CSE 2102: Introduction to Software Engineering

Objects & Classes

Assigned: October 5, 2022, Due: October 19, 2022

Problem A

Consider a class `MotorBoat` that represents motorboats. A motorboat contains the following attributes (all the attributes can accept real values):

- The capacity of the fuel tank.
- The amount of fuel in the tank.
- The maximum speed of the boat.
- The current speed of the boat.
- The efficiency of the boat's motor.
- The distance traveled so far.

Implement the class according to the following specifications:

- A constructor that creates an instance of the motorboat. The constructor accepts input values for all the parameters.
- A method to change the speed of the boat.
- A method to refuel the boat with some amount of fuel. This method should add more fuel to the existing fuel amount.
- A method to return the amount of fuel in the tank.
- A method to return the distance traveled so far.
- A method to operate the boat. This method accepts the amount of time for which the boat is to be operated as input. It then computes the total amount of fuel necessary to operate the boat for that time as:

  $$\text{fuelBurn} = (\text{current speed})^2 \times \text{efficiency} \times \text{time})/100$$

  Division by 100 accounts for efficiency expressed as a percentage. If the amount of fuel necessary to operate the boat is lower than the amount of fuel in the tank, it prints the message that the boat is operating. The method then reduces the amount of fuel in the tank by the fuel burnt in the operation of the motorboat. It calculates the distance traveled during the specified time as:

  $$\text{distance} = (\text{current speed}) \times \text{time}$$

  It updates the distance traveled so far with this computed distance.

  If the amount of the fuel in the tank is not sufficient to operate the motorboat, it prints an error message and then it exits.

Write a test client `TestMotorBoat.java` to the test the class. The test client should prompt the user to input:

- Capacity of the fuel tank.
- Amount of fuel in the tank.
- Maximum speed of the boat.
- Current speed of the boat.

- Efficiency of the boat's motor.
- Total distance traveled so far.

The test client should then create an instance of the motorboat by calling the constructor. Once the motorboat object is created, the test client should return the amount of fuel in the tank. The client should then ask the user if the user would like to add more fuel. If the user indicates that they wish to add more fuel, the user should be further prompted for the amount that the user wishes to add, and the fuel amount should be updated accordingly. The user should then be prompted for the time that the user would like to operate the motorboat. The client should also ask the user if the user wishes to change the speed of the motorboat. The test client should operate the motorboat (if there is sufficient fuel), and then print the updated distance traveled and the amount of remaining fuel. A sample interaction with the user can proceed as follows:

Input cmd:

```
java TestMotorBoat
```

Output:

Use Case #1:

```
Enter the capacity of the fuel tank: 50
Enter the amount of fuel in the tank: 10
Enter the maximum speed: 20
Enter the current speed: 4
Enter the efficiency (percentage): 80
Enter the total distance traveled so far: 100

Fuel Amount: 10.000000
Would you like to add more fuel (y/n): y
How much fuel would you like to add: 10
Fuel amount: 20.000000
How long would you like to operate the boat (hours): 1

Operating the motorboat

Remaining fuel: 7.200000
Updated distance: 104.000000
```

Use Case #2:

```
Enter the capacity of the fuel tank: 50
Enter the amount of fuel in the tank: 10
Enter the maximum speed: 20
Enter the current speed: 4
Enter the efficiency (percentage): 80
Enter the total distance traveled so far: 100

Fuel Amount: 10.000000
Would you like to add more fuel (y/n): y
How much fuel would you like to add: 10
Fuel amount: 20.000000
How long would you like to operate the boat (hours): 2

Not enough fuel, cannot operate the motorboat

Remaining fuel: 20.000000
Updated distance: 100.000000
```

Problem B

In this problem, we will upgrade the solution implemented in Problem A with input error checking. Broadly, we will consider two kinds of errors in user provided input.

1. **Ranges of input parameters:** Check whether the value of each input parameter lies within a range that represents the physical entity that is being modeled within the context of the problem. These constraints include (but not limited to):
   a) Each input parameter must be greater than zero.
   b) Maximum speed of the motorboat should be no more than 50 knots.
   c) Efficiency, expressed as a percentage should be less than or equal to 100.
   d) Time, expressed in hours, is no longer than 8 hours.
   e) Maximum capacity of the tank is less than or equal to 250 gallons.
2. **Logical relationships among parameters:** Check whether the logical relationship among the parameters holds. These constraints include:
   a) Current speed of the motorboat is lower than the maximum speed.
   b) Amount of fuel is lower than the capacity of the tank.
   c) Total amount of fuel, after refueling is lower than the capacity of the tank.

Each of these errors must be checked in a while loop, that is, the user should be allowed to enter the input until they get it right.

Submission

The following deliverables must be submitted on HuskyCT by midnight on October 19, 2022.

a) Well-documented code.
b) For Problem A, at least 2 use cases that you used to test the code (submit in a separate document as a txt file or a word document, in the form of an interaction shown above).
c) For Problem B, include one error-checking use case from class 1 (whether the value of an input parameter lies within its specified range), and one error-checking use case from class 2 (checks whether the logical relationship holds between a pair of parameters). You

have the discretion to choose an input parameter for class 1, and the pair of parameters for class 2.

d) Please make sure that your code compiles, we will test your code offline with specific test cases (common to all).
e) Late submissions (without any legitimate excuse) will incur a penalty of 10% per day.