install New Software...**
2. Click the **Add...** button
3. In the *Add Repository* dialog, enter the name "Subclipse" and the location

   `http://subclipse.tigris.org/update_1.8.x`
4. Select the Subclipse repository you just added from the drop-down list.
5. Click the check-boxes beside "Subclipse" and "SVNKit" in the tree view.
6. Click **Next** on this screen and the next one.
7. Read and accept the license agreements. Then click **Finish**.

Android Eclipse Plug-in. Finally, you need to hook up Eclipse with the Android SDK. Google provides a plug-in that makes writing Android applications in Eclipse very easy.

1. In Eclipse, from the menu-bar, go to: **Help > Install New Software...**
2. Click the **Add...** button
3. In the *Add Repository* dialog, enter the name "Android" and location

   `https://dl-ssl.google.com/android/eclipse/`
4. Select the Android repository you just added from the drop-down list.
5. Click the check-box beside "Developer Tools" in the tree view.
6. Click **Next** on this screen and the next one.
7. Read and accept the license agreements. Then click **Finish**.

[3]For Debian GNU/Linux, I also had to install `libsvnclientadapter-java` to get Subclipse to work.

4 Creating and Installing Your First App: Hello, World.

Now that you have installed the developer tools, you can create and install your first "Hello World" application. First, creation.

1. Select from the menu-bar **File > New > Other...**
2. Select "Android Application Project"
3. Name your Application **Lab0_SSS_XX** where SSS is your lab section (e.g. 201) and XX is your team number (e.g. 02). Unique naming helps us manage the project submissions. There may also be problems if you attempt to load two applications with the same package name on the same device.
4. Change the Package name to "**ca.uwaterloo.Lab0_SSS_XX**".
5. You will have to select Theme: None for the version of Android on the Nexus One phones.
6. Click **Next**.
7. Create an Icon for your project. You can use one of the provided clip arts or a custom picture. Your icon should be unique so that it is easy to recognize your application when it is loaded on the same device as other groups' applications.
8. Keep clicking **Next** until the wizard is done.

About emulators. The Android developer tools also allow you to run your application on an emulator. Emulators are very useful in engineering design. However, the labs in this course require sensor input, and we haven't recorded a set of sensor inputs for you. It is allegedly possible to do so using the following project:

```
code.google.com/p/openintents/wiki/SensorSimulator
```

To set up an emulator, follow these steps:

1. Open the Android Virtual Device Manager: **Window > AVD Manager**.
2. Select **New**.
3. Give your virtual device a unique name.
4. Select an API level (10 is good) from the Target drop-down.
5. Click **Create AVD**.

Now, whenever you run your application, if there is no physical device connected, Eclipse will launch the emulator you created. If you have multiple AVDs, you can choose which one is used by going to **Run > Run Configurations...**, switching to the **Target** tab and selecting the **Always Prompt to pick device** option.

Debug mode: Machine setup for running the app. When you normally plug your phone into your computer, your computer sees the phone as a USB drive. We need more capabilities. To enable loading applications onto phones and debugging them, you'll need to install debugging drivers. Each Android Phone has its own drivers. Your lab machines should already be set up to work with the Nexus One Phones we will provide. However, you'll need to enable debug mode on Android 4.2 and up; see these directions:

`http://dottech.org/87439/how-to-unlock-usb-debugging-mode-on-android-4-2-jelly-bean-and-higher-guide/`

A detailed guide to running your application on the Android hardware is at `http://developer.android.com/tools/device.html`. We've reproduced the core steps below:

1. On the Android device, go to **Settings > Applications > Development** and enable **USB debugging**.

2. Next, you will need to set up your system to recognize the phone. This step depends on your operating system.

   - Mac OS X
     (a) Do nothing, it should just work.
   - Ubuntu Linux: to set the proper permissions, you need to add a udev rules file that contains a USB configuration for each type of device you want to use for development. In the rules file, each device manufacturer is identified by a unique vendor ID, as specified by the ATTR{idVendor} property. Use the command-line tool `lsusb` to find the ID of your phone (after you've plugged it in.)
     (a) Log in as root and create the file `/etc/udev/rules.d/51-android.rules`

       Use this format to add each vendor to the file:
       ```
       SUBSYSTEM=="usb", ATTR{idVendor}=="0bb4", MODE="0666", GROUP="plugdev"
       ```

       In this example, the vendor ID is for HTC (which is the ID for the Nexus One phones). The MODE specifies read/write permissions, and GROUP defines which Unix group owns the device node.
     (b) Now execute: `chmod a+r /etc/udev/rules.d/51-android.rules`
   - Windows 7
     (a) Connect the Android device to your computer via USB.
     (b) Right-click on Computer from your desktop or Windows Explorer. Select Manage.
     (c) Select Devices in the left pane.
     (d) Locate and expand "Other device" in the right pane.
     (e) Right-click the device name (e.g. "Nexus One") and select "Update Driver Software". This will launch the Hardware Update Wizard.
     (f) Select "Browse my computer for driver software" and click Next.
     (g) Click Browse and locate the USB driver folder. (The Google USB Driver is located in `<sdk>\extras\google\usb_driver\`.)
     (h) Click Next to install the driver.

- Windows XP
  (a) Connect your Android-powered device to your computer's USB port. Windows will detect the device and launch the Hardware Update Wizard.
  (b) Select "Install from a list or specific location" and click Next.
  (c) Select "Search for the best driver in these locations"; un-check "Search removable media"; and check "Include this location in the search".
  (d) Click Browse and locate the USB driver folder. (The Google USB Driver is located in `<sdk>\extras\google\usb_driver\`.)
  (e) Click Next to install the driver.
- Windows Vista
  (a) Connect your Android-powered device to your computer's USB port. Windows will detect the device and launch the Found New Hardware wizard.
  (b) Select "Locate and install driver software".
  (c) Select "Don't search online".
  (d) Select "I don't have the disk. Show me other options".
  (e) Select "Browse my computer for driver software".
  (f) Click Browse and locate the USB driver folder. (The Google USB Driver is located in `<sdk>\extras\google\usb_driver\`.) As long as you specified the exact location of the installation package, you may leave "Include subfolders" checked or unchecked—it doesn't matter.
  (g) Click Next. Vista may prompt you to confirm the privilege elevation required for driver installation. Confirm it.
  (h) When Vista asks if you'd like to install the Google ADB Interface device, click Install to install the driver.

After you successfully install your phone. The first time you run your application, you will first need to select the .java file located in the `src` folder. To start the application, select **Run > Run**. When you do this, a dialog will pop up. Select **Android Application** and click **Next**. Hello, World!

5 Checking your code into source control

During this course, you will be submitting your labs and accessing sample code through a version control system called Subversion, abbreviated "SVN" after the command-line version of the tool. The Subclipse plug-in you installed will let you interact with the course repository in Eclipse.

How Subversion works Subversion uses a server-client model. The server is the ECE machine that stores a copy of your code. Your computer is the client; it stores your working copy of the code. A typical workflow looks like this:

1. If you have no version checked out, check out the latest version of the code from the server. If you do have a version checked out, update it. Other people may have worked on it since you last did.

2. Make changes to the copy of the code on your computer.

3. Add files you have made changes to *and new files* to the active change-list.

4. Commit the active change-list to the repository.

5. If someone else has edited the files you have changed between when you have checked them out and when you tried to commit them, you will need to resolve these conflicts. Then try to commit again.

6. Enter a meaningful message to go with your commit that explains what changes you have made.

In general, it is best to commit your work in logical chunks. A single commit should be focused around one feature or bug. When others look at your commits, they can understand which changes do what. Logically grouping commits is also useful if you want to roll-back a particular feature.

Connecting to the server First, you will need to connect to the repository you will be using for this course. To do this, follow these steps:

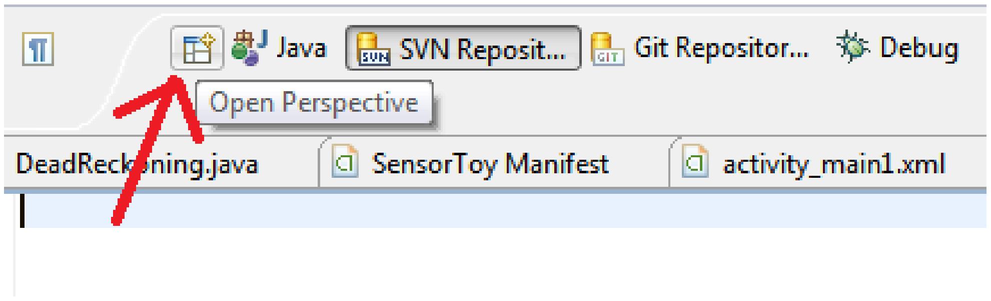


Figure 1: All of your active perspectives.

1. Click on the **Open Perspective** button at the top of your Eclipse window. (See Figure 1.)

2. Select **Other...** from the dropdown.

3. Select **SVN Repository Exploring** from the dialogue.

4. Switch to the new perspective tab that you just created.

5. Find the view labelled **SVN Repositories**. Right-click there and select **New > Repository Location...**

6. Enter the URL:

   `https://ecesvn.uwaterloo.ca/courses/ece155/w13`

7. A dialog will pop up asking for your username and password. Use your Quest login. You should also check the **Save Password** box, otherwise Subclipse will ask you for your password whenever you navigate to a different folder. A word of warning if you do this: **On some systems, the password is stored as plain text. Don't use this option on shared computers!**

   On Linux-based systems, passwords are stored in `~/.subversion/auth/svn.simple`. On Windows, they are stored in `c:\Documents and Settings\your_username\Application Data\subversion\auth\svn.simple`.

8. You will be able to check your code into the folder groups/group-SSS-NN where SSS is your lab section number and NN is your group number. We will only mark your submission after you have committed your code to that folder.

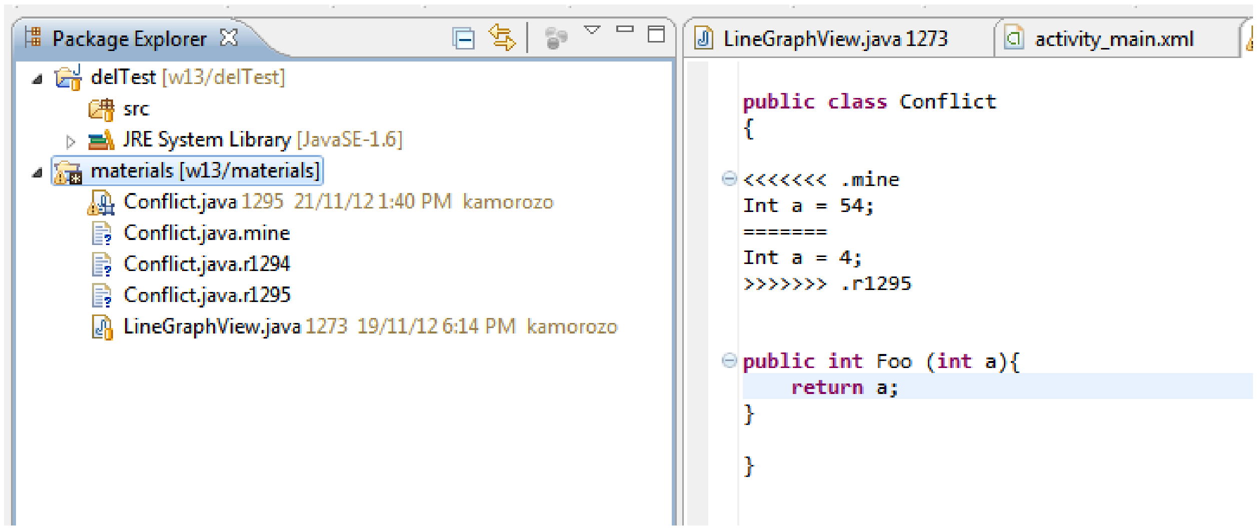
Getting files from the repository In this course, we will provide you with sample code for some labs. You can find this good stuff in the repository's `materials/` folder. You can view files in the repository by double-clicking them as normal. You can also download a local copy by right-clicking on a folder and selecting **Checkout...**. In the dialog, you can just click finish, and it will create a new project in your workspace that will contain all the files in the folder you checked out.

In the future, when you want to update your code to match the repository, you can do this by right-clicking on the top-level folder in your project and selecting **Team > Update to HEAD**.

Adding your project to the repository Once you have created a project, and you want to add it to the repository, follow these steps:

1. Right-click on the project and select **Team > Share Project...**.
2. From the Repository Type dialog, select **SVN** click **Next**.
3. Make sure the repository you just added is selected and click **Next**.
4. Select **Use specified folder name** and enter "groups/group-SSS-NN/lab0-SSS-NN". Then click **Finish**.
5. This will add all of the files in your project to the current change-list and ask you to open the synchronization perspective.
6. You can say **No** on the synchronization perspective because we don't need it now.
7. At this point, your don't need to do anything more. Just right-click on the top-level folder in your project and select **Team > Commit**.
8. In the future, when you make changes to your code, you can commit it to the server by right-clicking on the top-level folder and in the Java perspective selecting **Team > Commit**.
9. Remember: if you add a new file, you will need to explicitly put it under version control by right clicking it and selecting **Team > Add to Version Control**.

Conflicts While you are working on your code with your partner, sometimes you will modify the same file. Most of the time, Subversion is smart enough to detect that two changes are not on the same lines and will automatically merge these changes for you. However, sometimes you and your partner will edit the same lines. In this case, you have a conflict. When that happens, Subversion will not let you commit your changes. Here is how you fix it:


Package Explorer
delTest [w13/delTest]
src
JRE System Library [JavaSE-1.6]
materials [w13/materials]
Conflict.java 1295 21/11/12 1:40 PM kamorozo
Conflict.java.mine
Conflict.java.r1294
Conflict.java.r1295
LineGraphView.java 1273 19/11/12 6:14 PM kamorozo
LineGraphView.java 1273
activity_main.xml

```
public class Conflict
{

<<<<<<< .mine
Int a = 54;
=======
Int a = 4;
>>>>>>> .r1295


public int Foo (int a){
    return a;
}

}
```

Figure 2: Conflict Resolution.

1. You need to get the changes that are causing the conflict. You do this by right-clicking on the project and selecting **Team > Update to HEAD**.

2. You will see a little square icon beside any files that have conflicts. Look at Conflict.java in figure 2.

3. If you open the file, you will see the conflicting changes side by side in the code.

4. You should choose which version of each change to keep, and edit each conflicting file accordingly.

5. Once you are done, for each file with a conflict, you will need to right-click on it and select **Team > Mark Resolved**.

6. This will bring up a dialog that will let you choose if you want to keep one of the existing versions of the file or your manual resolutions.

7. Once you have resolved all the conflicts, try to commit again.