Computer Science 260

Lab: Introduction to Linux and Eclipse

**Important: Linux is case sensitive. "cs" != "CS"**

1. Login to Windows.
2. If you have used X2Go to log in to the computer science department's Linux server "ice" before (for example, if you are currently enrolled in CS 250), start X2Go and jump to step 11. Otherwise, continue with the next step.

1. We are going to use Windows to connect to another computer system: the computer science department's Linux server named "ice". The software we will be using to connect is called X2Go. Before we can do that, we need to create an X2Go session. A **session** is a collection of settings that allow you to connect from one computer to another computer in a specific way.
2. Click on the X2Go shortcut on the desktop. If you can't find the shortcut on the Windows desktop, click on the Start menu, find, and then start the X2Go client.
3. Once the X2Go client starts, perform the following steps to create a new session that will allow you to quickly connect to ice. You only have to create a session once for each computer that you want to connect to. After that, you can use the previously created session anytime you wish to connect.
	1. From the X2Go menu bar, select Session > New Session
	2. Under the Session tab, set the following text fields. Case is important.
		1. Session name: Ice
		2. Host: ice.truman.edu
		3. Login: <your TSU login name>
		4. Session type: XFCE
	3. Click the OK button.

The new session you just created appears on the left of the X2Go window.

1. Click on the ice session in the X2Go window to start the Linux login process.
2. Your username should already be filled. Type your password in the textbox and click OK. If you get a dialog box telling you that the server is unknown, and asking if you want to trust it, click "Yes". In a few moments, the Linux desktop should appear in a new window. Be patient, it can take a few seconds.
3. Notice that you now have two X2Go windows open – one is the X2Go session manager that shows all of the sessions that you have created, and the other is the Linux desktop. Minimize the X2Go session manager window – we won't need it for a while. The other window contains your Linux desktop. Make the window that contains the Linux desktop larger by dragging the lower right corner of the window to enlarge it.
4. The Linux system may ask you some initial configuration questions. It will only ask these questions once. If you are asked to choose a password for a "new keyring", click "cancel". If you are asked to configure panels, choose the default.
5. When you are finished with your Linux session, **do not click the close box on the X2Go window. It is very important that you properly end your session either by logging off of the Linux system *or* suspending the X2Go session when you are done**.
	1. The first way to end your X2Go session is by suspending it. To do this, maximize the X2Go Session manager, or make it the front most window. Click on the pause button  in the session manager. After a few seconds, the Linux window should disappear and the session manager login screen should appear. You have now properly suspended your X2Go session. Quit X2Go from the File menu.

The next time you log in, your X2Go session will appear exactly as you left it the last time. Log in again now.

* 1. The second way to end your session is by logging out of the Linux system. Go the Applications menu in Linux, and choose Log Out. When the "Log Out" menu appears, click the appropriate button. After a few seconds, the session manager should appear again. Quit X2Go from the File menu. You have now properly closed your X2Go session.
	2. If you do not log off of the Linux system or suspend X2Go when you are done, you will lose any unsaved work, and you can corrupt your Linux settings, which may make it impossible for you to login again without having the systems administrator reset your account. Simply closing the X2Go window is equivalent to turning off the power off on your desktop system instead of shutting it down; do not do this.

***If you access Linux using X2Go from a laptop, it is very important that you logout of Linux or suspend X2Go before closing your laptop.*** Closing the lid on most laptops puts the laptop to sleep. Before it sleeps, the laptop closes all of its network connections, including your X2Go network connection. This will destroy your Linux session.

1. If you are not logged in to ice, do so now. Once your connection is established and you see the Linux desktop, open the Applications menu at the top of the Linux desktop. Click on Applications > Development > Eclipse. Eclipse is an IDE (Integrated Development Environment). In this class you will edit, compile, and run your programs all from within Eclipse.
2. When you are interacting with Eclipse, all of the files that you create, along with the files that the compiler creates, are stored in a place called a "workspace". If you are not prompted for a workspace name, search through the Eclipse menu system until you find the "new workspace" command. Create a new workspace. Call the workspace cs260. Make sure you do not type any spaces in the workspace name.
3. When Eclipse opens, you might see a tab labeled "Welcome". If you see this tab, close it.
4. When you create a program in Eclipse, the files and the folders that hold the packages that make up the program are stored together in a unit called a project. The project is stored inside of the workspace. A project contains multiple classes and packages. This diagram shows the relationship between workspaces, projects, packages and classes.



 You can have multiple workspaces if you wish, but you only need one. However, *you must create a new project for every new program*, even if your program consists of a single file.

1. Let's write our first program. Create a new Java project by clicking File > New > Project from the Eclipse menu bar. The New Project dialog opens. Select Java Project and click Next. Name the project Speaker. (Project names may not contain any spaces. By convention, projects in Eclipse carry the same name as the name of program's main class. We will create the main class later.) The rest of the options for the project (default location, etc.) are OK. Click Finish. (If Eclipse asks you about a "Java Perspective", click the checkbox for "Remember my decision", and then click the "yes" button.)

 The Eclipse window contains different areas showing you information about your program.

* To the left is the package explorer panel. It contains all of the projects in your workspace. (Right now, there should only be one.) Click on the small triangle to the left of the project name to expand it. Your source files will eventually be stored inside the src folder, inside the Speaker project. You cannot expand the src folder now, because it is empty, but once you add a source file to the project, an expansion triangle will appear to the left of src.
* At the bottom of the Eclipse window is the information panel. Clicking on different tabs in this panel will give you information about problems with your project, Javadoc errors, and declarations.
* The main part of the window in the middle is the editing panel. Once we start creating classes, you will edit your code here.
* On the right is the Outline panel. We won't be using that panel this semester. You can close it by clicking on the "X". This will give you more room in the edit panel.
1. We will write a short program consisting of a main class and two helper classes. The helper classes will be located in a package, and the main program will import these classes from the package.

We will create the classes that go in the package first. In Eclipse, you must create a package before you put any classes in it. Whenever we create a new program component (project, package, class, interface, etc.) in Eclipse, we do this from the File > New menu.

 Create a new Java package: File > New > Package. Name your package

 edu.truman.cs260.*yourlastname*.lab

Substitute your last name in the appropriate place.(Remember that the convention in Java is for package names to consist only of lowercase alphabetic characters, digits, and underscores, with the words separated by periods.) Click Finish.

1. Now, create the first class of that goes in the package. We will call this class "Greeter".

Click File > New > Class. A dialog box appears. Check the following things in the "New Class" dialog box:

* Make sure that the correct package name is listed in the text box of the dialog.
* Enter "Greeter" for the class name.
* Make sure that the "public" modifier checkbox is selected, but no other checkboxes are.
* This class will not implement any interfaces, so leave that area blank.
* Look at the checkboxes below "Which method stubs would you like to create?" Since this class will not be our main class, make sure that the option to include a "public static void main" stub is *unchecked*. If we were creating a subclass, we could ask Eclipse to include templates for inherited abstract methods from the superclass, along with calls to the appropriate superclass constructor. These will eventually be useful, so make sure they are checked here.
* Finally, make sure that the "Generate Comments" option is checked. Eclipse will create places for Javadoc comments for the class and each method it creates.

 Click the "finish" button when you are sure that everything is correct. Java will create the new class in the edit pane, and fill in everything it knows about the class, following the information you gave it.

1. Type the following code for the Greeter class. You can delete, edit or leave alone the parts of the class that Eclipse created for you, as appropriate for your program.

 // Don't forget to put your name here in the package name.

 package edu.truman.cs260.turing.lab;

 /\*\*

 \* @author Turing

 \*

 \* A class to store and return a greeting.

 \*/

 public class Greeter

 {

 // The greeting message

 private String greeting;

 /\*\* Construct and initialize a new object.

 \* @param greeting what to say when greeting

 \*/

 public Greeter (String greeting)

 {

 this.greeting = greeting;

 }

 /\*\* Return the greeting.

 \* @return the greeting

 \*/

 public String getGreeting()

 {

 return greeting;

 }

 }

 Eclipse will analyze the syntax of your Java program as you type. Potential errors are noted in Eclipse by putting a red or orange mark in the left margin of the editor pane. Every Javadoc comment and method can be expanded or collapsed by clicking on the '+' or '-' symbol in the right margin. Try collapsing and expanding a few things now.

 Save your class by choosing File > Save. You do not have to specify a location when you save a class; Eclipse will put the class in the correct location inside of the project.

1. Create another class called Farewell. The class should be a part of the same package that you specified in the previous step. The Farewell class should work similarly to the Greeter class. It should contain
* a String instance variable named theFarewell,
* a constructor that initializes the String from a parameter, and
* an accessor method that returns the String instance variable.
1. Create a new class for the main program by clicking on File menu. The main program **should not** be part of the Lab package; make sure that nothing appears in the Package text box. Call the class Speaker. Because this class will contain our main method, make sure the "public static void main()" option **is** selected. The main class of a Java program does not have a constructor or inherit any methods, so make sure that the "Constructors from Superclass" and "Inherited Abstract Methods" checkboxes **are not** checked.

 When you are typing the class, you will need to import the Greeter and Farewell classes from the package you created using appropriate import statements at the top of the class. The main method should prompt the user to enter three things: the user's name, a greeting, and a farewell. It should then create new Greeter and Farewell objects. Initialize the new objects with the appropriate String entered by the user when you call the constructors. Finally, your program should print a greeting and a farewell message for the user.

 Here is an example of what your program should look like when it is run. User input is shown in bold.

 Please enter your name: **Fred**

 Please enter a greeting: **Hi**

 Please enter a farewell: **get lost**

 Hi, Fred!

 Nice of you to stop by.

 It is time for you to go, so get lost Fred.

1. Once you have finished with the three classes, test them out. You do not have to compile a Java program created in Eclipse. Eclipse checks the syntax of your program as you type, and compiles each class for you automatically when you execute the program.

 To execute the program, click on Run > Run. Eclipse will execute main(). You will interact with your program in a new tab that will appear in the information panel. Error messages (if any) will appear in the "problems" tab of the information panel.

1. When you are ready to submit a Java program created in Eclipse, you will need to assemble or "wrap up" the classes that make up your program into a Java ARchive (JAR) file. The JAR format is the standard method for distributing Java programs electronically. A JAR is a single file that can be unwrapped to reveal multiple source files or executable files. All of the JARs that we create will contain only source.

 Let's create a JAR file now for this program. Click File > Export. In the Export dialog window, do the following:

* 1. Expand the Java item, highlight JAR file (**not** Runnable Jar file) and click Next.
	2. Expand "Speaker" under "Resources to Export". Then, click the checkbox next to "src. Make sure that "settings" is **not** selected.
	3. Directly below the resources pane is a series of checkboxes. Make sure that "Export java source files and resources" is checked, and that *no other boxes* are checked.
	4. You will need to specify a location to create the JAR file. We will create a new directory (folder) under your home directory and save the JAR file here. In Linux, your home directory has the same name as your login name.

Find the "Select the export destination" text box, and click the "Browse" button next to it. Select your login name in the Places panel, then click "Create Folder". Call the folder "jars". Click OK.

* 1. In the "JAR file:" textbox, type the file name "Speaker.jar". (In Java, it is convention to name the JAR file after the name of the main class.) Below the "JAR file" text box is a series of options. Make sure that all options are unchecked.
	2. Click Finish.
	3. You are done. Exit Eclipse.
1. You now need to verify that the jar file you created contains all of the source files the classes of your program, and that it contains *only* the source classes. (Recall that Java source files end in ".java" and that compiled versions of the source end in ".class".)

 Go to the Applications menu on the Linux desktop, find the "File Manager", and start it. When the file manager starts, you should see a folder that represents your Eclipse workspace and the directory "jars" that you created in the previous step. Double-click on jars to go there.

 **Right-click** on the .jar file and choose "Open with XArchiver". XArchiver is a Linux tool that allows you to look inside different types of packaged programs, including JAR files, to see their contents.

 Use XArchiver to verify that the .jar file contains your main class source file (Speaker.java) and a folder called "edu", and that the source files for the classes in the package that you created are located in the appropriate subfolders. Important: verify that your JAR file contains source files only (.java) and not class files (.class). If the JAR file is not correct, go back to step 10 and repeat the JAR file creation process.

1. Log out of the Linux system by choosing "Log Out" from the Linux applications menu.

Remember that it is very important that you suspend or exit X2Go correctly. DO NOT JUST LOG OUT OF WINDOWS OR CLOSE THE X2Go WINDOW.

 Once you have logged out of the ice, don't forget to log out of Windows. You do not have to submit anything for this lab.