Due Feb 24 by 11:59p.m. **Points** 100 **Submitting** an external tool

Available Feb 12 at 12a.m. - Feb 24 at 11:59p.m.

Lab 3 -- Automatic Transmissions

Objective

To implement and thoroughly test a class using proper OO methodologies.

Description

A car's transmission controls how much the rotation of an engine affects the rotation of the axels By controlling the gear, the transmission prevents the car from speeding up too much when the engine is revved up. This is important for a smoothly

running car. In general there are two types of transmissions:

- Cars with a *manual transmission* require the driver to change gears as they are changing speed.
- Cars with *automatic transmission* let the car's computer choose how much rotation to transfer at certain speeds. Since cars cannot have negative speed (but can go as fast as their engine's allow), this choice of gear is a monotonically increasing function of speed.

In this lab, you will simulating this in a computer program. For our purposes, cars cannot have a negative speed, but can go as fast as needed.

What to do

Package name: `transmission`

Download **this provided `Transmission` interface (https://northeastern.instructure.com/courses/141347/files/19229909?wrap=1)** ⬇ **(https://northeastern.instructure.com/courses/141347/files/19229909/download?download_frd=1)** that represents a single car transmission. Your task is to implement this interface in a class called `AutomaticTransmission` which automatically determines the current gear by the current speed of the car. In addition to the methods in the interface, your implementation should:

- Include a constructor that takes 5 speed thresholds for the 6 possible gears in monotonically increasing order. The first threshold represents the speed that causes the transmission to change from gear 1 to gear 2, the second threshold represents the speed that causes the transmission to change from gear 2 to gear 3, and so forth. The constructor should ensure that the input values are valid and throw an `IllegalArgumentException` if any of the parameter values are not legal.
- Support the ability of a car to *idle* when its speed is 0. An idling car runs slowly while disconnected from a load; it is *out of gear* (i.e., gear = 0). Transmissions start in an *idle* state.
- Increasing or decreasing the speed should automatically adjust the gear if appropriate.
- Include a `toString` method that can be used to get the current state of the transmission (speed and gear), as follows:

```
Transmission (speed = 45, gear = 3)
Transmission (speed = 0, gear = 0)
```

Testing

Whenever you write a class, you should also write tests for that class that proves not only that your code CAN work but that it WILL ALWAYS work. Additionally, your tests should be sufficient to ***convince someone else that your code works correctly.***

What to Submit

1. Create a zip file that directly contains only your src/ and test/ folders. When you unzip the file, you must see only these two folders.
2. Submit the zip file on Gradescope.
3. Wait for a few minutes for the auto-grader's feedback to appear, and take action if needed.

Grading Criteria

This assignment will be assessed two ways (each with it's own grade):

1. Correctness and style will be assessed via *automatic testing* on Gradescope (50%).
2. Implementation and completeness of your testing suite will be graded manually (50%).
    - What we care about: **specifics about your implementation.** Did you write a test suite of your own that would convince someone else that your code works correctly?

CS5004 Vancouver Spring 2023 Spring 2023

Course ID: 483016

Description

The course provides an intensive tour of class-based program design and the design of abstractions that support the design of reusable software and libraries. It reviews typical object-oriented concepts such as information hiding, encapsulation and various forms of polymorphism. It contrasts the use of inheritance and composition as dual techniques for software reuse. It provides a deeper understanding of object-oriented design using the use of graphical design notations such as UML and object-oriented design patterns. It also examines the relationship between algorithms and data structures, as well as basic techniques for analyzing algorithm complexity. Finally, it emphasizes on testing, specifically unit testing of components.

Name	Status	Released	Due (EST)
			5 days, 8 hours left
Homework 2 - Chess Pieces	Submitted	Feb 06 at 12:00PM	Feb 18 at 3:00AM
Lab 2 - Interfaces and Abstract Classes	Submitted	Jan 29 at 12:00AM	Feb 11 at 3:00AM