Assignment 1

The xModz car corporation have designed a revolutionary new modular car. You can design the car online and choose a Car Body, a particular Engine, and Wheel types. XModz have designed a simple software simulation in Java to demonstrate how different car/engine/wheel configurations work. Your assignment is to demonstrate this simulation.

For example, to demonstrate a particular configuration, you might write the following code. [*This is assuming that all of the relevant classes are coded correctly*.]

```
Car car = new Car("X7");

Engine engine = new Engine("DR9", 43);

car.add(engine);

Wheel wheel = new Wheel("Wichelin15", 15);

car.add(wheel);

car.setFuel(100);

System.out.printf("Current fuel: %.2f\n",car.getFuel());

car.drive();

car.printState();

car.setFuel(50);

System.out.printf("Current fuel: %.2f\n",car.getFuel());

car.drive();

car.printState();
```

This program should output how far a particular Car configuration can travel given a full tank of fuel (the above example tests for both 100 and 50 units [output image below]).

Some notes to think about. These are only rough ideas; you should make your own design decisions and justify them with comments in the code:

1. Firstly, we must consider what classes we need:
    a. Car
    b. Engine
    c. Wheel
    d. TestCar

    [*You can write stub code for these or adapt the code from the lectures*]

2. Now, for each class we must consider some of the fields that are required
    a. Car
        i. name – a name describing the car
        ii. distance – the distance required to be travelled
        iii. totalKm – the total number of kilometres completed
        iv. fuelLevel – the level of fuel remaining
            1. Design decision: Does this belong here or in the Engine class?
    b. Engine
        i. name – a name describing the engine
        ii. tpl – wheel turns per litre (fuel efficiency rating)
        iii. totalNumTurns – increased every time the turn() method is called
    c. Wheel
        i. radius – the radius of the wheel
        ii. name – a name describing the wheel
    d. TestCar
        i. This class will be used to create instances of your Car class
        ii. It will have a single method with the following definition:
            public static void main(String[] args)
        iii. In this method, you will demonstrate a particular Car configuration with code similar to that shown on the first page of this document.

3. We must also consider the relationships between Car, Engine and Wheel
    a. A car **has an** Engine; An Engine **has a** Wheel
        i. These are composition relationships. Do you recall from the lecture how composition relationships were coded?
        ii. While automobiles generally have four wheels, we are going to use a single Wheel in this simplified model. Let's consider it to be one of the front two drive wheels. The other is identical so we won't bother about modelling it.
    b. Given a tank of 100 litres of fuel, each Car can call on the Engine to turn the Wheel a certain number of times. The number of turns will be based on the Engine's wheel turns per litre rating (tpl).
        i. For each full turn, a wheel will travel the length of its circumference. If you remember your geometry, you will recall that the circumference of a circle is $2\pi * r$, where r is the radius of the circle.
        ii. This means that larger wheels in our simulation will travel further as they have a larger radius.

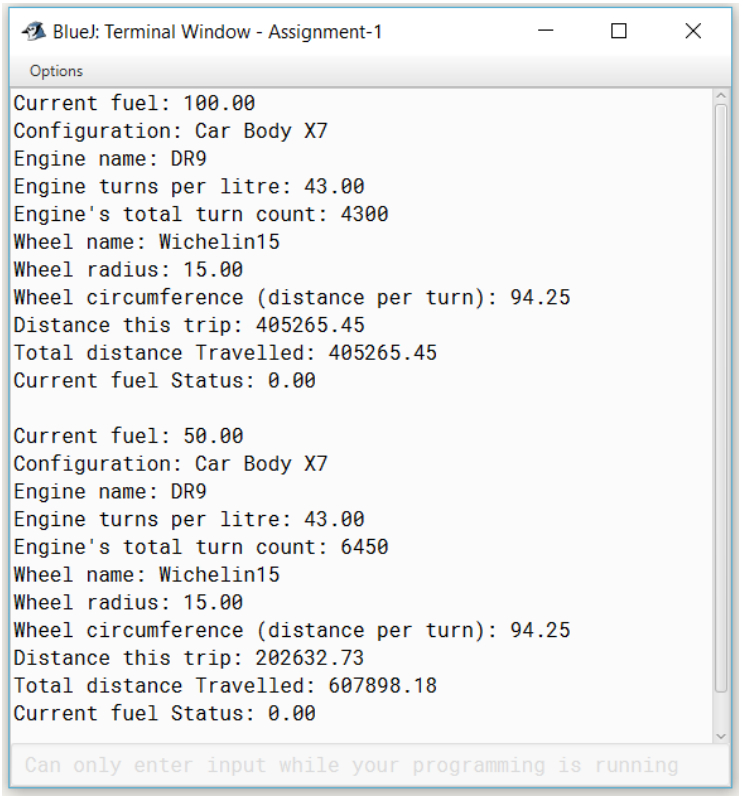
iii. In Java you can calculate this as:
double circumference = 2 * Math.PI * radius;
[*You should have the variable radius declared and initialised before using this piece of code.*]
You can add circumference as a field in the Wheel class.

c. An important thing to recall is that Engine does not know the radius of the Wheel, or the distance travelled. Your Wheel object should have a *turn()* method that returns the distance travelled (value stored in the circumference field) on every turn.

A key challenge for you in this assignment will be to decide on the methods that belong to each object and what each method should do/return. Here are some questions you can ask yourself:

- What value, if any, does the method return?
- Does the method functionality make sense in terms of the information that is held by the object? It should do.
- In order to deliver its own functionality, does the method need to call other method(s) from other object(s)?

Sample output from the above configuration code:



What to Submit:

1. A PDF with your code (copied and pasted; **not** a screenshot) and an outline of how the code works. It should contain screenshots of the output. If it does not work, explain your problems.
2. The individual .java files (not in a zip file). Do not include .class files

Note: You should include *meaningful* comments in the code to describe your design.

How this assignment will be graded:

We will be looking for the following features in your submission.

Basic: (these will get you a pass)

- Correct use of naming conventions (e.g Class names are capitalised, a lower-case letter for the first letter of method names, variables, fields etc.)
- Correct use of encapsulation: fields are private and are accessed via accessor/mutator methods only. Mutator method should be protected, if appropriate.
- Appropriate use of constructors to initialise fields.
- Appropriate use of any of the primitive types encountered so far: int, double, boolean
- Correct instantiation (creation) of a new object from a class and assigning a variable to point to the object.
- Your code compiles

More Advanced:

All basic requirements as well as:

- Appropriate use of composition.
- Appropriate creation of methods for each class - paying attention to the role the class is supposed to play and the data that its objects hold.
- Objects collaborating by calling each other's methods to achieve program functionality.
- Your code works by running the code example above (or very similar) and printing out the outputs from several Car/Engine/Wheel configurations (using clear, readable screenshots).

The deadline for submitting the assignment is **Friday the 30th of September @ 23:59pm**.

Plagiarism will be taken very seriously and **result in 0 marks for all parties involved**.

Any further instances will be reported to the plagiarism office of the School.

Only submit work that you have completed yourself.