Project 7: Semester Project - Write Up

Project Name: Car Customizer
Team Members: Abiriham Bitew & Nathan Mukooba

Final State of System Statement:

Our project, a car customizer built on Colsole, has several incorporated features. To help users make educated decisions, we first imported a CSV file from GitHub and performed parsing operations to display the retrieved information in the terminal. Data on 2020 car makes are included in the CSV file; the parser has divided the data into various groups according to model make and body type.

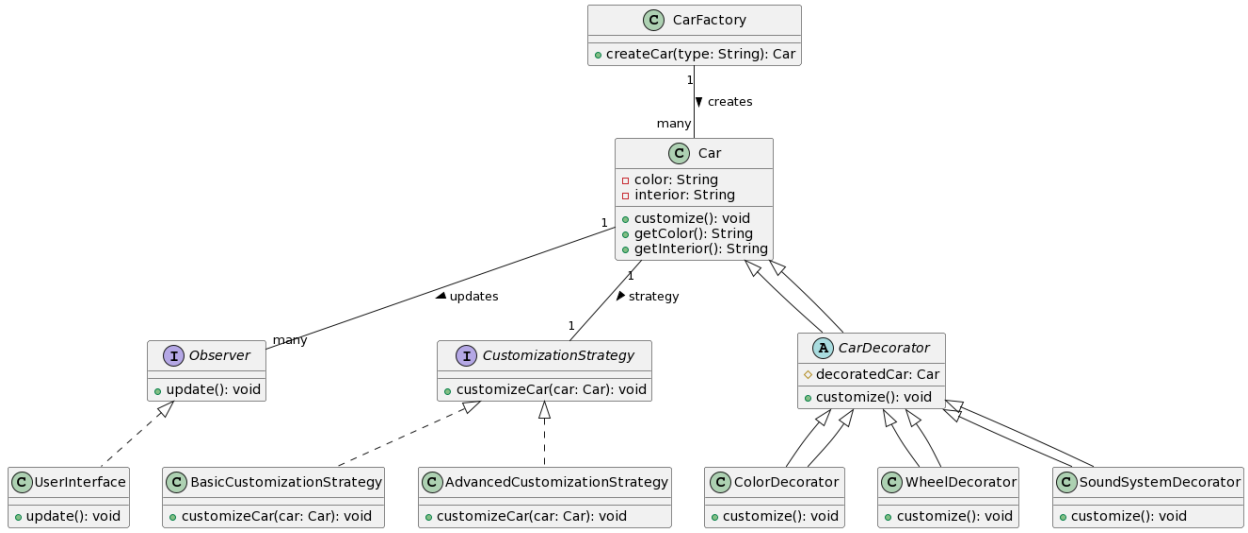
A significant benefit that gives users a great deal of customization possibilities is the ability to customize the car's interior, exterior, wheels, and sound system. Furthermore, a Strategy pattern was included to determine the cost of all the options and give consumers an overall sum for all the customizations they select.

Finally, to make the process of creating a text file easier, an Observer pattern was put into place. This file acts as a repository for the wide range of vehicles that users have personalized, listing the related expenses and options that have been chosen. This gives consumers a thorough history of their customization efforts.

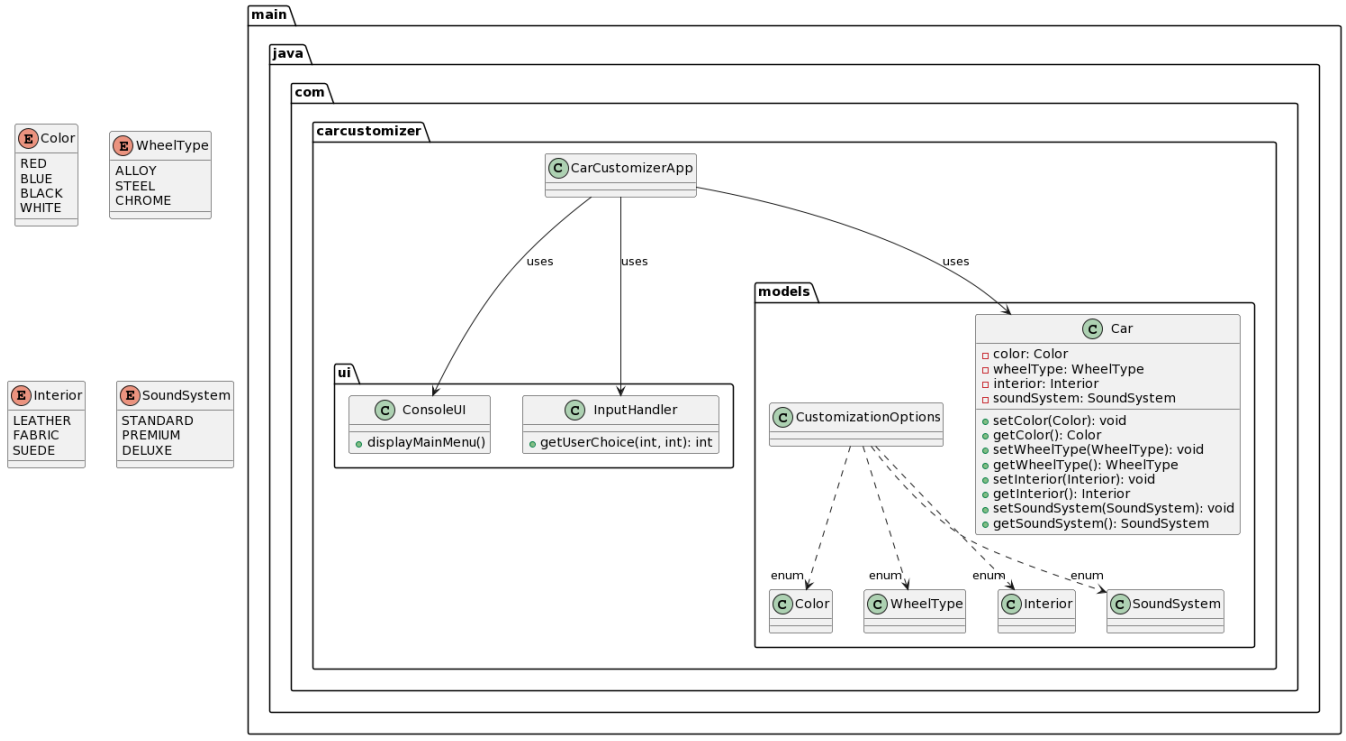
Our original plans for Project 6 involved developing a web application, but we ultimately opted to develop a console version instead. The ability to visually inspect the car was one of the elements from Project 6 that we were unable to complete. We attempted to use a third-party program or possibly an ASCII representation to display the actual car they created, but we were unable to accomplish that for this project. We were able to add more patterns and make the parser and text file function properly from project 6 to project 7.

Final Class Diagram and Comparison Statement:
UML Diagram creating a website: https://www.planttext.com/

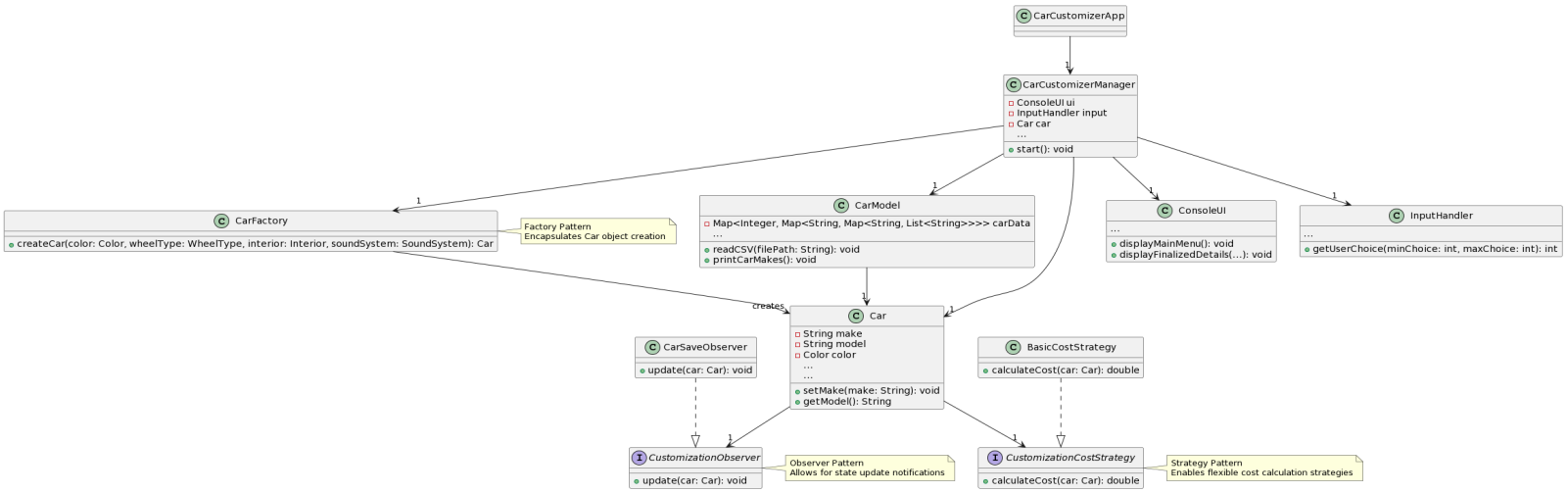
Project 5:



Project 6:



Project 7:



Key Changes:

- The main change from Project 6 and 7 is the incorporation of design patterns in the new system. Specifically, three patterns have been introduced: Factory Pattern, Observer Pattern, and Strategy Pattern. All these play a significant role in controlling what the system does and how it operates. We used the Factory pattern to encapsulate the creation of the Car objects. We used the Observer Pattern to allow the system to save the car configuration and its current state. Lastly, we used the Strategy pattern to calculate the cost of each option that the user can choose. From Project 5 the main changes are now that we have more options such as the cost feature being added as a feature, and also the car model class. We used that to parse through the CSV file and output the options, which was not part of our initial idea when we made the UML for Project 5.

Third-Party code vs. Original code Statement:

Most of the code in this project is original the parts of the code that aren't are based on the console-based outputs we found in the ARCANE projects. As for frameworks and Libraries we stuck with what was within Java such as importing packages like "java.lang" and "java.util".

Past Arcane Projects:

- Project 1
- Project 2
- Project 3
- Project 4

Git Repository of US Car Model Data:

- Repo of US Car Model, Make, and Year

Useful Resources for Java:
Java Programming for Beginners – Full Course | FreeCodeCamp YouTube Channel
Java Console Application Tutorial | YouTube Channel

Statement on the OOAD process for your overall Semester Project:

1. **Requirement Gathering and Analysis:**

When making the project plan we had originally wanted to create a web-based Java application. However, due to some personal setbacks as well as the time constraints we were then under we had to redesign how we wanted CarCustomize to look. For a console-based application, resources were plenty and easy to find. The challenge arrived at how we would maintain a user-friendly interface to the user while maintaining OOP principles. This was easy enough due to the resources we used to build up the project

2. **Iterative Design and Development:**

Another aspect of this project especially as we worked in a group was the various approaches we used when implementing a feature to fulfill a requirement. For example when creating the menu and output of data for the cars we wanted the user to see. The iteration process to arrive at a solid interface was time-consuming. Often relying on good debugging and testing to achieve an interface we were satisfied with. We often used pair programming as a means to be both on the same page and not have extremely diverging ideals.

3. **Design Patterns and Modularity**

We utilized 3 Patterns to better meet the requirements of the project and that best suited the need for how we wanted CarCustomizer to turn out. These patterns were well defined when making the requirements so as a team we knew when to make needed changes and when. However, the approach towards implementation led to refactoring on numerous occasions.