Lab 2 Data and Expressions

Names and Places

The goal in this exercise is to develop a program that will print out a list of student names together with other information for each. The tab character (an escape sequence) is helpful in getting the list to line up nicely. A program with only two names is in the file *Names.java*.

1. Compile and run *Names.java* to see how it works
2. Modify the program so that your name and hometown and the name and hometown of at least two classmates sitting near you in lab also are printed. Save, compile and run the program. Make sure the columns line up
3. Modify the program to add a third column with the intended major of each person (assume Feven's major is Computer Science and Yohannes' major is Math). Be sure to add a label at the top of the third column and be sure everything is lined up (use tab characters!).

A Table of Student Grades

Write a Java program called Grades.java that prints a table with a list of at least 5 students together with their grades earned (lab points, bonus points, and the total) in the format below.

```
//////////////////\\\\\\\\\\\\\\\\\\\
== Student Points ==
\\\\\\\\\\\\\\\\\\\//////////////////
Name          Lab           Bonus         Total
----          ---           -----         -----
Yohannes      43            7             50
Feven         50            8             58
Tomas         39            10            49
```

The requirements for the program are as follows:

1. Print the border on the top as illustrated (using the slash and backslash characters).
2. Use tab characters to get your columns aligned and you must use the + operator both for addition and string concatenation.
3. Make up your own student names and points—the ones shown are just for illustration purposes. You need 5 names.

Two Meanings of Plus

In Java, the symbol + can be used to add numbers or to concatenate strings. This exercise illustrates both uses. When using a string literal (a sequence of characters enclosed in double quotation marks) in Java the complete string must fit on one line. The following is NOT legal (it would result in a compile-time error).

```
System.out.println ("It is NOT okay to go to the next line in a LONG
string!!!");
```

The solution is to break the long string up into two shorter strings that are joined using the concatenation operator (which is the + symbol). So the following would be legal

```
System.out.println ("It is OKAY to break a long string into " +
      "parts and join them with a + symbol.");
```

So, when working with strings the + symbol means to concatenate the strings (join them). BUT, when working with numbers the + means what it has always meant—add!

1. **Observing the Behavior of +** To see the behavior of + in different settings do the following:
   a. Study the program in file *PlusTest.java*.
   b. Compile and run the program. For each of the last three output statements (the ones dealing with 8 plus 5) write down what was printed. Now for each explain why the computer printed what it did given that the following rules are used for +. Write out complete explanations.
      - If both operands are numbers + is treated as ordinary addition. (NOTE: in the expression a + b the a and b are called the operands.)
      - If at least one operand is a string the other operand is converted to a string and + is the concatenation operator
      - If an expression contains more than one operation expressions inside parentheses are evaluated first. If there are no parentheses the expression is evaluated left to right.
   c. The statement about when the computer was invented is too scrunched up. How should that be fixed?
2. **Writing Your Own Program With +** Now write a complete Java program called *MorePlusTest.java* that prints out the following sentence:

   ```
   Ten robins plus 13 canaries is 23 birds.
   ```

   Your program must use only one statement that invokes the *println* method. It must use the + operator both to do arithmetic and string concatenation.

Prelab Exercises

1. What is the difference between a variable and a constant?

2. Explain what each of the lines below does. Be sure to indicate how each is different from the others.
   a. int x;
   b. int x = 3;
   c. x = 3;

3. The program in *Average.java* reads three integers and prints the average. Fill in the blanks so that it will work correctly.
4. Given the declarations below, find the result of each expression

```
int a = 3, b = 10, c = 7;
double w = 12.9, y = 3.2;
```

a. `a + b * c`

b. `a - b – c`

c. `a / b`

d. `b / a`

e. `a - b / c`

f. `w / y`

g. `y / w`

h. `a + w / b`

i. `a % b / y`

j. `b % a`

k. `w % y`

5. Carefully study the program in *Errors.java* and find and correct all of the syntax errors.
6. Trace the execution of the program in *Teace.java* assuming the input stream contains the numbers 10, 3, and 14.3. Use a table that shows the value of each variable at each step. Also show the output (exactly as it would be printed).

Area and Circumference of a Circle

Study the program in *Circle.java*, which uses both variables and constants:

Some things to notice:

- The first three lines inside main are declarations for PI, radius, and area. Note that the type for each is given in these lines: final double for PI, since it is a floating point constant; int for radius, since it is an integer variable, and double for area, since it will hold the product of the radius and PI, resulting in a floating point value.
- These first three lines also hold initializations for PI, radius, and area. These could have been done separately, but it is often convenient to assign an initial value when a variable is declared.
- The next line is simply a print statement that shows the area for a circle of a given radius.
- The next line is an assignment statement, giving variable radius the value 20. Note that this is not a declaration, so the int that was in the previous radius line does not appear here. The

same memory location that used to hold the value 10 now holds the value 20—we are not setting up a new memory location.

- Similar for the next line—no double because area was already declared.
- The final print statement prints the newly computed area of the circle with the new radius.

Modify *Circle.java* as follows:

1. The circumference of a circle is two times the product of Pi and the radius. Add statements to this program so that it computes the circumference in addition to the area for both circles. You will need to do the following:
    - Declare a new variable to store the circumference.
    - Store the circumference in that variable each time you compute it.
    - Add two additional print statements to print your results.

    Be sure your results are clearly labeled.

2. When the radius of a circle doubles, what happens to its circumference and area? Do they double as well? You can determine this by dividing the second area by the first area. Unfortunately, as it is now the program overwrites the first area with the second area (same for the circumference). You need to save the first area and circumference you compute instead of overwriting them with the second set of computations. So, you'll need two area variables and two circumference variables, which means they'll have to have different names (e.g., area1 and area2). Remember that each variable will have to be declared. Modify the program as follows:
    - Change the names of the area and circumference variables so that they are different in the first and second calculations. Be sure that you print out whatever you just computed.
    - At the end of the program, compute the area change by dividing the second area by the first area. This gives you the factor by which the area grew. Store this value in an appropriately named variable (which you will have to declare).
    - Add a println statement to print the change in area that you just computed.
    - Now repeat the last two steps for the circumference.

    Look at the results. Is this what you expected?
3. In the program in *Circle.java*, you showed what happened to the circumference and area of a circle when the radius went from 10 to 20. Does the same thing happen whenever the radius doubles, or were those answers just for those particular values? To figure this out, you can write a program that reads in values for the radius from the user instead of having it written into the program ("hardcoded"). Modify your program as follows:
    - At the very top of the file, add the line

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

      This tells the compiler that you will be using methods from the Scanner class. In the main method create a Scanner object called scan to read from System.in.

- Instead of initializing the radius in the declaration, just declare it without giving it a value. Now add two statements to read in the radius from the user:
  - A prompt, that is, a print statement that tells the user what they are supposed to do (e.g., "Please enter a value for the radius.");
  - A read statement that actually reads in the value. Since we are assuming that the radius is an integer, this will use the nextInt() method of the Scanner class.
- When the radius gets it second value, make it be twice the original value.
- Compile and run your program. Does your result from above hold?

Painting a Room

File *Paint.java* contains a partial program which when complete will calculate the amount of paint needed to paint the walls of a room of the given length and width. It assumes that the paint covers 350 square feet per gallon.

Do the following:

1. Fill in the missing statements (the comments tell you where to fill in) so that the program does what it is supposed to. Compile and run the program and correct any errors.
2. Suppose the room has doors and windows that don't need painting. Ask the user to enter the number of doors and number of windows in the room, and adjust the total square feet to be painted accordingly. Assume that each door is 20 square feet and each window is 15 square feet.

Ideal Weight

Write a program to compute the ideal weight for both males and females. According to one study, the ideal weight for a female is 100 pounds plus 5 pounds for each inch in height over 5 feet. For example, the ideal weight for a female who is 5'3" would be 100 + 15 = 115 pounds. For a male the ideal weight is 106 pounds plus 6 pounds for each inch in height over 5 feet. For example, the ideal weight for a male who is 6'2" would be 106 + 14*6 = 190 pounds. Your program should ask the user to enter his/her height in feet and inches (both as integers—so a person 5'3" would enter the 5 and the 3). It should then compute and print both the ideal weight for a female and the ideal weight for a male. The general outline of your main function would be as follows:

- Declare your variables (think about what variables you need—you need to input two pieces of information (what?), then you need some variables for your calculations (see the following steps)
- Get the input (height in feet and inches) from the user
- Compute the total number of inches of height (convert feet and inches to total inches)
- Compute the ideal weight for a female and the ideal weight for a male (here you basically convert the "word" description above to assignment statements)
- Print the answers

Plan your program, then type it in, compile and run it. Be sure it gives correct answers.

Enhance the Program a Bit The weight program would be a bit nicer if it didn't just give one number as the ideal weight for each sex. Generally a person's weight is okay if it is within about 15% of the ideal. Add to your program so that in addition to its current output it prints an okay range for each sex—the range is from the ideal weight minus 15% to the ideal weight plus 15%. You may do this by introducing new variables and assignment statements OR directly within your print statements.

Lab Grades

Suppose your lab instructor has a somewhat complicated method of determining your grade on a lab. Each lab consists of two out-of-class activities—a pre-lab assignment and a post-lab assignment—plus the in-class activities. The in-class work is 60% of the lab grade and the out-of-class work is 40% of the lab grade. Each component of the grade is based on a different number of points (and this varies from lab to lab)—for example, the pre-lab may be graded on a basis of 20 points (so a student may earn 17 out of 20 points) whereas the post-lab is graded on a basis of 30 points and the in-class 25 points. To determine the out-of-class grade the instructor takes the total points earned (pre plus post) divided by the maximum possible number of points, multiplied by 100 to convert to percent; the in-class grade is just the number of points earned divided by the maximum points, again converted to percent.

The program *LabGrade.java* is supposed to compute the lab grade for a student. To do this it gets as input the number of points the student earned on the prelab assignment and the maximum number of points the student could have earned; the number of points earned on the lab itself and the maximum number of points; the number of points earned on the postlab assignment and the maximum number of points. The lab grade is computed as described above: the in-class and out-of-class grades (in percent) are computed separately then a weighted average of these is computed. The program currently assumes the out-of-class work counts 40% and the in-class counts 60%. Do the following:

1. First carefully hand trace the program assuming the input stream contains the values 17, 20, 23, 25, 12, 15. Trace the program exactly as it is written (it is not correct but it will compile and run so the computer would not know it isn't correct). Fill in the answers to the following questions:
    a. Show exactly how the computer would execute the assignment statement that computes the out of class average for this set of input. Show how the expression will be evaluated (the order in which the operations are performed) and what the result will be.
    b. Show how the computer would execute the assignment statement that computes the in-class average. What will the result be?
    c. Show how the computer would execute the assignment statement that computes the lab grade

2. Now run the program, typing in the input you used in your trace. Compare your answers to the output. Clearly the output is incorrect! Correct the program. This involves writing the expressions to do calculations correctly. The correct answers for the given input should be an out of class average of 82.857 (the student earned 29 points out of a possible 35 which is approximately 82.857%), an in-class average of 92 (23 points out of 25), and a lab grade of 88.34 (40% of 82.857 plus 60% of 92).
3. Modify the program to make the weights for the two components of the grade variable rather than the constants 0.4 and 0.6. To do this, you need to do four things:
    a. Change the declarations so the weights (IN_WEIGHT and OUT_WEIGHT) are variables rather than constants. Note that you should also change their names from all capital letters (the convention for constants) to lowercase letters with capitals starting new words (the convention for variables). So IN_WEIGHT should become inWeight. Of course, you'll also have to change it where it's used in the program.
    b. In the input section, add statements that will prompt the user for the weight (in decimal form—for example .4 for 40%) to be assigned to the in-class work, then read the input. Note that your prompt should explain to the user that the weight is expected to be in decimal form.
    c. In the section that calculates the labGrade add an assignment statement that calculates the weight to be assigned to the out of class work (this will be 1 minus the in-class weight).

Compile and run your program to make sure it is correct.

Base Conversion

One algorithm for converting a base 10 number to another base b involves repeatedly dividing by b. Each time a division is performed the remainder and quotient are saved. At each step, the dividend is the quotient from the preceding step; the divisor is always b. The algorithm stops when the quotient is 0. The number in the new base is the sequence of remainders in reverse order (the last one computed goes first; the first one goes last).

In this exercise you will use this algorithm to write a program that converts a base 10 number to a 4-digit number in another base (you don't know enough programming yet to be able to convert any size number). The base 10 number and the new base (between 2 and 9) will be input to the program. The start of the program is in the file *BaseConvert.java*. Modify it one step at a time as follows:

1. The program will only work correctly for base 10 numbers that fit in 4 digits in the new base. We know that in base 2 the maximum unsigned integer that will fit in 4 bits is 1111_2 which equals 15 in base 10 (or $2^4 - 1$). In base 8, the maximum number is 7777_8 which equals 4095 in base 10 (or $8^4 - 1$). In general, the maximum base 10 number that fits in 4 base b digits is $b^4 - 1$. Add an assignment statement to the program to compute this value for the base that is input and assign it to the variable maxNumber. Add a statement that

prints out the result (appropriately labeled). Compile and run the program to make sure it is correct so far.

2. Now add the code to do the conversion. The comments below guide you through the calculations—replace them with the appropriate Java statements.

```
// First compute place0 -- the units place. Remember this comes
// from the first division so it is the remainder when the
// base 10 number is divided by the base (HINT %).
// Then compute the quotient (integer division / will do it!) -
// You can either store the result back in base10Num or declare a
// new variable for the quotient
// Now compute place1 -- this is the remainder when the quotient
// from the preceding step is divided by the base.
// Then compute the new quotient
// Repeat the idea from above to compute place2 and the next quotient
// Repeat again to compute place3
```

3. So far the program does not print out the answer. Recall that the answer is the sequence of remainders written in reverse order— note that this requires concatenating the four digits that have been computed. Since they are each integers, if we just add them the computer will perform arithmetic instead of concatenation. So, we will use a variable of type String. Note near the top of the program a variable named baseBNum has been declared as an object of type String and initialized to an empty string. Add statements to the program to concatenate the digits in the new base to baseBNum and then print the answer. Compile and run your program. Test it using the following values: Enter 2 for the base and 13 for the base 10 number—the program should print 1101 as the base 2 value; enter 8 for the base and 1878 for the number— the program should print 3526 for the base 8 value; enter 3for the base and 50 for the number—the program should print 1212.