МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики
Кафедра програмного забезпечення комп’ютерних систем

Лабораторна робота № 1

з дисципліни “Програмування”
тема “C# .Net. Реалізація основних принципів ООП мовою C#”

Виконав студент
ІІ курсу групи КП-01
Тітов Єгор Павлович

Перевірив
“____” “_____” 20___р
викладач
Заболотня Тетяна Миколаївна

Київ 2021

Мета роботи

Ознайомитися з основами об’єктно-орієнтованого підходу до створення ПЗ у мові C#, створенням класів, об’єктів, механізмами інкапсуляції, наслідування та поліморфізму. Вивчити механізм управління ресурсами, реалізований у .Net.

Постановка задачі

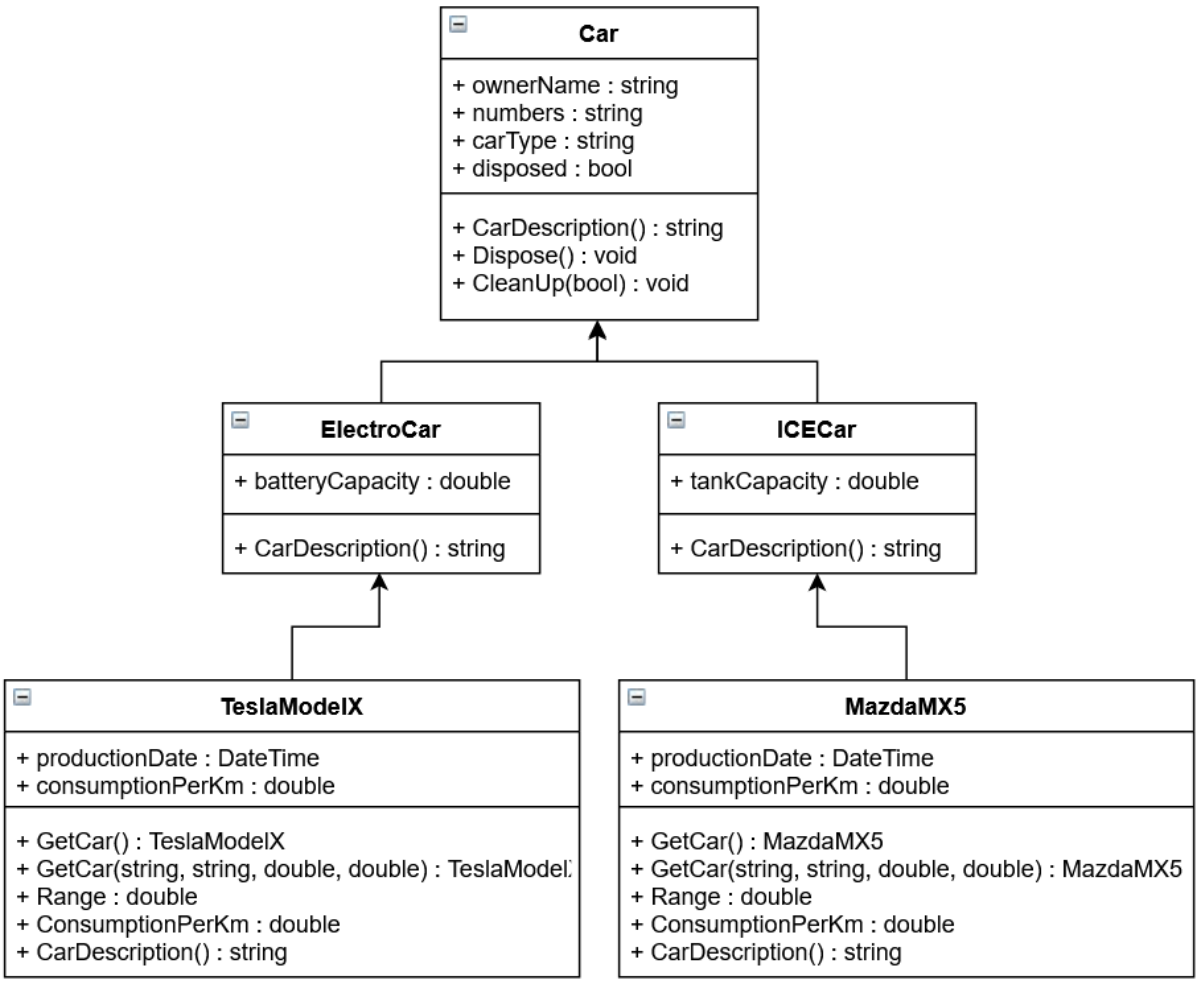
Побудувати ієрархію класів, що відтворюватимуть відношення наслідування між об’єктами реального світу (кількість класів >= 5). При цьому:

1. Забезпечити наявність у класах полів та методів з різними модифікаторами доступу, пояснити свій вибір **(1 бал)**.
2. Забезпечити наявність у класах властивостей: складніше, ніж просто get;set;, обґрунтувати доцільність створення властивості **(1 бал)**.
3. Створити для розроблюваних класів такі конструктори **(2 бали)**:
   - конструктор за замовчуванням;
   - конструктор з параметрами;
   - приватний конструктор;
   - статичний конструктор.

   Продемонструвати, яким чином викликаються конструктори базового та дочірнього класів.
4. Використати віртуальні та перевизначені методи **(1 бал)**.
5. Додати до класів методи, наявність яких дозволятиме управляти знищенням екземплярів цих класів **(2 бали)**:
   a. реалізувати інтерфейс IDisposable;
   b. створити деструктори;
   c. забезпечити уникнення конфліктів між Dispose та деструктором.
6. Забезпечити виклики методів GC таким чином, щоб можна було простежити життєвий цикл об’єктів, що обробляються (зокрема, використати методи Collect, SupressFinalize, ReRegisterForFinalize, GetTotalMemory, GetGeneration, WaitForPendingFinalizers). Створити ситуацію, яка спровокує примусове збирання сміття GC **(2 бали)**.

Протокол має містити: титульний аркуш, постановку задачі, UML діаграму класів, фрагменти коду, які демонструють виконання поставлених задач, висновки.

UML діаграма класів



Фрагменти коду

1.

```
class Car : IDisposable
    {
        protected string ownerName;
        protected string numbers;
        protected static string carType;

        protected bool disposed;
```

```
        public Car()
        {
            this.ownerName = "unknown";
            this.numbers = "unknown";
        }
```

```
class TeslaModelX : ElectroCar
    {
        private DateTime productionDate;
        private double consumptionPerKm;

        private TeslaModelX() : base()
        {
            this.productionDate = DateTime.Now;
            this.ConsumptionPerKm = 0;
        }
```

2.

```
public double Range
        {
            get
            {
                if (ConsumptionPerKm != 0)
                {
                    return batteryCapacity / ConsumptionPerKm;
                }
                else
                {
                    return -1;
                }
            }
        }
```

```
public double ConsumptionPerKm
        {
            get
            {
                return ConsumptionPerKm;
            }
```

```
            set
            {
                if (value > 0)
                {
                    this.consumptionPerKm = value;
                }
                else
                {
                    this.consumptionPerKm = 0;
                }
            }
        }
```

3.

```
public Car()
        {
            this.ownerName = "unknown";
            this.numbers = "unknown";
        }
```

```
public Car(string name, string numbers)
        {
            this.ownerName = name;
            this.numbers = numbers;
        }
```

```
private MazdaMX5() : base()
        {
            this.productionDate = DateTime.Now;
            this.ConsumptionPerKm = 0;
        }
```

```
static ElectroCar()
        {
            carType = "Electric Car";
        }
```

4.

```
public virtual string CarDescription()
        {
            return $"-- Car description: [Owner: {ownerName}]; [Numbers: {numbers}]";
        }
```

```
public override string CarDescription()
        {
            return $"-- Car description: [Owner: {ownerName}]; [Numbers: {numbers}];\n [Type: {carType}] [Battery capacity: {batteryCapacity} Watt * Hour]";
        }
```

5.

```
class Car : IDisposable
    {
        ...
        public virtual void Dispose()
        {
            CleanUp(true);

            GC.SuppressFinalize(this);
        }

        protected void CleanUp(bool disposing)
        {
            if (!this.disposed)
            {
                if (disposing)
                {
                    // Console.WriteLine("-- Managed resourses disposed --");
                }

                // Console.WriteLine("-- Unmanaged resourses disposed --");

                disposed = true;
```

```
            }
        }

        ~Car()
        {
            CleanUp(false);

            Console.WriteLine("-- Car destructor called --");
        }
    }
```

6.

```
MazdaMX5 mazda = MazdaMX5.GetCar("Ivan", "OO0000OO", 40, 4);

Console.WriteLine("\nMemory before creating 10_000 obj: " +
GC.GetTotalMemory(false));



            for (int i = 0; i < 10_000; i++)
            {
                Car car1 = new Car();

                car1.Dispose();
            }

            Console.WriteLine($"Memory after creating 10_000 obj:
{GC.GetTotalMemory(false)}\n");
            Console.WriteLine($"Generation of 'mazda' obj before collecting:
{GC.GetGeneration(mazda)}\n");

            GC.Collect(2);
            GC.WaitForPendingFinalizers();

            Console.WriteLine($"Memory after collecting garbage:
{GC.GetTotalMemory(false)}\n");
            Console.WriteLine($"Generation of 'mazda' obj after collecting:
{GC.GetGeneration(mazda)}\n");
            Console.WriteLine($"Garbage Collection count:
```

```
{GC.CollectionCount(0)}");
```

```
// Memory before creating 10_000 obj: 77776
// Memory after creating 10_000 obj: 490552

// Generation of 'mazda' obj before collecting: 0
...
// Memory after collecting garbage: 87352

// Generation of 'mazda' obj after collecting: 1

// Garbage Collection count: 1
```

Висновки

В результаті виконання лабораторної роботи відбулось ознайомлення з основами об’єктно-орієнтованого підходу програмування при створенні різного ПЗ на мові C#, створенням класів, об’єктів класів, механізмами інкапсуляції, наслідування та поліморфізму. Були випробувані механізми управління ресурсами, реалізованих у платформі .Net.