1. Code

Car.cpp

```
#include "Car.h"
Car::Car()
{
	raceCarStatus = false;
}

void Car::setRaceCarStatus(bool thisStatus)
{
	raceCarStatus = thisStatus;
}

bool Car::getRaceCarStatus()
{
	return raceCarStatus;
}
```

Car.h

```
#pragma once
#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
```

Vehicle.cpp

```
#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;
}
```

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
```

Stub_main_VehicleCar.cpp

```
#include "Car.h"

int main()
{
		Car Civic;

			int tempAge;
			float tempPrice;
```

```
	do {
		cout << "Civic's age: ";
		cin >> tempAge;
		if (tempAge < 0)
		{
			cout << endl << "Invalid input. Car cannot be negative
years old" << endl;
		}
	} while (tempAge < 0);
	Civic.setAge(tempAge);

	cout << "Age stored is " << Civic.getAge() << endl;


	do {
		cout << "How old is your Car now?: ";
		cin >> tempAge;
		if (tempAge < Civic.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 < Civic.getAge() || tempAge < 0);
	Civic.setAge(tempAge);
	cout << "Age of Car stored" << endl;




	do {
		cout << "Car's price: ";
		cin >> tempPrice;
		if (tempPrice < 0)
		{
			cout << endl << "Invalid input. Price value cannot be
negative" << endl;
		}
	} while (tempPrice < 0);
	Civic.setPrice(tempPrice);

	cout << "Price stored is $" << Civic.getPrice() << endl;



	do {
		cout << endl << "How much is your Car worth now?: ";
		cin >> tempPrice;
		if (tempPrice > Civic.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 > Civic.getPrice() || tempPrice < 0);
                Civic.setPrice(tempPrice);
                cout << "Price of Car stored" << endl;


        int tempStatus;

        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);
        Civic.setRaceCarStatus(tempStatus);
        cout << "Race Car Status stored" << endl;


        if (Civic.getRaceCarStatus())
        {
                cout << "Your car, CIvic, IS a race car" << endl;
        }
        else {
                cout << "Your car, Civic, IS NOT a race car" << endl;
        }

    return 0;
}
```

2. Test Plan

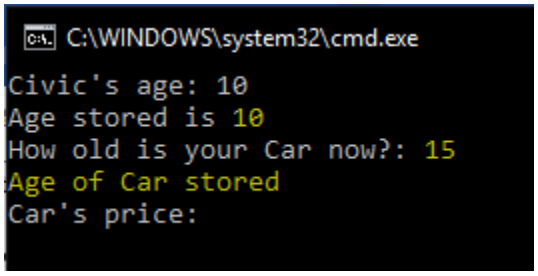
Test Strategy	Test Number	Description	Input	Expected Output	Actual Output	Pass/Fail
Valid	1	Age of car is greater than previously stored value	Previously stored variable = 10 New = 15	"Age of car stored"	"Age of car stored"	Pass
Valid	2	Price of car is less than previously stored value unless storing it 1st time	Previously stored variable = 50000 New = 12000	"Price of car stored"	"Price of Car stored"	Pass

Valid	3	Price value is always positive	Price = 50,000	"Price of car stored"	"Price stored is $50000"	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	Age value is always positive	Age = 10	"Age of car stored"	"Age stored is 10"	Pass
Invalid	1	Age of car 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 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	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	Pass

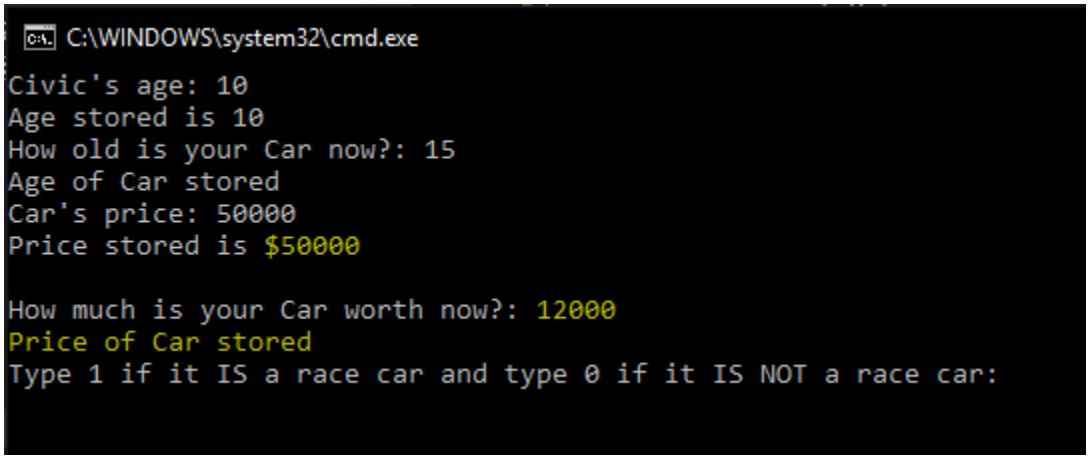
					number 0”	
Invalid	5	Age value is negative	Age = -50	“Invalid input. Car cannot be negative years old”	“Invalid input. Car cannot be negative years old”	Pass

3. Screenshots

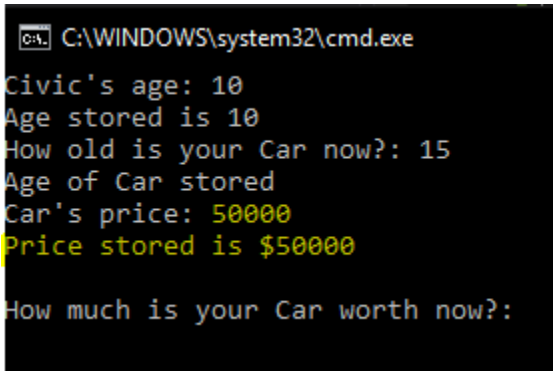
Valid Test Case 1:



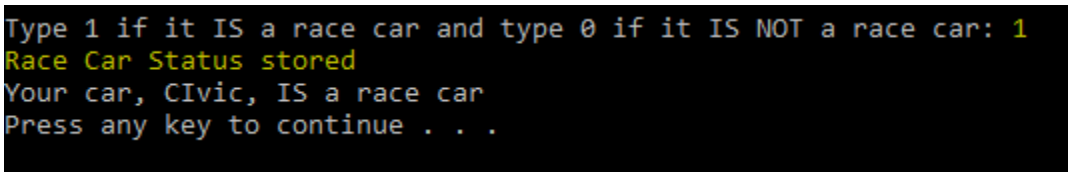
Valid Test Case 2:



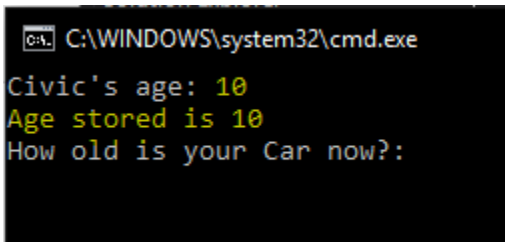
Valid Test Case 3:



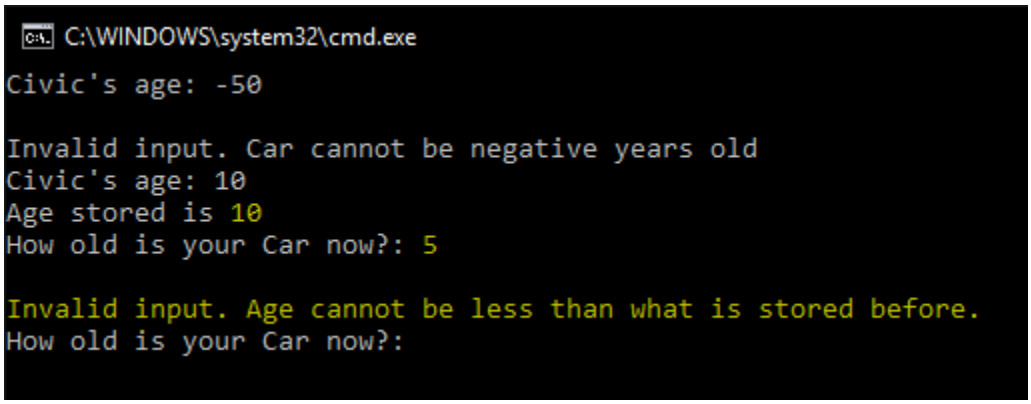
Valid Test Case 4:



Valid Test Case 5:



Invalid Test Case 1:



Invalid Test Case 2:

```
Car's price: 20000
Price stored is $20000

How much is your Car worth now?: 30000

Invalid input. You cannot sell your Car more than its
previous worth

How much is your Car worth now?:
```

Invalid Test Case 3:

```
Car's price: -50000

Invalid input. Price value cannot be negative
Car's price:
```

Invalid Test Case 4:

```
Type 1 if it IS a race car and type 0 if it IS NOT a race car: 10

Invalid input. Please try again with number 1 or number 0
Type 1 if it IS a race car and type 0 if it IS NOT a race car:
```

Invalid Test Case 5:

```
C:\WINDOWS\system32\cmd.exe
Civic's age: -50

Invalid input. Car cannot be negative years old
Civic's age:
```