

# 16.410: Jump Starting With Java

*by Robert Effinger and Shannon Dong*

## Introduction

This jumpstart shows you the basics of getting Java started, running simple programs, and simple editing and debugging. The jumpstart is oriented towards Windows installations, but can be adapted for Unix and other installations. Note that the Java SDK installed in the 16.410 computer lab is the Windows version.

Please note that this jumpstart will give you only the most rudimentary skills in Java. To get through the course, you will have to obtain a bit more information. The online Sun Tutorial and the Java in a Nutshell book are excellent resources. The url for the Sun Tutorial is: <http://java.sun.com/docs/books/tutorial/information/download.html>

## Installation

Please follow these steps to install Java.

- 1.) Download the file *tutorial.zip* from the url listed above, and extract them into a folder.
- 2.) Click on the file *index.html*
- 3.) Click on First Steps (Microsoft Windows)
- 4.) Click on A Checklist
- 5.) This Checklist provides all of the info you will need to install the SDK.

## Hello World

Once you have successfully installed Java, step 2 of the tutorial shows how to create a hello world program.

The Hello World code:

```
/**
 * The HelloWorldApp class implements an application that
 * displays "Hello World!" to the standard output.
 */
public class HelloWorldApp {
    public static void main(String[] args) {
        // Display "Hello World!"
        System.out.println("Hello World!");
    }
}
```

Now you need to open a MS-DOS command prompt on your Windows machine:

Start->All Programs->Accessories->Command Prompt

Then, get into the directory where your HelloWorld file is located using the cd command. The online tutorial shows you how to do this.

To compile the program, type: `javac HelloWorldApp.java`

To run the program, type: `java HelloWorldApp`

If you haven't set your path variable yet, you will get the error message:

**The name specified is not recognized as an internal or external command, operable program or batch file**

Instructions for how to set your PATH variable are in the pink box titled Error Explanation.

When the program runs correctly, the computer should print out:

**Hello, World!**

For additional practice, please try to compile and run the simple Factorial Example Program on page 7 of Java in a Nutshell. There are also many other examples in the book and online tutorial to help you get familiar with Java.

## Using the Eclipse SDK

Many people prefer to use an integrated software development kit (SDK). One SDK that you might find useful is Eclipse SDK 3.1, which can be downloaded from <http://www.eclipse.org/downloads/index.php>. Some advantages of using this SDK include: the user interface is very nice, the SDK organizes your files into project folders, and the debugger is somewhat more user-friendly. If you would like to use Eclipse, follow these directions to get started:

- 1.) Download and unzip Eclipse to your Program Files folder. Open Eclipse.
- 2.) When asked, enter in the path of the directory in which you want to work. A typical one could be: C:\Documents and Settings\\My Documents\java.
- 3.) You should see a window titled “Welcome to Eclipse 3.1” (If you don’t, click on the Help menu, select Welcome). Go to the Tutorials link (an image with “1 2 3”).
- 4.) Click on Java Development.
- 5.) Follow the steps through the Simple Java Application.
- 6.) Once you have a HelloWorld program running, try creating an error such as deleting a semicolon. Run it again to see what the error looks like.
- 7.) Then run it with the debug option (Run -> Debug As -> Java Application).
- 8.) After you terminate the debugger, you can get back to the java perspective by going to Windows -> Open Perspective -> Java.

## J-Unit

JUnit is an automated testing program that will be used to grade the problem sets. To install junit go to <http://www.junit.org/index.html>, and click on Download. Select a location, and unzip the files preferably into the directory C:\junit3.8.1 If you use a different directory make sure it doesn’t contain spaces, such as “Program Files”.

You will need to set the CLASSPATH environment variable. Don’t confuse CLASSPATH with the PATH environment variable, they are separate variables. Next try to run the sample test programs at the end of this tutorial to see that everything is working. Now, please open the README file in the junit3.8.1 directory. Click Getting Started, and then Test Infected – Programmers Love Writing Tests. This gives you a brief introduction to junit.

For your convenience, we have provided some example code to help you get started with junit. The file below implements a junit test for the first part of Problem 1 of Pset1. We show how to create a test for the reverse() function, and then run that test. You will need to create similar tests for the append() function, and also for Problems 2 and 3 of Pset1.

Name this file: FunWithLinkedListsTest.java

```
/*
 * This junit class tests the FunWithLinkedLists reverse() and append()
 * functions
 */

import junit.framework.*;
import java.util.List;
import java.util.LinkedList;

public class FunWithLinkedListsTest extends TestCase
{
    // this class contains tests for the FunWithLinkedLists reverse()
    // and append() functions

    public void testFunWithLinkedListsReverse()
    {
        LinkedList test_ll = new LinkedList();
        FunWithLinkedLists having_fun = new FunWithLinkedLists();

        String s_one = new String("one");
        String s_two = new String("two");
        String s_three = new String("three");
        test_ll.add(s_one);
        test_ll.add(s_two);
        test_ll.add(s_three);

        test_ll = having_fun.reverse( test_ll );

        //asserts that the elements have been reversed!
        Assert.assertTrue(test_ll.getFirst() == s_three);
        Assert.assertTrue(test_ll.getFirst() == s_two);
        Assert.assertTrue(test_ll.getLast() == s_one);
    }

    public void testFunWithLinkedListsAppend()
    {
        // add code here to test the append() function

        // replace these bogus assert statements with
        // something that actually tests your append() function

        Assert.assertTrue( true );
        Assert.assertTrue( true );
    }
}
```

The simplest way to run this junit test (and the others) is to make sure that FunWithLinkedLists.java and FunWithLinkedListsTest.java are in the same directory. Compile them both, and then type:

```
> java.textui.TestRunner FunWithLinkedListsTest
```

You should see the result:

```
..  
Time: 0.01  
OK (2 tests)
```

☺ Have Fun Programming in Java!!! ☺