Kevin Siraki

Professor Shroyer

CS/IS 137

16 December 2019

Final Project: Car Dealership

**Part 1:**

My final project for CS/IS 137 is a program that works as a database for the inventory of a car dealership. Using the program is outlined in these steps:

1. There are three text documents that must be put in the default directory for the code to work
   1. car.txt
   2. colors.txt
   3. inventory.txt

-car.txt and colors.txt are databases of cars and colors for user input validation when the user is picking a car and color. Inventory.txt is a file where the current inventory of cars can be written to if the user chooses (can have an inventory of cars either overwrite the current file or add an inventory to the existing file).

1. When the program runs, there will be a menu with the following options.
2. Show current inventory: shows the current inventory of cars in the dealership.
3. Add a car: allows you to choose between Sedan, Sports Car, Truck, or SUV.
   1. Each car has a special attribute. The sedan has a luxury option, the sports car can have a custom horsepower, the truck can have a custom bed length, and the SUV can have more than 5 seats. These special attributes come into play when determining the final price of the vehicle.
   2. Also, every car has these attributes in common which you must enter:
      1. Model
      2. Color
      3. Platform (awd/rwd/fwd)
      4. Manual or Automatic
      5. Number of Gears
      6. Number of Cylinders
      7. Year Made
      8. Number of Doors
      9. VIN number (omitted this from input because it must be 17 digits… may uncomment on lines 346-352 in Car.cpp).
      10. Mileage
      11. Average MPG
      12. Electric/Non Electric
      13. Amenities
      14. Initial Asking price (before taking the above features into account).
4. Remove a car: lets you pick a car number to remove. Will only remove it from the inventory during runtime, not from the actual file itself.
5. Add inventory to text file: APPENDS the inventory.txt file with the current inventory.
6. Overwrite text file w/ current inventory: OVERWRITES the inventory.txt file with the current inventory.
7. Clear the text file: erases the entire invetory.txt file.
8. View the inventory file: print the inventory in the file.
9. Quit the program
10. To summarize, this program works as a database/inventory of cars for a dealership and also decides the price of a vehicle based on its base asking price and the information the user provides about the car.

**Part 2:**

This project uses the following concepts from the class:

1. Files (both I/O)

2. Polymorphism

3. Inheritance

4. Composition

5. Operator Overloads

6. Pointers and Dynamic Memory Allocation (used this in one class as well)

7. Vectors

8. Abstract classes/pure virtual functions

1. Files are used to check if a user enters a valid car model/color and are also used to store the inventory of cars to a file.

2. Polymorphism is used since the 4 derived classes SUV, SportsCar, Sedan, and Truck share a member function for calculating their final price but the final price is calculated VERY differently for each type of car. Dynamic binding is used to print the information for each car by calling the calculateCost() function on a pointer of either a SUV, SportsCar, Truck, or Sedan.

3. PUBLIC Inheritance is used because SUV, Sedan, SportsCar, and Truck are all derived from the base class Car.

4. Composition is used a few times here in the project. First, every Car has a list of amenities that the user must choose. Moreover, the car class has a fileParser for checking the validity of the entered model and color based on the files car.txt and colors.txt. Lastly, the Car Dealership class has a vector of Car pointers (Car is abstract).

5. Operator overloads are used for >> and << operators. A Car can have all of its information entered by a user. This information can be outputted to either the console or a file. Moreover, a CarDealership can also be printed (basically printing a vector of cars).

6. I used dynamic memory allocation in main to create instances of Sedans, SportsCars, Trucks, or SUVs during runtime. I also was sure to free up the allocated memory in the vector of Car\*s at the end of the program as well. (Used delete and nullptr).

7. As stated above, the CarDealership class has a vector of car pointers. There is also a vector of previous user choices in the Amenities class that checks to see if the user has chosen an amenity more than once.

8. Car is an abstract class with a pure virtual function that is redefined in all of its derived classes. This is because Car is too general to be instantiated.

9. From out of class, I used For-each loops to iterate over vectors, file truncation to clear files, and atoi(char) to prevent crashes in the program if the user enters a string/character instead of an int/float.

**Part 3:**

In my proposal, I had said I was just going to make a base abstract class Vehicle with derived classes Sedan, Truck, and Tank. This was too easy so I added the following things that the professor recommended:

1. Files
2. CarDealerShip class with a vector of Car(\*)s.
3. Did not use exception handling but used atoi to prevent the program from crashing if characters are entered instead of ints.
4. Added an Amenities class, fileParser class for input validation, SUV class, SportsCar class, and several member variables/functions for each class.

Overall, I added quite a lot more to my final project than what I said in the proposal in order to make it more challenging.

**Part 4:**

To conclude, I believe my final project deserves a good grade due to the fact that I tried to incorporate everything that we learned this semester into the project and more. A couple of things I wish I could add are cascading, BINARY files, or try/catch exception handling. Other than that, the project uses almost everything we learned in CS/IS 137, from the basics in HW 1 to Polymorphism, I/O with Files, and Vectors. Moreover, the project has a total of 8 classes, 9 .cpp files, and 3 .txt files (20 files total). This is well over the minimum of 5 classes. Lastly, the use of atoi(char) made it that this project is not very susceptible to crashes if the user enters bad values such as a string instead of an integer. Some classes such as fileParser and Amenities can also be used for different projects, as can other classes, thus making the project reusable. Overall, I am content with my work and I believe that it does deserve a good grade.