CMPSC 111
Introduction to Computer Science I
Spring 2006
Bob Roos
Home page: http://cs.allegheny.edu/~rroos/cs111s2006/
Lab 1
Due Friday, 3 February, at the beginning of class

Purpose: Practice with input, variables, and expressions

Summary: You will write a Java program that implements a basic input-process-output algorithm.

Details:

File Management
Don’t forget to create a new subdirectory for each lab. For instance, once you log on you can type
“mkdir lab1” to create a directory named “lab1” and then “cd lab1” to change to this directory.

Suggestion: If you expect to use your computer science account for anything other than CMPSC 111, create a separate directory for your CMPSC 111 work (“mkdir cs111”, for example). Within your CMPSC 111 folder, create separate directories for each lab, e.g., “lab0”, “lab1”, etc., and keep all your work from each lab in its own folder. You’ll be glad you did.

The Input/Process/Output Algorithmic Pattern
Review the definition of “algorithm” given in your book on page 17. Today you will write several programs that implement one of the simplest algorithms possible. Each one will prompt the user for input, accept the input, perform some sort of processing of the input data, and print out labelled results.

Comments
Every Java program you write should be documented with comments. There are two styles of comments (see pages 99–100 in your textbook):

- Whenever two forward slashes appear, not as part of a String constant, they begin a comment which extend to the end of the line. Here are some examples:

  ```java
  int years; // number of years for this mortgage
dooble rate; // annual interest rate
dooble principal; // amount of the mortgage loan
  System.out.println("This // is not a comment!"); //...but this is!
  ...
  ```

- Any text that lies between the pair of characters “/\*” and the pair of characters “*/” is a comment. Such comments can extend over multiple lines. A special kind of comment is called a Javadoc comment — this begins with “/\*\*”, ends with “*/”, and has a row of “*/”’s running down the left edge of the comment. Here’s an example of a Javadoc comment:

```java

```
/**  
* Lab 1 Tuesday, 31 January 2006  
* Author Bob Roos  
* This program reads in a name, a quantity, and a unit price  
* for a purchase of some interociters. It prints the total  
* cost, including sales tax and shipping, for the purchase.  
*/  
public class Lab1  
...

Review of the Scanner class  
See section 2.3 of your textbook. To perform input, you must import the Scanner class (using import java.util.*; at the top of your program). In your program, before you read anything in you must declare a Scanner object. The book always uses the name “keyboard” for this, but any variable name is okay. I like names such as “userInput” or “input” or “fromUser” rather than keyboard, but this is a matter of individual choice.  

The various methods of the Scanner class are described on pages 92–93 of your textbook. Here are some examples:  

...  
Scanner userIn = new Scanner(System.in);  
System.out.print("Enter your first and last names: ");  
String first = userIn.next();  
String last = userIn.next();  
System.out.print("Hello, " + first + " " + last + "!");  
System.out.print("How many interociters do you wish to purchase? ");  
int quantity = userIn.nextInt();  
...

You will need nextInt(), nextDouble(), and either next() or nextLine(). The next() method reads in a word, the nextLine() method reads in an entire line, including spaces but not including the end of line character.  

Part 1  
The first program is completely up to you. You must make up an application for your program. It could be a simulation of an ATM machine, or an online shopping application, or a robotic financial advisor, or anything else. Please be imaginative and fun (but your program should be inoffensive and in good taste) — make me laugh. Don’t use anyone else’s idea — make up your own.  

Your program must:  
• print out your name and the date (see last week’s lab for how to do this); be sure to put a space between your name and the date  
• prompt the user for at least three different input values (more than three would be even better), then read them in using the Scanner class. At least one of these must be a String, at least one must be an int, and at least one must be a double  

Be sure you read the “Gotcha!” on page 95 about the “nextLine()” method!
• perform a nontrivial numeric computation on your input values. It should involve several different arithmetic operators and should “make sense” in the context of the application (don’t just do random arithmetic). Your computations should involve at least one int variable and at least one double variable. Computations may be spread out over several assignment statements.

• display the user’s input values, together with any results of computations, in a labelled, neatly-formatted output. Don’t just print the answers — print the user’s input values back out as well, appropriately labelled.

In addition, your program should begin with a Javadoc-style comment giving your name, the lab number, and a description of what the program does. Throughout your program you should have comments explaining the major sections of your program or describing, in English, what the calculations mean. Don’t overdo — one comment per program line is probably too much. But you generally need to have comments every few lines or so to “summarize” what’s going on.

Test your program to make sure that it works. Print out the program using the a2ps command, after being sure that it is properly indented and commented and has no line wraps. Run the program and print out the output using the script command as shown in last week’s lab.

Part Two

There is a remarkable formula that gives the area of any polygon as long as you know the coordinates of the polygon’s vertices. In class I’ll demonstrate this, but here’s a description:

Assume the vertices of the polygon are given as \((x, y)\) values, in counterclockwise order. Name these points \((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\). Then the area is equal to one half of the sum:

\[
x_1(y_2 - y_n) + x_2(y_3 - y_1) + x_3(y_4 - y_2) + \cdots + x_n(y_1 - y_{n-1})
\]

Write a program that finds the area of a 5-sided polygon (in other words, \(n = 5\) in the above formula), given the coordinates of the five vertices. Your program should print out your name, the date, and should ask the user to enter five pairs of \(x\) and \(y\) values. It should print these out as ordered pairs (the output should have parentheses and commas), then it should compute and print the area.

Print the program and output as you did in part 1.

Optional: Ways to Make Your Program More Interesting

If you would like to make your program more interesting, feel free to use any of the built-in mathematical functions described in the table on page 335; these give you the ability to compute square roots, raise things to powers, etc. (I believe there is an error in that table — the last example should say “Math.sqrt(4.0)”, not “sqrt(4.0)”.)

You can also generate random values between 0 and any integer \(n\) by using the expression “new Random().nextInt(n)”. I used this in my sample program (see Web page) to generate a “bid number” for the user.

Finally, if you would like to make your program even more flexible, consider using an “if” statement (our next topic; see page 133 for some examples).
Checklist:

- All requirements met (PG-rated application in good taste; inputs `String`, `int`, and `double`; labelled output, etc. — see above)
- Header comments with the assignment number, your name, your section
- Comments explaining program
- Correctly indented code; no wrapped lines
- Output “looks good” (no run-together words, no lines wrapped, etc.)
- Hard copies (printed using `a2ps`) of both and output, signed with honor code pledge
- Be sure you are logged off before you leave the lab.