Homework 9: Participant in a Study

Due: 5:00pm, Friday, December 2nd

Set Up
- Open your home directory (double click on your cs8fxx icon; 2nd icon from the top) and create a new directory in your home directory called hw9 (all lowercase).
- Copy the folder bookClasses from your Desktop into the hw9 directory. Right click on bookClasses > Copy > Right click on hw9 > Paste Into Folder.
- When you work in Dr. Java be sure you are editing files in the hw9/bookClasses directory.

Problems

1. Participant

In this assignment you will create a Participant Class. The Participant class so created will contain some private instance variables, some getXXX() methods, some setXXX() methods and some functional methods, i.e. methods which provide some functionality. In the context of this homework, an object of the Participant class represents a person taking part in a study, with a certain height, weight and name. In addition, it will have methods to return a string representation of the object and its BMI (Body-mass Index).

Once you have written the code for the Participant class, you will go ahead and test it out with a ParticipantTester class (which will be provided to you free of charge). This ParticipantTester class will contain the main() method and from within its main() method, it will instantiate objects of the Participant class and call its methods. For each call it makes, it will have expectations based on what the method is supposed to do. If your code fulfills those expectations, then your test case will pass. The goal of this homework is to ensure that *all* your test cases pass. Every successful test case will contribute towards your grade and every failed test case will reduce your grade.

You should read the descriptions of the Participant class methods below, but first open and read the tester application's main() method in detail before you write your code (so you understand how your methods will be used and what is expected out of them.)

The Participant class will have the following features.

- **Private Instance Variables:**
  - `name` - the name of the participant, this should be a **String**
  - `weight` - the weight of the participant, this should be an **int**
  - `height` - the height of the participant, this should be a **double**

- You are responsible for creating **three constructors** appropriate for the instance variables.
  - The first constructor should have no parameters and set the instance variables to default values. (default name: "Joe Bob", default weight: 1, default height: 1.0).
  - The second constructor should have only one parameter (a String to set the name), the other two instance variables should be set to default values.
  - The third construction should have an appropriate number of parameters to set every instance variable. If weight or height is negative, this constructor should not set weight or height, instead it should print "**Incorrect height, height must be positive. Height will not be set.**" or "**Incorrect weight, weight must be positive. Weight will not be set.**" and set the weight and height to their defaults.
  - Remember all the three constructors must not have any return type, not even void and should be declared simply public. If in doubt, check the constructors within the Sound or Picture class.

- You will create **accessor/mutator methods** for your instance variables.
  - For every instance variable - create two methods:
    - **setXXX()** which takes an appropriate parameter and sets the appropriate instance variable to the new value. (NOTE: If weight or height are negative, their set methods should not set the variables to the negative values, instead it should print "**Incorrect height, height must be positive. Height will not be updated.**" or "**Incorrect weight, weight must be positive. Weight will not be updated.**" and leave the weight and height unchanged. Here XXX is the name of the instance variable.

  **Please Note:** setXXX() methods will return void, but will accept an argument that will of the same type as of the instance variable it is setting.
Thus, for the instance variable ‘name’ which is of type String, the corresponding setXXX() method would be public void setName(String participantName)

- getXXX() which returns the value of the instance variable. This does the opposite of the setXXX() method.

Please Note: The return type of a getXXX() method should be the type of the object that it is returning and it should not take any arguments.

For example, for the instance variable ‘name’ of type String, it should be declared as

```
public String getName()
```

And once again, XXX is the instance variable name.

- Both the setXXX() and getXXX() should be declared public.
- The setXXX() and getXXX() methods for instance variable ‘name’ are shown above as examples. You need to figure out the setXXX() and getXXX() methods for the remaining two instance variables.

- You will also create a toString() method which has no parameters and returns a String. The String returned by toString() should match the output the ParticipantTester class expects. This should be declared as:

```
public String toString()
```

This method should return a String which will contain a summary of the Name, weight and height of the Participant object. Check the ParticipantTester class main() function to figure out exactly in what format it expects the output from the toString() method and then go ahead and implement your toString() method exactly like that.

Please Note: Your toString() method should provide the output *exactly* as expected in the main() method of the Participant class. Even a single space mismatch would render the test case failed. So pay attention to the testcase where it executes the toString() method of your Participant class.

- Lastly, you will create a getBMI() method which will use BMI calculation from http://en.wikipedia.org/wiki/Body_mass_index, specifically use the top "Imperial Units" formula. This should be declared as:

```
public double getBMI()
```

- With the above description, your Participant class outline would look something like:

```
public class Participant {
    private String name;
    private int weight;
    private double height;

    // 3 Constructors here

    // 3 getXXX() accessor methods here

    // 3 setXXX() mutator methods here

    // toString() functional method

    // getBMI functional method
}
```

- Testing your method (in an application): For this assignment, you MUST use our tester application. Incorrect execution of the tester application will result in a potentially low grade. You may view it on the assignment webpage when you click on ParticipantTester.java. We recommend you review the testing application prior to starting the assignment - it will help you in understanding what is required of you. Copy the file ~/.public/hw9/ParticipantTester.java into your hw9/bookClasses folder so that you may run it to check the correctness of your Participant class.

Please note: This tester does NOT test for all possible failure conditions and represents a basic testing method. More extensive automated testing is often applied in industry.
The output should match the output below exactly. If your implementation is incorrect, the tester will help you determine why. You can use the tester to determine default values, and mistakes you may have made.

Below is a sample output for a run, where all test cases have passed. This should be your goal. To help you determine whether you have reached this goal or not, the ParticipantTester class will print a summary of passed, failed and total run test cases after it has executed all its test cases. You will know your goal is reached when there are zero failed test cases and all are passed test cases, which in turn will ensure a full grade for the homework. If there are any failures, that will result in a grade that would be less than 100% of the maximum grade, which in turn would indicate that you should fix your Participant class code.

> java ParticipantTester
Creating a default participant

Testing correct default name using the getName method
[Correct]

Testing correct default weight using the getWeight method
[Correct]

Testing correct default height using the getHeight method
[Correct]

Testing the getBMI method on the first participant
[Correct]

Testing each set method using a default participant

Testing setName method using the getName method
[Correct]

Testing setWeight method using the getWeight method
[Correct]

Testing setHeight using the getHeight method
[Correct]

Testing the getBMI method on the first participant after changing weight and height
[Correct]

Default testing complete

**********************************************************************************************************************

Creating a new participant

Testing correct name using the getName method
[Correct]
Testing correct default weight using the getWeight method
[Correct]

Testing correct default height using the getHeight method
[Correct]

Testing the getBMI method on the second participant
[Correct]

Single parameter constructor testing complete

***************************************************************************

Creating a new participant

Testing correct name using the getName method
[Correct]

Testing correct weight using the getWeight method
[Correct]

Testing correct height using the getHeight method
[Correct]

Testing the getBMI method on the third participant
[Correct]

Three parameter constructor testing complete

***************************************************************************

Testing the toString method
[Correct]

toString testing complete

***************************************************************************

Creating a new participant with negative height

Should print "Incorrect height, height must be positive. Height will not be updated."
Incorrect height, height must be positive. Height will not be updated.

[Correct]

Testing setWeight method with negative weight

Should print "Incorrect weight, weight must be positive. Weight will not be updated."
Incorrect weight, weight must be positive. Weight will not be updated.

[Correct]
Creating a new participant with negative weight

Should print "Incorrect weight, weight must be positive. Weight will not be updated."
Incorrect weight, weight must be positive. Weight will not be updated.

[Correct]

Testing setHeight method with negative height

Should print " Incorrect height, height must be positive. Height will not be updated."
Incorrect height, height must be positive. Height will not be updated.

[Correct]

Negative height and weight testing complete.

Test cases run: 20
Test cases failed: 0
Test cases Passed: 20
*****Minimal testing passed*****

> 

**How to turnin your homework electronically**

Make sure the program works correctly in your cs8fxx login on the workstations in the labs in the basement of the CSE building when the workstations are booted under Linux (CentOS).

When you are ready to turn your program in, first open a terminal window by right clicking on the empty area of your computer’s desktop and select “Open a new Terminal.”

If you don't have one of these files present or have misspelled the name of one of these files, the turnin script will complain! Make sure all the files that you want to submit are in your `hw9/bookClasses` folder.

Type in the following command

turnin hw9

One file is required to be submitted for hw9:
  • Participant.java

You can verify your turnin with the following command

verify hw9

It is your responsibility to make sure you properly turned in your assignment.

**START EARLY!!!!**