Bozhidar Mindov

COS221a

Vladimir Georgiev

11 Dec. 2022

**Project Documentation**

**Car Dealership**

**Project idea:**

The idea project is to make a car dealership program. The aim of the program is to create different types of vehicle objects, using the implemented classes, place them in a collection and later sort this collection.

The objects of the classes are initialized while reading the data for them from a file.

The user can choose how the collection of items is sorted and later displayed. The collections of vehicles can be sorted by: total price, monthly insurance, and warranty. The sorting is done using the Merge Sort algorithm.

The user also has the option to look for a vehicle in the collection by typing the brand of the vehicle, or its model. Every match that exists in the collection will be displayed on screen. The matching is done using the Boyer-Moore pattern matching algorithm.

The program will have 8 classes in total. The functionality of the classes is later tested in a **main (driver) function**.

**Class Hierarchy**

Diagram

Description automatically generated

**Remark**: Each class is separated into a **header file** and a **cpp file.** There are 8 header files and 8 cpp files for each of the classes, as well as an additional cpp file, which contains the main function. (Total files: **17** (8 header files, 9 cpp files)).

1. **Vehicle** (abstract class) – the base class

**Remark:** This is the base class of the program, and it only contains pure virtual methods. This requires every class that inherits this one to make its own implementation of the methods.

enum FuelType { petrol, diesel, hybridDiesel, hybridPetrol, electric }; - this is an Enum that specifies the different fuel types that can exist in the classes. They can be: petrol, diesel, hybridDiesel, hybridPetrol or electric. The Enum is used in each of the classes that inherit the Vehicle one.

*Methods (public*)

* virtual int CalculateWarranty() = 0; - meant to calculate and return the ***years*** of warranty that a certain vehicle has, usually based on its fuel type and the type of vehicle.
* virtual double CalculateTotalPrice() = 0; - It will calculate and return the total price of a vehicle. It will be based on the type of the vehicle, its power, size and other properties, such as fuel type.
* virtual double CalculateInsurance() = 0; - It will calculate and return the price of the monthly insurance that will have to be paid for the vehicle. It will again be based on the type of the vehicle, its power, size, and other properties, such as fuel type.
* virtual void DisplayInfo() = 0; - meant to display information about each vehicle. For example, it will display their brand, model, total price, insurance per month, warranty, etc.
* virtual string getBrand() = 0; - meant to return the name of the vehicle’s brand. It is required and used primarily in the pattern matching method of the main function.
* virtual string getModel() = 0; - meant to return the name of the vehicle’s model. It is required and used primarily in the pattern matching method of the main function.

1. **Car** (inherits vehicle)

*Data members (private):*

* string model; - stores the model of the car
* string brand; - stores the brand of the car
* string color; - stores the color of the car
* int year; - stores the year of the car
* int horsePower; - stores the amount of horsepower the car has
* double price; - stores the price of the car
* double miles; - store the number of miles the car has
* double powerUnitSize; - stores the size of the Car’s power unit (in liters, ex: 1.7, or in Kilowatts, ex: 80)
* FuelType fuelType; - variable of type enum FuelType. Stores the fuel type of the car. Can be petrol, diesel, hybridDiesel, hybridPetrol or electric.
* bool used; - if ‘false’ – that means the car is new, if its ‘true’ – that means the car is used.

*Methods (public):*

* Car(); - a default constructor. If invoked, an object of the class will be created with default values.
* Car(string model, string brand, string color, int year, int horsePower, double price, double miles, double powerUnitSize, FuelType fuelType, bool used); - a constructor with parameters, which if invoked, initializes an object of the class. The constructor contains setters with validity checks in order to ensure that variables of type integer and double are not negative.
* int CalculateWarranty(); - this method computes the years of warranty that the car will have. The warranty is based on the fuel type of the car. The number of years of warranty is returned as type int.
* double CalculateTotalPrice(); - this method computes the total price of a car. The total price of each car depends on its fuel type, or whether it is used or not. The car’s total price is returned as type double.
* double CalculateInsurance(); - this method computes the monthly insurance that will have to be paid for each car. The monthly insurance depends on the fuel type of the car, whether it is used or not, and also on its year of production. The car’s monthly insurance is returned as type double.
* void DisplayInfo(); - displays information about the car (such as brand, model, year, total price, Insurance, Warranty, etc.)

**Remark**: Getters and setters are implemented for each data member of the class. The polymorphic methods string getBrand() and string getModel() are implemented as getters.

1. **SUV** (inherits Car)

The “SUV” class has access to the methods of the “Car” class, but also implements some of its own.

*Data members (private):*

* int numOfSeats; - stores the number of seats of the SUV

*Methods (public):*

* SUV(); - a default constructor. If invoked, an object of the class will be created with default values.
* SUV(string model, string brand, string color, int year, int horsePower, double price, double weight, double powerUnitSize, FuelType fuelType, bool used, int numOfSeats); - a constructor with parameters, which if invoked, initializes an object of the class. The constructor contains setters with validity checks in order to ensure that variables of type integer and double are not negative.
* int CalculateWarranty(); - this method computes the years of warranty that the SUV will have. The warranty is based on the fuel type of the SUV. The number of years of warranty is returned as type int.
* double CalculateTotalPrice(); - this method computes the total price of an SUV. The total price of each SUV depends on its fuel type, or whether it is used or not. The number of seats is also considered, because when an SUV has 7 or more seats, its price gets discounted. The SUV’s total price is returned as type double.
* double CalculateInsurance(); - this method computes the monthly insurance that will have to be paid for each SUV. The monthly insurance depends on the fuel type of the SUV, whether it is used or not, and also on its year of production. If the SUV has 7 or more seats, it gets a discount on its insurance payments. The SUV’s monthly insurance is returned as type double.
* void DisplayInfo(); - displays information about the SUV (such as brand, model, year, total price, Insurance, Warranty, etc.)

**Remark**: Getters and setters are implemented for each data member of the class. The polymorphic methods string getBrand() and string getModel() are implemented as getters, which are inherited from the Car class.

1. **Van** (inherits Car)

The “Van” class has access to the methods of the “Car” class, but also implements some of its own.

enum VanType {mini, micro, minibus, panel, step, box, fullSize, MPV }; - This

is an Enum which specifies the different types of vans. They can be: mini, micro, minibus, panel, step, box, fullSize or MPV.

*Data members (private):*

* VanType vanType; - stores the type of the van
* int numOfSeats; - stores the number of seats of the SUV

*Methods (public):*

* Van(); - a default constructor. If invoked, an object of the class will be created with default values.
* Van(string model, string brand, string color, int year, int horsePower, double price, double miles, double powerUnitSize, FuelType fuelType, bool used, VanType vanType, int numOfSeats); - a constructor with parameters, which if invoked, initializes an object of the class. The constructor contains setters with validity checks in order to ensure that variables of type integer and double are not negative.
* int CalculateWarranty(); - this method computes the years of warranty that the van will have. The warranty is based on the fuel type of the van. The number of years of warranty is returned as type int.
* double CalculateTotalPrice(); - this method computes the total price of a van. The total price of each van depends on its fuel type, or whether it is used or not. The number of seats is also considered, because when a van has 7 or more seats, its price gets discounted. The van’s total price is returned as type double.
* double CalculateInsurance(); - this method computes the monthly insurance that will have to be paid for each van. The monthly insurance depends on the fuel type of the van, whether it is used or not, and also on its year of production. If the van has 7 or more seats, it gets a discount on its insurance payments. If a van is of type micro, mini, box or step, a further discount is applied. The van’s monthly insurance is returned as type double.
* void DisplayInfo(); - displays information about the van (such as brand, model, year, total price, Insurance, warranty, etc.)

**Remark**: Getters and setters are implemented for each data member of the class. The polymorphic methods string getBrand() and string getModel() are implemented as getters, which are inherited from the Car class.

1. **Truck** (inherits Vehicle)

*Data members (private):*

* string model; - stores the model of the truck
* string brand; - stores the brand of the truck
* string color; - stores the color of the truck
* int year; - stores the year of the truck
* int horsePower; - stores the amount of horsepower the truck has
* double price; - stores the price of the truck
* double miles; - store the number of miles the truck has
* double powerUnitSize; - stores the size of the truck’s power unit (in liters, ex: 5.7)
* FuelType fuelType; - variable of type enum FuelType. Stores the fuel type of the truck. Can be petrol, diesel, hybridDiesel, hybridPetrol or electric.
* double weight – stores the weight of the truck in tons (ex: 9.5)
* bool used; - if ‘false’ – that means the truck is new, if its ‘true’ – that means the truck is used.
* double towingCappacity – stores the towing capacity that the truck has (in pounds, ex: 5500)

*Methods (public):*

* Truck(); - a default constructor. If invoked, an object of the class will be created with default values.
* Truck(string model, string brand, string color, int year, int horsePower, double price, double miles, double powerUnitSize, FuelType fuelType, double weight, bool used, double towingCappacity); - a constructor with parameters, which if invoked, initializes an object of the class. The constructor contains setters with validity checks in order to ensure that variables of type integer and double are not negative.
* int CalculateWarranty(); - this method computes the years of warranty that the truck will have. The warranty is based on the fuel type of the truck. The number of years of warranty is returned as type int.
* double CalculateTotalPrice(); - this method computes the total price of a truck. The total price of each truck depends on its fuel type, or whether it is used or not. The truck’s total price is returned as type double.
* double CalculateInsurance(); - this method computes the monthly insurance that will have to be paid for each truck. The monthly insurance depends mainly on the fuel type of the truck. Also, if the truck has a towing capacity of more than 7000, a discount on its insurance payment is applied. The truck’s monthly insurance is returned as type double.
* void DisplayInfo(); - displays information about the truck (such as brand, model, year, total price, Insurance, warranty, etc.)

**Remark**: Getters and setters are implemented for each data member of the class. The polymorphic methods string getBrand() and string getModel() are implemented as getters.

1. **PickupTruck** (inherits Truck)

The “PickupTruck” class has access to the methods of the “Truck” class, but also implements some of its own.

enum Use {offRoad, hauling, construction, personalUse}; - This

is an Enum which specifies the different uses that a pickup truck may have. They can be: offRoad, hauling, construction and personalUse.

*Data members (private):*

* int numOfSeats; - stores the number of seats of the pickup truck
* Use use; - stores the use of the pickup truck.

*Methods (public):*

* PickupTruck(); - a default constructor. If invoked, an object of the class will be created with default values.
* PickupTruck(string model, string brand, string color, int year, int horsePower, double price, double miles, double powerUnitSize, FuelType fuelType, double weight, bool used, double towingCappacity, int numOfSeats, Use use); - a constructor with parameters, which if invoked, initializes an object of the class. The constructor contains setters with validity checks in order to ensure that variables of type integer and double are not negative.
* int CalculateWarranty(); - this method computes the years of warranty that the pickup truck will have. The warranty is based on the fuel type of the pickup truck. The number of years of warranty is returned as type int.
* double CalculateTotalPrice(); - this method computes the total price of a pickup truck. The total price of each pickup truck depends on its fuel type, or whether it is used or not. The use of the pickup truck is also taken into account, because if the pickup truck is used for construction or hauling, its total price gets discounted. The number of seats is considered, too, because when a pickup truck has 7 or more seats, its price gets discounted further. The pickup truck’s total price is returned as type double.
* double CalculateInsurance(); - this method computes the monthly insurance that will have to be paid for each pickup truck. The monthly insurance depends mainly on the fuel type of the pickup truck. Also, if the pickup truck has a towing capacity of more than 7000, a discount on its insurance payment is applied. Various other discounts can be applied on the pickup truck’s insurance. For example, if it has 7 or more seats or is used for hauling or construction, discounts are applied to its monthly insurance price. The pickup truck’s monthly insurance is returned as type double.
* void DisplayInfo(); - displays information about the pickup truck (such as brand, model, year, total price, Insurance, warranty, etc.)

**Remark**: Getters and setters are implemented for each data member of the class. The polymorphic methods string getBrand() and string getModel() are implemented as getters, which are inherited from the Truck class.

1. **Motorcycle** (inherits Vehicle)

*Data members (private):*

* string model; - stores the model of the motorcycle
* string brand; - stores the brand of the motorcycle
* string color; - stores the color of the motorcycle
* int year; - stores the year of the motorcycle
* int horsePower; - stores the amount of horsepower the motorcycle has
* double price; - stores the price of the motorcycle
* double miles; - store the number of miles the motorcycle has
* double powerUnitSize; - stores the size of the motorcycle’s power unit (in liters, ex: 1.7, or in Kilowatts, ex: 80)
* FuelType fuelType; - variable of type enum FuelType. Stores the fuel type of the motorcycle. Can be petrol, diesel, hybridDiesel, hybridPetrol or electric.
* bool used; - if ‘false’ – that means the motorcycle is new, if its ‘true’ – that means the motorcycle is used.

*Methods (public):*

* Motorcycle(); - a default constructor. If invoked, an object of the class will be created with default values.
* Motorcycle(string model, string brand, string color, int year, int horsePower, double price, double miles, double powerUnitSize, FuelType fuelType, bool used); - a constructor with parameters, which if invoked, initializes an object of the class. The constructor contains setters with validity checks in order to ensure that variables of type integer and double are not negative.
* int CalculateWarranty(); - this method computes the years of warranty that the motorcycle will have. The warranty is based on the fuel type of the motorcycle. The number of years of warranty is returned as type int.
* double CalculateTotalPrice(); - this method computes the total price of a motorcycle. The total price of each motorcycle depends on its fuel type, or whether it is used or not. The motorcycle’s total price is returned as type double.
* double CalculateInsurance(); - this method computes the monthly insurance that will have to be paid for each motorcycle. The monthly insurance depends mainly on the fuel type of the motorcycle. The motorcycle’s monthly insurance is returned as type double.
* void DisplayInfo(); - displays information about the motorcycle (such as brand, model, year, total price, Insurance, warranty, etc.)

**Remark**: Getters and setters are implemented for each data member of the class. The polymorphic methods string getBrand() and string getModel() are implemented as getters.

1. **ElectricScooter**(inherits Motorcycle)

*Data members (private):*

* int batteryHealth – stores the battery health of the electric scooter (in percentages, ex: 100)

*Methods (public):*

* ElectricScooter (); - a default constructor. If invoked, an object of the class will be created with default values.
* ElectricScooter(string model, string brand, string color, int year, int horsePower, double price, double miles, double powerUnitSize, FuelType fuelType, bool used, int batteryHealth); - a constructor with parameters, which if invoked, initializes an object of the class. The constructor contains setters with validity checks in order to ensure that variables of type integer and double are not negative. The fuel type of the electric scooter is always set to electric.
* int CalculateWarranty(); - this method computes the years of warranty that the electric scooter will have. The number of years of warranty is returned as type int.
* double CalculateTotalPrice(); - this method computes the total price of an electric scooter. Each electric scooter comes with a 15% discount on its original price. If the scooter has less than 80 percent battery health, a further 30% discount is applied. The electric scooter’s total price is returned as type double.
* double CalculateInsurance(); - this method computes the monthly insurance that will have to be paid for each electric scooter. The electric scooter’s monthly insurance is returned as type double.
* void DisplayInfo(); - displays information about the electric scooter (such as brand, model, year, total price, Insurance, warranty, etc.)

**Remark**: Getters and setters are implemented for each data member of the class. The polymorphic methods string getBrand() and string getModel() are implemented as getters, which are inherited from the Motorcycle class.

**Source.cpp –** stores the main method, as well as the functions that are related from the reading of information from files, the merge sort algorithm and the Boyer-Moore pattern matching algorithms.

The file contains the following functions:

* bool match(string text, string pattern) – this function takes in a string that is a piece of text, and another string that is used as a pattern. The function returns true or false depending on whether the pattern is found somewhere in the text string.
* void mergeSort(vector<Vehicle\*> &items, string sortType) – This function takes a vector by reference and sorts it by the specified sortType, using the Merge sort algorithm. The sort types can be the following numbers:
  + 1 – sort the collection of vehicles by total price
  + 2 – sort the collection of vehicles by warranty
  + 3 – sort the collection of vehicles by monthly insurance
* void merge(vector<Vehicle\*> &items, vector< Vehicle \*> leftHalf, vector< Vehicle \*> rightHalf, string sortType) – This function represents the merging part of the Merge Sort algorithm. It is used inside the ‘mergeSort’ function above.
* FuelType checkFuelType(string fuelType) – This function takes in a string representation of a fuel type, and converts it to a variable of type Enum FuelType.
* Use checkPickupTruckUse(string use) - This function takes in a string representation of a use (used in the PickupTruck class), and converts it to a variable of type Enum Use.
* VanType checkVanType(string vanType) - This function takes in a string representation of a van type (used in the Van class), and converts it to a variable of type Enum VanType.
* bool stob(string str) – converts a string to Boolean (Example: “true” is converted to the Boolean true).
* vector<Vehicle\*> readVehicleDataFromFile(string filename) – this function takes in s filename and reads the data from that file. With that data it creates different vehicle objects. The first string of each line represents the type of the vehicle, and with it, the constructor of the corresponding class is invoked. The strings on each line must be separated by a comma with no spaces between different commas.
* int main() – The driver function

It contains the following steps:

1. A vector is created, which stores items of class Vehicle. The data for the vector is read from a text file.
2. Then the user is asked how they would like to see the items from the vector sorted – by total price, warranty or insurance.
3. After they select their choice, the vehicle collection is sorted and displayed on screen.
4. Finally, the user is asked if they want to look for a vehicle in the “database”. If they answer with “no” – the program will exit. If they select “yes”, they will be prompted to enter the model or the brand of the vehicle. With their input, the program will look at each vehicle and try to match the user’s input with their brand or model using the Boyer-Moore pattern matching algorithm. If matches occur, they will be displayed on screen, if not - the user will be notified that there were no matches found.