МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Звіт

про виконання лабораторної роботи № 4

на тему:

«Успадкування в C#»

Викона(в/ла):

Студент(ка) групи ФеП-12

Шита М.О.

Перевірив:

Щербак С.С

Львів 2020

***Мета роботи*:** розглянути з один з базових принципів об'єктно-орієнтованого програмування – успадкування, вивчити поняття класів, навчитися їх застосовувати.

Обладнання: ноутбук, інтегроване середовище розробки програмного забезпечення Microsoft Visual Studio (2019).

Теоретичні відомості

У програмуванні спадкування дозволяє створювати новий клас на базі іншого. Клас, на базі якого створюється новий клас, називається базовим, а базується новий клас - спадкоємцем або похідним класом. В клас-спадкоємець з базового класу переходять поля, властивості, методи і інші члени класу.

Оголошення нового класу, який буде наслідувати інший клас, виглядає так:

```
class [ім'я_класу]: [імя_базового_класса]

{

  // Тіло класу

}
```

Конструктор базового класу буде створювати ту частину об'єкта, яка належить базовому класу, а конструктор з спадкоємця буде створювати свою частину.

Коли конструктор визначений тільки в спадкоємця, то при створенні об'єкта спочатку викликається конструктор за замовчуванням базового класу, а потім конструктор спадкоємця.

Коли конструктори оголошені і в базовому класі, і в спадкоємця - нам необхідно викликати їх обидва. Для виклику конструктора базового класу використовується ключове слово base. Оголошення конструктора класу-спадкоємця з викликом базового конструктора має наступну структуру:

[імя_конструктора_класса-спадкоємця] ([аргументи]): base ([аргументи])

{

  // Тіло конструктора\

}

У базовий конструктор передаються всі необхідні аргументи для створення базової частини об'єкта.

Віртуальний метод - це метод, який може бути перевизначений в класі спадкоємця.

Перевизначення методу - це зміна його реалізації в класі спадкоємця. Перевизначивши метод, він працюватиме по-різному в базовому класі і класі спадкоємця, маючи при цьому одне і те ж ім'я та аргументи і тип повернення.

Віртуальний метод оголошується за допомогою ключового слова virtual:

[модифікатор доступу] virtual [тип] [ім'я методу] ([аргументи])

{

  // Тіло методу

}

Статичний метод не може бути віртуальним.

Оголосивши віртуальний метод, можемо перевизначити його в класі спадкоємця. Для цього використовується ключове слово override:

[модифікатор доступу] override [тип] [ім'я методу] ([аргументи])

{

  // Нове тіло методу

}

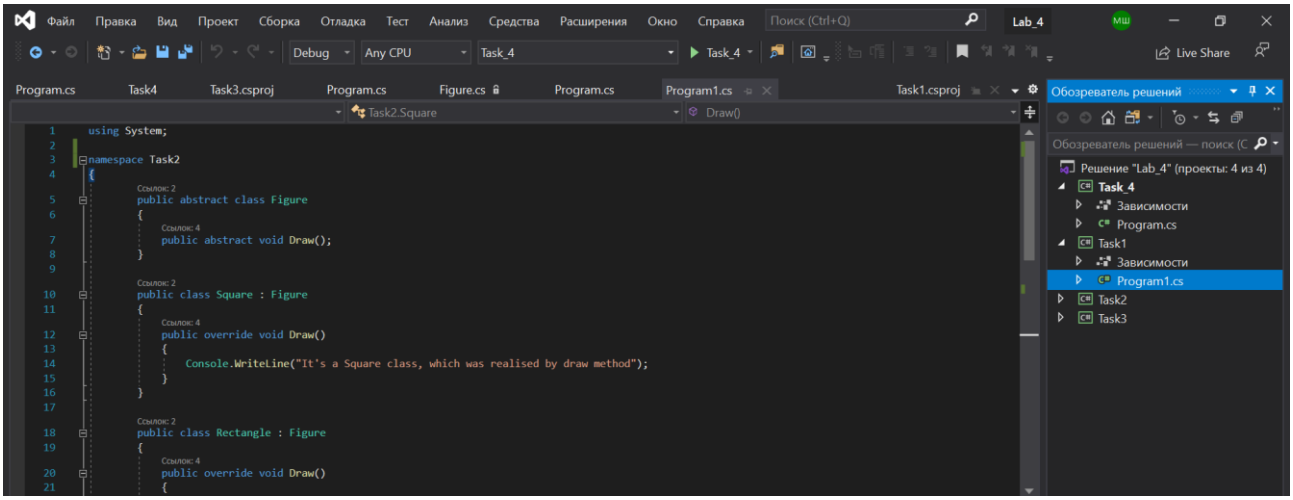
Виклик базового методу

Якщо функціонал методу, який перевизначається, в базовому класі мало відрізняється від функціоналу, який повинен бути визначений в класі спадкоємця. У такому випадку, при перевизначенні, можна викликати спочатку цей метод з базового класу, а далі дописати необхідний функціонал. Це робиться за допомогою ключового слова base

Хід роботи

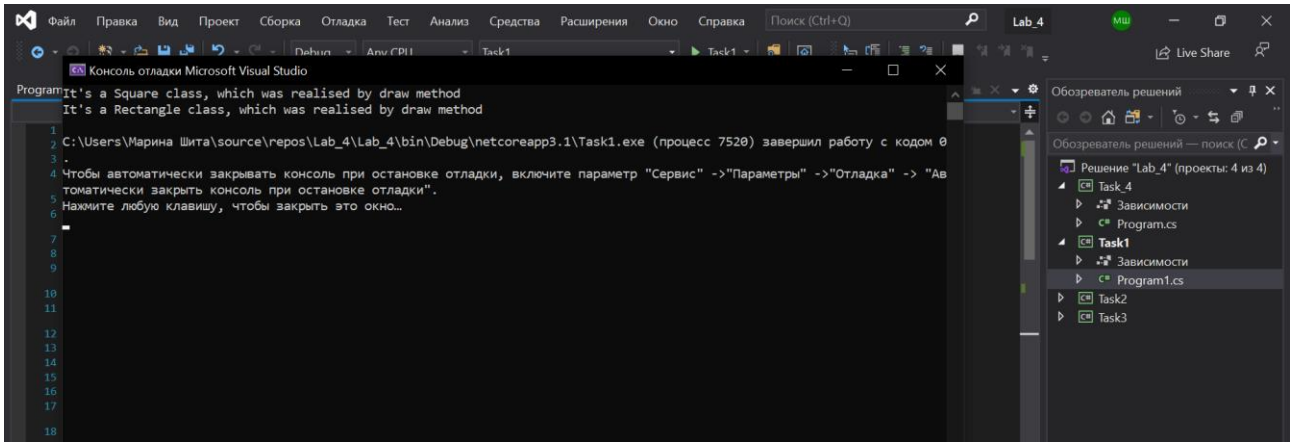
Завдання 1:

Реалізувала базовий клас Figure з абстрактним методом Draw(). Створила класи Square та Rectangle (унаслідовані від Figure) в яких перегружена імплементація методу Draw(), виводить на консоль назву класу в якому даний метод реалізований.



```
using System;

namespace Task2
{
    public abstract class Figure
    {
        public abstract void Draw();
    }

    public class Square : Figure
    {
        public override void Draw()
        {
            Console.WriteLine("It's a Square class, which was realised by draw method");
        }
    }

    public class Rectangle : Figure
    {
        public override void Draw()
        {
```

зображення коду



```
It's a Square class, which was realised by draw method
It's a Rectangle class, which was realised by draw method

C:\Users\Марина Шита\source\repos\Lab_4\Lab_4\bin\Debug\netcoreapp3.1\Task1.exe (процесс 7520) завершил работу с кодом 0
.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Ав
томатически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно…
```

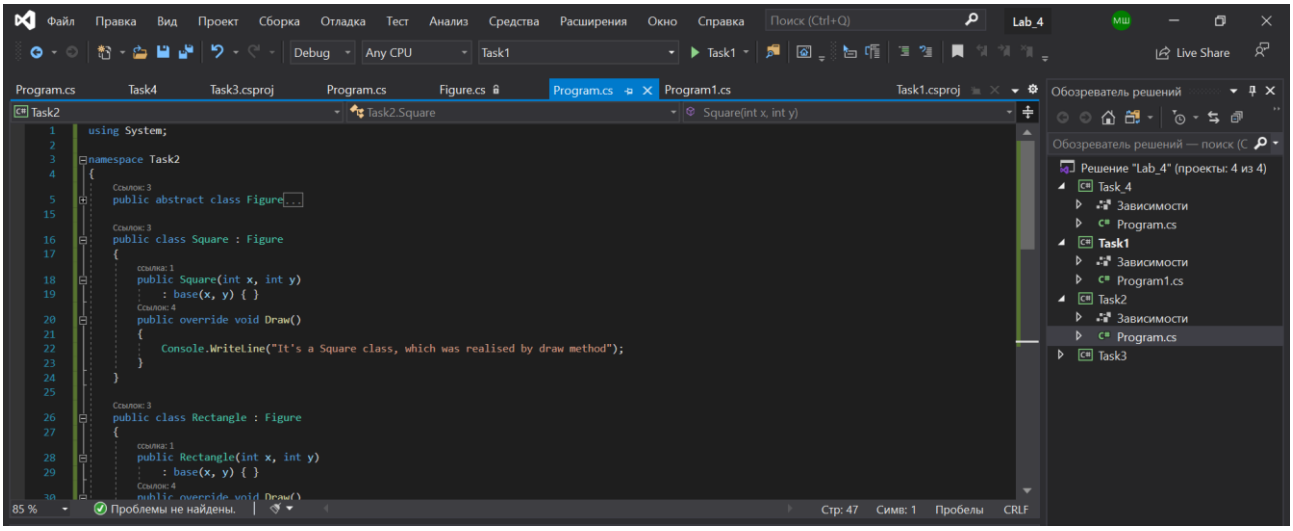
вигляд програми

Приклад коду:

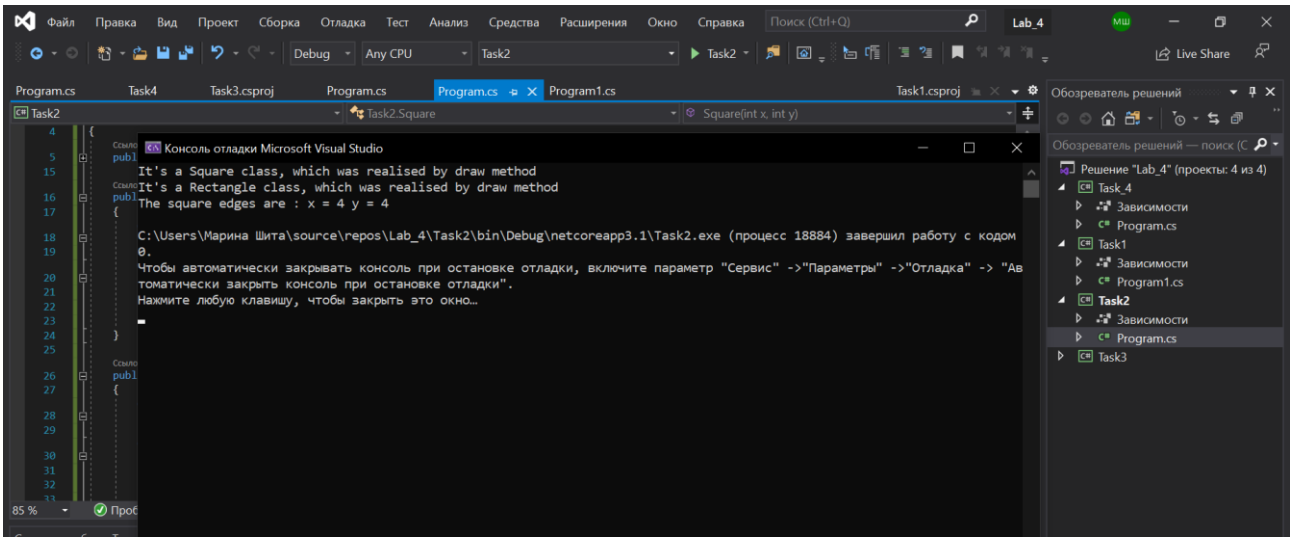
```
using System;

namespace Task1
{
    public abstract class Figure
    {
        public abstract void Draw();
    }

    public class Square : Figure
    {
        public override void Draw()
        {
            Console.WriteLine("It's a Square class, which was realised by draw
method");
        }
    }

    public class Rectangle : Figure
    {
        public override void Draw()
        {
            Console.WriteLine("It's a Rectangle class, which was realised by draw
method");
        }
    }
    class Program1
    {
        static void Main()
        {
            Square squarefigure = new Square();
            Rectangle rectanglefigure = new Rectangle();
            squarefigure.Draw();
            rectanglefigure.Draw();
        }
    }
}
```

Завдання 2:

Використовуючи попереднє завдання, я добавила в клас Figure readonly властивості X та Y. Також, в клас Figure добавила параметризований конструктор, що установлює значення в цих властивостях. Поправила унаслідовані класи щоб вони коректно працювали викликаючи базовий конструктор.



зображення коду



вигляд програми

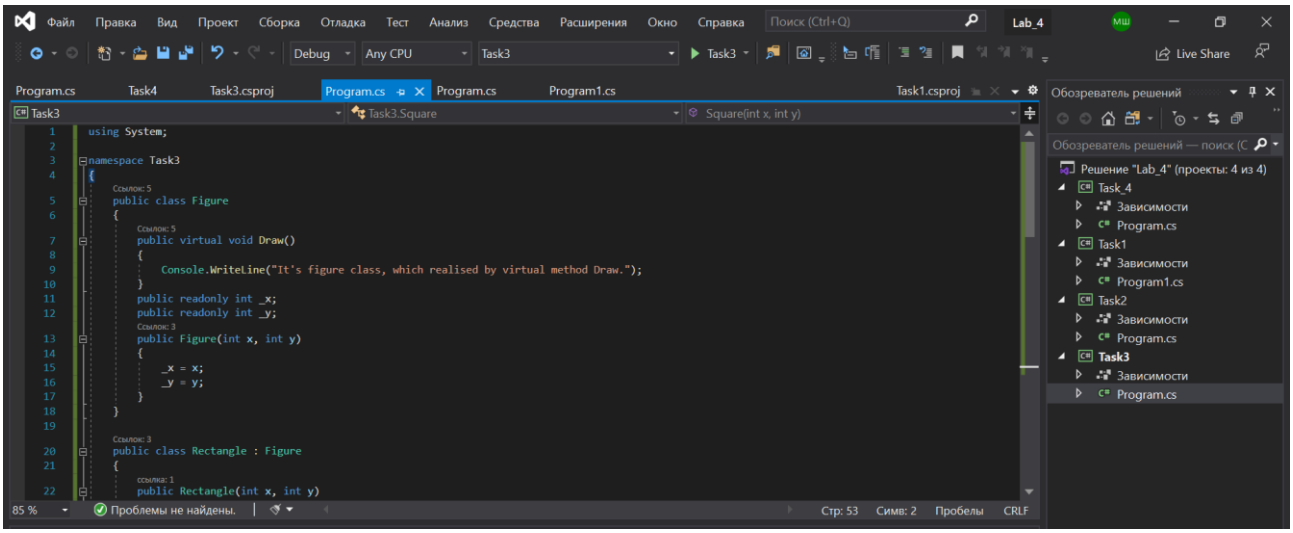
Приклад коду:

```
using System;

namespace Task2
{
    public abstract class Figure
    {
        public abstract void Draw();
        public readonly int _x, _y;
        protected Figure(int x, int y)
        {
            _x = x;
            _y = y;
        }
    }

    public class Square : Figure
    {
        public Square(int x, int y)
            : base(x, y) { }
        public override void Draw()
        {
            Console.WriteLine("It's a Square class, which was realised by draw method");
        }
    }

    public class Rectangle : Figure
    {
        public Rectangle(int x, int y)
            : base(x, y) { }
        public override void Draw()
        {
            Console.WriteLine("It's a Rectangle class, which was realised by draw
method");
        }
    }
    class Program
    {
        static void Main()
        {
            Square squarefigure = new Square(4, 4);
            Rectangle rectanglefigure = new Rectangle(2, 3);
            squarefigure.Draw();
            rectanglefigure.Draw();
```

```
            Console.WriteLine("The square edges are : x = {0} y = {1}", squarefigure._x,
squarefigure._y);
        }
    }
}
```

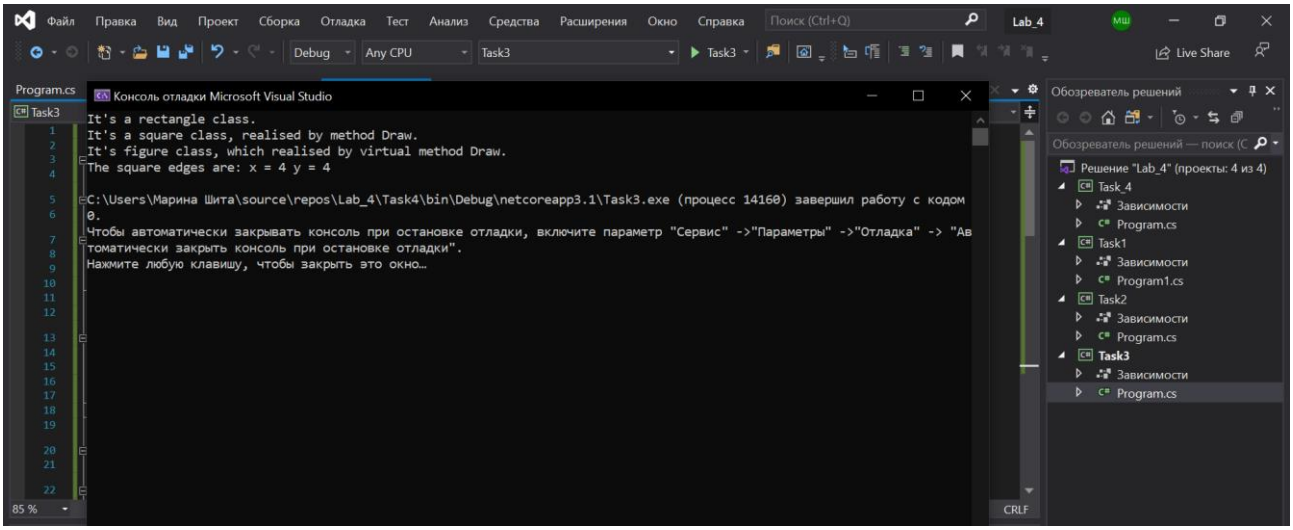
Завдання 3:

Використовуючи попереднє завдання змінила абстрактний метод Draw() на віртуальний і реалізувала його аналогічно, як у класах Square та Rectangle. Об'єкт класу Figure тепер може бути ініціалізований і виклики методу Draw() на об'єктах створених класів приведуть до виклику трьох різних реалізацій.



```
using System;

namespace Task3
{
    public class Figure
    {
        public virtual void Draw()
        {
            Console.WriteLine("It's figure class, which realised by virtual method Draw.");
        }
        public readonly int _x;
        public readonly int _y;
        public Figure(int x, int y)
        {
            _x = x;
            _y = y;
        }
    }

    public class Rectangle : Figure
    {
        public Rectangle(int x, int y)
```

зображення коду



```
It's a rectangle class.
It's a square class, realised by method Draw.
It's figure class, which realised by virtual method Draw.
The square edges are: x = 4 y = 4

C:\Users\Марина Шита\source\repos\Lab_4\Task4\bin\Debug\netcoreapp3.1\Task3.exe (процесс 14160) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Автоматически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно…
```

вигляд програми

Приклад коду:

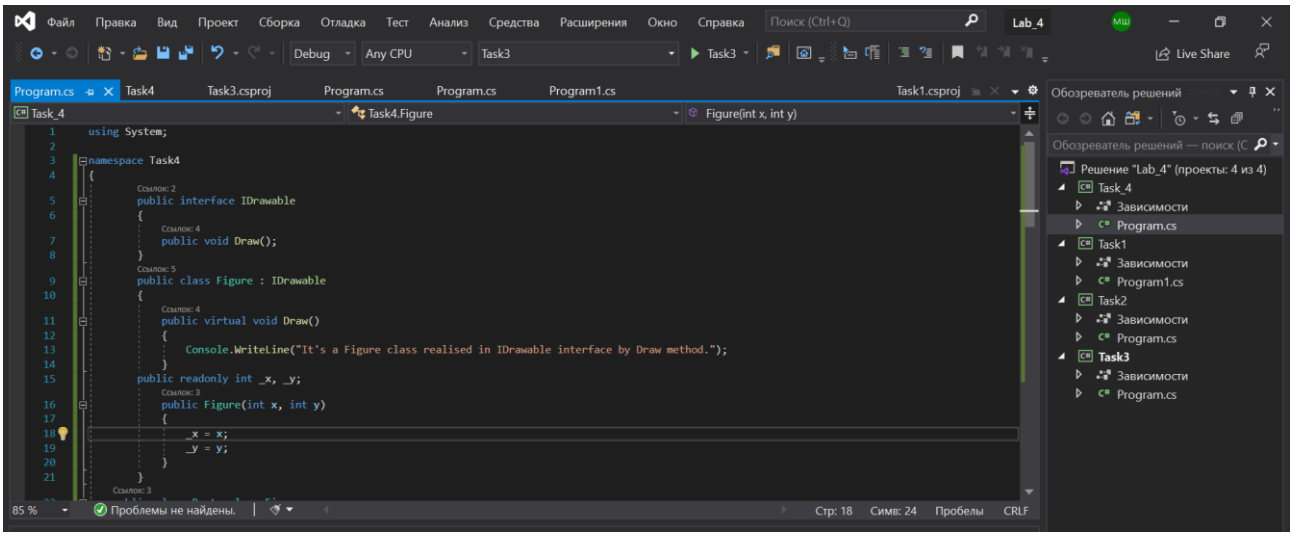
```
using System;

namespace Task3
{
    public class Figure
    {
        public virtual void Draw()
        {
            Console.WriteLine("It's figure class, which realised by virtual method
Draw.");
        }
        public readonly int _x;
        public readonly int _y;
        public Figure(int x, int y)
        {
            _x = x;
            _y = y;
        }
    }

    public class Rectangle : Figure
    {
        public Rectangle(int x, int y)
            : base(x, y) { }
        public override void Draw()
        {
            Console.WriteLine("It's a rectangle class.");
        }
    }

    public class Square : Figure
    {
        public Square(int x, int y)
            : base(x, y) { }
        public override void Draw()
        {
            Console.WriteLine("It's a square class, realised by method Draw.");
        }
    }

    class Program
    {
        static void Main()
        {
            Rectangle rectangle = new Rectangle(2, 3);
```

```
            Square square = new Square(4, 4);
            Figure justfigure = new Figure(8, 12);
            rectangle.Draw();
            square.Draw();
            justfigure.Draw();
            Console.WriteLine("The square edges are: x = {0} y = {1}", square._x,
square._y);
        }
    }
}
```

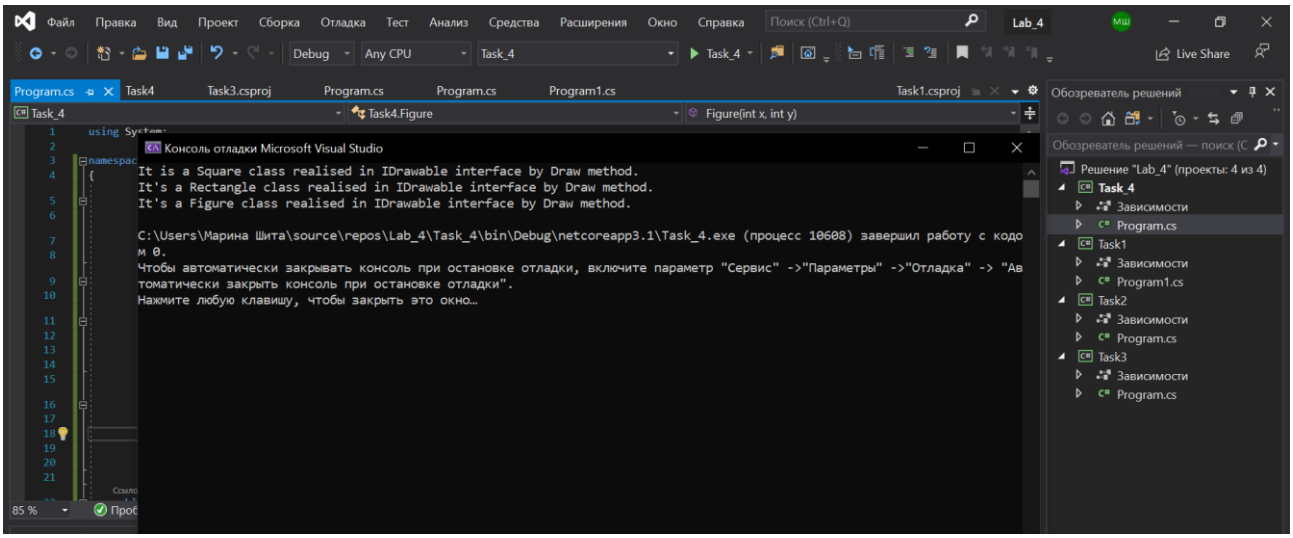
Завдання 4:

Використовуючи попереднє завдання винесла метод Draw() у інтерфейс IDrawable.

Створила метод DrawAll(params IDrawable[] array) в який передала об'єкти створених класів. І в циклі викликала метод Draw() для кожного з них.



зображення коду



вигляд програми

Приклад коду:

```
using System;

namespace Task4
{
        public interface IDrawable
        {
            public void Draw();
        }
        public class Figure : IDrawable
        {
            public virtual void Draw()
            {
                Console.WriteLine("It's a Figure class realised in IDrawable interface by
Draw method.");
            }
        public readonly int _x, _y;
            public Figure(int x, int y)
            {
                _x = x;
                _y = y;
            }
        }
    public class Rectangle : Figure
    {
        public Rectangle(int x, int y)
            : base(x, y) { }
        public override void Draw()
        {
            Console.WriteLine("It's a Rectangle class realised in IDrawable interface by
Draw method.");
        }
    }

    public class Square : Figure
        {
            public Square(int x, int y)
                : base(x, y) { }
            public override void Draw()
            {
                Console.WriteLine("It is a Square class realised in IDrawable interface by
Draw method.");
            }
        }
```

```
    class Program
    {
        static void Main()
        {
            Square square = new Square(2, 2);
            Rectangle rectangle = new Rectangle(2, 3);
            Figure justfigure = new Figure(4, 12);
            DrawAll(square, rectangle, justfigure);
        }
        private static void DrawAll(params IDrawable[] array)
        {
            for (int i = 0; i < array.Length; i++)
            {
                array[i].Draw();
            }
        }
    }
}
```

Висновок: протягом виконання даної лабораторної роботи я ознайомилася з одиним з базових принципів об'єктно-орієнтованого програмування – успадкуванням, вивчила поняття класів та навчилася їх застосовувати.