Lab 2: Data and Expressions

Welcome back!

Now that we've installed Java and Eclipse and familiarized ourselves with our development environment, we can focus on learning how to program. Today we'd like you to:

1. **Open Eclipse**. It's a good idea to keep all your projects in the same workspace. Think of a workspace as a shelf, and each project is a binder of stuff on the shelf.
2. **Create a new Java project**. Call it Comp1510Lab02*LastNameFirstInitial*. For example, if your name were Fred Bloggs, you would name the project **Comp1510Lab02BloggsF**.
3. **Complete the following tasks**. Remember you can right-click the src file in your lab 2 project to quickly create a new Java class.
4. When you have completed the exercises, **hand them in** or **show them to your lab instructor**. Be prepared to answer some questions about your code and the choices you made.
5. Although you may cut and paste from the lab document, it will help you learn if you **type in the code by hand**. The tactile effort of typing in the code will help reinforce the Java syntax.

What will you DO in this lab?

In this lab, you will practice:

1. Using String literals and String concatenation
2. Declaring and assigning values to variables
3. Performing computations using primitive data types
4. Converting data from one primitive type to another
5. Accepting input from the user.

Table of Contents

1. Prelab Questions

1. Can you explain the difference between the following three lines of code to your lab instructor?

```
a. int x;
b. int x = 3;
c. x = 3;
```

2. Given the declarations below, find the result of each expression:

```
int a = 3;
int b = 9;
int c = 7;
double w = 12.9;
double 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

2. Circle

Let's create a program that calculates some important values for a Circle.

1. Create a new class called **Circle** inside package ca.bcit.comp1510.lab02:
   a. Include a **Javadoc** comment at the top of the class. The Javadoc comment should contain:
      i. The name of the class and a (very) short description
      ii. An @author tag followed by your name
      iii. An @version tag followed by the version number.
   b. Include a **main method** inside the class definition. Remember the main method is what gets executed first when we start our program. Include a Javadoc comment for the main method. The Javadoc comment should contain:
      i. A description of the method. In this case, "Drives the program." is a good idea.
      ii. An @param tag followed by args. We will learn more about parameters and arguments in a few weeks. For now, you can just write @param followed by args followed by unused.

2. A Circle isn't a Circle without some PI. Declare a **constant called PI** inside the main method and assign the value 3.14159. What data type should you use? How do we know it's a constant? Does it compile? If not, fix the error.
3. Every Circle has a radius. Declare a **variable called radius** of type double inside the main method. Don't give it a value yet. We will ask the user to provide one.
4. Let's use a **Scanner object** to ask the user to enter a radius. Remember to import the Scanner class at the top of the class (after the package) statement. You can do this by adding this line of code:
   ```
   import java.util.Scanner;
   ```
5. Declare and initialize a Scanner object that reads from the keyboard inside the main method. You can do this by adding this line of code:
   ```
   Scanner myScanner = new Scanner(System.in);
   ```
6. Print a message to the screen that asks the user to enter a radius value for the Circle. We call this a **prompt**.
7. Use the scanner object to **acquire the user's input** and assign it to the radius variable. The scanner object is waiting for the user to enter a double for the radius, so you can use the nextDouble() method, like this:
   ```
   radius = myScanner.nextDouble();
   ```
8. The formula for the circumference of a Circle is 2 x PI x radius. **Calculate the circumference** of the Circle and store the result in a new variable called circumference. Print the circumference for the user.
9. The formula for the area of a Circle is PI * radius * radius. **Calculate the area** of the Circle and store the result in a new variable called area. Print the area for the user.
10. Let's have some fun. Suppose we double the radius. What happens to the area, does it also double? What about the circumference, does it double?
    a. **Modify your program** so that when the user enters a value for the radius, a variable called doubleRadius stores double the value stored in radius.
    b. **Calculate** the area and circumference of a circle using the doubleRadius value and store these values in new variables (you can choose the names).
    c. **Use division** to determine how many times the area and circumference increase when the radius doubles. Print a useful message for the user.

3. A Table of Student Grades

Create a Java program called **Students** inside package ca.bcit.comp1510.lab02 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. Tabs are helpful here. Make sure you:

1. Print the **border** on the top using the slash and backslash characters
2. **Use tab characters** to align your columns and **use the + operator** for addition and for String concatenation
3. **Comment your code**. Your class needs a Javadoc comment and your main method needs a Javadoc comment.

Here is the format we'd like you to use (how it looks depends on your tab stops):

```
//////////////////\\\\\\\\\\\\\\\\\\
==          Student Points          ==
\\\\\\\\\\\\\\\\\\//////////////////
Name  Lab   Bonus Total
----  ---   ----- -----
Joe   43    7     50
William     50    8     58
Mary Sue    39    10    49
Peng  87    6     93
Kwon  99    0     99
```

4. Painting a Room

Let's write a program to determine how many cans of paint we need to paint a room. We will consider the length, width, and height of the room, and the number of coats we want to apply:

1. Create a Java program called **Paint** inside package ca.bcit.comp1510.lab02. Make sure you include a Javadoc comment for the class. The Paint class needs a main method. The main method needs a Javadoc comment, too.
2. Let's start by declaring a constant. Let's assume that a 4 litre (≈1 gallon) can of paint will cover 400 square feet. Declare a constant called COVERAGE to store this value as an int.
3. Import the Scanner class. You can do this the same way you imported it for the Circle class.
4. Declare and initialize a Scanner object. You can do this the same way you initialized it in the Circle class.
5. Prompt the user for the length of the room in feet. Store the result in a variable called length.
6. Prompt the user for the width of the room in feet. Store the result in a variable called width.
7. Prompt the user for the height of the room in feet. Store the result in a variable called height.
8. Prompt the user for the number of coats to enter. Store the result in a variable called coats.
9. Calculate the total surface area to be painted in square feet and store the result in a variable called surfaceArea. You can imagine the room is a cube, and we want to calculate the area of the 4 sides and the top.
10. Multiple surfaceArea by the number of coats to apply and store the result in a new variable called coverageNeeded.
11. Divide coverageNeeded by COVERAGE, the amount each can will cover, and store the result in a variable called cansOfPaintNeeded.
12. Print a message to the user telling them how many cans of paint they need to buy.

5. Base Conversion

On the next page you will find a partial program called BaseConvert.java:

1. Create a Java program called **BaseConvert** inside package ca.bcit.comp1510.lab02.
2. Delete everything inside the file Eclipse made
3. Replace the content you deleted with the partial program we have provided.

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 provided by the user.

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 $b^4 - 1$

We've provided you with a partial program on the next page. You can copy it to Eclipse and complete the calculations. Study the code first. Note the names of the variables. We have placed comments where you need to add lines of code.

When it's time to 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. We should use a variable of type String. Note near the bottom of the program a variable named baseBNumber 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.

6. You're done! Show your lab instructor your work.

If your instructor wants you to submit your work in the learning hub, export it into a Zip file in the following manner:

1. Right click the project in the Package Explorer window and select export…
2. In the export window that opens, under the General Folder, select Archive File and click Next
3. In the next window, your project should be selected. If not click it.

4. Click *Browse* after the "to archive file" box, enter in the name of a zip file (the same as your project name above with a zip extension, such as Comp1510Lab02BloggsF.zip if your name is Fred Bloggs) and select a folder to save it. Save should take you back to the archive file wizard with the full path to the save file filled in. Then click Finish to actually save it.
5. Submit the resulting export file as the instructor tells you.

```
package ca.bcit.comp1510.lab02;

import java.util.Scanner;

/**
 * BaseConvert.
 *
 * @author BCIT
 * @version 1.0
 */
public class BaseConvert {
    /**
     * Drives the program.
     *
     * @param args
     *            arguments
     */
    public static void main(String[] args) {
        int base10number; // the number in base 10
        int base; // the new base
        int maximumNumber = 0; // 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;
        int place2;
        int place3;

        Scanner scan = new Scanner(System.in);

        System.out.println("Base Conversion Program");
        System.out.print("Please enter a base (2-9): ");

        // Assign the user's input to the base variable

        // Calculate the correct value to store in maxNumber

        System.out.println("The max base 10 number to convert for that base is " +
maximumNumber);

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

        // Assign the user's input to the base10number variable

        // Do the conversion
        // 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

        // Print the result
        String baseBNumber = new String(""); // the number in the new base

        scan.close();
    }
```