Hassan Harajly

10/21/17

Cis200

Assignment 2

**Car Program**

1. **Problem Statement**

Create functional classes in order to store information about a given car, or vehicle. Will create a base class of type vehicle for future vehicles to inherit from. The program will store information in various classes.

1. **Requirements**
   1. **Assumptions**

The input given to the classes will be as follows:

Age, must be a integer

Price, the price of the vehicle must be a float.

If the vehicle is a car, will ask if the car is a racecar then store the answer as true or false.will enter answer as a Boolean variable.

If the vehicle is a truck will ask for all the vehicle information include if the truck is a diesel truck or not input must also be of type boolean

* 1. **Specifications**

The program will prompt the user to enter information about the vehicle.

Depending on the type a specific class will used to store the information.

All information input about the car will be stored using various methods of different classes.

The user must input the an integer denoting the vehicles age.

Then the user will input a float to signify the vehicles price.

If the is a car it will ask if the car is a race car and store that information as a Boolean

If the vehicle is a truck, the program will need to get information on if it’s a diesel or gasoline car. Will be stored as Boolean.

1. **Decomposition Diagram** (Used to break program down into components visually. Diagram can have as many components as needed. Defines functionality that will solve the problem – does NOT define a flow of actions)

Vehicle information program

input

process

Output

User inputs integer for age by command line

Methods are used to return data

Display error messages if input is incorrect

ask user if they would like to rerun program

Methods are used to store data inside of class

Validate input

User inputs if car is a race car

User inputs if truck is diesel or not

user inputs type of vehicle such as car or truck

User inputs a float for the price of the vehicle

1. **Test Strategy**

* Valid Data
* Invalid Data

1. **Test Plan Version 1**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Invalid Data | 1 | User inputs wrong date for age of car |  |  |  |  |
| Invalid Data | 2 | User inputs incorrect price <0 |  |  |  |  |
| Valid Data | 3 | User enters 1995 for age of vehicle |  |  |  |  |

1. **Initial Algorithm**

Create a class that has 1 integer, and 1 float

Class vehicle

Private

Integer age

Float price

Public

Vehicle() constructor sets variables = to 0

Setage() sets age equal to the parameter

Setprice() sets price = to float parameter

Getage() returns age

Getprice() returns price

Class car inherits from vehicle

Car() sets racecar bool to false

Setracecarstatus() sets racecar status with a bool parameter

Getracecarstatus() returns status of the car

End class car

Class truck inherits from vehicle

Truck() constructor sets diesel Boolean to false

Setdieseltype() sets Boolean of diesel type to parameter

Getdieseltype() returns the diesel type

End class truck

Main function

Print “enter age of vehicle”

Input age

Print “enter price of vehicle “

Input price

Print”is the vehicle a car or truck”

Input vehicle type

se

If (type == car)

{

Car carclass

Carclass. Setage(age)

Carclass.setprice(price)

Print “is the car a race car?”

Input racecar

Carclass.setracecarstatus(racecar)

}

If (type== truck)

{

truck truckclass

truckclass. Setage(age)

truckclass.setprice(price)

print “is the truck a diesel engine?”

input gastype

truckclass.setdieseltype(gastype)

}

End main function

1. **Test Plan Version 2**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Invalid Data | 1 | User inputs wrong date for age of car | Ij9 | “error please enter a year like 1995” |  |  |
| Invalid Data | 2 | User inputs incorrect price <0 | -1 | “please enter a price of 0 or larger” |  |  |
| Valid Data | 3 | User enters 1995 for age of vehicle | 1995 | “what is the price of the vehicle” |  |  |
| Invalid data | 4 | Set function for diesel type gets an integer number | 2 | “please enter true or false” |  |  |

1. **Code**

//hassan harajly

//10/21/17 cis 200

//description this program stores vehicle information inside of a series of inhereited classes

#include <iostream>

using namespace std;

class vehicle {

private:

int age;

float price;

public:

//default constructor that initializes the values

vehicle() { age = 0; price = 0; }

//precondition:integer denoting age

//postcondition:age variable assignment

//description:sets private variable age

void setage(int agehold) { age = agehold; }

//precondition: float denoting price

//postcondition: price variable assignment

//description: sets private variable price

void setprice(float pricehold) { price = pricehold; }

int getage() { return age; }//returns age variable

float getprice() { return price; }//returns price variable

};

//class inherits from vehicle has the same attributes

class car :public vehicle

{

private:

bool raceCarStatus;

public:

//constructor must call constructor in base class since it isnt inherited

//precondition:racecar bool

//postcondition:racecarstatus assignment

//description: sets racecarstatus

car() { vehicle::vehicle(); raceCarStatus = false; }

void setRaceCarStatus(bool race) { raceCarStatus = race; }

bool getRaceCarStatus() { return raceCarStatus; }//returns the racecar status as a 0 or 1

};

class truck :public vehicle

{

private:

bool dieselTypeStatus;

public:

//constructor must call constructor in base class since it isnt inherited

truck() { vehicle::vehicle(); dieselTypeStatus = false; }//preconditon dieseltype status initialization

//postcondition private variable initialization

void setDieselTypeStatus(bool status) { dieselTypeStatus = status; }//sets dieseltype status as given bool by input

bool getDieselTypeStatus() { return dieselTypeStatus; }//returns a 0 or 1

};

int main()

{//stub main to test all functions and inheritance of various methods

vehicle v;

v.setage(5);

v.setprice(95.32);

cout << "vehicle age is " << v.getage() << endl;

cout << "vehicle price is " << v.getprice() << endl;

car c;

c.setage(8);

c.setprice(4932.02);

c.setRaceCarStatus(false);

cout << "age of the CAR: " << c.getage() << endl;

cout << "price of the CAR: " << c.getprice() << endl;

cout << "racecar status(1 for yes 0 for no): " << c.getRaceCarStatus() << endl;

truck t;

t.setage(53342);

t.setprice(124932.02);

t.setDieselTypeStatus(true);

cout << "age of the TRUCK: " << t.getage() << endl;

cout << "price of the TRUCK: " << t.getprice() << endl;

cout << "diesel fuel status(1 for yes 0 for no): "

<< t.getDieselTypeStatus() << endl;

system("pause");

return 0;

}

1. **Updated Algorithm**

create class vehicle

{

declare variables age and price

create constructor and initialize age and price

create getters and setters for age and price

//end of class vehicle

}

create class car and inherit class vehicle

{

declare variable raceCarStatus

create constructor and initialize raceCarStatus to false

create getters and setters for raceCarStatus

}

create class truck and inherit class vehicle

{

declare variable dieselTypeStatus

create constructor and initialize dieselTypeStatus to false

create getters and setters for dieselTypeStatus

}

create object v vehicle

set v.setage(5)

set v.setprice(5)

cout::vehicle age, price

create an object c car

set c.setage(532)

set c.setprice(4932.02)

set c.setRaceCarStatus(false)

cout::vehicle age, price,racecar status

create an object t truck

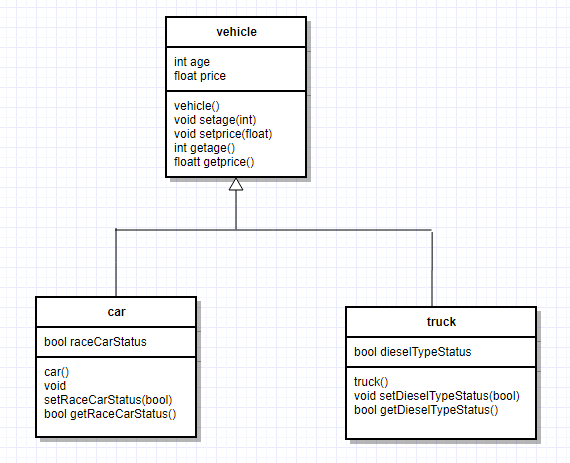
set t.setage(532)

set t.setprice(4932.02)

set t.setdieselTypeStatus(false)

cout::vehicle age, price,dieselType status

**uml:**



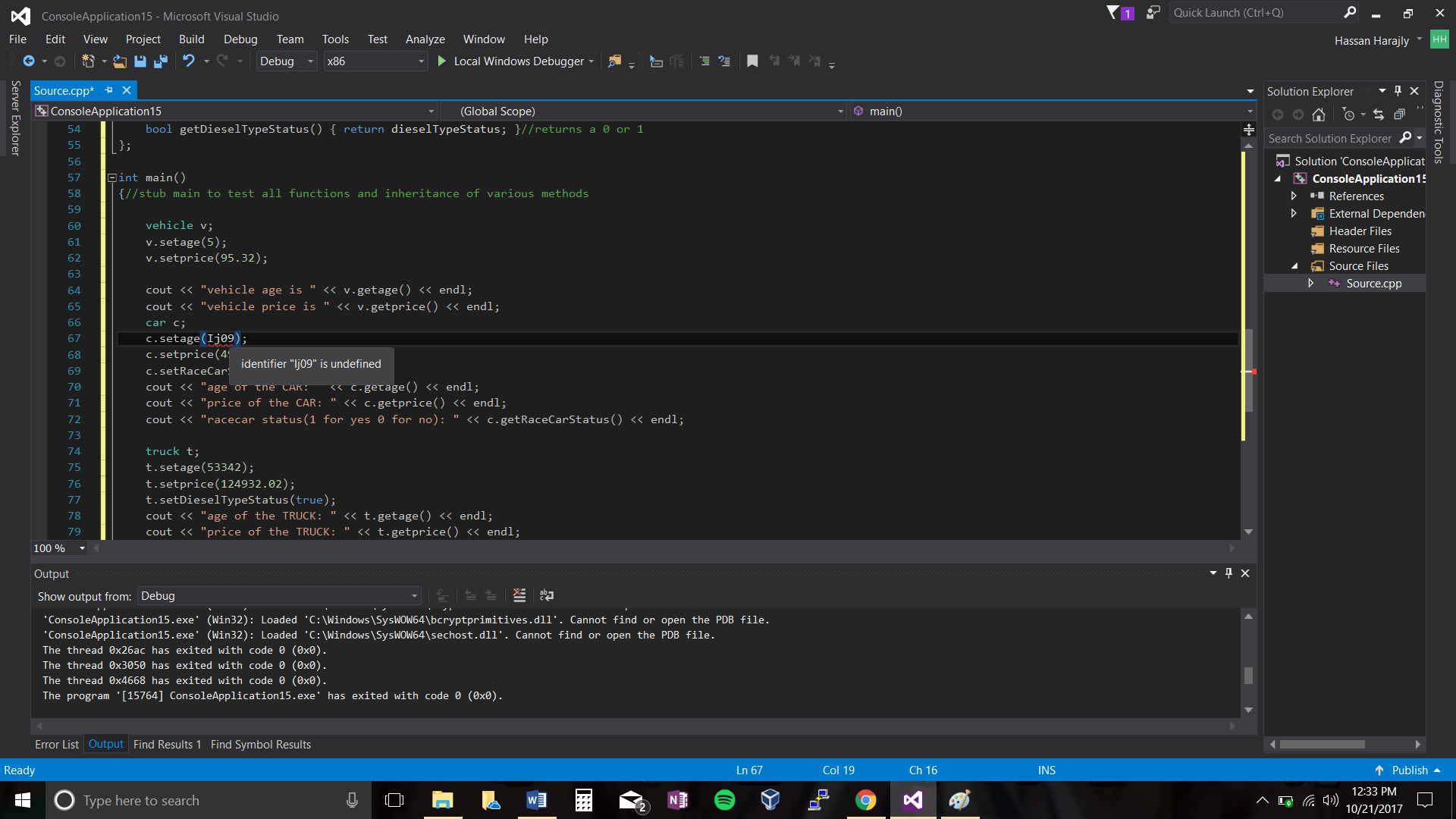
1. **Test Plan Version 3**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Invalid Data | 1 | User inputs wrong date for age of car | Ij9 | “error please enter a year like 1995” | Compiler error | Pass |
| Invalid Data | 2 | User inputs incorrect price <0 | -1 | “please enter a price of 0 or larger” | Price can actually be negative since for example if someone is receiving a car they can be paid to take the car so technically it can be negative | fail |
| Valid Data | 3 | User enters 1995 for age of vehicle | 1995 | “what is the price of the vehicle” | None correct input is set . | pass |
| Invalid data | 4 | Set function for diesel type gets an integer number | 2 | “please enter true or false” | None correct input is set | pass |
| Stub main | See below |  |  |  |  |  |

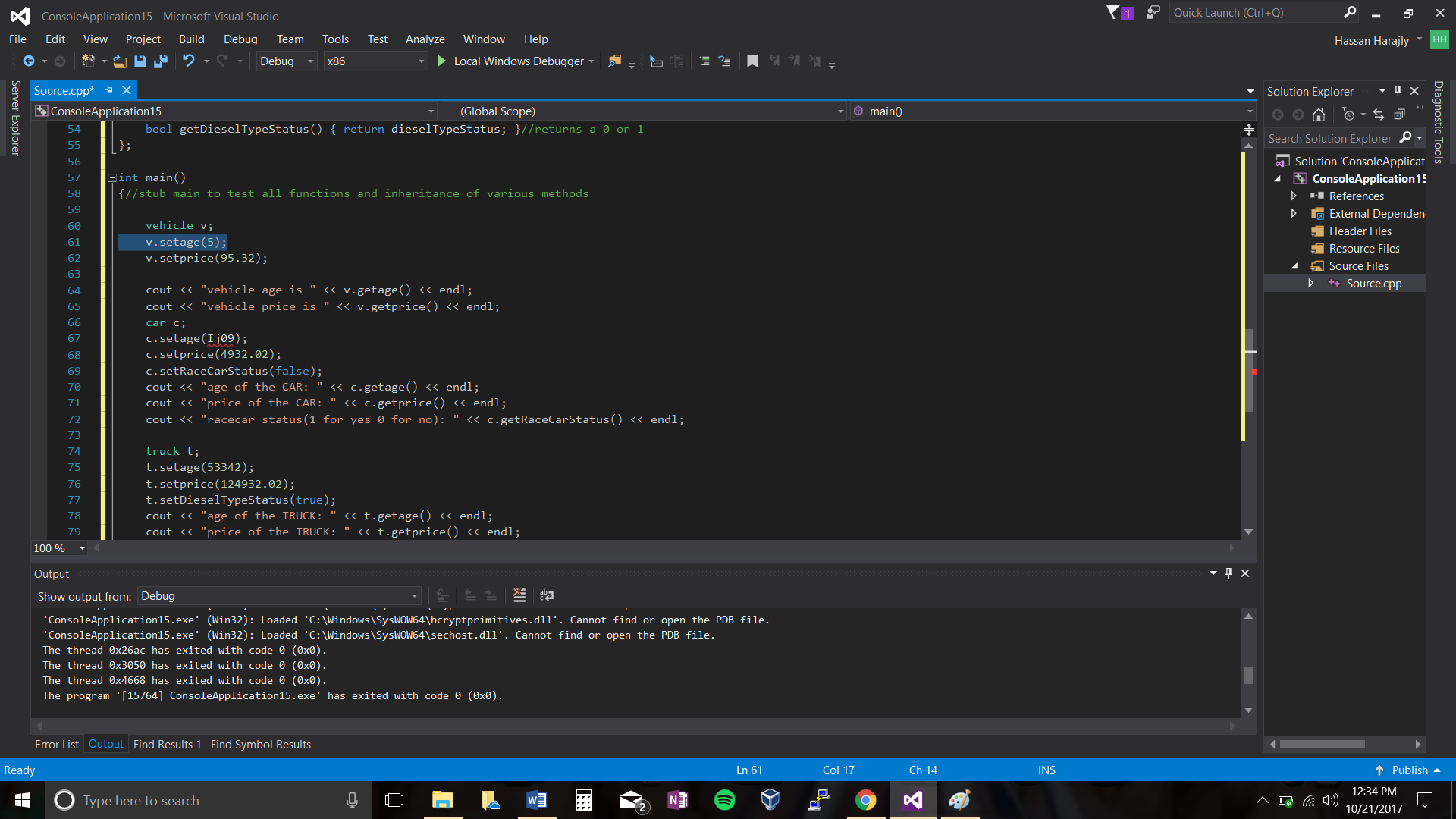
1. **Screenshots**

Test Cases 1-9

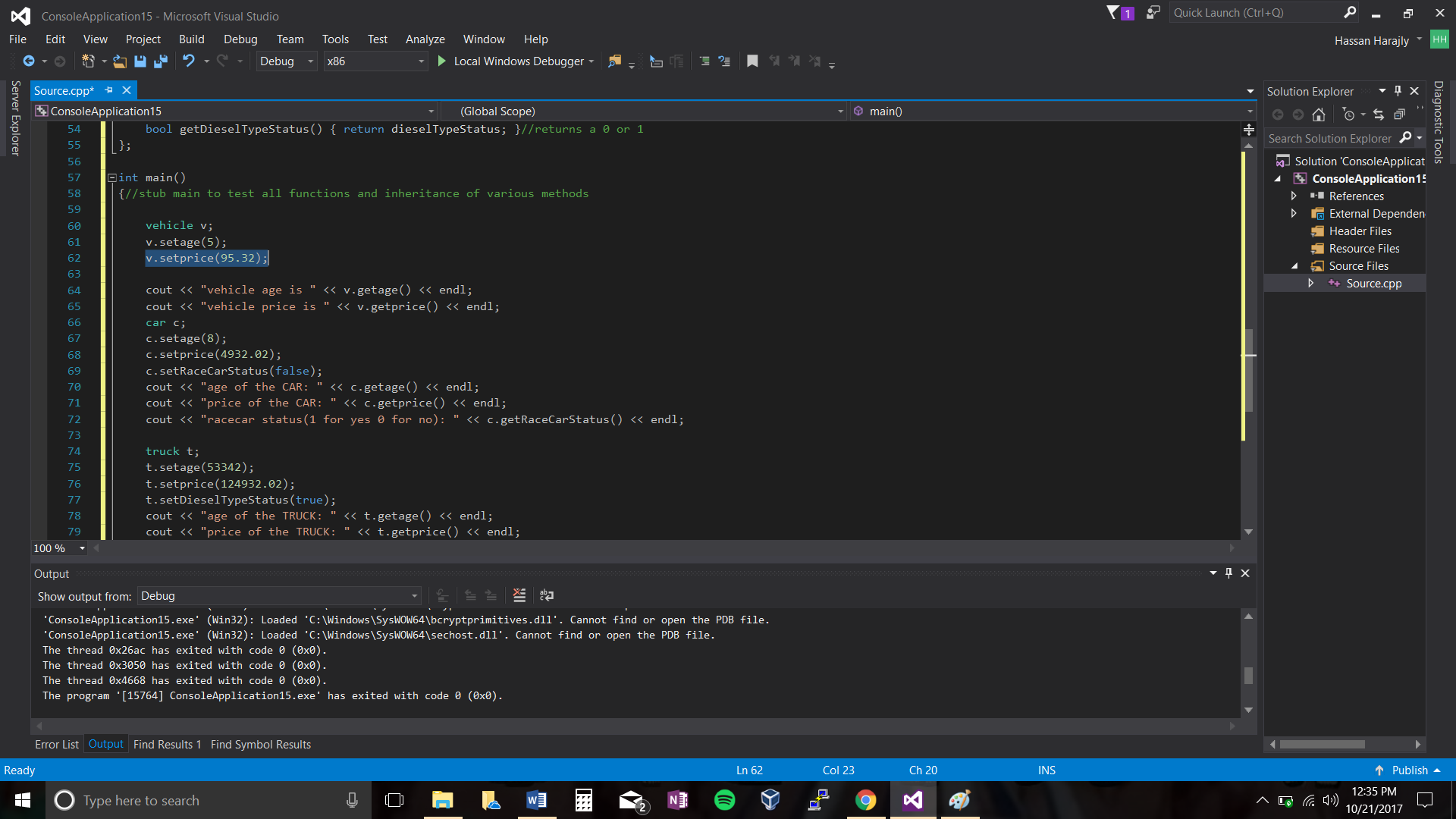
1:



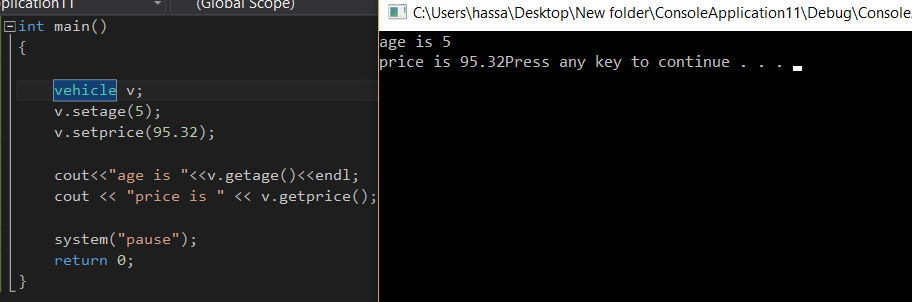
2:

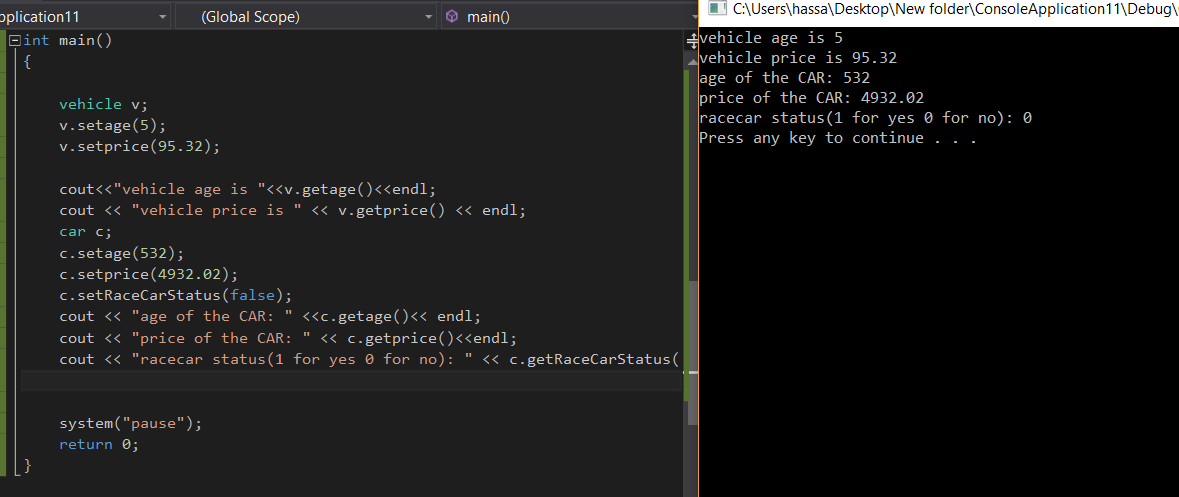


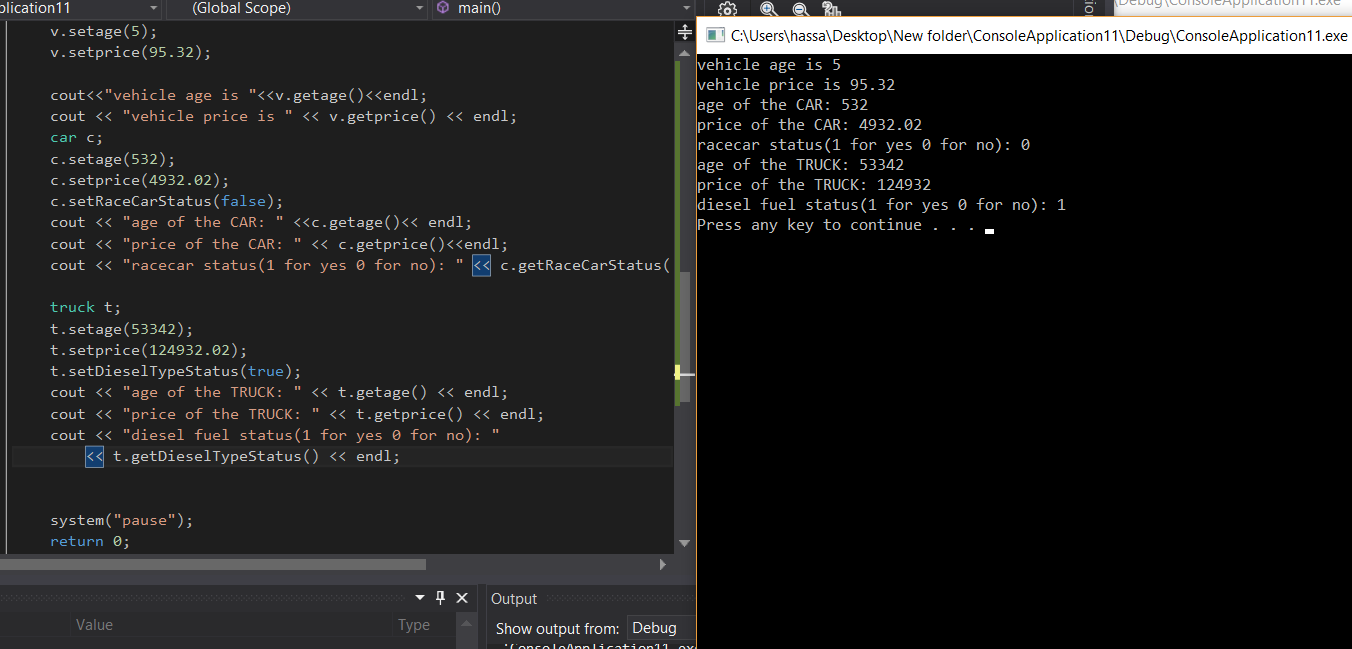
Test case 4:



Stub main:







1. **Error Log**

|  |  |  |
| --- | --- | --- |
| Error Type | Cause of Error | Solution to Error |
|  |  |  |

1. **Status**

The program works 100% with assumptions in place