

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.

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?

3. Save the program above as Circle2 and start over with the directions below to modify program...

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 its second value, make it be twice the original value.
- Compile and run your program. Does your result from above hold?

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.

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 3 for 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)

    }
}

```