CSE114 Fall 2023 : Assignment 2

Due: Sep 18, 2023 at 11:59 PM [KST]

Read This Right Away

For the due date of this assignment, don't go by the due date that you see on Brightspace unless you have set your timezone to KST. By default, Brightspace shows times in EST/EDT in the US.

Directions

- At the top of every file you submit, include the following information in a comment
  - Your first and last name
  - Your Stony Brook email address
- Please read carefully and follow the directions exactly for each problem.
- Your files, Java classes, and Java methods must be named as stated in the directions (including capitalization). Work that does not meet the specifications may not be graded.
- Your source code is expected to compile and run without errors. Source code that does not compile will have a heavy deduction (i.e. at least 50% off).
- You should create your program using a text editor (e.g. emacs, vim, atom, notepad++, etc).
- Your programs should be formatted in a way that is readable. In other words, indent appropriately, use informative names for variables, etc. If you are uncertain about what a readable style is, see the examples from class and textbook as a starting point for a reasonable coding style.

What Java Features to Use

For this assignment, you are *not* allowed to use more advanced features in Java than what we have studied ***at the time I post the assignment to Brightspace***. If you have a question about features you may use, please ask!

What to Submit

Combine all your .java files from the problems below into a single zip file named as indicated in the **Submission Instructions** section at the end of this assignment. Upload this file to **Brightspace** as described in that section.

Multiple submissions are allowed before the due date. Only the last submission will be graded.

Please do **not** submit `.class` files or any that I did not ask for.

Partial vs. Complete Solutions

Your programs should compile and run without errors, both compile-time and run-time errors! Please do not submit programs that do *not* even compile! Its better to submit a partial solution that compiles and runs as opposed to an almost complete solution that does not even compile. A program that does not compile even if it is very close to being perfect would only receive less than 50% maximum of the possible score for the program! This policy applies not only to this problem set but also to all the other ones in the remainder of the semester. I will not repeat this in each problem set though. So, please remember this.

Naming Conventions In Java And Programming Style In General

Please use these conventions as you establish your *programming style.* Programming professionals use these, too.

- **Names:** Choose informative names,
  - e.g. `hourlyRate` rather than `hr` for a variable name.
  - `CheckingAccount` rather than `CA` for a Java class name.
- **Class names:** We start a class name with an uppercase letter as I have in my examples: `Hello` not `hello` or `HELLO`. For multi-word names you should capitalize the first letter of each word, This is called 'Pascal' case.
  - e.g. `UndergraduateStudent`, `GraduateStudent`, `CheckingAccount`
- **Variable names:** We start a variable name with a lowercase letter. This is called 'Camel' case.
  - e.g. `count`, `hourlyRate`, averageMonthlyCost
- **Method names:** Naming convention for method names is the same as that for variable names.
- **Use of white space:** Adopt a good indentation style using white spaces and be consistent throughout to make your program more readable. Most text editors (like emacs and vim) should do the indentation automatically for you. If this is not the case, see me and I can help you configure your setup.

I will not repeat these directions in each assignment, but you should follow this style throughout the semester.

Introduction

This assignment will give you practice in the use of methods with parameters, Reading from the console. Printing to the console. Performing basic mathematical calculations.

There are 4 problems wit hthe following point values (totaling 25 points):

Problem 1 : BMI – 5 Points
Problem 2 : Distance – 8 Points
Problem 3 : Param – 5 Points
Problem 4 : CornTower – 7 Points

Problem 1 (5 Points)

Write a program BMI in the file BMI.java.

Your body mass index (BMI) is a measure of weight vs height. It is not an absolute measure of health as it doesn't account for body composition, muscle mass, and distribution of fat. However, in general, a lower BMI is an indicator of better overall health.

Some categories of BMI are:

- Underweight: <18.5
- Normal weight: 18.5-24.9
- Overweight: 25-29.9
- Obese: >30

The calculation for this indicator is the person's weight (in kg) divided by their height (in meters) squared.

Your application will ask the user for their weight in kg and their height in meters. It will compute the BMI and report it back to the console.

Use System.out.printf() to make sure the output BMI values are limited to 2 decimal places.

Example output:

```
java BMI
Enter the person's weight in kg: 90
Enter the person's height in meters: 1.5
BMI:    40.00
java BMI
Enter the person's weight in kg: 80
Enter the person's height in meters: 1.65
BMI:    29.38
java BMI
Enter the person's weight in kg: 70
Enter the person's height in meters: 1.75
BMI:    22.86
```

Hand in BMI.java.

Problem 2 (8 Points)

Write a Java class named Distance in a file named Distance.java that sums two distance values as follows.

Distances are measured in fitbits, qubits, and pings.

1 fitbit = 6 qubits

1 qubit = 20 pings

Your application will read separate values (integers) for each unit for each of the two distance measures. You must then sum the two distance values accounting for overflow (i.e. 18 + 8 pings = 1 qubit, 6 pings).

Print the two distances on separate lines (with unit names) and finally, print the sum on a 3rd line. See the example output below.

Prompt the user for each value (first fitbits, qubits, and pings, then second fitbits, qubits and pings.)

Note: You can convert the measures entirely into 'pings, add the two values for total pings, and then convert the difference (in pings) back into fitbits, qubits and pings before printing the result.

Finally, print the three distance values with some annotations ('fitbits', 'qubits', 'pings') so that the result will be meaningful to the user.

Example output:

First distance: Enter Fitbits: 3
First distance: Enter Qubits: 4
First distance: Enter Pings: 18
Second distance: Enter Fitbits: 4
Second distance: Enter Qubits: 2
Second distance: Enter Pings: 8
3 fitbits, 4 qubits, 18 pings +
4 fitbits, 2 qubits, 8 pings =
8 fitbits, 1 qubits, 6 pings.

Hand in Distance.java

Problem 3 (5 Points)

The point of this exercise is to help you understand how to write and invoke methods that take parameters.

Create a class named Param in a file named Param.java and do the following:\

- Introduce a method named pars that takes five formal parameters: an int, a double, another double, a boolean, and a String in that order. This method then prints those values to the standard output device (screen) one on each line with some annotations that describe what your function is printing. The method does not return anything useful, which means the return type of the function is to be **void**.
- Introduce a main method that establishes five values that would be needed to call pars and actually calls it. The four values that you will use are a year that went particularly well for you, the current 15-year fixed mortgage rate in the US (obtained from any credible source), the current currency exchange rate in number of Korean Won for a US dollar, whether the statement (Today is Sunday.) is true or not, and the name of a country you want to visit.

These values are to be assigned at compile-time. That is, they will be hard-coded in he declarations. In other words, you will not be reading the values from keyboard at run time (so you do not need a Scanner). Unless I specifically asked that you read input from keyboard, you are not expected to do so here as well as in other problems in this problem set and future problem sets. Compile your program and run it to make sure that it works without any compile-time or run-time errors.

Here is sample output:

java Param
Year that went well:: 1982
Fixed 15 year Mortgage Rate: 6.55
KRW to USD: 1289.72
Today is Sunday: false
Country I want to visit: Italy

Hand in Param.java.

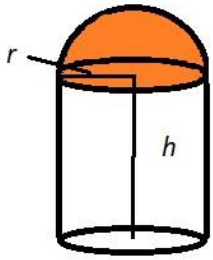
Problem 4 (7 Points)

Write a class called **CornTower** in a file named **CornTower.java** . This will compute the volume of a corn 'elevator'. As you can see in the diagram below, the structure used to store corn is a tall cylinder with a dome on the top. Thus, the total volume of corn held is the volume of the hemisphere plus the volume of the cylinder.

The following diagram shows a corn elevator. Imgine as an example that the height of the elevator (cylinder) is 40 meters and the radius of the cylinder (and dome) is 15 meters. The computed volume should be 35325 cubic meters.

Your program must prompt for the two values, which are taken from the keyboard *as integers*:

- the hemisphere/cylinder radius in meters, and
- the cylinder height in meters



The hemisphere (orange) and the cylinder (white) have the same radius. The program should report the total volume of the corn tower in cubic meters. Use 3.14 as the value of π rather than the built-in constant Math.PI for this assignment. The value's output should be formatted and limited to 2 decimal places.

Here is a sample run of the program, showing how input and output must be formatted:

Enter radius of Dome and cylinder (in meters): 18
Enter height of cylinder (in meters): 80
Volume of elevator (in cubic meters): 93597.12

Needed formulas:

cylinderVolume = $\pi r^2 h$

sphereVolume = $4\pi r^3 / 3$

Submission Instructions

Please follow this procedure for submission:

1. Place the deliverable source files (BMI.java, Distance.java, Param.java, and CornTower.java) into a folder by themselves. The folder's name should be CSE114_PS2_<yourname>_<yourid>. So if your name is Alice Kim and your id is 12345678, the folder should be named 'CSE114_PS2_AliceKim_12345678.

2. Compress the folder and submit the zip file.
    a. On windows, do this by pressing the right-mouse button while hovering over the folder. Select 'Send to -> Compressed (zipped) folder'. The compressed folder will have the same name with a .zip extension. You will upload that file to the Brightspace dropbox for **Assignment2**
    b. On mac, move the mouse over the folder then right-click (or for single button mouse, use Control-click) and select **Compress**. There should now be a file with the same name and a .zip extension. You will upload that file to the Blackboard dropbox for **Assignment2.**
3. Navigate to the course blackboard site. Click **Assignments** in the left column menu. Click **Assignment2** in the content area. Under **ASSIGNMENT SUBMISSION**, click **Browse My Computer** next to **Attach Files.** Find the zip file and click it to upload it to the web site.