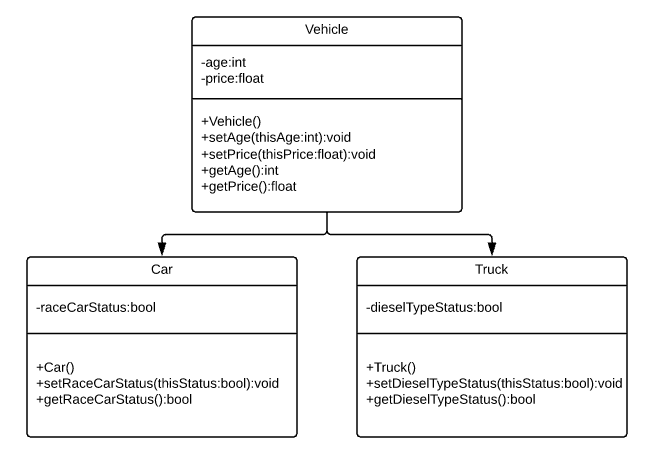
1. **Problem Statement**

Program has different and specific functions for a driver depending on the type of vehicle involved; Car or Truck. Driver will have ability to see and store data of one or more of these two types of vehicles.

1. **Requirements**
   1. **Assumptions**
      1. User enters integer value for age variable
      2. Driver owns Lamborghini Aventador car and/or Ford F-150 Pick Up Truck
      3. User selected appropriate option in Menu Selections, without giving out of bound inputs.
   2. **Specifications**
      1. Welcome message to the driver
      2. Store data
         1. Age of car and truck (default = 0)
         2. Price of car and truck (default = 0.0)
         3. True or false, is car a race car? (default = false)
         4. True or false, does truck have diesel type? (default = false)
      3. Manipulate data
         1. Get and set age of car and truck
         2. Get and set price of car and truck
         3. Get and set true or false whether or not car is a race car
         4. Get and set true or false whether or not truck is diesel type
      4. Inheritance
         1. Class Car and class Truck inherits all public attributes and behaviors of class Vehicle
      5. Thank you message to the driver
      6. Perform checks
         1. Only accept values for age more than what was previously stored
         2. Only accept values for price less than what was previously stored
2. **UML Class Design**



1. **Decomposition Diagram**

|  |  |  |
| --- | --- | --- |
| **Main** | | |
| **Input** | **Process** | **Output** |
| Age | Check if it is more than previously stored value.  Store it in the variable age | “Age of car/truck stored” |
| Price | Check if it is less than previously user-stored value. Store it in variable price. | “Price of car/truck stored” |
| Menu selection to get age | Make an appropriate print statement with variable age | Print the statement with age |
| Menu selection to get price | Make an appropriate print statement with variable print | Print the statement with price |
| Race car status | Set the passed in status to the variable raceCarStatus | “Race Car Status stored” |
| Menu selection to get race car status | Make an appropriate print statement with variable raceCarStatus | Print the statement with race car status |
| Diesel type status | Set the passed in status to the variable dieselTypeStatus | “Diesel Type Status stored” |
| Menu Selection of either owning a car or truck | Make an object and call functions of that particular class selected by user | Confirmation saying user selected either car or truck |

1. **Test Strategy**
   1. Valid Data
   2. Invalid Data
2. **Test Plan Version 1**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Valid | 1 | Age of car/truck is greater than previously stored value |  |  |  |  |
| Valid | 2 | Price of car/truck is less than previously stored value unless storing it 1st time |  |  |  |  |
| Valid | 3 | Price value is always positive |  |  |  |  |
| Valid | 4 | User enters corresponding number for choosing either true or false for race car status |  |  |  |  |
| Valid | 5 | User enters corresponding number for choosing either true or false for diesel type status |  |  |  |  |
| Valid | 6 | User enters corresponding number for choosing either car or truck |  |  |  |  |
| Invalid | 1 | Age of car/truck is more than previously stored value |  |  |  |  |
| Invalid | 2 | Price of car/truck is more than previously stored value unless storing it 1st time |  |  |  |  |
| Invalid | 3 | Price value is negative |  |  |  |  |
| Invalid | 4 | User enters number not corresponding to choosing either true or false for race car status |  |  |  |  |
| Invalid | 5 | User enters number not corresponding to choosing either true or false for race diesel type status |  |  |  |  |
| Invalid | 6 | User enters number not corresponding to choosing either Car or Truck |  |  |  |  |

1. **Initial Algorithm**
   1. In main function
      1. Make an instance for Car class and for Truck class
      2. Ask user is they have a Lamborghini Aventador (input 1) or Ford F-150 Pickup (input 2)
      3. If user inputs 1
         1. Ask user for car’s price and car’s age and store it in appropriate variables
         2. Do this and loop while user does not input 0 to end
            1. Let user select from a menu of

1. Change age

Ask user for car’s age and check if it is greater than previously stored value

If greater, then store it in age variable

Else loop until user selects correct age

2. Change price

Ask user for car’s price and check if it is smaller than previously stored value

If smaller, then store it in price variable

Else loop until user selects correct price

3. Change race car status

Ask user if the car is a race car.

If user types 1, set race car status to true

Else if user types 0, set race car status to false

Else tell user “Invalid input, try again” and loop until correct value is given

4. Get age

Call function to get age and make print statement “Your car, Lamborghini Aventador’s age is “ + age

5. Get price

Call function to get price and make print statement “Your car, Lamborghini Aventador’s price is “ + price

6. Get race car status

Call function to get race car status.

If value is true, make a print statement “Your car Lamborghini Aventador IS a race car”

If value is false, make a print statement “Your car Lamborghini Aventador IS NOT a race car”

0. Done with Car Maintenance

Thank you message

* + 1. If user inputs 2
       1. Ask user for truck’s price and truck’s age and store it in appropriate variables
       2. Do this and loop while user does not input 0 to end
          1. Let user select from a menu of

1. Change age

Ask user for truck’s age and check if it is greater than previously stored value

If greater, then store it in age variable

Else loop until user selects correct age

2. Change price

Ask user for truck’s price and check if it is smaller than previously stored value

If smaller then store it in price variable

Else loop until user selects correct price

3. Change diesel type status

Ask user if the truck is a diesel type.

If user types 1, set diesel type status to true

Else if user types 0, set diesel type status to false

Else tell user “Invalid input, try again” and loop until correct value is given

4. Get age

Call function to get age and make print statement “Your truck, Ford F-150’s age is “ + age

5. Get price

Call function to get price and make print statement “Your truck, Ford F-150’s price is “ + price

6. Get diesel type status

Call function to get diesel type status.

If value is true, make a print statement “Your truck Ford F-150 IS a diesel type”

If value is false, make a print statement “Your truck Ford F-150 IS NOT a diesel type”

0. Done with Truck Maintenance

Thank you message

* 1. In class Vehicle
     1. Make private variables
        1. Age (int)
        2. Price (float)
     2. Constructor
        1. Set age to 0
        2. Set price to 0.0
     3. In *setAge()* function
        1. Set vehicle’s age to be the value passed in
           1. Validate the vehicle’s age passed in is greater than previously stored value if not setting the age for 1st time
           2. If setting the age for 1st time, validate that value passed in is positive
     4. In *setPrice()* function
        1. Set vehicle’s price to be the value passed in
           1. Validate the vehicle’s price passed in is smaller than previously stored value if not setting the price for 1st time
           2. If setting the price for 1st time, validate that value passed in is positive
     5. In *getAge()* function
        1. Return vehicle’s age
     6. In *getPrice()* function
        1. Return vehicle’s price
  2. In class Cars (inherits class Vehicle)
     1. Private variable for race car status (bool)
     2. Constructor
        1. Set race car status to false
     3. In *setRaceCarStatus()* function
        1. Set race car status to be the value passed in
     4. In *getRaceCarStatus()* function
        1. Return race car status
  3. In class Truck (inherits class Vehicle)
     1. Private variable for diesel type status (bool)
     2. Constructor
        1. Set diesel type status to false
     3. In *setDieselTypeStatus()* function
        1. Set diesel type status to be the value passed in
     4. In *getDieselTypeStatus()* function
        1. Return diesel type status

1. **Test Plan Version 2**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Valid | 1 | Age of car/truck is greater than previously stored value | Previously stored variable = 2  New = 5 | “Age of car/truck stored” |  |  |
| Valid | 2 | Price of car/truck is less than previously stored value unless storing it 1st time | Previously stored variable = 300,000  New = 250,000 | “Price of car/truck stored” |  |  |
| Valid | 3 | Price value is always positive | Price = 50,000 | “Price of car/truck stored” |  |  |
| Valid | 4 | User enters corresponding number for choosing either true or false for race car status | User enter “1” for true for car status | “Race Car Status stored” |  |  |
| Valid | 5 | User enters corresponding number for choosing either true or false for diesel type status | User enters “1” for true for diesel type status | “Diesel Type Status” |  |  |
| Valid | 6 | User enters corresponding number for choosing either car or truck | User enters “2” for truck | “You have chosen to maintain truck right now” |  |  |
| Invalid | 1 | Age of car/truck is more than previously stored value | Previously stored variable = “10”  New = 5:” | “Invalid input. Age cannot be less than what is stored before |  |  |
| Invalid | 2 | Price of car/truck is more than previously stored value unless storing it 1st time | Previously stored variable = “20,000”  New = “30,000” | “Invalid input. You cannot sell your car more than its previous worth |  |  |
| Invalid | 3 | Price value is negative | Price value = -50, 000 | “Invalid input. Price value cannot be negative” |  |  |
| Invalid | 4 | User enters number not corresponding to choosing either true or false for race car status | User enters “10” for true for race car status | “Invalid input. Please try again with number 1 or number 0” |  |  |
| Invalid | 5 | User enters number not corresponding to choosing either Car or Truck | User enters “2” to choose truck | “You have chosen to maintain truck now” |  |  |

1. **Code**

Program\_3A.cpp

// Program\_3A.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include "Car.h"

#include "Truck.h"

int main()

{

int tempVehicle; // to store vehicle type

int tempMenu; // to store menu selection

int tempAge;

float tempPrice;

int tempStatus;

int countCar = 0, countTruck = 0; //To check if it is the first time user is adding data

Car Lamborghini;

Truck Ford;

cout << "Welcome, stranger. ";

do {

cout << "Do you want to work on Lamborghini Aventador or Ford F-150 Pick Up?\nChoose 1 for the car, 2 for the truck, and 0 to shut me off" << endl;

cin >> tempVehicle;

//Lamborghini

if (tempVehicle == 1) // Lamborghini Aventador

{

system("CLS");

cout << "...Currently maintaining Lamborghini Aventador..." << endl;

if (countCar == 0)

{

do {

cout << "Car's age: ";

cin >> tempAge;

if (tempAge < 0)

{

cout << endl << "Invalid input. Car cannot be negative years old" << endl;

}

} while (tempAge < 0);

Lamborghini.setAge(tempAge);

do {

cout << "Car's price: ";

cin >> tempPrice;

if (tempPrice < 0)

{

cout << endl << "Invalid input. Price value cannot be negative" << endl;

}

} while (tempPrice < 0);

Lamborghini.setPrice(tempPrice);

system("pause");

system("CLS");

}

countCar++;

do {

cout << "1. Update car's age - " << Lamborghini.getAge() << " year(s)" << endl;

cout << "2. Update car's price - $" << Lamborghini.getPrice() << endl;

cout << "3. Update race car status" << endl;

cout << "4. Forgot the age, stranger?" << endl;

cout << "5. Forgot the price, stranger?" << endl;

cout << "6. Forgot whether or not your car is a race car?" << endl;

cout << "0. Done with Car Maintenance?" << endl << endl;

cout << "Alright, select a number corresponding to the option you want to choose: ";

cin >> tempMenu;

switch (tempMenu)

{

case 1:

do {

cout << "How old is your car now?: ";

cin >> tempAge;

if (tempAge < Lamborghini.getAge())

{

cout << endl << "Invalid input. Age cannot be less than what is stored before." << endl;

}

else if (tempAge < 0)

{

cout << endl << "Invalid input. Car cannot be negative years old" << endl;

}

} while (tempAge < Lamborghini.getAge() || tempAge < 0);

Lamborghini.setAge(tempAge);

cout << "Age of car stored" << endl;

system("pause");

system("CLS");

break;

case 2:

do {

cout << endl << "How much is your car worth now?: ";

cin >> tempPrice;

if (tempPrice > Lamborghini.getPrice())

{

cout << endl << "Invalid input. You cannot sell your car more than its\nprevious worth" << endl;

}

else if (tempPrice < 0)

{

cout << endl << "Invalid input. Price value cannot be negative" << endl;

}

} while (tempPrice > Lamborghini.getPrice() || tempPrice < 0);

Lamborghini.setPrice(tempPrice);

cout << "Price of car stored" << endl;

system("pause");

system("CLS");

break;

case 3:

do {

cout << "Type 1 if it IS a race car and type 0 if it IS NOT a race car: ";

cin >> tempStatus;

if (tempStatus < 0 || tempStatus > 1)

{

cout << endl << "Invalid input. Please try again with number 1 or number 0" << endl;

}

} while (tempStatus < 0 || tempStatus > 1);

Lamborghini.setRaceCarStatus(tempStatus);

cout << endl << "Race Car Status stored" << endl;

system("pause");

system("CLS");

break;

case 4:

cout << "Your car, Lamborghini Aventador, is " << Lamborghini.getAge() << " year(s) old" << endl;

system("pause");

system("CLS");

break;

case 5:

cout << "Your car, Lamborghini Aventador, is worth $" << Lamborghini.getPrice() << endl;

system("pause");

system("CLS");

break;

case 6:

if (Lamborghini.getRaceCarStatus())

{

cout << "Your car, Lamborghini Aventador, IS a race car" << endl;

}

else {

cout << "Your car, Lamborghini Aventador, IS NOT a race car" << endl;

}

system("pause");

system("CLS");

break;

case 0:

cout << "Lamborghini looks great. Move on to your truck now?" << endl;

system("pause");

system("CLS");

break;

default:

system("CLS");

cout << "Come on, please don't put some random number.\nSelect a number corresponding to the option you want to choose: " << endl;

break;

}

} while (tempMenu != 0);

}

//Ford F-150

else if (tempVehicle == 2) // Ford F-150 Pick Up Truck

{

/\*system("CLS");\*/

cout << "...Currently maintaining Ford F-150 Pick Up Truck..." << endl;

if (countTruck == 0)

{

do {

cout << "Truck's age: ";

cin >> tempAge;

if (tempAge < 0)

{

cout << endl << "Invalid input. Truck cannot be negative years old." << endl;

}

} while (tempAge < 0);

Ford.setAge(tempAge);

cout << "Age of truck is stored" << endl;

do {

cout << "Truck's price: ";

cin >> tempPrice;

if (tempPrice < 0)

{

cout << "Price cannot be negative. Please try again" << endl;

}

} while (tempPrice < 0);

Ford.setPrice(tempPrice);

cout << "Price of truck is stored" << endl;

system("pause");

system("CLS");

}

countTruck++;

do {

cout << "1. Update truck's age - " << Ford.getAge() << " year(s)" << endl;

cout << "2. Update truck's price - $" << Ford.getPrice() << endl;

cout << "3. Update race truck status" << endl;

cout << "4. Forgot the age, stranger?" << endl;

cout << "5. Forgot the price, stranger?" << endl;

cout << "6. Forgot whether or not your truck eats diesel?" << endl;

cout << "0. Done with Truck Maintenance?" << endl << endl;

cout << "Alright, select a number corresponding to the option you want to choose: ";

cin >> tempMenu;

switch (tempMenu)

{

case 1:

do {

cout << "How old is your Truck now?: ";

cin >> tempAge;

if (tempAge < Ford.getAge())

{

cout << endl << "Invalid input. Age cannot be less than what is stored before." << endl;

}

} while (tempAge < Ford.getAge());

Ford.setAge(tempAge);

cout << "Age of Truck stored" << endl;

system("pause");

system("CLS");

break;

case 2:

do {

cout << endl << "How much is your Truck worth now?: ";

cin >> tempPrice;

if (tempPrice > Ford.getPrice())

{

cout << endl << "Invalid input. You cannot sell your Truck more than its\nprevious worth" << endl;

}

else if (tempPrice < 0)

{

cout << endl << "Invalid input. Price value cannot be negative" << endl;

}

} while (tempPrice > Ford.getPrice() || tempPrice < 0);

Ford.setPrice(tempPrice);

cout << "Price of Truck stored" << endl;

system("pause");

system("CLS");

break;

case 3:

do {

cout << "Type 1 if it IS a diesel-type and type 0 if it IS NOT a diesel-type: ";

cin >> tempStatus;

if (tempStatus < 0 || tempStatus > 1)

{

cout << endl << "Invalid input. Please try again with number 1 or number 0" << endl;

}

} while (tempStatus < 0 || tempStatus > 1);

Ford.setDieselTypeStatus(tempStatus);

cout << endl << "Diesel Type Status stored" << endl;

system("pause");

system("CLS");

break;

case 4:

cout << "Your Truck, Ford F-150 Pick Up, is " << Ford.getAge() << " year(s) old" << endl;

system("pause");

system("CLS");

break;

case 5:

cout << "Your truck, Ford F-150, is worth $" << Ford.getPrice() << endl;

system("pause");

system("CLS");

break;

case 6:

if (Ford.getDieselTypeStatus())

{

cout << "Your truck, Ford F-150, IS a diesel-type" << endl;

}

else {

cout << "Your truck, Ford F-150, IS NOT a diesel-type" << endl;

}

system("pause");

system("CLS");

break;

case 0:

cout << "Ford Pick Up looks great. Move on to your car now?" << endl;

system("pause");

system("CLS");

break;

default:

system("CLS");

cout << "Come on, please don't put some random number.\nSelect a number corresponding to the option you want to choose: " << endl;

break;

}

} while (tempMenu != 0);

}

//Quit program

else if (tempVehicle == 0) // Quit

{

cout << "Thank you for using me to maintain your vehicle(s), stranger.\nSee you next time" << endl;

}

//Invalid input

else // Invalid input

{

cout << "Sorry stranger, you have a whole different kind of vehicle\nor you just typed in a wrong number" << endl;

}

} while (tempVehicle != 0);

return 0;

}

Vehicle.h

#pragma once

#ifndef VECHICLE\_H

#define VEHICLE\_H

#include <iostream>

#include <string>

using namespace std;

class Vehicle

{

public:

Vehicle();

void setAge(int thisAge);

void setPrice(float thisPrice);

int getAge();

float getPrice();

private:

int age;

float price;

};

#endif

Vehicle.cpp

#include "stdafx.h"

#include "Vehicle.h"

using namespace std;

Vehicle:: Vehicle()

{

age = 0;

price = 0.0;

}

void Vehicle::setAge(int thisAge)

{

age = thisAge;

}

void Vehicle::setPrice(float thisPrice)

{

price = thisPrice;

}

int Vehicle::getAge()

{

return age;

}

float Vehicle::getPrice()

{

return price;

}

Car.h

#pragma once

#ifndef CAR\_H

#define CAR\_H

#include "Vehicle.h"

class Car: public Vehicle

{

public:

Car();

void setRaceCarStatus(bool thisStatus);

bool getRaceCarStatus();

private:

bool raceCarStatus;

};

#endif // !CAR\_H

Car.cpp

#include "stdafx.h"

#include "Car.h"

Car::Car()

{

raceCarStatus = false;

}

void Car::setRaceCarStatus(bool thisStatus)

{

raceCarStatus = thisStatus;

}

bool Car::getRaceCarStatus()

{

return raceCarStatus;

}

Truck.h

#pragma once

#ifndef TRUCK\_H

#define TRUCK\_H

#include "Vehicle.h"

class Truck: public Vehicle

{

public:

Truck();

void setDieselTypeStatus(bool thisStatus);

bool getDieselTypeStatus();

private:

bool dieselTypeStatus;

};

#endif // !TRUCK\_H

Truck.cpp

#include "stdafx.h"

#include "Truck.h"

Truck::Truck()

{

dieselTypeStatus = false;

}

void Truck::setDieselTypeStatus(bool thisStatus)

{

dieselTypeStatus = thisStatus;

}

bool Truck::getDieselTypeStatus()

{

return dieselTypeStatus;

}

1. **Updated Algorithm**
   1. In main function
      1. Make an instance for Car class and for Truck class
      2. Ask user is they have a Lamborghini Aventador (input 1) or Ford F-150 Pickup (input 2) and loop until user wants to quit by pressing 0
      3. If user inputs 1
         1. Ask user for car’s price and car’s age and store it in appropriate variables
            1. Keep looping if user gives a negative number for both
         2. Do this and loop while user does not input 0 to end
            1. Let user select from a menu of

1. Change age and show current age next to it

Ask user for car’s age and check if it is greater than previously stored value

Check if it is positive

If greater and positive, then store it in age variable

Else loop until user selects correct age

2. Change price and show current price next to it

Ask user for car’s price and check if it is smaller than previously stored value

Check if it is positive

If smaller and positive, then store it in price variable

Else loop until user selects correct price

3. Change race car status

Ask user if the car is a race car.

If user types 1, set race car status to true

Else if user types 0, set race car status to false

Else tell user “Invalid input, try again” and loop until correct value is given

4. Get age

Call function to get age and make print statement “Your car, Lamborghini Aventador’s age is “ + age

5. Get price

Call function to get price and make print statement “Your car, Lamborghini Aventador’s price is “ + price

6. Get race car status

Call function to get race car status.

If value is true, make a print statement “Your car Lamborghini Aventador IS a race car”

If value is false, make a print statement “Your car Lamborghini Aventador IS NOT a race car”

0. Done with Car Maintenance

Thank you message

* + 1. If user inputs 2
       1. Ask user for truck’s price and truck’s age and store it in appropriate variables
          1. Keep looping if user gives a negative number for both
       2. Do this and loop while user does not input 0 to end
          1. Let user select from a menu of

1. Change age and show current age next to it

Ask user for truck’s age and check if it is greater than previously stored value

Check if it is positive

If greater and positive, then store it in age variable

Else loop until user selects correct age

2. Change price and show current price next to it

Ask user for truck’s price and check if it is smaller than previously stored value

Check if it is positive

If smaller and positive, then store it in price variable

Else loop until user selects correct price

3. Change diesel type status

Ask user if the truck is a diesel type.

If user types 1, set diesel type status to true

Else if user types 0, set diesel type status to false

Else tell user “Invalid input, try again” and loop until correct value is given

4. Get age

Call function to get age and make print statement “Your truck, Ford F-150’s age is “ + age

5. Get price

Call function to get price and make print statement “Your truck, Ford F-150’s price is “ + price

6. Get diesel type status

Call function to get diesel type status.

If value is true, make a print statement “Your truck Ford F-150 IS a diesel type”

If value is false, make a print statement “Your truck Ford F-150 IS NOT a diesel type”

0. Done with Truck Maintenance

Thank you message

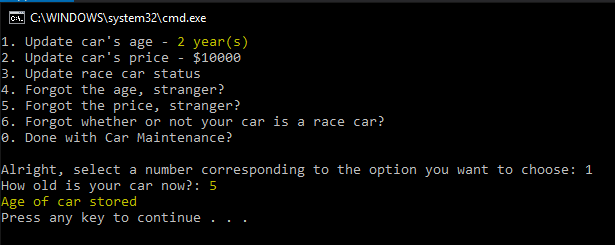
* + 1. If user inputs 0
       1. Print a thank you message and end the loop
    2. If user enters some other number
       1. Print a message saying there is an invalid input and try again
  1. In class Vehicle
     1. Make private variables
        1. Age (int)
        2. Price (float)
     2. Constructor
        1. Set age to 0
        2. Set price to 0.0
     3. In *setAge()* function
        1. Set vehicle’s age to be the value passed in
           1. Validate the vehicle’s age passed in is greater than previously stored value if not setting the age for 1st time
           2. If setting the age for 1st time, validate that value passed in is positive
     4. In *setPrice()* function
        1. Set vehicle’s price to be the value passed in
           1. Validate the vehicle’s price passed in is smaller than previously stored value if not setting the price for 1st time
           2. If setting the price for 1st time, validate that value passed in is positive
     5. In *getAge()* function
        1. Return vehicle’s age
     6. In *getPrice()* function
        1. Return vehicle’s price
  2. In class Cars (inherits class Vehicle)
     1. Private variable for race car status (bool)
     2. Constructor
        1. Set race car status to false
     3. In *setRaceCarStatus()* function
        1. Set race car status to be the value passed in
     4. In *getRaceCarStatus()* function
        1. Return race car status
  3. In class Truck (inherits class Vehicle)
     1. Private variable for diesel type status (bool)
     2. Constructor
        1. Set diesel type status to false
     3. In *setDieselTypeStatus()* function
        1. Set diesel type status to be the value passed in
     4. In *getDieselTypeStatus()* function
        1. Return diesel type status

1. **Test Plan Version 3**

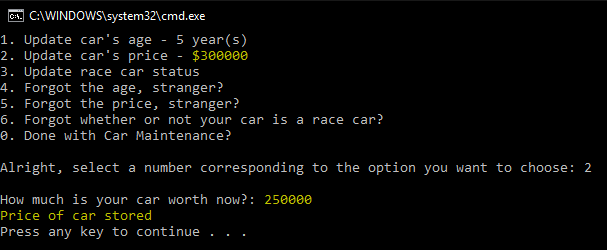
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Valid | 1 | Age of car/truck is greater than previously stored value | Previously stored variable = 2  New = 5 | “Age of car/truck stored” | “Age of car stored” | Pass |
| Valid | 2 | Price of car/truck is less than previously stored value unless storing it 1st time | Previously stored variable = 300,000  New = 250,000 | “Price of car/truck stored” | “Price of car stored” | Pass |
| Valid | 3 | Price value is always positive | Price = 50,000 | “Price of car/truck stored” | “Price of car stored” | Pass |
| Valid | 4 | User enters corresponding number for choosing either true or false for race car status | User enter “1” for true for car status | “Race Car Status stored” | “Race Car Status stored” | Pass |
| Valid | 5 | User enters corresponding number for choosing either true or false for diesel type status | User enters “1” for true for diesel type status | “Diesel Type Status stored” | “Diesel Type Status stored” | Pass |
| Valid | 6 | User enters corresponding number for choosing either car or truck | User enters “2” for truck | “You have chosen to maintain truck right now” | “Currently maintaining Ford F-150 Pick up truck” | Pass |
| Invalid | 1 | Age of car/truck is more than previously stored value | Previously stored variable = “10”  New = 5:” | “Invalid input. Age cannot be less than what is stored before” | “Invalid input. Age cannot be less than what is stored before” | Pass |
| Invalid | 2 | Price of car/truck is more than previously stored value unless storing it 1st time | Previously stored variable = “20,000”  New = “30,000” | “Invalid input. You cannot sell your car more than its previous worth” | “Invalid input. You cannot sell your car more than its previous worth” | Pass |
| Invalid | 3 | Price value is negative | Price value = -50,000 | “Invalid input. Price value cannot be negative” | “Invalid input. Price value cannot be negative” | Pass |
| Invalid | 4 | Age value is negative | Age value = -5 | “Invalid input. Car/Truck cannot be negative years old” | “Car/Truck cannot be negative years old” | Pass |
| Invalid | 5 | User enters number not corresponding to choosing either true or false for race car status | User enters “10” for true for race car status | “Invalid input. Please try again with number 1 or number 0” | “Invalid input. Please try again with number 1 or number 0” | Pass |
| Invalid | 6 | User enters number not corresponding to choosing either Car or Truck | User enters “5” to choose truck | “Sorry stranger, you have a whole different kind of vehicle or you just typed in a wrong number” | “Sorry stranger, you have a whole different kind of vehicle or you just typed in a wrong number” | Pass |

1. **Screenshots**

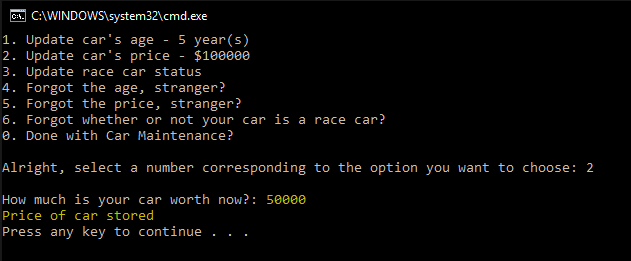
Valid Test Case 1



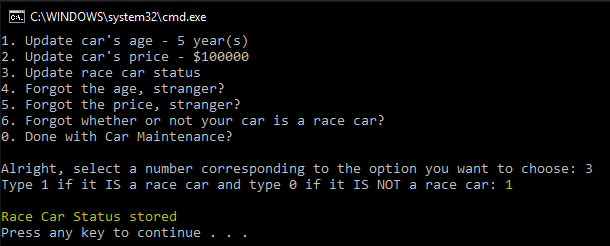
Valid Test Case 2



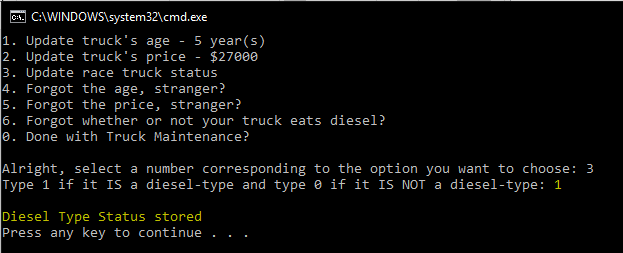
Valid Test Case 3



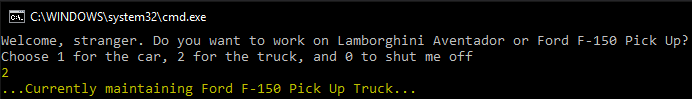
Valid Test Case 4



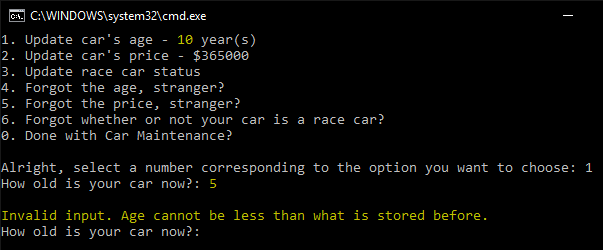
Valid Test Case 5



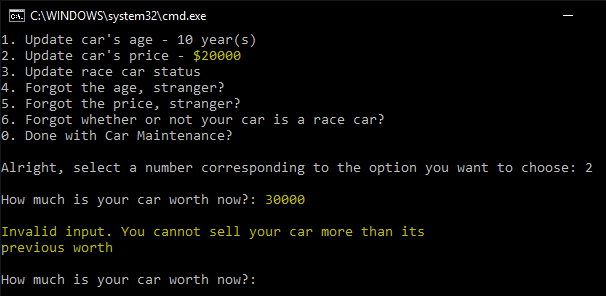
Valid Test Case 6



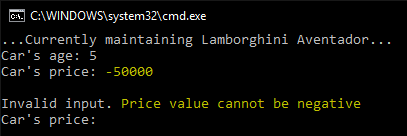
Invalid Test Case 1



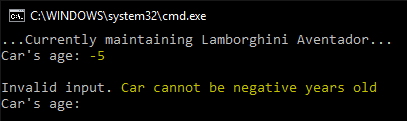
Invalid Test Case 2



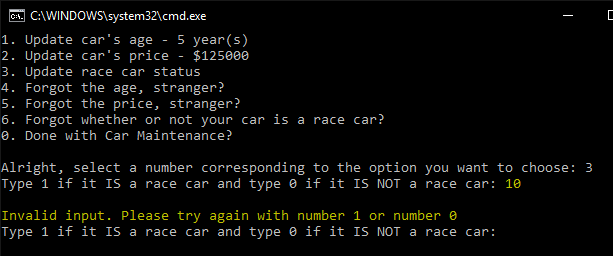
Invalid Test Case 3



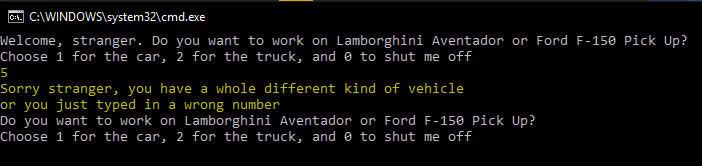
Invalid Test Case 4



Invalid Test Case 5



Invalid Test Case 6



1. **Status**

Program works perfectly with assumptions in mind.