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 № 2

з дисципліни “Програмування”
тема “C# .Net. Розширені можливості реалізації ООП у мові C#. Події.”

Виконав студент
ІІ курсу групи КП-01
Тітов Єгор Павлович

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

Київ 2021

Мета роботи

Ознайомитися з такими можливостями мови програмування C# як абстрактні класи, інтерфейси, делегати. Вивчити механізми оброблення подій у C#, а також можливості, які мають методи-розширення.

Постановка задачі

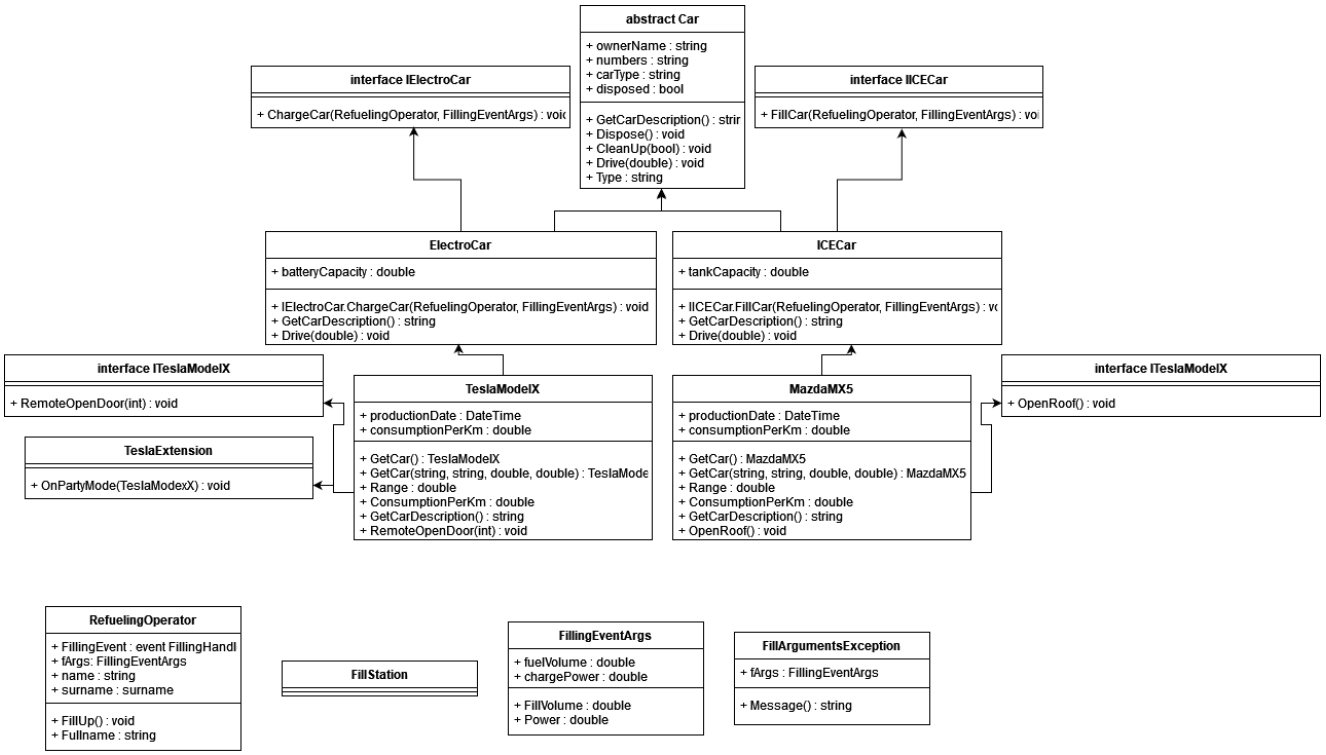
Для ієрархії класів, побудованої в лабораторній роботі №1, реалізувати:

1. Множину інтерфейсів. При чому один з класів повинен реалізовувати щонайменше 2 інтерфейси. Також продемонструвати реалізацію explicit implementation інтерфейса, обгрунтувати її використання **(1 бал)**.
2. Абстрактний клас. Забезпечити його наслідування. Наявність в цьому класі абстрактних методів - обов'язкова **(1 бал)**.
3. Механізм «делегат – подія – обробник події» **(2 бали)**.
4. Перетворити код, який забезпечує роботу з подіями та обробниками подій, на код, що використовує (*) **(2 бали)**:
   a. анонімні методи;
   b. lambda-вирази;
   c. типи Action та Func (кожен з них).

   (*) - допускається реалізація коду однієї події різними способами, необов'язково різних подій.
5. Механізм створення та оброблення власних помилок **(2 бали)**:
   a. створити новий клас виключної ситуації;
   b. створити новий клас аргументів для передачі їх до обробника виключної ситуації;
   c. забезпечити ініціювання створеної виключної ситуації та продемонструвати, як працює обробник даної помилки;
   d. реалізувати різні сценарії оброблення помилки.
6. Метод-розширення будь-якого класу **(1 бал)**.

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

UML діаграма класів



Фрагменти коду

1.

```
interface IElectroCar
    {
        void ChargeCar(RefuelingOperator oper, FillingEventArgs fArgs);
    }
interface IICECar
    {
        void FillCar(RefuelingOperator oper, FillingEventArgs fArgs);
    }
interface IMazdaMX5
    {
        void OpenRoof();
    }
```

```
interface ITeslaModelX
    {
        void RemoteOpenDoor(int doorNumber);
    }
```

```
class TeslaModelX : ElectroCar, IElectroCar, ITeslaModelX
    {
      ...
    }
```

```
void IElectroCar.ChargeCar(RefuelingOperator oper, FillingEventArgs fArgs)
        {
          ...
        }
```

2.

```
abstract class Car : IDisposable
    {
      public abstract void Drive(double distance);
    }
```

```
class ElectroCar : Car, IElectroCar
    {
      ...
    }
```

3.

```
public delegate void FillingHandle(RefuelingOperator oper, FillingEventArgs
fArgs);
```

```
public class FillingEventArgs : EventArgs
    {
     ...
    }
```

```
class FillStation
    {
```

```
        public FillStation(RefuelingOperator oper)
        {
            IElectroCar[] eCars = new ElectroCar[2];
            eCars[0] = TeslaModelX.GetCar();
            eCars[1] = TeslaModelX.GetCar("Gregory", "OO0000OO", 100_000,
24_000);

            IICECar[] iceCars = new ICECar[2];
            iceCars[0] = MazdaMX5.GetCar();
            iceCars[1] = MazdaMX5.GetCar("StickMan", "JJ9999JJ", 40, 4);

            for (int i = 0; i < 2; i++)
            {
                oper.FillingEvent += new FillingHandle(eCars[i].ChargeCar);

                oper.FillingEvent += new FillingHandle(iceCars[i].FillCar);
            }
        }
    }
```

```
public class RefuelingOperator
    {
     public event FillingHandle FillingEvent;
     public FillingEventArgs fArgs;

       …
         public void FillUp()
         {
          double volume, power;
            …
          fillArgs = new FillingEventArgs(volume, power);
          fArgs = fillArgs;

            …

          if (FillingEvent != null)
              {
                 FillingEvent((RefuelingOperator)this, fillArgs);
              }
          }
       …
```

```
    }
```

4.

```
FillingHandle currentFillInfo = delegate(RefuelingOperator oper,
FillingEventArgs fArgs)
            {
                Console.WriteLine($"Current filling/charging operator:
[{oper.Fullname}]; Current filling volume: [{fArgs.FillVolume}] litres;
Current charging power: [{fArgs.Power}] Watt");
            };
```

```
Action<RefuelingOperator> newClientReact = oper =>
Console.WriteLine($"[{oper.Fullname}] see new client car, it's [Electric
car]");

Func<double, double, double> calcPercent = (currValue, maxValue) =>
currValue / maxValue * 100;
```

5.

```
class FillArgumentsException : Exception
    {
        private FillingEventArgs fArgs;

        public FillArgumentsException(FillingEventArgs fArgs)
        {
            this.fArgs = fArgs;
        }

        public override string Message => $"Incorrect filling values
entered. Only positive arguments allowed";
    }
```

```
public class FillingEventArgs : EventArgs
    {
        private double fuelVolume;
        private double chargePower;
```

```
        …
    }
```

```
try
            {
                Console.Write("Enter filling volume: ");
                volume = Double.Parse(Console.ReadLine());

                Console.Write("Enter charge power: ");
                power = Double.Parse(Console.ReadLine());

                fillArgs = new FillingEventArgs(volume, power);
            }
            catch (Exception ex)
            {
                Console.WriteLine("".PadRight(83, '#'));
                Console.WriteLine(ex.Message);
                Console.WriteLine("Set to default values: [Fill volume: 1
liter]; [Charge power: 1_000 Watt]");
                Console.WriteLine("".PadRight(83, '#'));

                fillArgs = new FillingEventArgs();
            }
            finally
            {
              …
            }
```

6.

```
static class TeslaExtension
    {
        public static void OnPartyMode(this TeslaModelX tesla)
        {
            Console.WriteLine("Party mode is on");
        }
    }
```

Висновки

В результаті виконання лабораторної роботи я ознайомився з такими можливостями мови C#: абстрактні класи, інтерфейси, делегати. Також зрозумів механізми оброблення подій у C# і можливості, що надають методи-розширення.