Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут”

Кафедра АСОІУ

**ЗВІТ**

про виконання комп’ютерного практикуму №1

з дисципліни

«Сучасні технології програмування»

Тема: «Шаблони проектування. Породжуючі шаблони»

Варіант 2

|  |  |
| --- | --- |
| Прийняв: | Виконав: |
| Ліщук  Катерина  Ігорівна | студент 3-го курсу  гр. ІП-51 ФІОТ  Гладишко  Валентин  Юрійович |

Київ – 2017

**ЗМІСТ:**

1. Мета РОБОТИ 3
2. Постановка задачі 4
3. Виконання завдання 5
4. Висновок 11
5. ЛІСТИНГ КОДУ ПРОГРАМИ 12

# Мета РОБОТИ

Ознайомитися з основними шаблонами проектування, навчитися застосовувати їх при проектуванні і розробці ПЗ.

# Постановка задачі

При виконанні комп'ютерного практикуму слід реалізувати наступні дії:

1. Реалізувати завдання згідно варіантів, наведених нижче (можна запропонувати реалізацію власного завдання).
2. Коротко описати архітектуру проекту.
3. Наведіть UML-діаграму класів.
4. Описати використовувані породжуючі паттерни. Для кожного паттерну потрібно вказати складові класи і їх призначення.

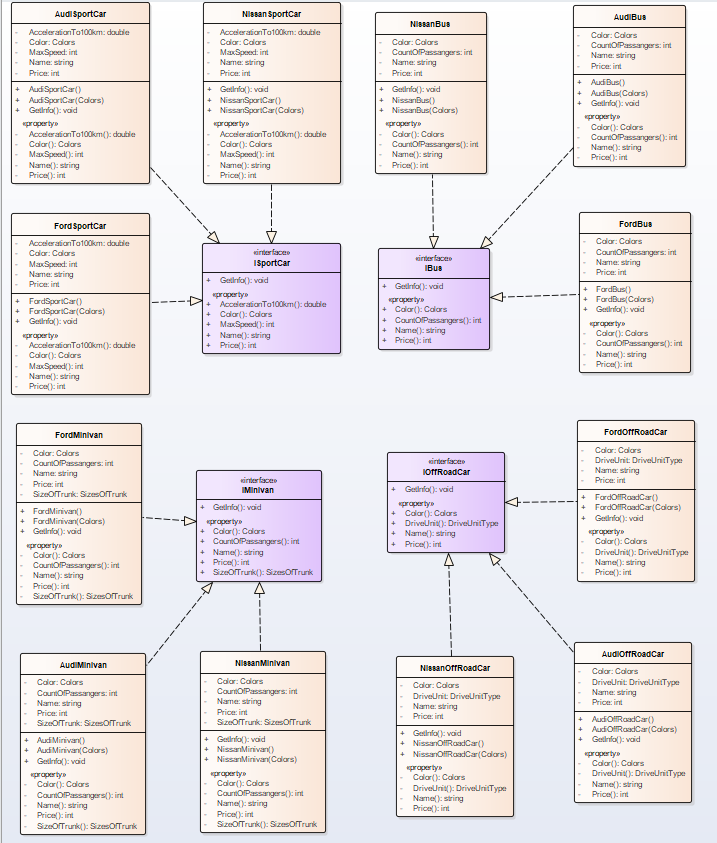
Реалізувати завдання "Заводи з виробництва автомобілів". Повинна бути реалізована можливість створювати автомобілі різних типів на різних заводах.

# Виконання завдання

* 1. **Короткий опис архітектури проекту**

В програмі є основний клас Клієнт. Він містить в собі фабрику з виробництва автомобілів. Всі дії з фабрикою (наприклад створити авто) виконуються через клас Клієнта. Клас Фабрика створює екземпляри класів Спортивний Автомобіль, Позашляховик, Автобус, Мінівен.

* 1. **UML-діаграми класів**

****

# C:\Users\Kappi\Source\Repos\KPI-Csharp-labs\lab1.1\ClassDiagram2.png

* 1. **Опис Паттерну, який був використаний для ПЗ.**

Для даної задачі був використаний паттерн **Абстрактна фабрика**.

**Абстрактна фабрика** - це породжучий паттерн проектування, який дозволяє створювати сімейства пов'язаних об'єктів, не прив'язуючись до конкретних класів створюваних об'єктів.

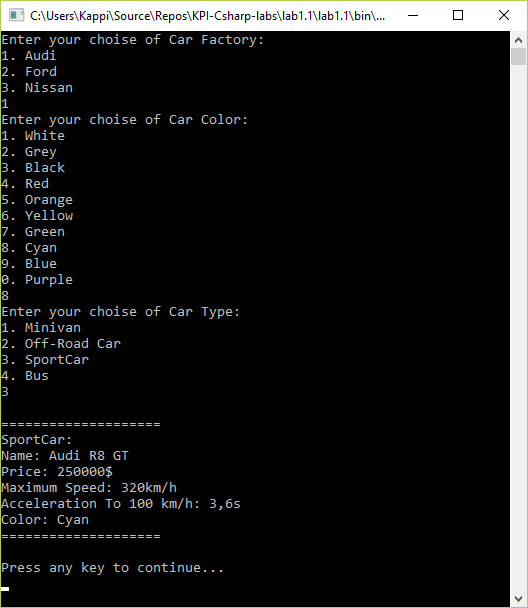
Складові класи:

* **Абстрактні продукти** оголошують інтерфейси продуктів, які пов'язані один з одним за змістом, але виконують різні функції.
* **Конкретні продукти** - великий набір класів, які відносяться до різних абстрактних продуктів (Спортивний автомобіль / Автобус), але мають одні і ті ж варіації (Nissan / Audi).
* **Абстрактна фабрика** оголошує методи створення різних абстрактних продуктів (Спортивний автомобіль / Автобус).
* **Конкретні фабрики** відносяться кожна до своєї варіації продуктів (Nissan / Audi) і реалізують методи абстрактної фабрики, дозволяючи створювати всі продукти певної варіації.

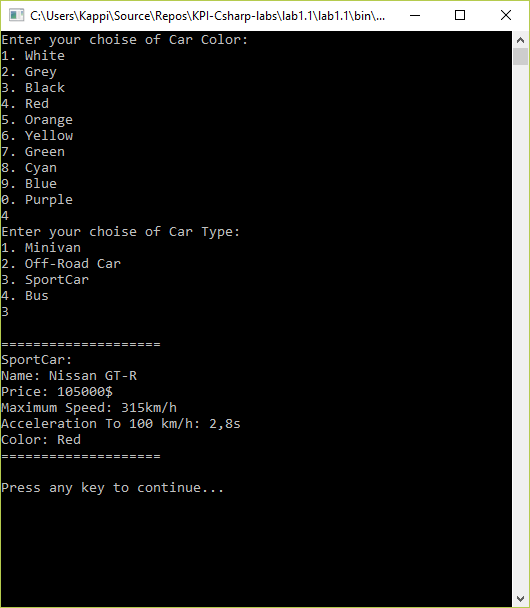
Цей паттерн був використаний тому що на різних заводах ми повинні мати можливість виготовляти автомобілі однакового типу.

* 1. **Результати виконання програми**

*Результат виконання 1:*



*Результат виконання 2:*



# Висновок

Отже, я ознайомився з основними шаблонами проектування і навчився застосовувати їх при розробці ПЗ. За допомогою шаблону Абстрактна фабрика я реалізував своє завдання згідно варіанту. Також я побудував діаграму класів для своєї предметної області враховуючи вимоги шаблону.

Посилання на виконану лабораторну:

<https://github.com/ValentineGladyshko/KPI-Csharp-labs/tree/master/lab1.1>

1. **ЛІСТИНГ КОДУ ПРОГРАМИ**

«Program.cs»

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab1.\_1

{

class Program

{

static void Main(string[] args)

{

Client client = new Client();

client.ChoiseCarType(client.ChoiseColor());

Console.WriteLine("Press any key to continue...");

Console.ReadKey();

}

}

}

«Client.cs»

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab1.\_1

{

enum Colors

{

White,

Grey,

Black,

Red,

Orange,

Yellow,

Green,

Cyan,

Blue,

Purple

}

class Client

{

private ICarFactory CarFactory;

public Client()

{

ChoiseCarFactory();

}

public void ChoiseCarFactory()

{

while (true)

{

Console.WriteLine("Enter your choise of Car Factory:\n1. Audi\n2. Ford\n3. Nissan");

char choise = Convert.ToChar(Console.ReadLine());

switch (choise)

{

case '1':

CarFactory = new AudiCarFactory();

return;

case '2':

CarFactory = new FordCarFactory();

return;

case '3':

CarFactory = new NissanCarFactory();

return;

default:

Console.WriteLine("Incorrect input. Try again");

break;

}

}

}

public Colors ChoiseColor()

{

while (true)

{

Console.WriteLine("Enter your choise of Car Color:\n1. White\n2. Grey\n3. Black" +

"\n4. Red\n5. Orange\n6. Yellow\n7. Green\n8. Cyan\n9. Blue\n0. Purple");

char choise = Convert.ToChar(Console.ReadLine());

switch (choise)

{

case '1':

return Colors.White;

case '2':

return Colors.Grey;

case '3':

return Colors.Black;

case '4':

return Colors.Red;

case '5':

return Colors.Orange;

case '6':

return Colors.Yellow;

case '7':

return Colors.Green;

case '8':

return Colors.Cyan;

case '9':

return Colors.Blue;

case '0':

return Colors.Purple;

default:

Console.WriteLine("Incorrect input. Try again");

break;

}

}

}

public void ChoiseCarType(Colors Color)

{

while (true)

{

Console.WriteLine("Enter your choise of Car Type:\n1. Minivan\n2. Off-Road Car\n3. SportCar\n4. Bus");

char choise = Convert.ToChar(Console.ReadLine());

switch (choise)

{

case '1':

CarFactory.CreateMinivan(Color).GetInfo();

return;

case '2':

CarFactory.CreateOffRoadCar(Color).GetInfo();

return;

case '3':

CarFactory.CreateSportCar(Color).GetInfo();

return;

case '4':

CarFactory.CreateBus(Color).GetInfo();

return;

default:

Console.WriteLine("Incorrect input. Try again");

break;

}

}

}

}

}

«CarFactory.cs»

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab1.\_1

{

interface ICarFactory

{

IMinivan CreateMinivan();

IOffRoadCar CreateOffRoadCar();

ISportCar CreateSportCar();

IBus CreateBus();

IMinivan CreateMinivan(Colors Color);

IOffRoadCar CreateOffRoadCar(Colors Color);

ISportCar CreateSportCar(Colors Color);

IBus CreateBus(Colors Color);

}

class AudiCarFactory : ICarFactory

{

public IMinivan CreateMinivan()

{

return new AudiMinivan();

}

public IOffRoadCar CreateOffRoadCar()

{

return new AudiOffRoadCar();

}

public ISportCar CreateSportCar()

{

return new AudiSportCar();

}

public IBus CreateBus()

{

return new AudiBus();

}

public IMinivan CreateMinivan(Colors Color)

{

return new AudiMinivan(Color);

}

public IOffRoadCar CreateOffRoadCar(Colors Color)

{

return new AudiOffRoadCar(Color);

}

public ISportCar CreateSportCar(Colors Color)

{

return new AudiSportCar(Color);

}

public IBus CreateBus(Colors Color)

{

return new AudiBus(Color);

}

}

class FordCarFactory : ICarFactory

{

public IMinivan CreateMinivan()

{

return new FordMinivan();

}

public IOffRoadCar CreateOffRoadCar()

{

return new FordOffRoadCar();

}

public ISportCar CreateSportCar()

{

return new FordSportCar();

}

public IBus CreateBus()

{

return new FordBus();

}

public IMinivan CreateMinivan(Colors Color)

{

return new FordMinivan(Color);

}

public IOffRoadCar CreateOffRoadCar(Colors Color)

{

return new FordOffRoadCar(Color);

}

public ISportCar CreateSportCar(Colors Color)

{

return new FordSportCar(Color);

}

public IBus CreateBus(Colors Color)

{

return new FordBus(Color);

}

}

class NissanCarFactory : ICarFactory

{

public IMinivan CreateMinivan()

{

return new NissanMinivan();

}

public IOffRoadCar CreateOffRoadCar()

{

return new NissanOffRoadCar();

}

public ISportCar CreateSportCar()

{

return new NissanSportCar();

}

public IBus CreateBus()

{

return new NissanBus();

}

public IMinivan CreateMinivan(Colors Color)

{

return new NissanMinivan(Color);

}

public IOffRoadCar CreateOffRoadCar(Colors Color)

{

return new NissanOffRoadCar(Color);

}

public ISportCar CreateSportCar(Colors Color)

{

return new NissanSportCar(Color);

}

public IBus CreateBus(Colors Color)

{

return new NissanBus(Color);

}

}

}

«Bus.cs»

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab1.\_1

{

interface IBus

{

string Name

{

get;

set;

}

int CountOfPassangers

{

get;

set;

}

int Price

{

get;

set;

}

Colors Color

{

get;

set;

}

void GetInfo();

}

class AudiBus : IBus

{

private string Name;

private int CountOfPassangers;

private int Price;

private Colors Color;

public AudiBus()

{

Name = "Audi Bus";

CountOfPassangers = 60;

Price = 200000;

this.Color = Colors.Grey;

}

public AudiBus(Colors Color)

{

Name = "Audi Bus";

CountOfPassangers = 60;

Price = 200000;

this.Color = Color;

}

string IBus.Name { get => Name; set => Name = value; }

int IBus.CountOfPassangers { get => CountOfPassangers; set => CountOfPassangers = value; }

int IBus.Price { get => Price; set => Price = value; }

Colors IBus.Color { get => Color; set => Color = value; }

public void GetInfo()

{

Console.WriteLine("\n====================\nBus:\nName: {0}\nCount Of Passangers: {1}\n" +

"Price: {2}$\nColor: {3}\n====================\n", Name, CountOfPassangers, Price, Color);

}

}

class FordBus : IBus

{

private string Name;

private int CountOfPassangers;

private int Price;

private Colors Color;

public FordBus()

{

Name = "Ford Bus";

CountOfPassangers = 48;

Price = 90000;

this.Color = Colors.Red;

}

public FordBus(Colors Color)

{

Name = "Ford Bus";

CountOfPassangers = 48;

Price = 90000;

this.Color = Color;

}

string IBus.Name { get => Name; set => Name = value; }

int IBus.CountOfPassangers { get => CountOfPassangers; set => CountOfPassangers = value; }

int IBus.Price { get => Price; set => Price = value; }

Colors IBus.Color { get => Color; set => Color = value; }

public void GetInfo()

{

Console.WriteLine("\n====================\nBus:\nName: {0}\nCount Of Passangers: {1}\n" +

"Price: {2}$\nColor: {3}\n====================\n", Name, CountOfPassangers, Price, Color);

}

}

class NissanBus : IBus

{

private string Name;

private int CountOfPassangers;

private int Price;

private Colors Color;

public NissanBus()

{

Name = "Nissan Bus";

CountOfPassangers = 90;

Price = 150000;

this.Color = Colors.Cyan;

}

public NissanBus(Colors Color)

{

Name = "Nissan Bus";

CountOfPassangers = 90;

Price = 150000;

this.Color = Color;

}

string IBus.Name { get => Name; set => Name = value; }

int IBus.CountOfPassangers { get => CountOfPassangers; set => CountOfPassangers = value; }

int IBus.Price { get => Price; set => Price = value; }

Colors IBus.Color { get => Color; set => Color = value; }

public void GetInfo()

{

Console.WriteLine("\n====================\nBus:\nName: {0}\nCount Of Passangers: {1}\n" +

"Price: {2}$\nColor: {3}\n====================\n", Name, CountOfPassangers, Price, Color);

}

}

}

«Minivan.cs»

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab1.\_1

{

enum SizesOfTrunk

{

Small,

Medium,

Large

}

interface IMinivan

{

string Name

{

get;

set;

}

int CountOfPassangers

{

get;

set;

}

SizesOfTrunk SizeOfTrunk

{

get;

set;

}

int Price

{

get;

set;

}

Colors Color

{

get;

set;

}

void GetInfo();

}

class AudiMinivan : IMinivan

{

private string Name;

private int CountOfPassangers;

private SizesOfTrunk SizeOfTrunk;

private int Price;

private Colors Color;

public AudiMinivan()

{

Name = "Audi A3 Van";

CountOfPassangers = 5;

SizeOfTrunk = SizesOfTrunk.Medium;

Price = 50000;

this.Color = Colors.Orange;

}

public AudiMinivan(Colors Color)

{

Name = "Audi A3 Van";

CountOfPassangers = 5;

SizeOfTrunk = SizesOfTrunk.Medium;

Price = 50000;

this.Color = Color;

}

string IMinivan.Name { get => Name; set => Name = value; }

int IMinivan.CountOfPassangers { get => CountOfPassangers; set => CountOfPassangers = value; }

SizesOfTrunk IMinivan.SizeOfTrunk { get => SizeOfTrunk; set => SizeOfTrunk = value; }

int IMinivan.Price { get => Price; set => Price = value; }

Colors IMinivan.Color { get => Color; set => Color = value; }

public void GetInfo()

{

Console.WriteLine("\n====================\nMinivan:\nName: {0}\nCount Of Passangers: {1}\n" +

"Size Of Trunk: {2}\nPrice: {3}$\nColor: {4}" +

"\n====================\n", Name, CountOfPassangers, SizeOfTrunk, Price, Color);

}

}

class FordMinivan : IMinivan

{

private string Name;

private int CountOfPassangers;

private SizesOfTrunk SizeOfTrunk;

private int Price;

private Colors Color;

public FordMinivan()

{

Name = "Ford Transit Connect";

CountOfPassangers = 7;

SizeOfTrunk = SizesOfTrunk.Large;

Price = 24000;

this.Color = Colors.Black;

}

public FordMinivan(Colors Color)

{

Name = "Ford Transit Connect";

CountOfPassangers = 7;

SizeOfTrunk = SizesOfTrunk.Large;

Price = 24000;

this.Color = Color;

}

string IMinivan.Name { get => Name; set => Name = value; }

int IMinivan.CountOfPassangers { get => CountOfPassangers; set => CountOfPassangers = value; }

SizesOfTrunk IMinivan.SizeOfTrunk { get => SizeOfTrunk; set => SizeOfTrunk = value; }

int IMinivan.Price { get => Price; set => Price = value; }

Colors IMinivan.Color { get => Color; set => Color = value; }

public void GetInfo()

{

Console.WriteLine("\n====================\nMinivan:\nName: {0}\nCount Of Passangers: {1}\n" +

"Size Of Trunk: {2}\nPrice: {3}$\nColor: {4}" +

"\n====================\n", Name, CountOfPassangers, SizeOfTrunk, Price, Color);

}

}

class NissanMinivan : IMinivan

{

private string Name;

private int CountOfPassangers;

private SizesOfTrunk SizeOfTrunk;

private int Price;

private Colors Color;

public NissanMinivan()

{

Name = "Nissan Passenger";

CountOfPassangers = 9;

SizeOfTrunk = SizesOfTrunk.Small;

Price = 33800;

this.Color = Colors.Yellow;

}

public NissanMinivan(Colors Color)

{

Name = "Nissan Passenger";

CountOfPassangers = 9;

SizeOfTrunk = SizesOfTrunk.Small;

Price = 33800;

this.Color = Color;

}

string IMinivan.Name { get => Name; set => Name = value; }

int IMinivan.CountOfPassangers { get => CountOfPassangers; set => CountOfPassangers = value; }

SizesOfTrunk IMinivan.SizeOfTrunk { get => SizeOfTrunk; set => SizeOfTrunk = value; }

int IMinivan.Price { get => Price; set => Price = value; }

Colors IMinivan.Color { get => Color; set => Color = value; }

public void GetInfo()

{

Console.WriteLine("\n====================\nMinivan:\nName: {0}\nCount Of Passangers: {1}\n" +

"Size Of Trunk: {2}\nPrice: {3}$\nColor: {4}" +

"\n====================\n", Name, CountOfPassangers, SizeOfTrunk, Price, Color);

}

}

}

«OffRoadCar.cs»

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab1.\_1

{

enum DriveUnitType

{

FourWheelDrive,

FrontWheelDrive,

RearDrive

}

interface IOffRoadCar

{

string Name

{

get;

set;

}

int Price

{

get;

set;

}

DriveUnitType DriveUnit

{

get;

set;

}

Colors Color

{

get;

set;

}

void GetInfo();

}

class AudiOffRoadCar : IOffRoadCar

{

private string Name;

private DriveUnitType DriveUnit;

private int Price;

private Colors Color;

public AudiOffRoadCar()

{

Name = "Audi Q7";

Price = 79000;

DriveUnit = DriveUnitType.FrontWheelDrive;

this.Color = Colors.Grey;

}

public AudiOffRoadCar(Colors Color)

{

Name = "Audi Q7";

Price = 79000;

DriveUnit = DriveUnitType.FrontWheelDrive;

this.Color = Color;

}

string IOffRoadCar.Name { get => Name; set => Name = value; }

int IOffRoadCar.Price { get => Price; set => Price = value; }

DriveUnitType IOffRoadCar.DriveUnit { get => DriveUnit; set => DriveUnit = value; }

Colors IOffRoadCar.Color { get => Color; set => Color = value; }

public void GetInfo()

{

Console.WriteLine("\n====================\nOff-Road Car:\nName: {0}\nPrice: {1}$\nDrive Unit: {2}\nColor: {3}" +

"\n====================\n", Name, Price, DriveUnit, Color);

}

}

class FordOffRoadCar : IOffRoadCar

{

private string Name;

private DriveUnitType DriveUnit;

private int Price;

private Colors Color;

public FordOffRoadCar()

{

Name = "Ford Raptor F-150";

Price = 51000;

DriveUnit = DriveUnitType.FourWheelDrive;

this.Color = Colors.Grey;

}

public FordOffRoadCar(Colors Color)

{

Name = "Ford Raptor F-150";

Price = 51000;

DriveUnit = DriveUnitType.FourWheelDrive;

this.Color = Color;

}

string IOffRoadCar.Name { get => Name; set => Name = value; }

int IOffRoadCar.Price { get => Price; set => Price = value; }

DriveUnitType IOffRoadCar.DriveUnit { get => DriveUnit; set => DriveUnit = value; }

Colors IOffRoadCar.Color { get => Color; set => Color = value; }

public void GetInfo()

{

Console.WriteLine("\n====================\nOff-Road Car:\nName: {0}\nPrice: {1}$\nDrive Unit: {2}\nColor: {3}" +

"\n====================\n", Name, Price, DriveUnit, Color);

}

}

class NissanOffRoadCar : IOffRoadCar

{

private string Name;

private DriveUnitType DriveUnit;

private int Price;

private Colors Color;

public NissanOffRoadCar()

{

Name = "Nissan Frontier";

Price = 18900;

DriveUnit = DriveUnitType.FourWheelDrive;

this.Color = Colors.White;

}

public NissanOffRoadCar(Colors Color)

{

Name = "Nissan Frontier";

Price = 18900;

DriveUnit = DriveUnitType.FourWheelDrive;

this.Color = Color;

}

string IOffRoadCar.Name { get => Name; set => Name = value; }

int IOffRoadCar.Price { get => Price; set => Price = value; }

DriveUnitType IOffRoadCar.DriveUnit { get => DriveUnit; set => DriveUnit = value; }

Colors IOffRoadCar.Color { get => Color; set => Color = value; }

public void GetInfo()

{

Console.WriteLine("\n====================\nOff-Road Car:\nName: {0}\nPrice: {1}$\nDrive Unit: {2}\nColor: {3}" +

"\n====================\n", Name, Price, DriveUnit, Color);

}

}

}

«SportCar.cs»

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab1.\_1

{

interface ISportCar

{

string Name

{

get;

set;

}

int Price

{

get;

set;

}

int MaxSpeed

{

get;

set;

}

double AccelerationTo100km

{

get;

set;

}

Colors Color

{

get;

set;

}

void GetInfo();

}

class AudiSportCar : ISportCar

{

private string Name;

private int Price;

private int MaxSpeed;

private double AccelerationTo100km;

private Colors Color;

public AudiSportCar()

{

Name = "Audi R8 GT";

Price = 250000;

MaxSpeed = 320;

AccelerationTo100km = 3.6;

this.Color = Colors.White;

}

public AudiSportCar(Colors Color)

{

Name = "Audi R8 GT";

Price = 250000;

MaxSpeed = 320;

AccelerationTo100km = 3.6;

this.Color = Color;

}

string ISportCar.Name { get => Name; set => Name = value; }

int ISportCar.Price { get => Price; set => Price = value; }

int ISportCar.MaxSpeed { get => MaxSpeed; set => MaxSpeed = value; }

double ISportCar.AccelerationTo100km { get => AccelerationTo100km; set => AccelerationTo100km = value; }

Colors ISportCar.Color { get => Color; set => Color = value; }

public void GetInfo()

{

Console.WriteLine("\n====================\nSportCar:\nName: {0}\nPrice: {1}$\nMaximum Speed: {2}km/h" +

"\nAcceleration To 100 km/h: {3}s\nColor: {4}" +

"\n====================\n", Name, Price, MaxSpeed, AccelerationTo100km, Color);

}

}

class FordSportCar : ISportCar

{

private string Name;

private int Price;

private int MaxSpeed;

private double AccelerationTo100km;

private Colors Color;

public FordSportCar()

{

Name = "Ford GT";

Price = 450000;

MaxSpeed = 350;

AccelerationTo100km = 2.9;

this.Color = Colors.White;

}

public FordSportCar(Colors Color)

{

Name = "Ford GT";

Price = 450000;

MaxSpeed = 350;

AccelerationTo100km = 2.9;

this.Color = Color;

}

string ISportCar.Name { get => Name; set => Name = value; }

int ISportCar.Price { get => Price; set => Price = value; }

int ISportCar.MaxSpeed { get => MaxSpeed; set => MaxSpeed = value; }

double ISportCar.AccelerationTo100km { get => AccelerationTo100km; set => AccelerationTo100km = value; }

Colors ISportCar.Color { get => Color; set => Color = value; }

public void GetInfo()

{

Console.WriteLine("\n====================\nSportCar:\nName: {0}\nPrice: {1}$\nMaximum Speed: {2}km/h" +

"\nAcceleration To 100 km/h: {3}s\nColor: {4}" +

"\n====================\n", Name, Price, MaxSpeed, AccelerationTo100km, Color);

}

}

class NissanSportCar : ISportCar

{

private string Name;

private int Price;

private int MaxSpeed;

private double AccelerationTo100km;

private Colors Color;

public NissanSportCar()

{

Name = "Nissan GT-R";

Price = 105000;

MaxSpeed = 315;

AccelerationTo100km = 2.8;

this.Color = Colors.Grey;

}

public NissanSportCar(Colors Color)

{

Name = "Nissan GT-R";

Price = 105000;

MaxSpeed = 315;

AccelerationTo100km = 2.8;

this.Color = Color;

}

string ISportCar.Name { get => Name; set => Name = value; }

int ISportCar.Price { get => Price; set => Price = value; }

int ISportCar.MaxSpeed { get => MaxSpeed; set => MaxSpeed = value; }

double ISportCar.AccelerationTo100km { get => AccelerationTo100km; set => AccelerationTo100km = value; }

Colors ISportCar.Color { get => Color; set => Color = value; }

public void GetInfo()

{

Console.WriteLine("\n====================\nSportCar:\nName: {0}\nPrice: {1}$\nMaximum Speed: {2}km/h" +

"\nAcceleration To 100 km/h: {3}s\nColor: {4}" +

"\n====================\n", Name, Price, MaxSpeed, AccelerationTo100km, Color);

}

}

}