Chapter 2: Data and Expressions
Lab Exercises

Topics	Lab Exercises
Print and println	
String literals	Names and Places
String concatenation	A Table of Student Grades
Escape sequences	Two Meanings of Plus
Variables	Prelab Exercises
Constants	Area and Circumference of a Circle
Assignment	Painting a Room
Integers and Floating point	Ideal Weight
Arithmetic Expressions	Lab Grades
Operator Precedence	Base Conversion
Input using the Scanner class	

Do the lab exercises in this order:
1) Area and Circumference of a Circle, Part 3
2) Painting a Room
3) Ideal Weight
4) Lab Grades
5) Base Conversion

The Chapter and Section numbers from the textbook that are mentioned in the Labs do not always match (apparently, the Labs were written for an earlier edition of the textbook and never updated to match later editions). I have tried to add a Note with a reference to the correct Chapter, Section, and/or page numbers of the edition we are using.

Area and Circumference of a Circle

Study the program below, which uses both variables and constants:

```
//********************************************************
//  Circle.java
//
//  Print the area of a circle with two different radii
//********************************************************

public class Circle
{
    public static void main(String[] args)
    {
     final double PI = 3.14159;

     int radius = 10;
     double area = PI * radius * radius;

     System.out.println("The area of a circle with radius " + radius +
                        " is " + area);

     radius = 20;
     area = PI * radius * radius;

     System.out.println("The area of a circle with radius " + radius +
                        " is " + area);

    }
}
```

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.

Save this program, which is in file *Circle.java*, into your directory and modify it 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?

Before you start Part 3, make a copy of your Circle.java program and rename it Circle3.java. Remember to change the name of the class inside the program, too.

3. In the program above, 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?

   Note: Do not prompt for two values! Prompt for one value, and your program should use code to double it.

Painting a Room

File *Paint.java* contains the partial program below, 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.

```
//*************************************************************
//File: Paint.java
//
//Purpose: Determine how much paint is needed to paint the walls
//of a room given its length, width, and height
//*************************************************************
import java.util.Scanner;

public class Paint
{
    public static void main(String[] args)
    {
        final int COVERAGE = 350;  //paint covers 350 sq ft/gal
        //declare integers length, width, and height;
        //declare double totalSqFt;
        //declare double paintNeeded;
        //declare and initialize Scanner object

        //Prompt for and read in the length of the room

        //Prompt for and read in the width of the room

        //Prompt for and read in the height of the room

        //Compute the total square feet to be painted--think
        //about the dimensions of each wall

        //Compute the amount of paint needed

        //Print the length, width, and height of the room and the
        //number of gallons of paint needed.
    }
}
```

Save this file to your directory and 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

You have to write this one from scratch. Name it IdealWeight.java. Follow the framework in Paint.java to help you with the structure.

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.

```
// ********************************************************************
//   LabGrade.java
//   This program computes a student's lab grade from
//   the grades on the three components of lab:  the pre-lab
//   assignment, the lab itself, and the post-lab assignment.
// ********************************************************************

import java.util.Scanner;
```



```
public class LabGrade
{
    public static void main (String[] args)
    {
      // Declare constants
      final double IN_WEIGHT = 0.6;  // in-class weight is 60%
      final double OUT_WEIGHT = 0.4; // out-of-class weight is 40%

      // Declare variables
      int preLabPts;    //number of points earned on the pre-lab assignment
      int preLabMax;    //maximum number of points possible for pre-lab
      int labPts;       //number of poitns earned on the lab
      int labMax;       //maximum number of points possible for lab
      int postLabPts;   //number of points earned on the post-lab assignment
      int postLabMax;   //maximum number of points possible for the post-lab
      int outClassAvg;  //average on the out of class (pre and post) work
      int inClassAvg;   //average on the in-class work
      double labGrade;  //final lab grade

      Scanner scan = new Scanner(System.in);

      // Get the input
      System.out.println("\nWelcome to the Lab Grade Calculator\n");
      System.out.print("Enter the number of points you earned on the pre-lab: ");
      preLabPts = scan.nextInt();
      System.out.print("What was the maximum number of points you could have earned? ");
      preLabMax = scan.nextInt();
      System.out.print("Enter the number of points you earned on the lab: ");
      labPts = scan.nextInt();
      System.out.print("What was the maximum number of points for the lab? ");
      labMax = scan.nextInt();
      System.out.print("Enter the number of points you earned on the post-lab: ");
      postLabPts = scan.nextInt();
      System.out.print("What was the maximum number of points for the post-lab? ");
      postLabMax = scan.nextInt();
      System.out.println();

      // Calculate the average for the out of class work
      outClassAvg = preLabPts + postLabPts / preLabMax + postLabMax * 100;

      // Calculate the average for the in-class work
      inClassAvg = labPts / labMax * 100;

      // Calculate the weighted average taking 40% of the out-of-class average
      // plus 60% of the in-class
      labGrade = OUT_WEIGHT * outClassAvg + IN_WEIGHT * inClassAvg;

      // Print the results
      System.out.println("Your average on out-of-class work is " + outClassAvg + "%");
      System.out.println("Your average on in-class work is " + inClassAvg + "%");
      System.out.println("Your lab grade is " + labGrade + "%");
      System.out.println();
    }
}
```

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*. Save this file to your directory, then 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.


```
// ********************************************
//   BaseConvert.java
//
//   Converts base 10 numbers to another base
//   (at most 4 digits in the other base).  The
//   base 10 number and the base are input.
// ********************************************

import java.util.Scanner;

public class BaseConvert
{
   public static void main (String[] args)
   {
      int base;        // the new base
      int base10Num;   // the number in base 10
      int maxNumber;   // the maximum number that will fit
                       // in 4 digits in the new base

      int place0;      // digit in the 1's (base^0) place
      int place1;      // digit in the base^1 place
      int place2;      // digit in the base^2 place
      int place3;      // digit in the base^3 place

      String baseBNum = new String (""); // the number in the new base
      Scanner scan = new Scanner(System.in);

      // read in the base 10 number and the base
      System.out.println();
      System.out.println ("Base Conversion Program");
      System.out.println();
      System.out.print ("Please enter a base (2 - 9): ");
      base = scan.nextInt();

      // Compute the maximum base 10 number that will fit in 4 digits
      // in the new base and tell the user what range the number they
      // want to convert must be in

      System.out.print ("Please enter a base 10 number to convert: ");
      base10Num = scan.nextInt();


      // Do the conversion (see notes in lab)

      // Print the result (see notes in lab)

    }
}
```