COLBY COLLEGE
Department of Mathematics
and Computer Science

CS115 **Allen B. Downey**
Spring 1998

Assignment 1: Java compilation and execution

Due: Wednesday, February 11, 1pm

Reading: Kamin, Mickunas and Reingold, Chapter 1, pages 3-13, Chapter 2, pages 19-31.

1 Programming assignment

The purpose of this assignment is to allow you to practice using Roaster, which is the Java interactive development environment (IDE) we will be using in this class. Also, you will practice writing and debugging small Java programs.

In general, I will hand out assignments during lab time on Wednesdays and they will be due the following Wednesday at the beginning of lab time. You should start the assignment and get as much done during lab time as you can. Some people may finish some assignments during the lab, but most people will probably have to finish on their own time during the week.

Start Roaster

1. Each time you sit down at a Macintosh, you should start by shutting down all running applications, closing folders, and organizing the desktop if necessary.

2. Find the icon for the Roaster IDE on the hard disk, probably in a folder called Roaster.

3. Double-click on the icon to start the Roaster IDE.

Create a new project folder

1. From the Project menu, select "New Project".

2. In the dialog box that appears, click on the "Desktop" button on the right and then the "New Folder" just below it.

3. In the dialog box that appears, type in a name for the folder, such as "Hello World," and click on the "Create" button. This creates a new folder on the desktop; this folder will contain all the files associated with this project.

4. Back in the original dialog box, type in a name for the new project, such as "Hello.π". It is standard to give projects names that end in π. To type a π on a Macintosh, hold down the Option key and hit "p". Click the "Save" button.

5. The Project Wizard will appear to walk you through the process of creating a new project. First, it asks for the type of the project, either an application or an applet. We will be doing some of each this semester, but we will start with an application. Select the application button and press Next.

6. You will be prompted for a class name. Within each project, there many be many classes, but this project will only have one. You should call it Hello. Ignore the other options and press OK to continue.

A project window entitled "Hello.π" should appear on the screen. One of the lines in the window should be "Hello.java." The suffix `.java` means that this file contains Java source code.

Roaster expects all the files associated with a project to be in a folder together, and it can get quite snippy if they are not. Every time you start a new project, you should follow the steps above to create the new folder and the new project.

Generate some java code

1. Open Hello.java by double-clicking on the arrow to the right of the file name. A window should appear with a few lines of Java code. This program was created by the Wizard so that you can always start with a working program. Every time you start a project, you will start with the "Hello, world" program and modify it a little at a time until it does whatever you want it to do.

2. Compare the program on your screen with the program on page 19 of the textbook (which I have provided, in case you don't have the book yet). Which things are the same? Which are different?

Run the program

1. Save the program by selecting "Save" from the File menu.

2. Run the program by selecting "Run" from the Project menu.

   A window will appear that monitors the activity of the Java compiler. You will see an animated coffee grinder and various messages. If a message window appears indicating that there are errors in the code, you will have to fix the program and try again. At this point, though, there should be no errors.

   Assuming that your program compiles correctly, Roaster will run it. There is actually a separate program, called the Roaster Runner that runs the program. It will create an I/O window with the words "Hello, world!" in the corner.

   If anything goes wrong up to this point, feel free to summon a lab denizen for assistance. Otherwise, continue.

QUIT THE ROASTER RUNNER!

One of the most annoying things about Roaster is the following bug: if you leave the Roaster Runner in the background and return to the Roaster IDE, and then edit your program and run it again, it pretends to recompile the program, but then IT RUNS THE OLD VERSION AGAIN!!! If you are trying to debug a program, this can be really mystifying and frustrating. Therefore, you should get in the habit of quitting the Roaster Runner every time you run a program.

IMPORTANT: Closing the I/O window by clicking the close box does not quit the Roaster Runner!

1. Quit the Roaster Runner by selecting "Quit" from the File menu or pressing Command-Q. It will ask you if you want to save the contents of the I/O window. You don't.

Edit the program

1. Read pages 20 and 21 of the textbook so that you understand what each line of this program does.
2. Add a comment line to the program (anywhere) that says something like, "Modified by Your Name, 2-4-1998".
3. Run the program again. The new comment should not affect the execution of the program. QUIT THE ROASTER RUNNER!
4. Add a second print statement that prints a second message after the "Hello, world!". Something witty like, "How are you?"
5. Run the program again. Don't forget to QUIT THE ROASTER RUNNER!

Type in another program

1. Return to the Hello.java window and modify your program so that it looks like the program on page 21. You should have an idea by now what things have to be exactly the same (like capitalization) and which things can vary without breaking the program (like comments and most spacing).
2. Before you run the program you have to make a couple of changes. First, you can't call the class Temperature; you have to call it Hello, because that's the name of the file, and the name of the class has to be the same as the name of the file. This is a dumb rule that the otherwise sensible Java people came up with during a caffeine-induced frenzy. Nevertheless, it can be accomodated easily enough by making sure the name of the class is Hello.

   IMPORTANT: If you ever forget this rule, and the class name is different from the file name, confusing things will happen. Avoid this situation!
3. The second change you have to make is to get rid of the stuff that takes input from the keyboard. Instead, we are just going to set the temperature to 32. Delete line 8, and change line 9 to: `temperature = 32;`
4. Run the program. Check that the output is correct (0 deg C), and quit the Roaster Runner.
5. Modify the program to find the Celcius equivalent of -40F.

Make mistakes

It is a good idea to commit as many errors as you can think of with this program, so that you see what error messages the compiler produces. Sometimes the compiler will tell you exactly what is wrong, and all you have to do is fix it. Sometimes, though, the compiler will produce wildly misleading messages. You will develop a sense for when you can trust the compiler and when you have to figure things out yourself.

1. Do the exercise on page 23 of the textbook, numbers 1a through 1d.
2. Remove line 6.
3. Bring line 6 back, but spell temperature wrong.
4. Instead of calling it temperature, call it class, or void, or int, or any of the other reserved keywords in table 2.1 on page 25.
5. Delete one of the quotation marks that end a string. Add an extra one.
6. Delete one of the parentheses in an expression. Add an extra one.

   How many of the syntax rules of Java can you guess just by the error messages from the compiler? If you are ever unsure whether something is legal in Java or not, ask the compiler!

Preserve your program for posterity

1. If you want to print your program, select "Print" from the File menu.
2. When you are done working on the program, close the source code window and the project window. Open the project folder on the desktop. You should see the project file (Hello.π), the source code file (Hello.java), and the byte code file (Hello.class). The byte code file was created by the compiler when you compiled the program. You can keep it around or you can discard it.
3. Move your folder from the desktop onto a floppy disk or over the network to somewhere where you can get at it later. Please clean up after yourself. Do not leave files lying around on the lab machines. They will be deleted (eventually).
4. IMPORTANT: when you run your program later, always copy the entire project folder onto the local hard disk where you are running. If you run your program off a floppy disk or over the network, it will run incredibly slowly.

Exercises

1. On pages 48 and 49 of the textbook, choose ANY TWO of exercises 4, 6 and 8. Write and debug your programs and test them with a variety of inputs.
2. Add comments to your code to explain what each part of it does. Add a comment that has your name, the date, the number of the assignment, and the number of the exercise you chose to do.
3. Although the spaces, tabs and newlines in your program do not (generally) affect its execution, they do make the program easier to read. There are some standard conventions for how to layout a Java program. By reading and copying the programs in the book, you can figure out what these conventions are. Please check your program to make sure that lines are indented properly, etc., before you turn it in.
4. Print your program and write your name, the number of the assignment, and the number of the exercise in the upper-right corner of the page.
5. You should have two pages, one for each program. Staple them together and hand them in.

Congratulations on completing the first assignment. Only ten more to go!