

**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN**



FINAL PROJECT OF DESIGN PATTERN

BUILDING CAR PROJECT

Người hướng dẫn: Thầy **NGUYỄN THANH PHƯỚC**

Người thực hiện: **NGUYỄN TIẾN ĐẠT - 520H0527**

NGUYỄN TIẾN DŨNG - 518H0340

PHẠM PHƯỚC TẤN - 520H0418

Lớp : **504077**

Khoá : **24**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN**



FINAL PROJECT OF DESIGN PATTERN

BUILDING CAR PROJECT

Người hướng dẫn: Thầy **NGUYỄN THANH PHƯỚC**

Người thực hiện: **NGUYỄN TIẾN ĐẠT - 520H0527**

NGUYỄN TIẾN DŨNG - 518H0340

PHẠM PHƯỚC TẤN - 520H0418

Lớp : **504077**

Khoá : **24**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

LỜI CẢM ƠN

Cảm ơn thầy và khoa đã cho chúng em thực hiện bài báo cáo cuối kì để ứng dụng những kiến thức đã được học trong học kì học vừa qua.

CÔNG TRÌNH ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là công trình nghiên cứu của riêng chúng tôi và được sự hướng dẫn khoa học của Thầy Nguyễn Thanh Phước. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong luận văn còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung luận văn của mình. Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày 25 tháng 04 năm 2023

Tác giả

(ký tên và ghi rõ họ tên)

Đạt

Nguyễn Tiến Đạt

Dũng

Nguyễn Tiến Dũng

Tấn

Phạm Phước Tấn

TEACHER'S CONFIRMATION AND ASSESSMENT SECTION

The confirmation part of the instructor

City. Ho Chi Minh, May Day
(signature and full name)

The evaluation part of the teacher marks the test

City. Ho Chi Minh, May Day
(signature and full name)

TÓM TẮT

Thực hiện bài báo cáo với đề tài xây dựng ứng dụng hỗ trợ “Lắp ráp xe” áp dụng các loại Design Pattern khác nhau.

TABLE OF CONTENT

DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT	2
CÁC KÝ HIỆU	2
CÁC CHỮ VIẾT TẮT	2
DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ	3
DANH MỤC HÌNH	3
DANH MỤC BẢNG	3
Part A: Functional Requirements	4
1. Screenshots of completed construction	4
1.1 Singleton Pattern	4
1.2 Strategy Pattern	5
1.3 Factory Pattern	6
1.4 Builder Pattern	7
1.5 Decorator Pattern	9
1.6 Observer Pattern	10
1.7 UI In Windows form	11
Part B: Architectural Design	13
1. Describe the project	13
2. Describe the technology	13
Part C: Detailed design	14
1. ERD or DB Schema	14
2. List of design patterns	14
3. Class diagram	15
3.1 Class diagram of Singleton Pattern	15
3.2 Class diagram of Strategy Pattern	16
3.3 Class diagram of Factory Pattern	16
3.4 Class diagram of Builder Pattern	17
3.5 Class diagram of Prototype Pattern	17
3.6 Class diagram of Decorator Pattern	18
3.7 Class diagram of Observer Pattern	18
3.8 Class diagram of system	19
4. Repository	19
Part D: Self-assessment	20

DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT

CÁC KÝ HIỆU

CÁC CHỮ VIẾT TẮT

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ
DANH MỤC HÌNH
DANH MỤC BẢNG

Part A: Functional Requirements

1. Screenshots of completed construction

1.1 Singleton Pattern

```

1  using MySql.Data.MySqlClient;
2  using System;
3  using System.Collections.Generic;
4  using System.Data;
5  using System.Linq;
6  using System.Text;
7  using System.Threading.Tasks;
8
9  namespace ck
10 {
11
12     public class DataProvider
13     {
14         private static DataProvider instance; // Ctrl + R + E
15
16         public static DataProvider Instance
17         {
18             get { if (instance == null) instance = new DataProvider(); return DataProvider.instance; }
19             private set { DataProvider.instance = value; }
20         }
21
22         private DataProvider() { }
23
24         private string strconn = "Server=localhost; Database=cardp; User Id=root;Password=kocomatkau;Character Set=UTF8";
25

```

```

26     public DataTable ExecuteQuery(string query, object[] parameter = null)
27     {
28         DataTable data = new DataTable();
29         using (MySqlConnection connection = new MySqlConnection(strconn))
30         {
31             connection.Open();
32             MySqlCommand command = new MySqlCommand(query, connection);
33
34             if (parameter != null)
35             {
36                 string[] listPara = query.Split(' ');
37                 int i = 0;
38                 foreach (string item in listPara)
39                 {
40                     if (item.Contains('@'))
41                     {
42                         command.Parameters.AddWithValue(item, parameter[i]);
43                         i++;
44                     }
45                 }
46             }
47             MySqlDataAdapter adapter = new MySqlDataAdapter(command);
48             adapter.Fill(data);
49
50             connection.Close();
51         }
52
53         return data;
54     }
55

```

1.2 Strategy Pattern

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ck.Strategy
8  {
9      public class StrategyPattern
10     {
11         private IStrategy strategy;
12         public StrategyPattern(IStrategy strategy)
13         {
14             this.strategy = strategy;
15         }
16         public float SetStrategy(IStrategy strategy)
17         {
18             return strategy.maxCapacity();
19         }
20     }
21 }
22
23
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ck.Strategy
8  {
9      public class GasolineStrategy : IStrategy
10     {
11         public float maxCapacity()
12         {
13             return 100;
14         }
15     }
16 }
17
```

1.3 Factory Pattern

```
124     switch (carType)
125     {
126         case CarType.Diesel:
127             DieselCarBuilder dCarBuilder = new DieselCarBuilder();
128             dCarBuilder.setWheel(wheel);
129             dCarBuilder.setEngine(engine);
130             dCarBuilder.setColor(color);
131             dCarBuilder.setGear(gearBox);
132             dCarBuilder.setModel(modelCar);
133             dCarBuilder.setMaxTankVolume(maxFuelCapacity);
134             dCarBuilder.setFuelTankeVolume(fuelCapacity);
135             dCarBuilder.setRecommentGas(recommentGas);
136
137             return dCarBuilder.BuildCar();
138             //break;
139         case CarType.Electric:
140             ElectricCarBuilder eCarBuilder = new ElectricCarBuilder();
141             eCarBuilder.setWheel(wheel);
142             eCarBuilder.setEngine(engine);
143             eCarBuilder.setColor(color);
144             eCarBuilder.setGear(gearBox);
145             eCarBuilder.setModel(modelCar);
146             eCarBuilder.setMaxBetteryCapacity(maxFuelCapacity);
147             eCarBuilder.setBetteryCapacity(fuelCapacity);
148
149             return eCarBuilder.BuildCar();
150             //break;
151         case CarType.Gasoline:
152             GasolineCarBuilder gCarBuilder = new GasolineCarBuilder();
153             gCarBuilder.setWheel(wheel);
154             gCarBuilder.setEngine(engine);
155             gCarBuilder.setColor(color);
156             gCarBuilder.setGear(gearBox);
157             gCarBuilder.setModel(modelCar);
158             gCarBuilder.setMaxTankVolume(maxFuelCapacity);
159             gCarBuilder.setFuelTankeVolume(fuelCapacity);
160             gCarBuilder.setRecommentGas(recommentGas);
161
162             return gCarBuilder.BuildCar();
163             //break;
164     }
165     return null;
166 }
167 }
168 }
```

1.4 Builder Pattern

```
1  using Cuoiky_DP.DataObject.Colors;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace Cuoiky_DP.DataObject.Produce
9  {
10     internal abstract class CarBuilder
11     {
12         public Wheel _wheel;
13         public Engine _engine;
14         public CarColor _color;
15         public string _gearBox;
16         public string _modelCar;
17
18         public void setWheel(Wheel wheel)
19         {
20             _wheel = wheel;
21         }
22
23         public void setEngine(Engine engine)
24         {
25             _engine = engine;
26         }
27
28         public void setColor(CarColor color)
29         {
30             _color = color;
31         }
32         public void setGear(string gearBox)
33         {
34             _gearBox = gearBox;
35         }
36         public void setModel(string modelCar)
37         {
38             _modelCar = modelCar;
39         }
40
41         public abstract Car BuildCar();
42     }
43 }
```

```

1  using Cuoiky_DP.DataObject.TypeCar;
2  using System;
3  using System.Collections.Generic;
4  using System.Drawing;
5  using System.Linq;
6  using System.Text;
7  using System.Threading.Tasks;
8
9  namespace Cuoiky_DP.DataObject.Produce
10 {
11     internal class DieselCarBuilder : CarBuilder
12     {
13         public string _RecommentGas;
14         public float _FuelTankVolume;
15         public float _MaxTankVolume;
16         public static DieselCar dieselCar = new DieselCar();
17
18         public void setRecommentGas(string recommentGas)
19         {
20             _RecommentGas = recommentGas;
21         }
22         public void setFuelTankeVolume(float fuelTankeVolume)
23         {
24             _FuelTankVolume = fuelTankeVolume;
25         }
26         public void setMaxTankVolume(float maxTankVolume)
27         {
28             _MaxTankVolume = maxTankVolume;
29         }
30         public override Car BuildCar()
31         {
32             DieselCar dCar = (DieselCar)dieselCar.Clone();
33
34             dCar.wheel = _wheel;
35             dCar.engine = _engine;
36             dCar.color = _color;
37             dCar.modelCar = _modelCar;
38             dCar.gearBox = _gearBox;
39             dCar.recommentGas = _RecommentGas;
40             dCar.maxTankVolume = _MaxTankVolume;
41             dCar.fuleTankVolume = _FuelTankVolume;
42
43             return dCar;
44         }
45     }
46 }

```


1.5 Decorator Pattern

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ck.Decorator
8  {
9      public abstract class CarDecorator:IDecorator
10     {
11         protected IDecorator decorator;
12         public CarDecorator(IDecorator decorator)
13         {
14             this.decorator = decorator;
15         }
16
17         public virtual string name()
18         {
19             return decorator.name();
20         }
21     }
22 }
23

```

```

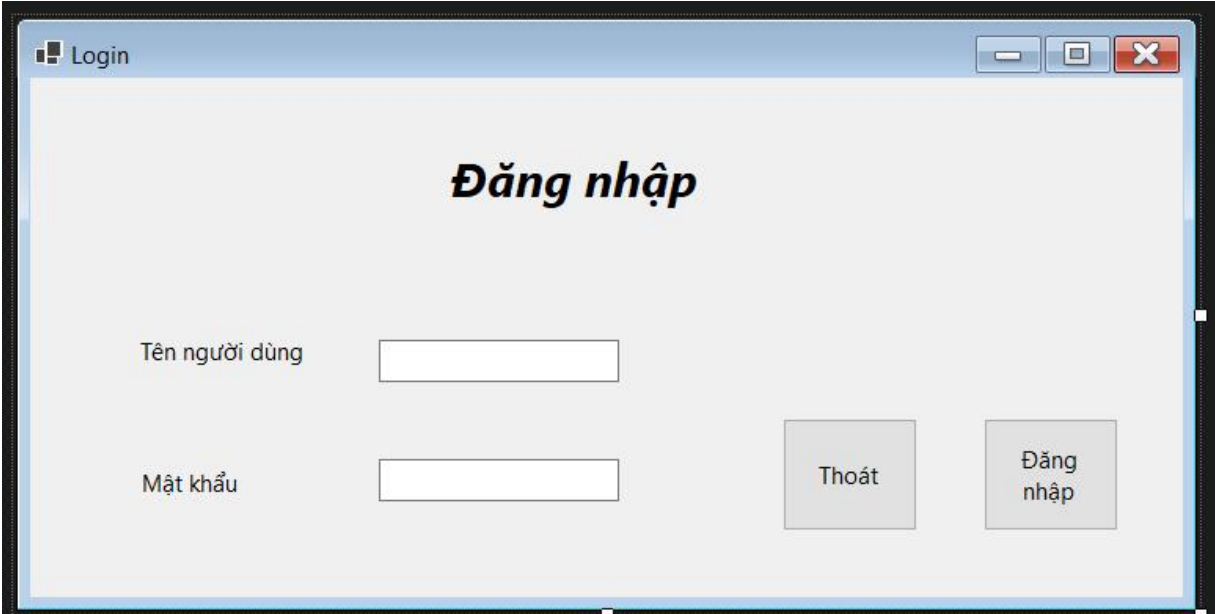
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ck.Decorator
8  {
9      public class DEngine01:CarDecorator
10     {
11         public DEngine01(IDecorator decorator) : base(decorator)
12         {
13         }
14         public override string name()
15         {
16             addEngine();
17             return base.name() + addEngine();
18         }
19         public string addEngine()
20         {
21             return "DEngine01";
22         }
23     }
24 }

```

1.6 Observer Pattern

```
public class Customer : Observer
{
    private string name;
    public string Name
    {
        get { return name; }
        set { name = value; }
    }
    private string phoneNumber;
    public string PhoneNumber
    {
        get { return phoneNumber; }
        set { phoneNumber = value; }
    }
    public override void ObserverInfo(string info)
    {
        MessageBox.Show("Thông báo tới khách hàng: " + Name + " Số điện thoại là: "
            + PhoneNumber + " Thông tin sản phẩm: " + info);
    }
    public Customer(string name, string phone)
    {
        this.Name = name;
        this.PhoneNumber = phone;
    }
}
```


1.7 UI In Windows form



The image shows a Windows form titled "Login". The window has a title bar with standard minimize, maximize, and close buttons. The main content area has a light gray background. At the top center, the text "Đăng nhập" is displayed in a bold, italicized font. Below this, there are two input fields. The first is labeled "Tên người dùng" (Username) and the second is labeled "Mật khẩu" (Password). To the right of the password field, there are two buttons: "Thoát" (Exit) and "Đăng nhập" (Login).

Login

Đăng nhập

Tên người dùng

Mật khẩu

Thoát Đăng nhập



The image shows a Windows form titled "Phần mềm bán xe" (Car Sales Software). The window has a title bar with standard minimize, maximize, and close buttons. The main content area has a light gray background. At the top center, the text "Chào mừng quý khách đã đến với cửa hàng chúng tôi" (Welcome to our store) is displayed in a bold, italicized font. Below this, there are four blue panels with white text and radio buttons. The "TypeCar" panel has three options: Gasoline, Diesel, and Electric, each with a corresponding dropdown menu. The "Color" panel has five options: Black, Green, White, Blue, and Red. The "GearBox" panel has two options: Manual and Automatic. The "Model Car" panel has four options: Sedan, Hatchback, SUV, and Sport. Below these panels, there is a "Wheel Type" panel with three options: DTDModel01_26, DTDModel02_27, and DTDModel03_29. At the bottom left, there is a "Đặt hàng" (Order) button and a black rectangular area. At the bottom right, there are two buttons: "Thông báo" (Notification) and "Thoát" (Exit).

Phần mềm bán xe

Chào mừng quý khách đã đến với cửa hàng chúng tôi

TypeCar

☐ Gasoline Gasoline

☐ Diesel Diesel

☐ Electric Electric

Color

☐ Black ☐ Green

☐ White ☐ Blue

☐ Red

GearBox

☐ Manual

☐ Automatic

Model Car

☐ Sedan ☐ Hatchback

☐ SUV

☐ Sport

Wheel Type

☐ DTDModel01_26

☐ DTDModel02_27

☐ DTDModel03_29

Đặt hàng

Thông báo Thoát

Ketqua

Sản phẩm đã đặt

label2

Ý kiến đóng góp


Trang chủ

Cảm ơn quý khách đã mua hàng

Gửi

Thoát

cosodulieu



Trở lại

Part B: Architectural Design

1. Describe the project

This is a project about building an application "Assembling and manufacturing vehicles". The application will be used to create car models from many different components such as chassis, engine, fuel type, ...

Including 3 main types of vehicles, namely petrol cars, electric cars and diesel cars. Each type of vehicle will be distinguished by using different engines, as well as different types of fuel.

And some details can be shared for all vehicles such as wheels, chassis, color, gearbox, ...

The application allows users to select the components of the car and then assemble it into a complete car. After creating a vehicle version, if there is a need to change, the user can also replace other vehicle details with the original design.

2. Describe the technology

In this project, the main programming language C# is used in Visual Studio.

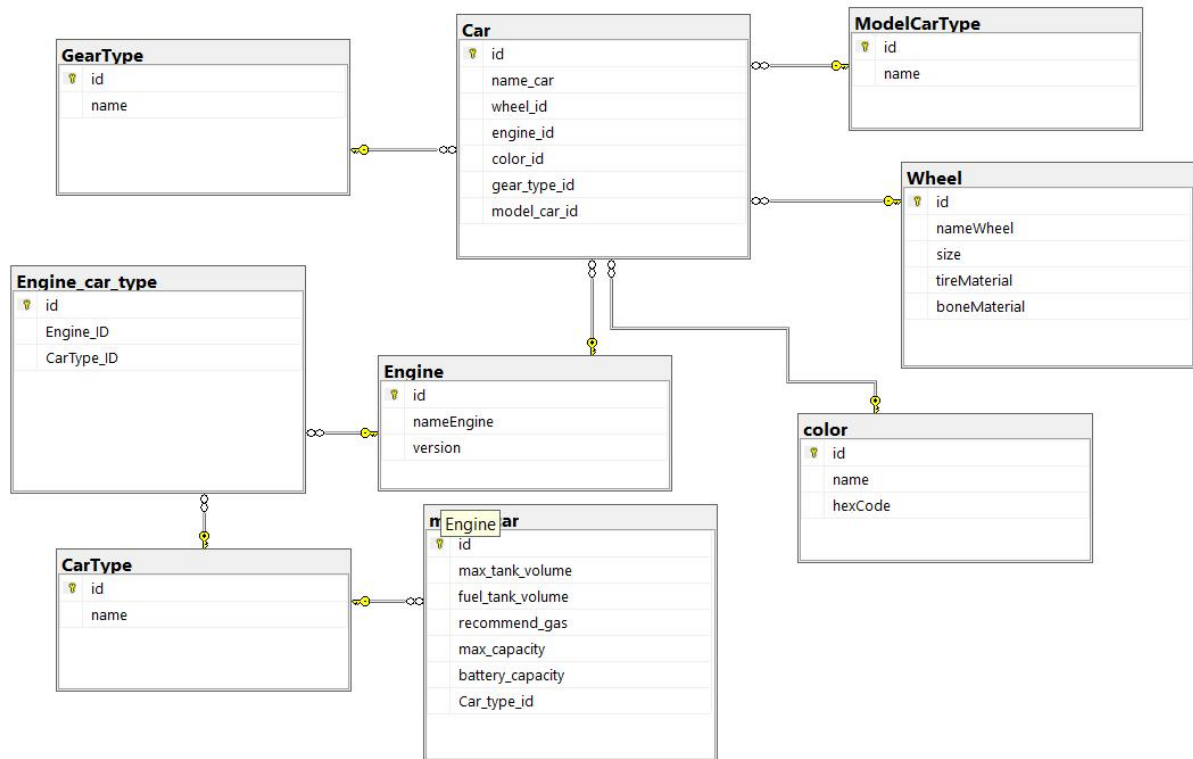
The UI part uses Window form technology to perform the interactions with the application.

Database part created with mySQL.

The project used a total of 7 design patterns: Singleton Pattern, Strategy Pattern, Builder Pattern, Factory Pattern, Prototype Pattern, Decorator Pattern and Observer Pattern.

Part C: Detailed Design

1. Database Schema



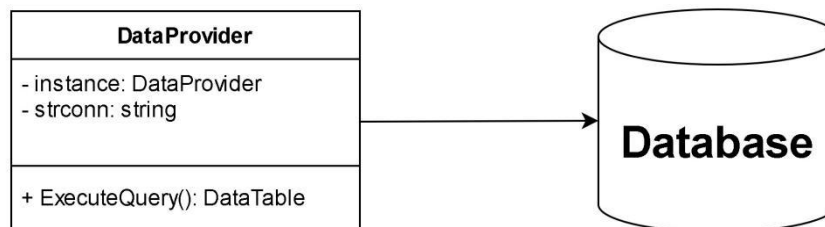
2. List of design patterns

STT	Design pattern	Describe
1	Singleton Pattern	Using Singleton Pattern to create objects to link databases.
2	Strategy Pattern	Using Strategy Pattern to classify the fueling methods of Gasoline, Electric and Diesel cars.
3	Factory Pattern	Using Factory Pattern to create different types of cars.

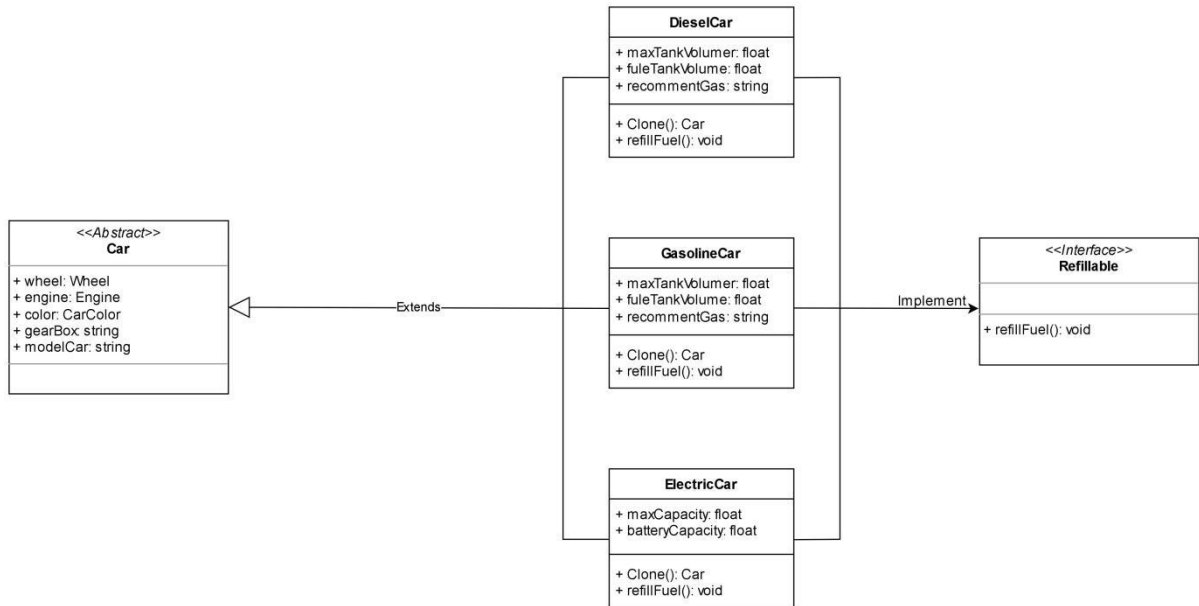
4	Builder Pattern	Using Builder Pattern to create a car according to the user's requirements.
5	Prototype Pattern	Using Prototype Pattern to create a new object by copying, this avoids the time and memory consumption like using "New".
6	Decorator Pattern	Using Decorator Pattern to customize previously created cars.
7	Observer Pattern	Use Observer Pattern to be able to notify users when using functions.

3. Class Diagram

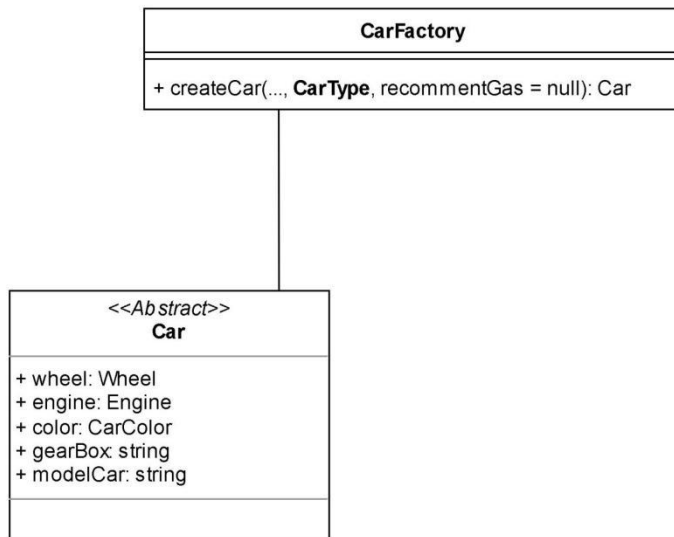
3.1 Class diagram of Singleton Pattern



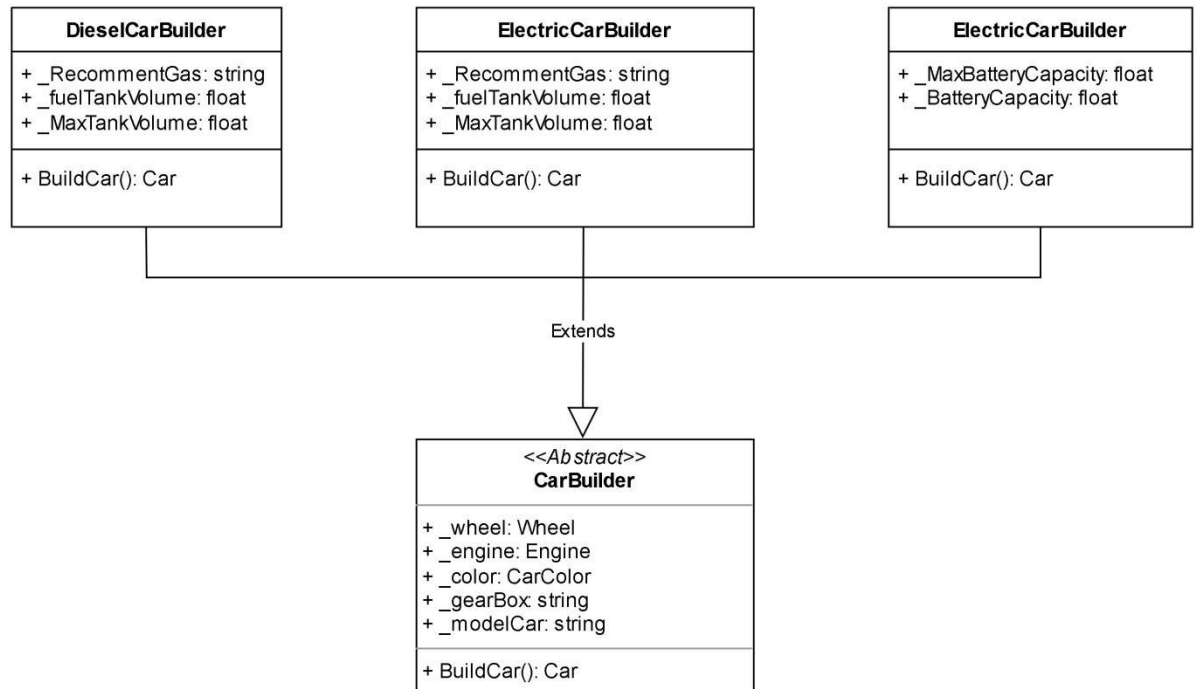
3.2 Class diagram of Strategy Pattern



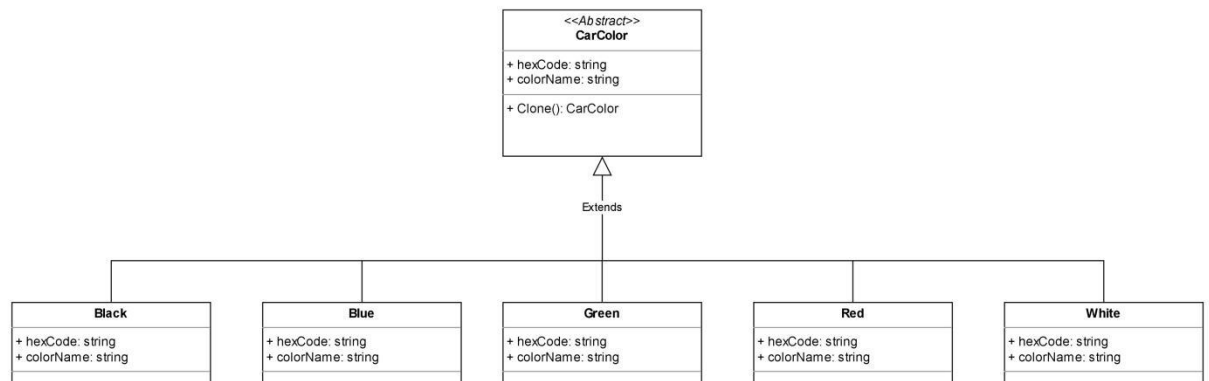
3.3 Class diagram of Factory Pattern



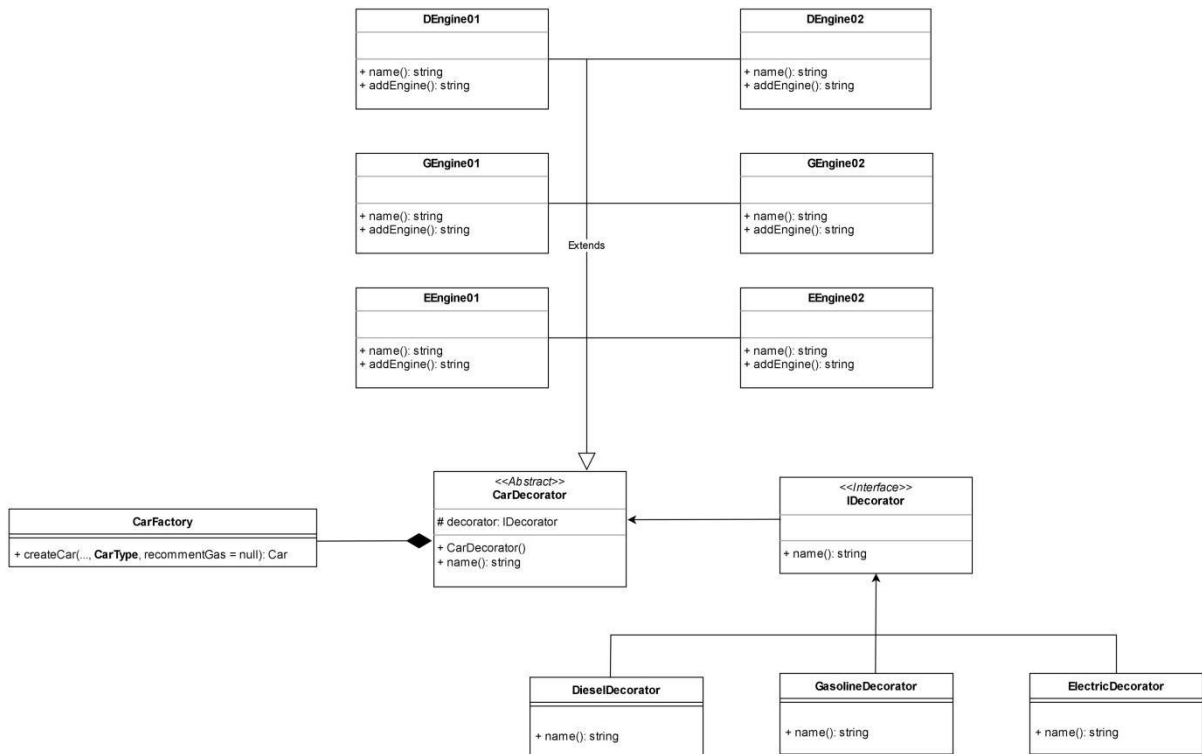
3.4 Class diagram of Builder Pattern



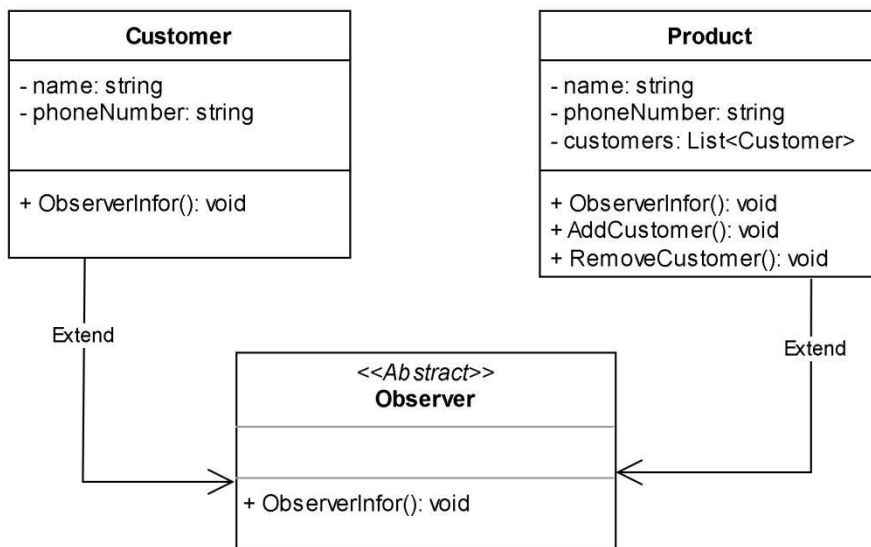
3.5 Class diagram of Prototype Pattern



3.6 Class diagram of Decorator Pattern

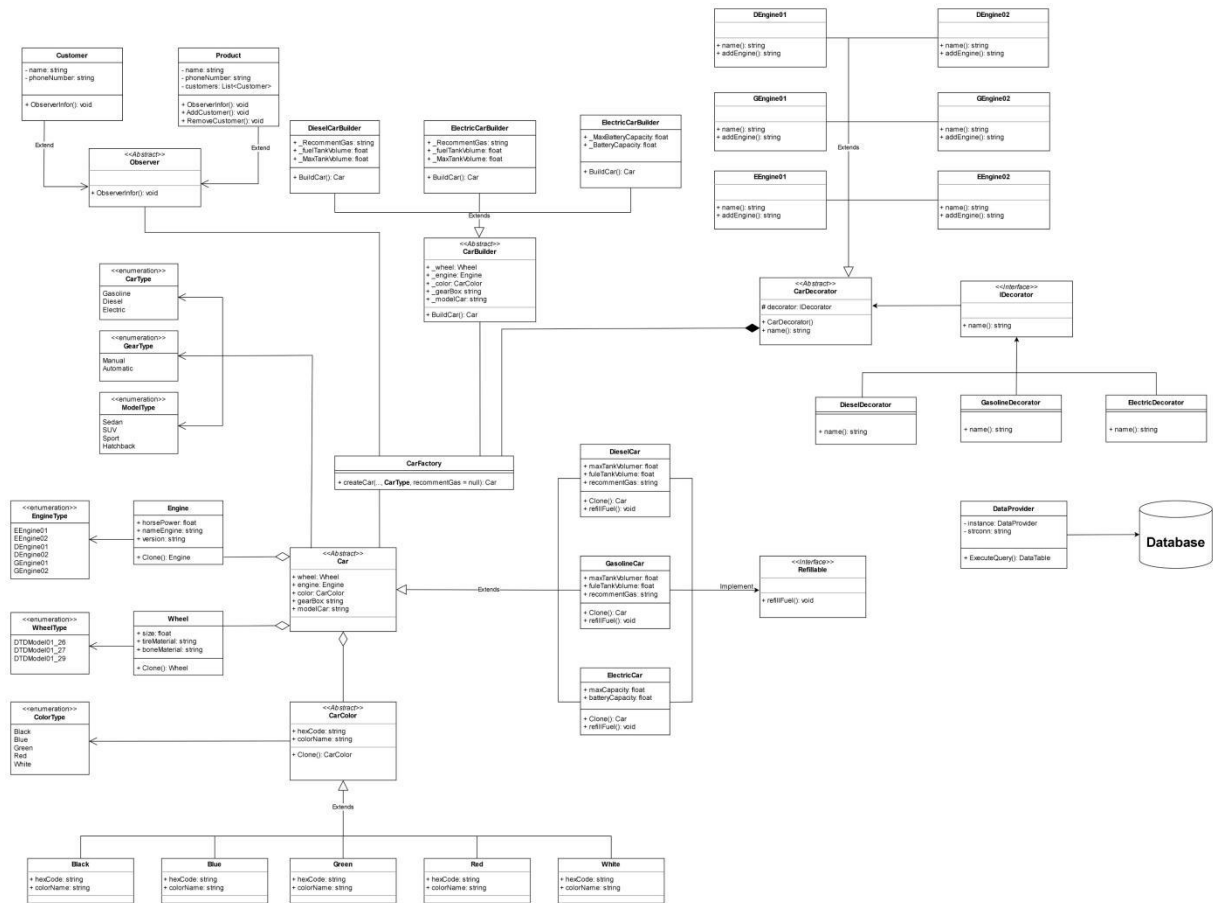


3.7 Class diagram of Observer Pattern



3.8 Class diagram of system

[Link](#) to see clear class diagram (using email TDTU to see).



4. Repository

To direct our repository: [Link](#)

Part D: Self-assessment

Nội dung tiêu chí	0 điểm	1/2 điểm	Trên 1/2 hoặc trọn điểm	Điểm
Lý do Pattern áp dụng	Không có hoặc không hợp lý hoặc gượng ép	Có nhưng chưa thể hiện rõ	Rõ ràng, hợp lý	0.3 đ
Sơ đồ lớp	Không có hoặc vẽ sai, vẽ không hợp lý	Vẽ hợp lý cho bài toán nhưng còn có điểm sai trong sơ đồ	Sơ đồ đúng, hợp lý	0.5đ
Code Pattern áp dụng	Không có hoặc code sai hoặc code bị lỗi	Có code nhưng chưa đủ pattern hoặc sai phần nhỏ	Code đúng	0.7đ

Pattern	Lý do Pattern áp dụng	Sơ đồ lớp	Code Pattern áp dụng	Tổng điểm
Singleton Pattern	0.15đ	0.25đ	0.7đ	1.1đ
Strategy Pattern	0.15đ	0.25đ	0.7đ	1.1đ
Factory Pattern	0.3đ	0.25đ	0.7đ	1.25đ
Builder Pattern	0.3đ	0.25đ	0.7đ	1.25đ
Prototype Pattern	0.3đ	0.25đ	0.7đ	1.25đ
Decorator Pattern	0.3đ	0.25đ	0.7đ	1.25đ
Observer Pattern	0.3đ	0.25đ	0.7đ	1.25đ
Điểm Pattern	8 đ			
Báo cáo	1.5 đ			
Điểm cộng	1 đ			
Tổng điểm	10.5 đ			