Lab Exercises 1

Dr. Sarvar Abdullaev
s.abdullaev@inha.uz

September 17, 2019

This problem is intended to be solved in a closed-lab session with a teaching assistant or instructor. Students can also collaborate to complete lab exercises after lab session.

Important: After completing this lab exercise, zip all your source code (i.e. all you `.java` files) into a single file named exactly as your student ID (e.g. `u180023.zip`) and upload it to eClass before the end of this week. Note, you do not have to upload compiled Java bytecode or screenshot of your program's output.

1 Body Mass Index Calculator

Create a BMI calculator that reads the user's weight in kilograms and height in meters, then calculates and displays user's BMI. Note, user input should be read from console using `Scanner` object. Also, display the following information after user's BMI on the screen, so user can evaluate his/her category.

```
BMI VALUES
Underweight: less than 18.5
Normal:      between 18.5 and 24.9
Overweight:  between 25 and 29.9
Obese:       30 or greater
```

Figure 1: BMI categories

The formula for BMI is given as follows: $BMI = \frac{weight}{height^2}$

2 Invoice Class

Create a class called `Invoice` that a hardware store might use to represent an invoice for an item sold at the store. An `Invoice` should include four pieces of information as instance variables—a part number (type `String`), a part description (type `String`), a quantity of the item being purchased (type `int`) and a price per item (`double`). Your class should have a constructor that initializes the four instance variables. Provide a `set` and a `get` method for each instance variable. In addition, provide a method named `getInvoiceAmount` that calculates the invoice amount (i.e., multiplies the quantity by the price per item), then returns the amount as a `double` value. If the quantity is not positive, it should be set to 0. If the price per item is not positive, it should be set to 0.0. Write a test app named `InvoiceTest` that demonstrates class `Invoice`'s capabilities.

3 Sales Commission Calculator

A large company pays its salespeople on a commission basis.The salespeople receive $200 per week plus 9% of their gross sales for that week. For example, a salesperson who sells $5,000 worth of merchandise in a week receives $200 plus 9% of $5000, or a total of $650. You've been supplied with a list of the items sold by each salesperson. The values of these items are as follows:

Item	Value
1	239.99
2	139.75
3	35.49
4	350.89

Develop a Java application that inputs one salesperson's items sold for last week and calculates and displays that salesperson's earnings. There's no limit to the number of items that can be sold.

4 Find the Largest Number

The process of finding the largest value is used frequently in computer applications. For example, a program that determines the winner of a sales contest would input the number of units sold by each salesperson. The salesperson who sells the most units wins the contest. Write a Java application that inputs a series of 10 integers and determines and prints the largest integer. Your program should use at least the following three variables:

1. `counter`: A counter to count to 10 (i.e., to keep track of how many numbers have been input and to determine when all 10 numbers have been processed).
2. `number`: The integer most recently input by the user.
3. `largest`: The largest number found so far.

In the same class, write a function which finds 2 largest numbers. In the same class, write a function which finds smallest number.

5 Palindromes

A palindrome is a sequence of characters that reads the same backward as forward. For example, each of the following five-digit integers is a palindrome: 12321, 55555, 45554 and 11611. Write an application that reads in a five-digit integer and determines whether it's a palindrome. If the number is not five digits long, display an error message and allow the user to enter a new value.

6 Triangle Printing Program

Write an application that displays the following patterns separately, one below the other. Use for loops to generate the patterns. All asterisks (*) should be printed by a single statement of the form `System.out.print('*');` which causes the asterisks to print side by side. A statement of the form `System.out.println();` can be used to move to the next line. A statement of the form `System.out.print(' ');` can be used to display a space for the last two patterns. There should be no other output statements in the program. [Hint: The last two patterns require that each line begin with an appropriate number of blank spaces.]

```
(a)          (b)          (c)          (d)
*            **********   **********         *
**           *********     *********        **
***          ********      ********       ***
****         *******       *******      ****
*****        ******        ******     *****
******       *****         *****    ******
*******      ****          ****   *******
********     ***           ***  ********
*********    **            **  *********
**********   *             *  **********
```

Figure 2: Triangles

7 Approximating π

Calculate the value of π from the infinite series:

$$\pi = 4 * (1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} \dots)$$

Print a table that shows the value of π approximated by computing the first 200,000 terms of this series. How many terms do you have to use before you first get a value that begins with 3.14159?

8 Pythagorean Triples

Find integers a, b and c such that following equation $a^2 + b^2 = c^2$ is satisfied. All a, b and c should be less than 500. Found triples should be displayed in a tabular format. Use a triple-nested for loop that tries all possibilities. This method is an example of "brute-force" computing. You'll learn in more advanced computer science courses that for many interesting problems there's no known algorithmic approach other than using sheer brute force.