COM1003 Java Programming, Spring Semester

Problem sheet 1: An introduction to OO

This is an <u>individual</u> assignment.
You are <u>not</u> allowed to work in groups and your code must be all <u>your own</u> work.
You may not use other peoples' code or ideas, e.g., taken from the Internet.
Any form of unfair means (plagiarism or collusion) will be treated as a serious academic offence and will be processed under our University Discipline Regulations. These might result in a loss of marks, a possible failure of the module or even expulsion from our University. For further details, please visit the UG Student Handbook:
<u>https://sites.google.com/sheffield.ac.uk/comughandbook/general-information/assessment/unfair-means</u>

Before you begin, you should have completed the week 1 practical sheet and be up-to-date with the week 1 self-assessment quiz available in MOLE.
You may ask demonstrators for help in understanding the problem, but this is intended to be an <u>individual assignment</u>, so you should not ask for help with the programming task.

Task / Requirements

You have been provided with an Eclipse project which contains the starting code for this problem sheet (classes `Distance`, `Steps`, `HeartRate`, and `MeasurementGenerator`).
This new project contains the `uk.ac.sheffield.com1003.problemsheet1` package. The classes `Distance`, `Steps`, and `HeartRate` are meant to help describing the measurements that fitness activity trackers are able to capture nowadays. One of the classes is called `Steps` and can be used to instantiate a `Steps` object by providing the number of steps an individual has walked. The second is called `HeartRate` and can be used to instantiate a `HeartRate` object by providing the heart rate of an individual. And the third is called `Distance` and can be used to instantiate a `Distance` object by providing the units (kilometres or miles) and the value of the distance walked by an individual. You will see that the methods for these classes are incomplete and that you will need to complete them so that you can get the requested functionality for the also incomplete class `MeasurementGenerator` which will have to:

1. Instantiate two `Steps`, two `HeartRate` and three `Distance` objects.
2. Display the number of steps of each object `Steps` on the console.
3. Display the heart rate of each object `HeartRate` on the console.
4. Display the distance of each object `Distance` on the console indicating the units.
5. Change the units of each `Distance` object (e.g. from kilometres to miles, and vice versa), and display their value in the correct units.

Testing your classes

You have been provided with 3 additional test classes (`TestDistance`, `TestSteps`, and `TestHeartRate`). They are meant to help you assess that your solution is 'correct' and that you are working in the right direction. We have not yet covered the topic of Unit testing in Java but your submitted code will need to successfully pass the tests provided in these 3 classes. The battery of tests provided doesn't fully test your solution, so even if your solution passes these tests, you need to be sure that your solution is actually fully correct, as during marking, other tests will be run on your code. Please, watch the supporting video showing how to verify that your code has passed all the tests in `TestDistance`, `TestSteps`, and `TestHeartRate`.

Recommended approach

- Study the three classes provided: `Steps`, `HeartRate` and `Distance`, and then complete the methods in comments, such as `getValue`, `setValue`. Each class has its own methods that need to be implemented. Read very carefully the comments provided in those classes as they will specify the desired functionality. Please, ask yourself why in the class `Distance` the methods `convertToMiles` and `convertToKilometres` are defined as `static`. Make sure that you use the constants that are available to you to perform those unit conversions (`KMS_PER_MILE` and `MILES_PER_KM`). Please, pay also attention at the enumeration `DistanceUnit` provided. You are also meant to use it.
- Complete the class `MeasurementGenerator` (you have been provided already with an empty class), gradually adding code to create the requested number of `Steps`, `HeartRate` and `Distance` objects, displaying the information about each of them, and then displaying the new values after either changing their values or units.
- Watch the video about how to run the tests in the provided test classes (`TestDistance`, `TestSteps`, and `TestHeartRate`) and confirm that your code passes all those tests.
- And last… once you are done and have uploaded your solution… ask yourself… was this the best implementation for the task at hand? By the submission deadline, you will have already seen inheritance… so… should inheritance have been handy? How?

Coding tips

Your code should be written using a good coding style, and you should note that ***readability*** is very important. The Java guidelines used by Google (https://google.github.io/styleguide/javaguide.html) are a good guide. In particular, you should:

- Use comments, but only when required (i.e. not excessively, but you should have a JavaDoc comment block at the head of each class).
- Use one code statement per line.
- Code your methods with as few lines as possible, and as many lines as needed. Ideally a complete method should be visible in a code editor without scrolling (see *Clean Cod*e for a longer discussion).
- Adhere to naming conventions; class names in *UpperCamelCase*, method names in *lowerCamelCase*, constant names in *CONSTANT_CASE*, and variable names in *lowerCamelCase*.
- Choose sensible, self-documenting, and descriptive names for all variables, methods, and classes.
- Use indentation consistently with either 2 or 4 whitespaces per indent (use whitespace because tabs can be interpreted differently on different operating systems).
- Before your code is checked, ask yourself whether the code is understandable by someone marking it.

Assessment and hand-in procedure

This problem sheet is worth 10% of your mark for the spring semester. How many marks you get depend on submitting a solution that:

- Implements the 5 requirements provided in the section 'Task/Requirements'.
- Passes all the tests provided in the classes `TestDistance`, `TestSteps`, and `TestHeartRate` (these classes should not be modified).

You must submit your code by the deadline (Friday 26th February, 3pm) <u>or earlier</u>. Follow the guidance about how to perform your upload in Blackboard. Start working on this problem sheet <u>as soon as possible</u> so that you can get help from the demonstrators or the lecturer during the labs. Please, use the discussion board dedicated to 'Problem sheet 1' to ask your questions, without sharing your solution. Late submissions will attract standard late penalties.

Maria-Cruz Villa-Uriol, February 2021