

НАВЧАЛЬНО-НАУКОВИЙ КОМПЛЕКС «ІНСТИТУТ ПРИКЛАДНОГО
СИСТЕМНОГО АНАЛІЗУ» НАЦІОНАЛЬНОГО ТЕХНІЧНОГО
УНІВЕРСИТЕТУ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені
ІГОРЯ СІКОРСЬКОГО»

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРНИХ СИСТЕМ

КУРСОВА РОБОТА

з дисципліни

Програмування

На тему: “Автосалон”

Студента 1 курсу групи КП-21

Спеціальність 121 Інженерія програмного забезпечення.

Матвієнко Артем Олегович

Керівник старший викладач Погорелов В.В.

Національна оцінка: _____

Кількість балів: ____ Оцінка ECTS ____

Члени комісії _____

(підпис)

(вчене звання, науковий ступінь, прізвище та ініціали)

(підпис)

(вчене звання, науковий ступінь, прізвище та ініціали)

(підпис)

(вчене звання, науковий ступінь, прізвище та ініціали)

Київ-2023 рік

ННК "ФПМ" НТУУ "КПІ"					
Кафедра		<i>Програмного забезпечення комп'ютерних систем</i>			
Дисципліна		<i>Основи програмування</i>			
Галузь знань		<i>12 Інформаційні технології</i>			
Курс	<i>перший</i>	Група	<i>КП-21</i>	Семестр	<i>другий</i>

ЗАВДАННЯ
на курсову роботу студента

Матвієнка Артема Олеговича	
1. Тема роботи	Автосалон
2. Строк задачі студентом закінченої роботи	09.06.2023
3. Вихідні дані до роботи	
4. Зміст розрахунково-пояснювальної записки	
<i>1. Постановка задачі.</i>	
<i>2. Метод розв'язку задачі.</i>	
<i>3. Загальна блок-схема алгоритму та опис алгоритму.</i>	
<i>4. Опис програмного продукту.</i>	
<i>5. Результати роботи.</i>	
<i>6. Висновки.</i>	
<i>7. Список використаної літератури</i>	
5. Перелік графічного матеріалу	
<i>1. Загальна блок-схема алгоритму</i>	
<i>2. Ілюстрації роботи програми.</i>	
6. Дата видачі завдання	

ВСТУП

РОЗДІЛ 1 Постановка задачі

- 1.1 Створити систему зберігання автомобілів.
- 1.2 Створити систему формування замовлень за певними критеріями.

РОЗДІЛ 2 Розробка програмного продукту

- 2.1 Метод розв'язку задачі
- 2.2 Алгоритм розв'язку задачі.

РОЗДІЛ 3 Опис розробленого програмного продукту

- 3.1 Опис головних структур і змінних програми
- 3.2 Опис головної функції програми
- 3.3 Опис інтерфейсу
- 3.4 Результат роботи програмного коду

ВИСНОВКИ

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

ВСТУП

Актуальність. В сучасному світі автосалони є невід'ємною частиною автомобільної індустрії. Зростаючий попит на автомобілі, широкий вибір моделей та різноманітність послуг ставлять перед автосалонами вимогу до автоматизації процесу продажу та обслуговування клієнтів. Розробка програмного забезпечення, що спрощує та оптимізує роботу автосалону, є актуальною задачею, яка допоможе підвищити ефективність та якість обслуговування клієнтів.

Мета. Метою даного дослідження є розробка програмного рішення для автосалону, яке спростить процес продажу автомобілів та покращить взаємодію з клієнтами. Програма повинна надати зручний та інтуїтивно зрозумілий інтерфейс для роботи з переліком доступних моделей автомобілів, фільтрації за параметрами, оформлення замовлення та обробки платежів. Основною метою є забезпечення ефективного управління продажами та збільшення задоволення клієнтів від обслуговування в автосалоні.

Завдання. Розробка консольного інтерфейсу для взаємодії з користувачем, що дозволить зручно вибирати та фільтрувати доступні моделі автомобілів. Реалізація функціоналу замовлення, який дозволить користувачу вибрати певну модель автомобіля та оформити замовлення на неї. Розробка системи управління базою даних, яка забезпечить збереження та оновлення інформації про наявні моделі автомобілів, їх характеристики та ціни.

Практичне значення одержаних результатів. Результати даного дослідження матимуть практичне значення для автосалонів, оскільки розроблений програмний продукт дозволить автоматизувати процес продажу автомобілів і полегшити взаємодію з клієнтами. Автосалони

зможуть використовувати це програмне забезпечення для пропозиції своїх продуктів, ефективного керування складом та обробки замовлень. Це призведе до збільшення продажів, покращення обслуговування клієнтів і збільшення задоволення від покупки автомобілів.

Використане програмне забезпечення. Для розробки даного проєкта було використане програмне забезпечення Visual Studio 2022, яке є потужним інтегрованим середовищем розробки (IDE) та надає розробникам ряд інструментів для створення програм на мові C#. Git (система контролю версій) для збереження станів проєкту та зручного маніпулювання з ним.

РОЗДІЛ 1 Постановка задачі

1.1 Створити систему зберігання автомобілів.

Головною метою даного розділу є створення системи зберігання автомобілів, яка буде використовуватись в автосалоні. Ця система буде забезпечувати зручне та ефективне управління інформацією про наявні автомобілі, їх характеристики та інші деталі. Основним завданням цього підрозділу є розробка структури даних та функцій, які дозволять фільтрувати інформацію про автомобілі та подальше створення замовлень..

1.2 Створити систему формування замовлень за певними критеріями.

Однією з ключових задач даного розділу є розробка системи формування замовлень в автосалоні на основі певних критеріїв. Головною метою цієї системи є надання зручного та ефективного інструменту для клієнтів автосалону для вибору та замовлення автомобілів з відповідними характеристиками.

РОЗДІЛ 2 Розробка програмного продукту

2.1 Метод розв'язку задачі

Аналіз вимог: На цьому етапі були проаналізовані вимоги до системи зберігання автомобілів та формування замовлень. Було визначено необхідний функціонал, структуру даних, а також критерії фільтрації та параметри замовлень.

Реалізація функціоналу: На цьому етапі було реалізовано функції для зберігання автомобілів, фільтрації за критеріями та формування замовлень.

Тестування та налагодження: Після реалізації функціоналу було проведено мануальне тестування системи, включаючи перевірку правильності отримання даних, коректність роботи фільтрації та формування замовлень. Були виправлені помилки.

2.2 Алгоритм розв'язку задачі.

1. Створення локальної “бази даних” для збереження інформації про автомобілі.
2. Створення програмного коду для виведення інформації з локальної “бази даних”.
3. Створення програмного коду для введення користувачем відповідних ознак (фільтрів) з пошуку авто.
4. Створення програмного коду для фільтрування та виведення можливих варіантів користувачеві.
5. Створення програмного коду для формування замовлення на основі обраного користувачем автомобіля.

РОЗДІЛ 3 Опис розробленого програмного продукту

3.1 Опис головних структур і змінних програми

```
Ссылка: 19
abstract class Car
{
    public string _model;
    public int _year;
    public string _brand;
    public int _power;
    public int _maxSpeed;
    public string _transmission;
    public double _price;
    public int _technicalCondition;
    Ссылка: 7
    abstract public string ToString();
}
```

Абстрактний клас Car - який є шаблоном для створення об'єктів (екземплярів) типу OldCar та NewCar, які є дочірніми класами від класу Car.

```
Ссылка: 3
class OldCar : Car
{
    public string _model;
    public int _year;
    public string _brand;
    public int _power;
    public int _maxSpeed;
    public string _transmission;
    public double _price;
    public int _technicalCondition;

    Ссылка: 1
    public OldCar(string csvLine)
    {
        string[] values = csvLine.Split(',');

        _brand = values[0];
        _model = values[1];
        _year = int.Parse(values[2]);
        _price = double.Parse(values[3]);
        _technicalCondition = int.Parse(values[4]);
        _power = int.Parse(values[5]);
        _maxSpeed = int.Parse(values[6]);
        _transmission = values[7];
    }

    Ссылка: 6
    public override string ToString()
    {
        return $"{_brand}, {_model}, {_year}, ${_pri
```

```
Ссылка: 3
class NewCar : Car
{
    public string _model;
    public int _year;
    public string _brand;
    public int _power;
    public int _maxSpeed;
    public string _transmission;
    public double _price;
    public int _technicalCondition;

    Ссылка: 1
    public NewCar(string csvLine)
    {
        string[] values = csvLine.Split(',');

        _brand = values[0];
        _model = values[1];
        _year = int.Parse(values[2]);
        _price = double.Parse(values[3]);
        _technicalCondition = int.Parse(values[4]);
        _power = int.Parse(values[5]);
        _maxSpeed = int.Parse(values[6]);
        _transmission = values[7];
    }

    Ссылка: 6
    public override string ToString()
    {
        return $"{_brand}, {_model}, {_year}, ${_pri
```

Класи NewCar та OldCar у свою чергу описують характеристики автомобіля: марка, модель, рік випуску, ціна, стан, потужність двигуна, максимальна швидкість та тип коробки передач.

```

Ссылка: 2
interface ICarList
{
    Ссылка: 4
    void Init(string path);

    Ссылка: 10
    void Sort(double budgetLow, double budgetHigh, int conditionLow, int conditionHigh, string transmission);

    Ссылка: 4
    void ShowList();

    Ссылка: 6
    void ShowProposed();

    Ссылка: 6
    Car GetProposed();
}

```

Інтерфейс ICarList - шаблон для створення дочірніх класів OldCarList та NewCarList, які у свою чергу створюють список об'єктів(екземплярів) Car. За замовчуванням мають бути методи: Init() - ініціалізує список зчитуючи інформацію про автомобіль з відповідного .csv файлу; Sort() - аналог фільтру у каталозі, відсортовує список за перними ознаками (аргументами що передані у метод); ShowList() - показує увесь список автомобілів з їх характеристиками; ShowProposed() - показує відфільтрований список автомобілів (разом з їхніми характеристиками). GetProposed() - повертає обраний користувачем автомобіль.

```

interface IOrder
{
    Ссылка: 2
    string MakeOrder();
}

Ссылка: 5
class Address
{
    Ссылка: 2
    public string HouseNumber { get; set; }
    Ссылка: 2
    public string Street { get; set; }
    Ссылка: 2
    public string City { get; set; }
    Ссылка: 2
    public int PostalCode { get; set; }
}

Ссылка: 2
class Buyer : Address
{
    Ссылка: 2
    public string Name { get; set; }
    protected Address _address;
}

Ссылка: 2
class Order : Buyer, IOrder
{
    private string _name;
    private Address _address;
    Ссылка: 2
    public Car Car { get; set; }
    Ссылка: 1
    public Order(string name, Address address) { ... }
    Ссылка: 2
    public string MakeOrder() { ... }
}

```


Класи Address, Buyer та Order - відповідають за формування “замовлення”. Структура наслідування: Address <= Buyer <= Order - де Address - батьківський елемент. Клас Address містить наступні властивості (property): назва міста, назва вулиці, номер будинку, номер замовлення. Клас Buyer містить властивість - ім’я замовника. Клас Order (який додатного наслідує інтерфейс IOrder) на основі попередніх даних формує замовлення за допомогою методу MakeOrder().

```
Ссылка: 2
class ConsoleInterface<T>
{
    private List<T> _menu = new List<T>();

    Ссылка: 22
    public void AddItem(T item)...

    Ссылка: 4
    public void Reset()...

    Ссылка: 5
    public int GetCheckedInput()...

    Ссылка: 1
    public double GetLowerValue()...

    Ссылка: 1
    public double GetHigherValue(double lower = 0)...

    Ссылка: 1
    public int GetLowerValueInt(int higher = 10)...

    Ссылка: 1
    public int GetHigherValueInt(int lower = 0, int higher = 10)...
}
```

Також важливим класом є клас ConsoleInterface, завдяки якому стає більш зручною розробка консольного інтерфейсу. Основні методи: AddItem() - додає у “меню” вказаний елемент; Reset() - скидає елементи меню для повторного використання; GetCheckedInput() (та похідні від нього) - дають змогу відразу обробляти коректність введення на основі “меню” (інтерфейсу).

3.2 Опис головної функції програми

Спочатку створюємо об'єкт menu - екземпляр класу ConsoleInterface для полегшення створення консольного інтерфейсу та покращення читабельності коду в цілому. Та зробимо нескінченний цикл за для симуляції додатку.

```
static void Main(string[] args)
{
    ConsoleInterface<string> menu = new ConsoleInterface<string>();

    while (true)
    {
        Console.ResetColor();
        Console.Clear();
        Console.WriteLine("Whitch type of car you want?\n");
        menu.AddItem("1. Old cars (retro) 1950-1970");
        menu.AddItem("2. New cars 2010-now");

        int input = menu.GetCheckedInput();
    }
}
```

Після чого створюємо об'єкти oldCarList та newCarList та перевіряємо що вибрав користувач (старі моделі чи нові).

```
OldCarList oldCarList = new OldCarList();
NewCarList newCarList = new NewCarList();

switch (input)
{
    case 1:
        oldCarList.Init(@"D:\Microsoft Visual Studio\Projects\Course_Work\Course_Work\Old-Cars.csv");
        isOldCar = true;
        Console.Clear();
        break;
    case 2:
        newCarList.Init(@"D:\Microsoft Visual Studio\Projects\Course_Work\Course_Work\New-Cars.csv");
        isNewCar = true;
        Console.Clear();
        break;
}
```

завдяки методу Init() відповідно ініціалізуємо список.

після чого будуємо інтерфейс, для цього використовується інтерфейс об'єкту menu аби коректно відображати елементи вибору та коректно перевіряти правильність введення.

```

menu.Reset();

menu.AddItem("1. Show all list");
menu.AddItem("2. Seletc filters");

input = menu.GetCheckedInput();
bool isEnd = false;

switch (input)
{
    case 1:
        isEnd = true;
        if (isNewCar)
            newCarList.ShowList();
            Console.WriteLine("\n");
        if (isOldCar)
            oldCarList.ShowList();
            Console.WriteLine("\n");
        Console.ReadLine();
        break;
    case 2:
        break;
}

if (isEnd) continue;

```

Додаємо елементи “меню” та в залежності від вводу користувача або показуємо йому повний список автомобілів, або починаємо робити пошук за фільтрами.

У випадку показу всього списку для ознайомлення після показу меню повертається до початкового меню.

Далі робимо фільтри за: ціною, станом авто (по 10-бальній шкалі) та типом коробки передач (КПП).

```

menu.Reset();

string type = isOldCar ? "Old car" : "New car";
double budgetLow;
double budgetHigh;
int conditionLow;
int conditionHigh;
string transmission;

Console.Clear();
Console.ForegroundColor = ConsoleColor.Green;
Console.WriteLine($"Type: {type}\n");
Console.ResetColor();

Console.WriteLine("Select price range\n");
menu.AddItem("0. Custom price");
menu.AddItem("1. Up to $5.000");
menu.AddItem("2. From $5.000 up to $20.000");
menu.AddItem("3. From $20.000 up to $50.000");
menu.AddItem("4. From $50.000 up to $100.000");
menu.AddItem("5. From $100.000 up to $200.000");
menu.AddItem("6. From $200.000 up to $500.000");
menu.AddItem("7. From $500.000 up to $1.000.000");
menu.AddItem("8. From $1.000.000");

input = menu.GetCheckedInput();

```

Створюємо змінні: мінімальна/максимальна ціна, мінімальна/максимальна оцінка стану, тип коробки передач (КПП).

Завдяки об'єкту `menu` додаємо елементи у консоль та ставимо обробку введення користувача. Після обрання користувачем відповідного пункту користувач або переходить далі на наступний фільтр, або виставляє свій діапазон цін та переходить на наступний фільтр.

```
switch (input)
{
    case 0:
        Console.WriteLine("Enter lower price:");
        budgetLow = menu.GetLowerValue();
        Console.WriteLine("Enter higher price:");
        budgetHigh = menu.GetHigherValue(budgetLow);
        break;
    case 1:
        budgetLow = 0;
        budgetHigh = 5000;
        break;
    case 2:
        budgetLow = 5000;
        budgetHigh = 20000;
        break;
    case 3:
        budgetLow = 20000;
        budgetHigh = 50000;
        break;
    case 4:
        budgetLow = 50000;
        budgetHigh = 100000;
        break;
    case 5:
        budgetLow = 100000;
        budgetHigh = 200000;
        break;
    case 6:
        budgetLow = 200000;
        budgetHigh = 500000;
        break;
    case 7:
        budgetLow = 500000;
        budgetHigh = 1000000;
        break;
    case 8:
        budgetLow = 1000000;
        budgetHigh = double.PositiveInfinity;
        break;
    default:
        budgetLow = 0;
        budgetHigh = double.PositiveInfinity;
        break;
}
```

Аналогічно робимо меню за обробку введення для фільтру якості:

```
menu.Reset();

Console.Clear();
Console.ForegroundColor = ConsoleColor.Green;
Console.WriteLine($"Type: {type}");
Console.WriteLine($"Price: {budgetLow}-{budgetHigh}\n");
Console.ResetColor();
Console.WriteLine("Enter wanted condition from 1 up to 10\n");
menu.AddItem("0. Custom condition");
menu.AddItem("1. (1-3) the worst condition (for parts)");
menu.AddItem("2. (4-7) mediocre condition (replace some details)");
menu.AddItem("3. (8-10) good condition (visual defect)");

input = menu.GetCheckedInput();
```

```

switch (input)
{
    case 0:
        Console.WriteLine("Enter lower condition:");
        conditionLow = menu.GetLowerValueInt(10);
        Console.WriteLine("Enter higher condition:");
        conditionHigh = menu.GetHigherValueInt(conditionLow, 10);
        break;
    case 1:
        conditionLow = 1;
        conditionHigh = 3;
        break;
    case 2:
        conditionLow = 4;
        conditionHigh = 7;
        break;
    case 3:
        conditionLow = 8;
        conditionHigh = 10;
        break;
    default:
        conditionLow = 1;
        conditionHigh = 10;
        break;
}

```

Та для коробки передач (КПП):

```

menu.Reset();

Console.Clear();
Console.ForegroundColor = ConsoleColor.Green;
Console.WriteLine($"Type: {type}");
Console.WriteLine($"Price: {budgetLow}-{budgetHigh}");
Console.WriteLine($"Condition: {conditionLow}-{conditionHigh}\n");
Console.ResetColor();
Console.WriteLine("Select which type of transmission you want\n");
menu.AddItem("0. All Transmissions");
menu.AddItem("1. Manual");
menu.AddItem("2. Automatic");
menu.AddItem("3. Robotic");
menu.AddItem("4. None");

input = menu.GetCheckedInput();

```

```

input = menu.GetCheckedInput();

switch (input)
{
    case 0:
        transmission = null;
        break;
    case 1:
        transmission = "Manual";
        break;
    case 2:
        transmission = "Automatic";
        break;
    case 3:
        transmission = "Robotic";
        break;
    case 4:
        transmission = "None";
        break;
    default:
        transmission = "Automatic";
        break;
}

```

Після встановлення фільтрів створюємо об'єкт car класу Car аби потім додати його у замовлення. Та, власне, сортуємо список автомобілів за фільтрами:

```
if (isOldCar && transmission != null)
{
    oldCarList.Sort(budgetLow, budgetHigh, conditionLow, conditionHigh, transmission);

    Console.Clear();
    Console.ForegroundColor = ConsoleColor.Green;
    Console.WriteLine($"Type: {type}");
    Console.WriteLine($"Price: {budgetLow}-{budgetHigh}");
    Console.WriteLine($"Condition: {conditionLow}-{conditionHigh}");
    Console.WriteLine($"Transmission: {transmission}\n");
    Console.ForegroundColor = ConsoleColor.Cyan;

    if (oldCarList.GetLength() == 0)
    {
        Console.WriteLine("Sorry, we don't have cars with this feature :(");
        Thread.Sleep(3500);
        continue;
    }

    oldCarList.ShowProposed();

    Console.ResetColor();
    Console.WriteLine("\n");

    car = oldCarList.GetProposed();
}
```

(аналогічно для newCarList (списку нових авто)).

Після всього, робимо “замовлення”:

```
var buyer = new Buyer();

Console.WriteLine("Enter your Name:");
buyer.Name = Console.ReadLine().Replace(" ", "");

var address = new Address();

Console.WriteLine("Enter the city where you want to deliver: ");
address.City = Console.ReadLine().Replace(" ", "");

Console.WriteLine("Enter the street where you want to deliver: ");
address.Street = Console.ReadLine().Replace(" ", "");

Console.WriteLine("Enter the house-number where you want to deliver: ");
address.HouseNumber = Console.ReadLine().Replace(" ", "");

address.PostalCode = new Random().Next(10000, 100000);

var order = new Order(buyer.Name, address);

order.Car = car;

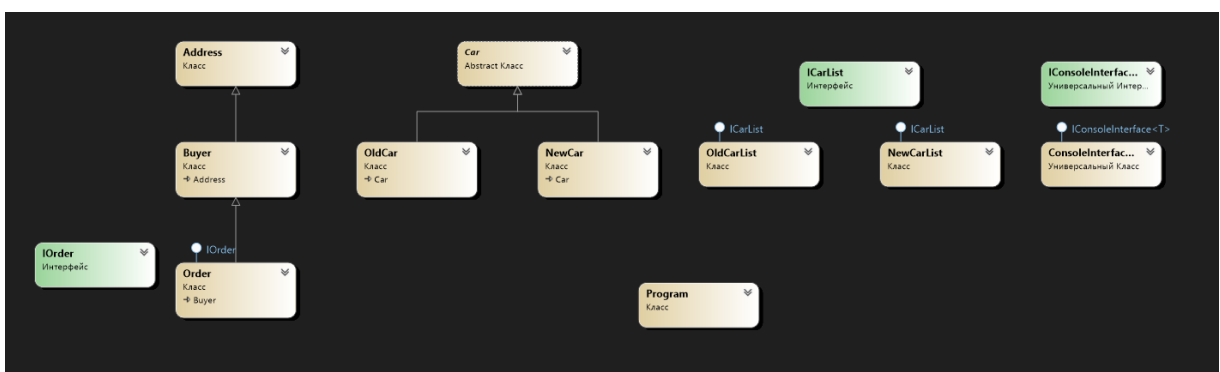
Console.ForegroundColor = ConsoleColor.DarkYellow;
Console.WriteLine("Your order:\n");
Console.WriteLine(order.MakeOrder());
Console.ResetColor();

Console.WriteLine();
Console.ReadLine();
```

створюємо об'єкт `buyer` (покупець) класу `Buyer` та надаємо його полю `Name` ім'я, після, створюємо об'єкт `address` класу `Address` та надаємо його полям `City`, `Street`, `HouseNumber`, `PostalCode` відповідні значення. У кінці створюємо об'єкт `order` класу `Order` та передаємо у конструктор ім'я "замовника" та об'єкт `address` (який вже містить відповідну інформацію про адресу). Використовуємо метод `MakeOrder()` аби створити та вивести у консоль замовлення. Після чого показується замовлення з відповідно введеними даними про адресу. Після натиснення на `Enter` цикл повторюється (тобто користувач потрапляє у головне меню ...).

3.3 Опис інтерфейсу

Діаграма класів:

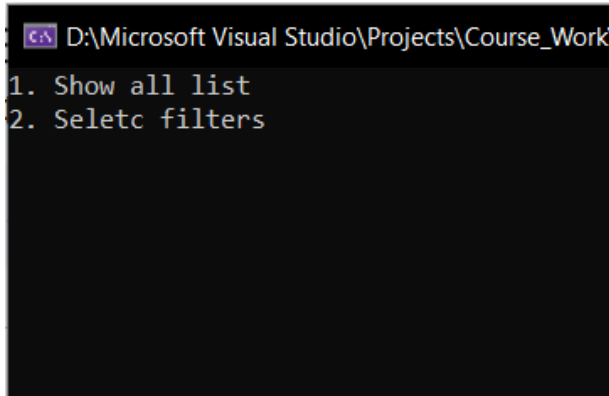


Інтерфейс програми:

1. Початкове меню з вибором типу авто, яке хоче придбати користувач.
(Виберемо наприклад нові авто (2))

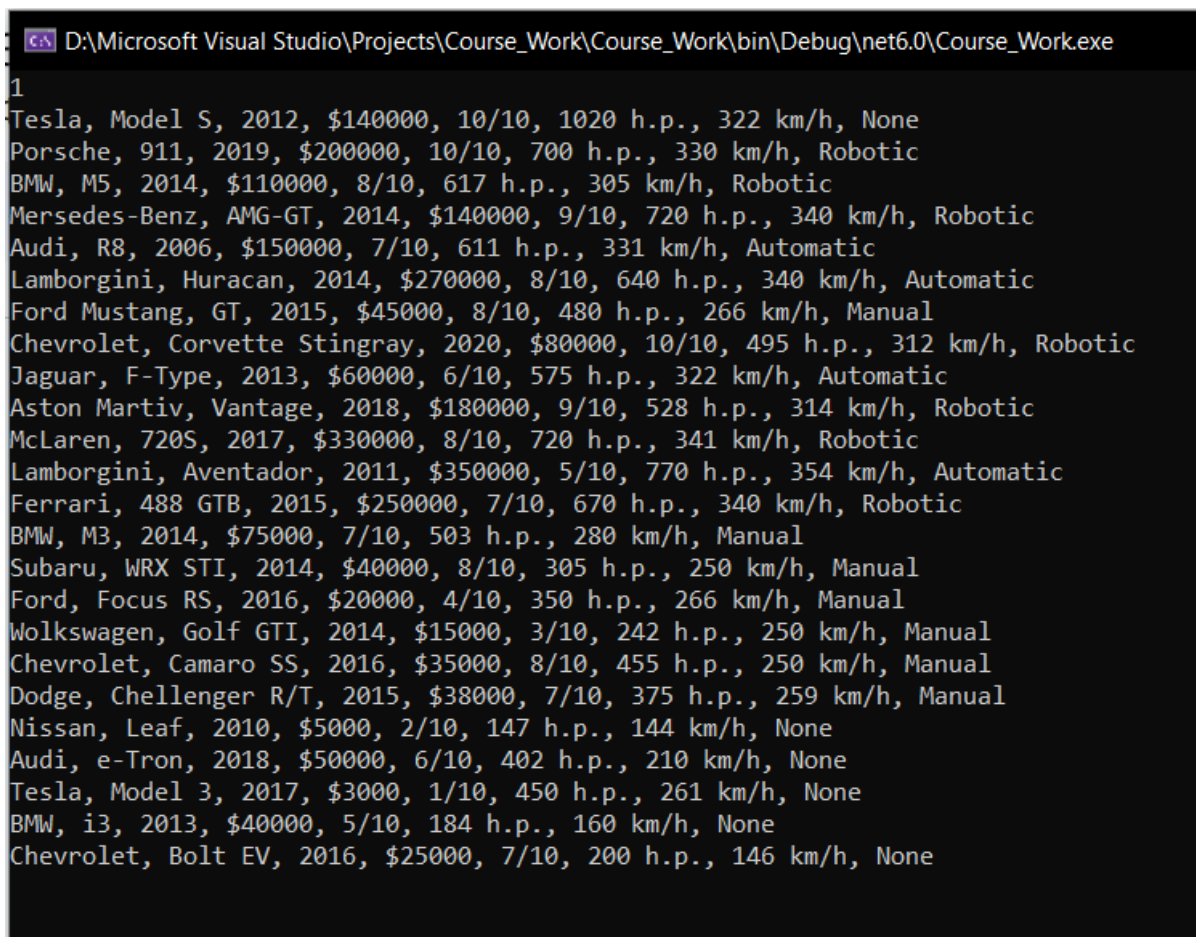
```
D:\Microsoft Visual Studio\Projects\Course_Work
Which type of car you want?
1. Old cars (retro) 1950-1970
2. New cars 2010-now
```

2. Після вибору типу авто пропонується обрати або повний список авто для ознайомлення, або вибір фільтрів для подальшого замовлення авто.



```
D:\Microsoft Visual Studio\Projects\Course_Work
1. Show all list
2. Seletc filters
```

3.а При виборі пункту “1” показується список усіх наявних авто. Після натискання на клавішу “Enter”, користувача буде повернено до головного меню.



```
D:\Microsoft Visual Studio\Projects\Course_Work\Course_Work\bin\Debug\net6.0\Course_Work.exe
1
Tesla, Model S, 2012, $140000, 10/10, 1020 h.p., 322 km/h, None
Porsche, 911, 2019, $200000, 10/10, 700 h.p., 330 km/h, Robotic
BMW, M5, 2014, $110000, 8/10, 617 h.p., 305 km/h, Robotic
Mersedes-Benz, AMG-GT, 2014, $140000, 9/10, 720 h.p., 340 km/h, Robotic
Audi, R8, 2006, $150000, 7/10, 611 h.p., 331 km/h, Automatic
Lamborghini, Huracan, 2014, $270000, 8/10, 640 h.p., 340 km/h, Automatic
Ford Mustang, GT, 2015, $45000, 8/10, 480 h.p., 266 km/h, Manual
Chevrolet, Corvette Stingray, 2020, $80000, 10/10, 495 h.p., 312 km/h, Robotic
Jaguar, F-Type, 2013, $60000, 6/10, 575 h.p., 322 km/h, Automatic
Aston Martiv, Vantage, 2018, $180000, 9/10, 528 h.p., 314 km/h, Robotic
McLaren, 720S, 2017, $330000, 8/10, 720 h.p., 341 km/h, Robotic
Lamborghini, Aventador, 2011, $350000, 5/10, 770 h.p., 354 km/h, Automatic
Ferrari, 488 GTB, 2015, $250000, 7/10, 670 h.p., 340 km/h, Robotic
BMW, M3, 2014, $75000, 7/10, 503 h.p., 280 km/h, Manual
Subaru, WRX STI, 2014, $40000, 8/10, 305 h.p., 250 km/h, Manual
Ford, Focus RS, 2016, $20000, 4/10, 350 h.p., 266 km/h, Manual
Wolkswagen, Golf GTI, 2014, $15000, 3/10, 242 h.p., 250 km/h, Manual
Chevrolet, Camaro SS, 2016, $35000, 8/10, 455 h.p., 250 km/h, Manual
Dodge, Challenger R/T, 2015, $38000, 7/10, 375 h.p., 259 km/h, Manual
Nissan, Leaf, 2010, $5000, 2/10, 147 h.p., 144 km/h, None
Audi, e-Tron, 2018, $50000, 6/10, 402 h.p., 210 km/h, None
Tesla, Model 3, 2017, $3000, 1/10, 450 h.p., 261 km/h, None
BMW, i3, 2013, $40000, 5/10, 184 h.p., 160 km/h, None
Chevrolet, Bolt EV, 2016, $25000, 7/10, 200 h.p., 146 km/h, None
```


3.6 При виборі пункту “2”, користувач перейде у меню фільтрув, де першим запропонують вказати ціну.

```
C:\> D:\Microsoft Visual Studio\Projects\Course_Work\Course_Work\bin\Debug\net6.0\
Type: New car
Select price range
0. Custom price
1. Up to $5.000
2. From $5.000 up to $20.000
3. From $20.000 up to $50.000
4. From $50.000 up to $100.000
5. From $100.000 up to $200.000
6. From $200.000 up to $500.000
7. From $500.000 up to $1.000.000
8. From $1.000.000
0
Enter lower price:
0
Enter higher price:
700 000
```

4. Після встановлення ціни програма потребує вказати стан авто по 10-ти бальній шкалі.

```
C:\> D:\Microsoft Visual Studio\Projects\Course_Work\Course_Work\bin\Debug\net6.0\
Type: New car
Price: 0-700000
Enter wanted condition from 1 up to 10
0. Custom condition
1. (1-3) the worst condition (for parts)
2. (4-7) mediocre condition (replace some details)
3. (8-10) good condition (visual defect)
0
Enter lower condition:
4
Enter higher condition:
10_
```

5. Далі система потребує вибрати тип коробки передач (КПП). Наприклад оборемо усі типи (0).

```
C:\> D:\Microsoft Visual Studio\Projects\Course_Work\Course_Work\bin\Debug\net6.0\Course_Work.exe
Type: New car
Price: 0-700000
Condition: 4-10

Select which type of transmission you want

0. All Transmissions
1. Manual
2. Automatic
3. Robotic
4. None
```

6. Тепер на основі заданих фільтрів система показує увесь список авто за заданими специфікаціями. Система відразу запропонує ввести номер авто який користувач бажає придбати.

```
C:\> D:\Microsoft Visual Studio\Projects\Course_Work\Course_Work\bin\Debug\net6.0\Course_Work.exe
Type: New car
Price: 0-700000
Condition: 4-10
Transmission: All

1. Ford Mustang, GT, 2015, $45000, 8/10, 480 h.p., 266 km/h, Manual
2. BMW, M3, 2014, $75000, 7/10, 503 h.p., 280 km/h, Manual
3. Subaru, WRX STI, 2014, $40000, 8/10, 305 h.p., 250 km/h, Manual
4. Ford, Focus RS, 2016, $20000, 4/10, 350 h.p., 266 km/h, Manual
5. Chevrolet, Camaro SS, 2016, $35000, 8/10, 455 h.p., 250 km/h, Manual
6. Dodge, Challenger R/T, 2015, $38000, 7/10, 375 h.p., 259 km/h, Manual
7. Audi, R8, 2006, $150000, 7/10, 611 h.p., 331 km/h, Automatic
8. Lamborghini, Huracan, 2014, $270000, 8/10, 640 h.p., 340 km/h, Automatic
9. Jaguar, F-Type, 2013, $60000, 6/10, 575 h.p., 322 km/h, Automatic
10. Lamborghini, Aventador, 2011, $350000, 5/10, 770 h.p., 354 km/h, Automatic
11. Porsche, 911, 2019, $200000, 10/10, 700 h.p., 330 km/h, Robotic
12. BMW, M5, 2014, $110000, 8/10, 617 h.p., 305 km/h, Robotic
13. Mercedes-Benz, AMG-GT, 2014, $140000, 9/10, 720 h.p., 340 km/h, Robotic
14. Chevrolet, Corvette Stingray, 2020, $80000, 10/10, 495 h.p., 312 km/h, Robotic
15. Aston Martiv, Vantage, 2018, $180000, 9/10, 528 h.p., 314 km/h, Robotic
16. McLaren, 720S, 2017, $330000, 8/10, 720 h.p., 341 km/h, Robotic
17. Ferrari, 488 GTB, 2015, $250000, 7/10, 670 h.p., 340 km/h, Robotic
18. Tesla, Model S, 2012, $140000, 10/10, 1020 h.p., 322 km/h, None
19. Audi, e-Tron, 2018, $50000, 6/10, 402 h.p., 210 km/h, None
20. BMW, i3, 2013, $40000, 5/10, 184 h.p., 160 km/h, None
21. Chevrolet, Bolt EV, 2016, $25000, 7/10, 200 h.p., 146 km/h, None

Enter number of car you want to buy:
```

7. Після введення номеру бажаного авто, система запитає: ім'я, місто, вулицю та номер будинку, що потрібно для доставлення замовлення. (також генерується код замовлення у діапазоні від 10000-99999)

```
Enter number of car you want to buy:
11
Enter your Name:
Artem
Enter the city where you want to deliver:
Kyiv
Enter the street where you want to deliver:
Raidujna
Enter the house-number where you want to deliver:
10
```

8. У консоль виводиться замовлення на основі введених даних.

```
Your order:

    Car purchase order
Order in the name: Artem
Deliver down the address:
    City: Kyiv
    Street: Raidujna
    House: 10
Postal code: 42106
Car: Porsche, 911, 2019, $200000, 10/10, 700 h.p., 330 km/h, Robotic
```

9. Після натиснення на “Enter” користувача поверне на головне меню.

3.4 Результат роботи програмного коду

Програму протестовано. Не було виявлено некоректностей у роботі. Всі некоректні введення користувача обробляються коректно не викликаючи помилок або виключень. Очікуваний результат співпадає з фактичним.

Система обробляє некоректні введення користувача. Окрім очевидних некоректних введень також при виставленні свого діапазону цін або стану

авто користувач при виборі верхньої межі не зможе її зробити нижчою за нижню межу.

ВИСНОВКИ

В результаті розробки системи зберігання автомобілів та формування замовлень на мові C# було досягнуто поставленої мети. Система надає зручний інструмент для управління замовленнями.

Дослідження мало актуальність у зв'язку з потребою автоматизувати процес зберігання автомобілів та формування замовлень. Розроблена система дозволяє клієнтам швидко вибирати автомобілі та робити замовлення.

Метою дослідження було розробити функціонал системи зберігання автомобілів та формування замовлень.

Результати дослідження мають практичне значення для автосалонів, які можуть використовувати розроблену систему для спрощення управління замовленнями та зберігання даних про автомобілі.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

- [Microsoft learn C#](#)
- [Metanit](#)