CIS 200 - Project 3 Provided File Analysis

Problem Statement

Take in another students Lab 0501 and finalize and broken or unfinished portions and provide a certain letter grade. Original files and documentation provided by Sherry Robbins. Corrections and verdict provided by Arthur Aigeltinger IV.

*All original files that I received will be included to show some of the formatting issues that existed with the original document and may back up some of the explanations I have included below. If you have any questions please contact me. The formatting of this new document should best show all of the portions that I received, reformatted, with verdicts and any applicable changes below that in the *Updated* categories - as opposed to the *Provided* category. I hope this simplifies the evaluation process. The grade is in the last section.

Provided Algorithm

```
1. Create a program with 3 classes,
    a. the first class is called Vehicle (parent class)
        i. Vehicle has two private variables:
            1. Age(int): the age of the vehicle
            2. Price(float) cost of vehicle
        ii. Vehicle has 4 methods and a constructor
            1. Vehicle()
                a. Default constructor
                b. Sets:
                    i. Age=0
                    ii. Price=0.0
            2. Void setAge(int a)
                a. Sets age=a;
            3. Void setPrice( float p)
                a. Sets price=p;
            4. Int getAge()
                a. Return age;
            5. Float getPrice()
                a. Return price;
    b. The second class is called Car (child class)
        i. Add Boolean variable RaceCarStatus=false
        ii. Is uses the same variables as Vehicle along with the same methods
            1. Car also has the following methods:
            2. A default constructor Car()
                a. RaceCarStatus=false
            3. Void setRaceCarStatus(bool s)
```

                a. RaceCarStatus=s;
            4. Bool getRaceCarStatus():
                a. return raceCarStatus
    c. The third class is called Truck (child class)
        i. Truck uses the same variables and methods as Vehicle however it includes the Boolean variable DieselTypeStatus set to false and the following methods/constructor:
            1. Truck() is a default constructor
                a. DieselTypeStatus=false
            2. Void setDieselTypeStatus(bool s)
                a. DieselTypeStatus=s;
            3. bool getDieselTypeStatus()
                a. return DieselTypeStatus
2. Int the main test all the methods and the classes

Provided Test Cases

Test Plan For Vehicle

Test Strategy	Test #	Description	Input	Expected Output	Actual Output	Pass/Fail
Valid Data	1	setAge()	setAge(25)	25	25	PASS
Valid Data	2	setPrice()	setPrice(10.00)	10.00	10	PASS
Valid Data	3	getAge()	--------------			
Valid Data	4	getPrice()	--------------			
Valid Data	5	setAge()	setAge()	0	0	PASS
Valid Data	6	setPrice()	setPrice()	0	0	PASS

Test Plan for Car

Test Strategy	Test #	Description	Input	Expected Output	Actual Output	Pass/Fail
Valid Data	1	setRaceCarStatus()	setRaceCarStatus(true)	1	1	PASS
Valid Data	2	setRaceCarStatus()	setRaceCarStatus()	0	0	PASS
Valid Data	3	getRaceCarStatus()	getRaceCarStatus()			

Test Plan for Truck

Test Strategy	Test #	Description	Input	Expected Output	Actual Output	Pass/Fail
Valid Data	1	setDieselStatus()	setDieselStatus(true)	1	1	PASS

Valid Data	2	SetDieselStatus()	SetDieselStatus()	0	0	PASS
ValidData	3	getDieselStatus()	getDieselStatus()			

Test Plan for Vehicle and Car

Test Strategy	Test #	Description	Input	Expected Output	Actual Output	Pass/Fail
Valid Data	1	setAge()	setAge(25)	25	25	PASS
Valid Data	2	setPrice()	setPrice(3000)	3000	3000	PASS
Valid Data	3	etRaceCarStatus()	setRaceCarStatus(true)	1	1	PASS
Valid Data	4	setAge()	setAge()	0	0	PASS
Valid Data	5	setPrice()	setPrice()	0	0	PASS
Valid Data	6	etRaceCarStatus()	setRaceCarStatus()	0	0	PASS

Test Plan for Vehicle, Car and Truck

Test Strategy	Test #	Description	Input	Expected Output	Actual Output	Pass/Fail
Valid Data	1	setAge()	setAge(30)	30	30	PASS
Valid Data	2	setPrice()	setPrice(6.00)	6.00	6	PASS
Valid Data	3	etRaceCarStatus()	setRaceCarStatus(true)	1	1	PASS
Valid Data	4	setDieselStatus()	setDieselStatus(true)	1	1	PASS
Valid Data	5	setAge()	setAge()	0	0	PASS
Valid Data	6	setPrice()	setPrice()	0	0	PASS
Valid Data	7	etRaceCarStatus()	setRaceCarStatus()	0	0	PASS
Valid Data	8	setDieselStatus()	setDieselStatus()	0	0	PASS

Provided UML Diagrams

UML Diagram for Vehicle

Vehicle
Int age
Float price

Void setAge(int a)
Void setPrice(float p)
Int getAge()
Float getPrice()

UML Diagram for Car

Car
Bool RaceCar
Void setRaceCarStatus(bool)
Bool getRaceCarStatus()

UML Diagram for Truck

Truck
Bool DieselTypeStatus
Void setDieselStatus(bool)
Bool getDieselTypeStatus()

Provided Code

[Vehicle.h]

```
/*PROGRAM: LAB 5
PROGRAMMER: SHERRY ROBBINS
DESCRIPTION:Create a program that use parent-child classes & inheritance
DATE CREATED:10/15/18
*/
#pragma once
#ifndef VEHICLE_H
#define VEHICLE_H
class Vehicle
{
public:
	Vehicle();
	void setAge(int a);
	void setPrice(float p);
	int getAge();
	float getPrice();
private:
	int age;
	float price;
};
#endif
```

[Vehicle.cpp]

```
/*PROGRAM: LAB 5
PROGRAMMER: SHERRY ROBBINS
DESCRIPTION:Create a program that use parent-child classes & inheritance
DATE CREATED:10/15/18
*/
#include<iostream>
#include "Vehicle.h"
Vehicle::Vehicle()
{
	age = 0;
	price = 0.0;
}
void Vehicle::setAge(int a)
{
	age=a;
}
void Vehicle::setPrice(float p)
{
	price=p;
}
int Vehicle::getAge()
{
	return age;
}
float Vehicle::getPrice()
{
	return price;
}
```

[Car.h]

```
#pragma once
#include "Vehicle.h"
/*PROGRAM: LAB 5
PROGRAMMER: SHERRY ROBBINS
DESCRIPTION:Create a program that use parent-child classes & inheritance
DATE CREATED:10/16/18
*/
class Car : public Vehicle
{
public:
	Car();
	void setRaceCarStatus(bool s);
	bool getRaceCarStatus();
private:
	bool RaceCarStatus;

};
```

[Car.cpp]

```
#include "Car.h"
# include <iostream>
/*PROGRAM: LAB 5
PROGRAMMER: SHERRY ROBBINS
DESCRIPTION:Create a program that use parent-child classes & inheritance
DATE CREATED:10/16/18
*/
Car::Car()
{
	RaceCarStatus = false;
}
void Car::setRaceCarStatus(bool s)
{
	RaceCarStatus = s;
}
bool Car::getRaceCarStatus()
{
	return RaceCarStatus;
}
```

[Truck.h]

```
#pragma once
#include"Vehicle.h"
/*PROGRAM: LAB 5
PROGRAMMER: SHERRY ROBBINS
DESCRIPTION:Create a program that use parent-child classes & inheritance
DATE CREATED:10/16/18
*/
class Truck : public Vehicle
{
public:
	Truck();
	void setDieselStatus(bool s);
	bool getDieselTypeStatus();
private:
	bool DieselTypeStatus;
};
```

[Truck.cpp]

```
/*PROGRAM: LAB 5
PROGRAMMER: SHERRY ROBBINS
DESCRIPTION:Create a program that use parent-child classes & inheritance
DATE CREATED:10/16/18
*/
#include "Truck.h"
Truck::Truck()
{
	DieselTypeStatus =  false;
}
void Truck::setDieselStatus(bool s)
{
	DieselTypeStatus = s;
}
bool Truck::getDieselTypeStatus()
{
	return DieselTypeStatus;
}
```

[main.cpp]

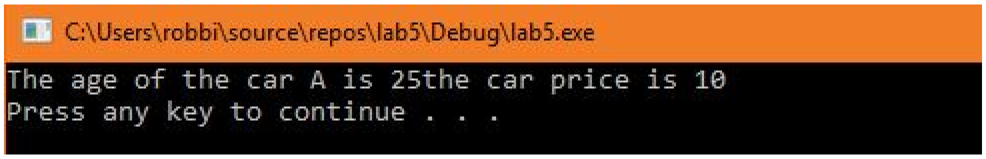
```
/*PROGRAM: LAB 5
PROGRAMMER: SHERRY ROBBINS
DESCRIPTION:Create a program that use parent-child classes & inheritance
DATE CREATED:10/15/18
*/
#include<iostream>
#include "Vehicle.h"
#include "Car.h"
#include "Truck.h"
/*This main test everything(VEHICLE CAR & TRUCK)*/
using namespace std;
int main()
{

	Vehicle a;
	a.setAge(30);
	a.setPrice(6.00);

	cout << "VEHICLE CLASS" << endl;
	cout << "The age of the car A is " << a.getAge() << "" << "the car price is "
<< a.getPrice() << endl;
	Vehicle b;
	cout << "The age of the car B is " << b.getAge() << "" << "the car price is "
<< b.getPrice() << endl;
```

```
	Car c;
	c.setAge(30);
	c.setPrice(6.00);
	c.setRaceCarStatus(true);
	cout << "CAR CLASS" << endl;
	cout << "The age of the car C is " << c.getAge() << "" << "the car price is "
<< c.getPrice() <<""<<"The race car status is"<<c.getRaceCarStatus()<< endl;
	Car d;
	cout << "The age of the car D is " << d.getAge() << "" << "the car price is "
<< d.getPrice() << "" << "The race car status is" << d.getRaceCarStatus() << endl;

	Truck e;
	e.setAge(30);
		e.setPrice(6.00);
	e.setDieselStatus(true);
	cout << "TRUCK CLASS" << endl;
	cout << "The age of the car E is " << e.getAge() << "" << "the car price is "
<< e.getPrice() << "" << "The race car status is" << e.getDieselTypeStatus()<< endl;
	Truck f;
	cout << "The age of the car F is " << f.getAge() << "" << "the car price is "
<< f.getPrice() << "" << "The race car status is" << f.getDieselTypeStatus() << endl;

	system("pause");
	return 0;
}
```

Provided Screenshots

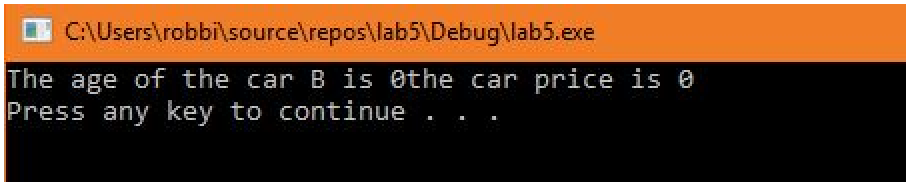
Screen Shot for Vehicle

Test 1 & Test 2



```
The age of the car A is 25the car price is 10
Press any key to continue . . .
```

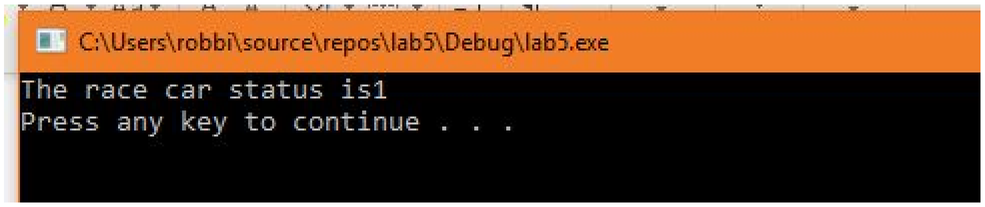
Test 5 & Test 6



```
The age of the car B is 0the car price is 0
Press any key to continue . . .
```

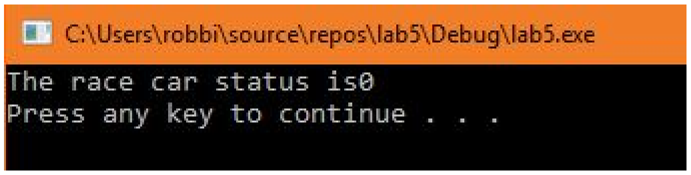
Screen Shot for Car

Test 1



```
C:\Users\robbi\source\repos\lab5\Debug\lab5.exe
The race car status is1
Press any key to continue . . .
```

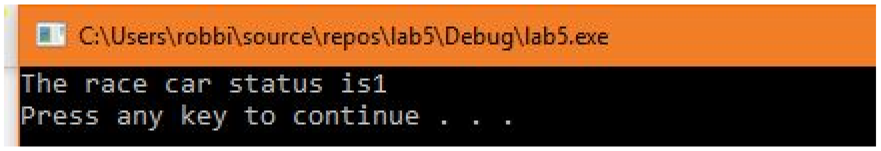
Test 2



```
C:\Users\robbi\source\repos\lab5\Debug\lab5.exe
The race car status is0
Press any key to continue . . .
```

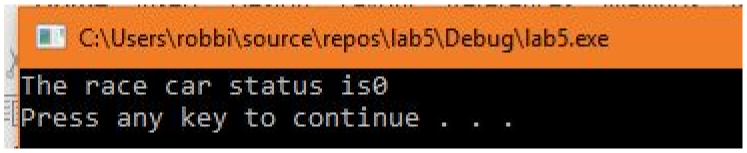
Screen Shot for Truck

Test 1



```
C:\Users\robbi\source\repos\lab5\Debug\lab5.exe
The race car status is1
Press any key to continue . . .
```

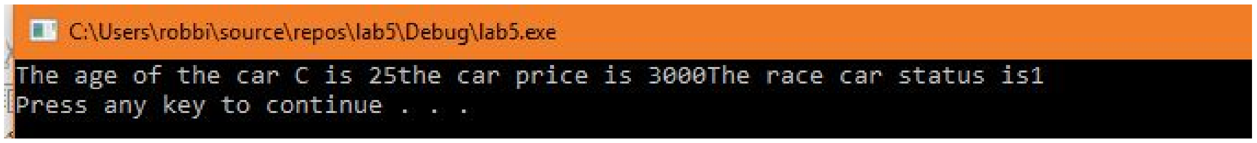
Test 2



```
C:\Users\robbi\source\repos\lab5\Debug\lab5.exe
The race car status is0
Press any key to continue . . .
```

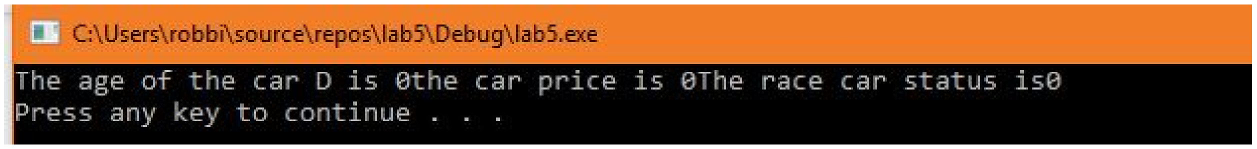
Screen Shot for Vehicle and Car

Test 1 - 3



```
C:\Users\robbi\source\repos\lab5\Debug\lab5.exe
The age of the car C is 25the car price is 3000The race car status is1
Press any key to continue . . .
```

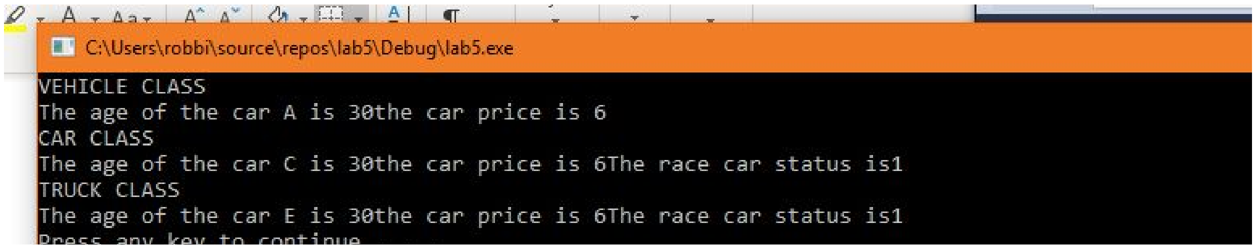
Test 4 - 6



```
C:\Users\robbi\source\repos\lab5\Debug\lab5.exe
The age of the car D is 0the car price is 0The race car status is0
Press any key to continue . . .
```

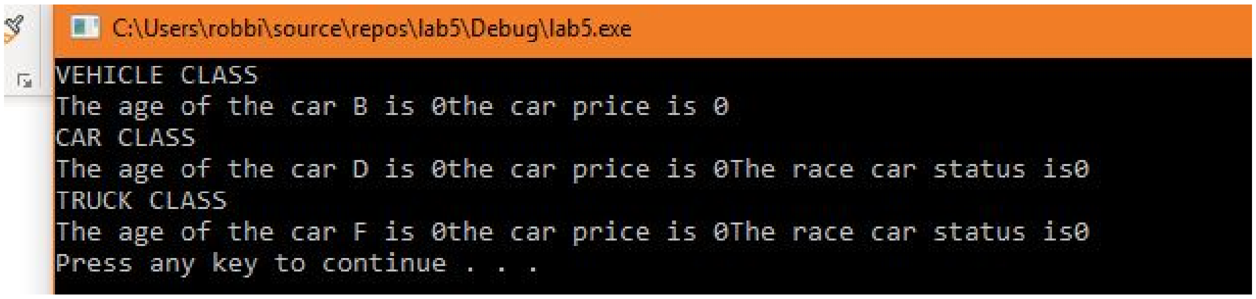
Screen Shot for Vehicle, Car and Truck

Test 1 - 4



```
C:\Users\robbi\source\repos\lab5\Debug\lab5.exe
VEHICLE CLASS
The age of the car A is 30the car price is 6
CAR CLASS
The age of the car C is 30the car price is 6The race car status is1
TRUCK CLASS
The age of the car E is 30the car price is 6The race car status is1
```

Test 5 - 6



```
C:\Users\robbi\source\repos\lab5\Debug\lab5.exe
VEHICLE CLASS
The age of the car B is 0the car price is 0
CAR CLASS
The age of the car D is 0the car price is 0The race car status is0
TRUCK CLASS
The age of the car F is 0the car price is 0The race car status is0
Press any key to continue . . .
```

Updated Algorithm

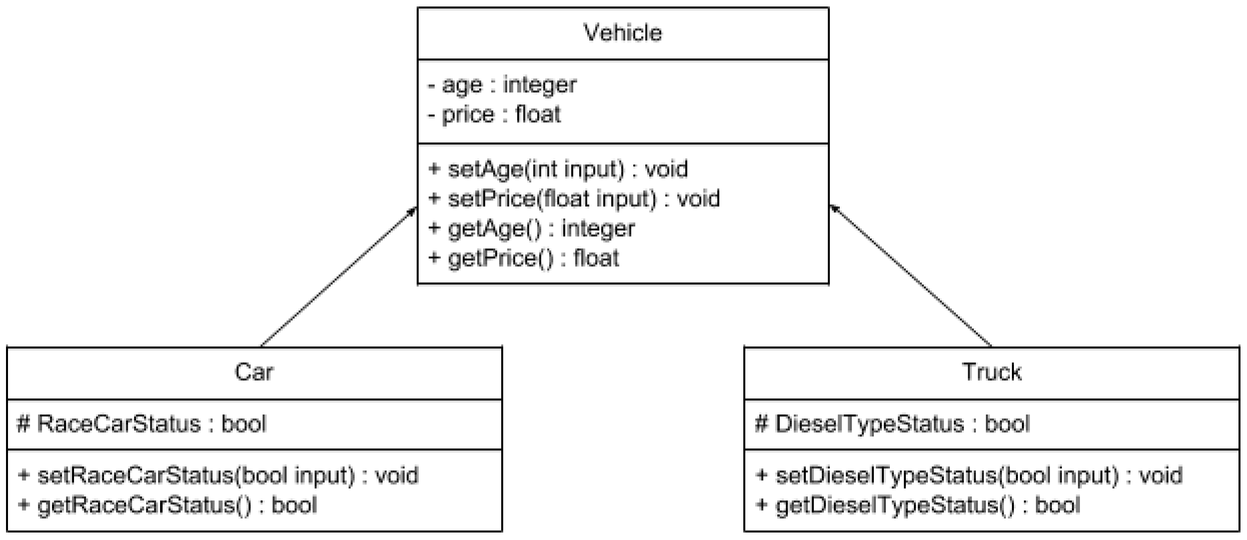
The algorithm checks out. No major revising is required. Probably could have been more detailed with the main() but given that it changes during testing I am not weighing that in my assessment.

Updated Test Cases

The test cases themselves all checked out correctly. The only issue that came up with each of these is that the screenshots and some of the string literals were inserted incorrectly making it appear that the screenshots are wrong even though they are not.

Updated UML / Generalization Diagram

The originals were incorrect with no reference to private/public/protected and they needed reformatting from the table based UML from the Microsoft Word DOCX. Also, the original Generalization Diagram was also corrupted from the conversion to gDoc so that is also unweighted (Although they can also be combined as they are below).


Vehicle
- age : integer
- price : float
+ setAge(int input) : void
+ setPrice(float input) : void
+ getAge() : integer
+ getPrice() : float
Car
# RaceCarStatus : bool
+ setRaceCarStatus(bool input) : void
+ getRaceCarStatus() : bool
Truck
# DieselTypeStatus : bool
+ setDieselTypeStatus(bool input) : void
+ getDieselTypeStatus() : bool

Updated Code

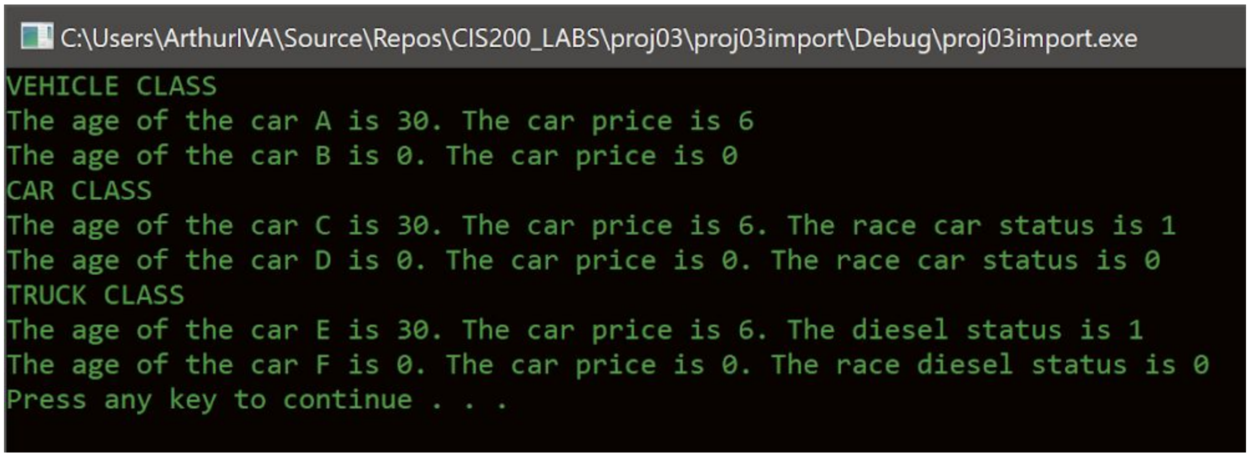
Every one of the files checked out except for my own nitpicking about the spacing of lines so those are not "updated" significantly enough to warrant showing those changes. Below is the *main()* that tested all of the different classes when brought together. This was kept just to show the minor change that was needed to fix the "incorrect" screenshots that contain *getDieselStatus()*.

[main.cpp]

```
/*PROGRAM: LAB 5
PROGRAMMER: SHERRY ROBBINS
DESCRIPTION:Create a program that use parent-child classes & inheritance
DATE CREATED:10/15/18
*/
#include<iostream>
#include "Vehicle.h"
#include "Car.h"
#include "Truck.h"
/*This main test everything(VEHICLE CAR & TRUCK)*/
using namespace std;
int main()
{

        Vehicle a;
        a.setAge(30);
        a.setPrice(6.00);

        cout << "VEHICLE CLASS" << endl;
        cout << "The age of the car A is " << a.getAge() << "" << ". the car price is " << a.getPrice() <<
endl;
        Vehicle b;
        cout << "The age of the car B is " << b.getAge() << "" << ". the car price is " << b.getPrice() <<
endl;

        Car c;
        c.setAge(30);
        c.setPrice(6.00);
        c.setRaceCarStatus(true);
        cout << "CAR CLASS" << endl;
        cout << "The age of the car C is " << c.getAge() << "" << ". the car price is " << c.getPrice()
<<""<<". The race car status is "<<c.getRaceCarStatus()<< endl;
        Car d;
        cout << "The age of the car D is " << d.getAge() << "" << ". the car price is " << d.getPrice() <<
"" << ". The race car status is " << d.getRaceCarStatus() << endl;

        Truck e;
        e.setAge(30);
                e.setPrice(6.00);
        e.setDieselStatus(true);
        cout << "TRUCK CLASS" << endl;
        cout << "The age of the car E is " << e.getAge() << "" << ". the car price is " << e.getPrice() <<
"" << "~~The race car status is~~. The diesel status is " << e.getDieselTypeStatus()<< endl;
        Truck f;
        cout << "The age of the car F is " << f.getAge() << "" << ". the car price is " << f.getPrice() <<
"" << "~~The race car status is~~. The diesel status is " << f.getDieselTypeStatus() << endl;

        system("pause");
        return 0;
}
```

Updated Screenshots

Below is the screenshot from the above *main()* to show the minor incorrect string literals and spacing that showed up in testing.



C:\Users\ArthurIVA\Source\Repos\CIS200_LABS\proj03\proj03import\Debug\proj03import.exe

```
VEHICLE CLASS
The age of the car A is 30. The car price is 6
The age of the car B is 0. The car price is 0
CAR CLASS
The age of the car C is 30. The car price is 6. The race car status is 1
The age of the car D is 0. The car price is 0. The race car status is 0
TRUCK CLASS
The age of the car E is 30. The car price is 6. The diesel status is 1
The age of the car F is 0. The car price is 0. The race diesel status is 0
Press any key to continue . . .
```

Status / Grade

All in all the code was written correctly and the algorithm was written adequately enough to match that. Everything including testing was a bit jumbled* in the original document so it was reformatted to fit this one - along with other issues that were caused by the conversion to gDoc from DOCX. Due to what I can only assume is rushing to test everything at the end of the assignment every screenshot that included *getDieselStatus*() was slightly incorrect, also formatting was an issue from this rush. Either way, after stepping through the code I discovered it was still operating correctly, just the string literals that described the tests were not accurate. Therefore not many changes were required in the end and it will be more than capable of scaling to the larger inherited class project in the following week. I give it a letter grade of C+ given my nitpicks, but a B- for actual usability/readability.

**Please check the original DOCX included in this file submission to see what I mean regarding formatting being the major issue with this assignment.*