ALGONQUIN COLLEGE

Lab #1

Activity: Vehicle Management System

Objective:

- Understand and implement the Singleton, Builder, and Simple Factory design patterns in Java.
- Apply these patterns in the context of a Vehicle Management System.

Scenario:

You are tasked with designing a system to manage different types of vehicles. Each vehicle has various attributes, and you need to ensure that the creation and management of these vehicles are done efficiently.

Tasks:

For all patterns, you are going to need an abstract class Vehicle with properties and methods like start(), stop(), etc. Feel free to add more. Only one main method is needed. Use Builder and Factory to instantiate vehicles to be added to the vehicle manager.

a. Singleton Pattern:

- **Objective:** Ensure that there is only one instance of a **VehicleManager** that manages all vehicles in the system.
- **Steps:**
    1. Create a **VehicleManager** class following the Singleton pattern.
    2. Implement a method in **VehicleManager** to add and retrieve vehicles (Hint: you can use an ArrayList to keep the Vehicles).
    3. Create a few instances of vehicles (using Builder and Factory) and add them to the **VehicleManager**.
    4. Demonstrate that the **VehicleManager** is a singleton by trying to create multiple instances and showing that they refer to the same object in the main method.

b. Builder Pattern:

- **Objective:** Implement a flexible way to construct different types of vehicles using the Builder pattern.
- **Steps:**
    1. Define an interface **VehicleBuilder** with methods for building different parts of a vehicle (e.g. buildEngine, buildWheels).
    2. Implement concrete builders (**CarBuilder**, **MotorcycleBuilder**, etc.) that implement **VehicleBuilder**.
    3. Create a **VehicleDirector** class that takes a builder and constructs a vehicle.
    4. Use the builder pattern to create instances of different vehicles.

c. Simple Factory Pattern:

- **Objective:** Implement a simple factory to create instances of vehicles based on user input.
- **Steps:**
    1. Create a **VehicleFactory** class with a method that takes a vehicle type (e.g., "car", "motorcycle") and returns an instance of the corresponding class.
    2. Implement concrete classes (**Car**, **Motorcycle**, etc.) based on **Vehicle**.
    3. Use the factory to create instances of vehicles based on user input.

d. JUnit

- **Objective:** Implement JUnit tests for **VehicleManager.java**, **VehicleDirector.java** and **VehicleFactory.java** classes.

Lab Deliverables

- This task constitutes 7.5% of your overall grade and due by **June 9, 2024 (Sunday), 11:59 PM.**
- Deliver a comprehensive coding solution.
- You code should include proper commenting and coding practice according to Java standards.
- Export your Java project (zip) that includes the implementation of the Singleton, Builder, and Simple Factory patterns + JUnit.
- You must demo your solution during the lab session and submit your code on **Brightspace**
- Violating academic integrity or missing the deadline will result in a grade of 0 for your submission.

Assessment Criteria:

- Correct implementation of Singleton, Builder, and Simple Factory patterns.
- Creativity and flexibility in handling different types of vehicles.
- Quality of code, including proper usage of design patterns and adherence to coding standards.
- Effectiveness and efficiency of the solution in managing and creating vehicles.

Individual Assignment

You may verbally discuss the general approach to solving this individual assignment with other students and that is the only extent of collaboration allowed for this assignment. You are not allowed to work together and you are not allowed to share or read each other's code/deliverables. If your code or any other deliverables for this assignment resemble with those of another student, you will be reported to the Academic Integrity Office for cheating/plagiarism investigation. Please refer to the Academic Integrity policy document (AA48) of the college at https://www.algonquincollege.com/policies/files/2021/09/AA48.pdf .

Statement on Generative Artificial Intelligence (AI)

You may consult any tools and information available external to the course but you must quote the reference in your submission. Failing to do so will result in Academic Integrity investigation. Furthermore, any content taken directly from these tools/information base and submitted will result in proportional reduction in grade. Any code obtained from generative AI tools such as ChatGPT cannot be submitted as your work and will be considered as plagiarized.