

Homework 0: Getting Started!

This homework will serve as a warm-up for ANTLR/Java development. In it, you will practice several skills:

- Setting up Eclipse IDE for Java Developers
- Setting up ANTLRWorks2
- Accessing a Subversion repository
- Generating parser code with ANTLR
- Running an app that uses the parser

The homework will introduce you to several tools that you will be using throughout the semester: Eclipse, ANTLR, and Subversion (SVN). These are professional-quality tools commonly used in practice.

Step 0. Send me your username + encrypted password

Although not needed for this homework, you will need a username and password to commit to the Subversion repository in future assignments.

To set up a username/password, do the following:

1. Go to <http://www.cs.memphis.edu/~sdf/util/htpasswd/>, and fill out the form. Use your UofM username (i.e., the name before the @ in your university email address). Once you've filled in your username and password, press the button.
2. Copy all of the **encrypted password** string, and paste it into an email to me.

Note that this username/password is independent from all your other university accounts. Don't forget this password! If you need to change it, simply repeat the above steps.

Step ½. Get Java (if you don't already have it)

You must have Java JRE version 7 (aka 1.7) or later installed on your system. If you don't already have it, get it here:

<http://www.java.com/>

Simply run the executable to install.

Step 1. Get Eclipse IDE for Java Developers

First, create a folder somewhere in your user directory (e.g., on your Desktop) called **java-dev**. (Actually, you can name it whatever you want.)

IMPORTANT! It's important that you pick a good spot for this folder. Moving it later will break things.

Download the latest version (*Kepler*) of *Eclipse IDE for Java Developers* and save it to your **java-dev** folder. Eclipse is the most widely used Integrated Development Environment (IDE) for Java. It includes facilities for editing, running, and debugging Java apps. Get it here:


<http://www.eclipse.org/downloads/>

IMPORTANT! You need the version called Eclipse IDE for Java Developers.

Once you've downloaded Eclipse, unpack the zip/tar.gz into your **java-dev** folder. Thus, you should have an **eclipse** folder inside your **java-dev** folder. The **eclipse** folder contains the Eclipse executable among other things.

Next, create a folder **workspace** inside your **java-dev** folder. Eclipse will use this folder to store your project files.

Now, it's time to fire Eclipse up. To do so, run the executable **eclipse** within the **eclipse** folder (i.e., with the same name). When Eclipse prompts you to select a workspace, choose the **workspace** folder you created earlier.

Once Eclipse has launched, click the **Workbench** button (that looks like this: ) to open the **Java** perspective.

Now set Eclipse aside for a bit, while we move on to the next step...

Step 2. Get ANTLRWorks 2

Download and unpack ANTLRWorks 2 on your machine. You will use it to create ANTLR4 grammars and to generate parser code for your grammars. Get it here:

<http://tunnelvisionlabs.com/products/demo/antlrworks>

Download ANTLRWorks and unpack the zip into your **java-dev** folder. Thus, your **java-dev** folder should now contain three folders: **eclipse**, **workspace**, and **antlrworks2**.


To run ANTLRWorks, run one of the executables in the **antlrworks2/bin/** folder. The **.exe** files should work for Windows users, whereas the **antlrworks2** file (i.e., with no suffix) should work for Linux/Mac users.

Step 3. Install the Eclipse Subversion plugin

To download the example ANTLR/Java app, you will need a Subversion plugin for Eclipse. Subversion is a tool that enables you to store your project files in a central repository and to keep track of the different revisions of your files. Later, you will use Subversion to turn in homework/project assignments and to collaborate on code with your team.

Follow these steps to install Eclipse's Subversive SVN Team Provider plugin. In Eclipse:

1. Click **Help** → **Install New Software...** This should pop up an **Install** window.

2. In the **Install** window's **Work with:** dropdown, select **--All Available Sites--**.
3. In the filter field (which initially contains the words "type filter text"), enter *Subversive*. This should cause the **Name/Version** columns to update. Note: Eclipse may be *very slow* in updating the table. Be patient and wait for it!
4. Check the box next to **Subversive SVN Team Provider**, and click the **Next** button. Click through the installation wizard by clicking **Next/Finish/etc.**, and restart Eclipse when prompted.
5. When Eclipse starts, click the **Workbench** button () if necessary to get back to the **Java** perspective.
6. Click **Window** → **Open Perspective** → **Other...** This should pop up a list of perspectives.
7. Select **SVN Repository Exploring** from the list to open the perspective. Doing this should pop up an **Install Connectors** window.
8. From the **Install Connectors** window, select the version of **SVN Kit** (*do not use JavaHL*) that is compatible with SVN 1.6.x, and click **Finish**. This should cause an installation wizard to pop up.
9. Click through the installation wizard by clicking **Next/Finish/etc.**, and restart Eclipse when prompted. When Eclipse restarts, you should be back at the **SVN Repository Exploring** perspective. The left column of the main Eclipse window should have an **SVN Repositories** view instead of the usual **Package Explorer** view.


Note: If you mess up the **Install Connectors** bit and can't get that window to pop up again, try removing your workspace folder (I assume you have no valuable work in there at this point), and then restarting.

Step 4. Checkout the example ANTLR/Java app

To download the example ANTLR/Java app, you'll need to use Subversion to "checkout" a copy of the example source code.

Go to the **SVN Repository Exploring** perspective (see Step 4), and perform the following steps to do the checkout:

1. At the top of the **SVN Repositories** view is a **New Repository Location** button—click it. This should open a **New Repository Location** window with fields for you to fill in.
2. Fill in the fields as follows, and click **Finish**.
 - **URL:** <https://utopia.cs.memphis.edu/course/comp4040-2013fall/examples/>
 - **User:** anonymous
 - Leave the **Password:** field blank.
This should add the repository to the **SVN Repositories** view. You may get a message about "master password" and/or "password recovery". This is your OS trying to help you keep track of your passwords. Whatever option you choose is probably fine.
3. Expand the contents of the repository by clicking the triangle. You should see a **hello-antlr** folder.
4. Expand the **hello-antlr** folder, and you should see three folders: **trunk**, **branches**, and **tags**.
5. Right click on **trunk**, and select **Check Out**. This should cause a copy of the **hello-antlr** project to download.

6. Switch to the **Java** perspective (by clicking the button that looks like  near the upper right of the Eclipse window). If you see the **hello-antlr** project in the **Package Explorer**, the checkout was successful. The project will be broken initially, indicated by a small red **x** on the **hello-antlr** project in the **Package Explorer**.

Step 5. Get the build working

This is a Maven project (i.e., the build is managed by Maven), and you must first tell Maven to update the project:

1. Right-click on **hello-antlr** in the **Package Explorer**, and select **Maven → Update Project...** An **Update Maven Project** window should pop up. Click **OK**.

For the Java project to build properly, you must generate the parser code using ANTLRWorks:

2. In ANTLRWorks, open this file from your **hello-antlr** workspace:
src/main/java/edu/memphis/hello_antlr/parser/Hello.g4
3. Click **Run → Generate Recognizer...** A **Generate Recognizer** wizard should pop up.
4. Click through the wizard, making sure to check **Generate listener** and **Generate visitor** when the opportunity presents itself. This should generate several Java files in the **edu.memphis.hello_antlr.parser** package (along with a couple of **.token** files).
5. In Eclipse, right-click on **hello-antlr** in the **Package Explorer**, and click **Refresh**. The project should build, and the red **x** should disappear.

Step 6. Run the app

To run the app, perform the following steps in the Eclipse **Java** perspective.

1. Right-click on the **hello-antlr** project in the **Package Explorer**, and then click **Run As → Java Application**. This should bring up the **Select Java Application** window.
2. Select the **App** application, and click **OK**. The app should run in Eclipse's **Console** view, and display a **HELLO>** prompt.
3. Under the prompt, type **hello foo** and hit enter. Then enter an EOF character (typically by pressing **Ctrl-D** on Unix/Linux/OSX systems or **Ctrl-Z** on Windows systems). The app should output "(r hello foo)". Congratulations! You did it!

Step 7. One last thing to fix

For Maven to work properly in the future, you must rebuild its repository index:

1. Click **Window → Show View → Other...** This should pop up a **Show View** window.
2. Expand the **Maven** folder, and select **Maven Repositories**. Click **OK**. A **Maven Repositories** view should appear near the bottom of the Eclipse window.
3. In the **Maven Repositories** view, expand **Global Repositories** item to reveal the **central** item.
4. Right-click on **central**, and select **Rebuild Index**. This should pop up a confirmation dialog; click **OK**. This should cause Eclipse to rebuild its Maven index. Note: Eclipse

may be *very slow* at rebuilding the index. It may take on the order of minutes. Furthermore, the progress (bottom right corner of Eclipse window) may sit at 0% for a very long time. Once it completes, you should be able to expand **central** and see many items, such as **abbot**, **acegisecurity**, etc.

Submitting

To receive credit for this homework, you must

- demonstrate that you can run the web app at the start of class on the day the homework is due, and
- email me your encrypted password before class on the day the homework is due.