Introduction to Java

Objectives:

1. Learn to use a2ps to print Java program.
2. To set up your Linux account to compile and run Java applications.
3. To practice writing simple Java applications.

Preparation: Before the exercise read the following chapters in *Java: How to Program* by Deitel and Deitel, sixth edition: Preface, Chapters 1, 2, 3, 4, 5 and 6. Since these chapters are written for the beginning programmer, many sections can be skimmed. We recommend that you study the programs and if you can explain each line then skip over the text. Otherwise, read enough to understand the programs.

Laboratory Assignment:

1. Use a2ps for printing Java Programs If a file has a `.java` extension, a2ps assumes it is a Java file and bolds all reserved words. All Java files handed in for this course must be printed with a2ps.

2. Checking for Java 1.5: We will be using Java 2 version 1.5 in this course. To check which version of Java you are using, type

```
% java -version
```

If your Linux account is not set up to compile and run Java applications, seek help.

3. Compile and Run a Simple java Application: Using emacs editor, type in the following Java application program. The java code should be colorized by emacs. If not seek help.

```
// A first program in Java

public class Welcome {

   public static void main ( String args []) {

      System.out.println( "Welcome to Java Programming!" );
   }
 }
```

Save the file as **Welcome.java**. Note the file name **MUST** have the same name as the class in the file plus the `.java` extension!

For the above application, the class name is ”Welcome”.

To compile a java application, you type

```
% javac Welcome.java
```

which creates a file of Java byte code called `Welcome.class`.

To run a java application, you use the classname not the file name! In this case,

```
% java Welcome
```

If you have are having trouble compiling and running the above program, ask for help!

4. Using `Scanner` Class for Keyboard Input Input and output in Java can be a bit tricky for beginners. A simple approach is to use the `Scanner` class for input and the `System.out.println()` method for output.

To read input, you first import the proper package:

```
import java.util.Scanner;
```

create a `input` object as follows

```
Scanner input = new Scanner(System.in);
```

then to read an integer you call the `nextInt()` method such as

```
a = input.nextInt();
```

Here is a working programming using the `Scanner` class.

```
// Using Scanner class for input

import java.util.Scanner;  // need for Scanner class

/**
 * Test program to use Scanner class to read input
 * Scanner class in Java 1.5 and later
 *
 * Created: Thu Dec 16 11:58:40 2004
 *
 * @author Dr. Daniel C. Hyde
 * @version 1.0
 */
public class ScannerTest {

    /**
     * main method to test Scanner Class
     *
     * @param args a String[] value
     */
    public static void main(String[] args) {
```

```
Scanner input = new Scanner(System.in);
int a;
double x;
String word;
String line;

System.out.print("Enter int, double, word, line: ");
a = input.nextInt();
x = input.nextDouble();
word = input.next();     // returns a String up to next whitespace
        line = input.nextLine(); // returns rest of line & eats the '\n'

        System.out.println(a + ":" + x + ":" + word + ":" + line);
    }
}

/* run
Enter int, double, word, line: -12 34.56 Great time had by all!
-12:34.56:Great: time had by all!
*/
```

Write a Java application to input three floating point numbers as doubles using the `Scanner` class and print out their sum using `System.out.println()` method for output.

5. Using JOptionPane from Swing API: The Java 2 **Swing** *Application Programming Interface* (API) provides us with a second easy way to do input and output. The class **JOptionPane** provides simple dialog boxes for input and output much like Visual Basic.

Write a Java application to input three floating point numbers as doubles and print out their sum. Use the **JOptionPane** class in the **Swing** API for input. See example on pages 104-107 in sixth edition of Java text. Print out the answer using `System.out.println()` method for output.

6. Formatting Output for doubles: Write a Java application to input ten floating point numbers as doubles and print out their average with two decimal places. To control the outputting of doubles, see page 102 in sixth edition of Java text. Print out the answer using `System.out.println()` method for output.

7. Read "Java for C++ Programmers": Read carefully, the tutorial by Marvin Solomon at

`http://www.cs.wisc.edu/~solomon/cs537/java-tutorial.html`

Skip the section on **Threads** for now.

If you have trouble accessing Marvin Solomon's web page, a local Bucknell copy is available on the CSCI 355 web page.

8. Reading from a file: Write a Java application to read the lines in a text file and print them to the shell window using the `System.out.println()` method.

Hint: You will find the information needed by a careful reading of Solomon's web pages.

9. Explore the CSCI 355 Java Resources Web Page: Spend some time exploring the CSCI 355 Java Resources web page at

`http://www.eg.bucknell.edu/~cs355/2007-spring/javaResources.html`

Since we will be using it later, we recommend you bookmark it.

Hand In: For Exercises 4, 5, 6 and 8, combine all the Java listings and outputs from runs into one handin file with a **.java** extension. Print using a2ps command.