

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Звіт

про виконання лабораторної роботи № 3

на тему:

«Класи в мові C#»

Викона(в/ла):

Студент(ка) групи Фел-12

Шита М.О.

Перевірив:

Щербак С.С

Львів 2020

Мета роботи: ознайомитися з класами та методами мови програмування C#.

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

Теоретичні відомості

Клас - це абстрактний тип даних, деякий шаблон, на основі якого будуть створюватися його екземпляри - об'єкти.

У Сі-шарп класи оголошуються за допомогою ключового слова `class`. Загальна структура оголошення виглядає наступним чином:

```
[модифікатор доступу] class [ім'я_класу]
{

    // тіло класу

}
```

Модифікаторів доступу для класів є два:

- **Public** - доступ до класу можливий з будь-якого місця однієї збірки або з іншої збірки, на яку є посилання;
- **Internal** - доступ до класу можливий тільки з збірки, в якій він оголошений.

Збірка (assembly) - це готовий функціональний модуль у вигляді exe або dll файлу (файлів), який містить скомпільований код для .NET. Збірка надає можливість повторного використання коду.

При оголошенні класу модифікатор доступу можна не вказувати, при цьому застосовуватиметься режим за замовчуванням `internal`.

Клас слід оголошувати всередині простору імен `namespace`, але за межами іншого класу (можливо також оголошення класу всередині іншого - вкладені типи).

Приклад оголошення класів Student і Pupil:

```
namespace HelloWorld
```

```
{  
    class Student // без вказівки модифікатор доступу, клас буде internal  
    {  
        // тіло класу  
    }  
    public class Pupil  
    {  
        // тіло класу  
    }  
}
```

Класи в Сі-шарп можуть містити наступні члени:

- Поля;
- Константи;
- Властивості;
- Конструктори;
- Методи;
- Події;
- Оператори;
- Індексатори;
- Вкладені типи.

Всі члени класу, як і сам клас, мають свій рівень доступу. Тільки у членів їх може бути вже п'ять:

- **Public** - доступ до члена можливий з будь-якого місця однієї збірки, або з іншої збірки, на яку є посилання;
- **Protected** - доступ до члена можливий тільки усередині класу, або в класі-спадкоємця (при спадкуванні);

- **Internal** - доступ до члена можливий тільки з збірки, в якій він проголошений;
- **Private** - доступ до члена можливий тільки всередині класу;
- **Protected internal** - доступ до члена можливий з однієї збірки, або з класу-спадкоємця іншої збірки.

Не вказавши модифікатор доступу для члена, за замовчуванням йому буде присвоєно режим `private`.

За допомогою модифікаторів доступу в Сі-шарп реалізується один із базових принципів ООП - **інкапсуляція**.

Поля класу

Поле - це змінна, оголошена усередині класу. Як правило, поля оголошуються за модифікаторами доступу `private` або `protected`, щоб заборонити прямий доступ до них. Для отримання доступу до полів слід використовувати властивості або методи.

Приклад оголошення полів у класі:

```
class Student
{
    private string firstName;
    private string lastName;
    private int age;

    public string group;
}
```

Створення об'єктів відбувається за допомогою ключового слова `new` і імені класу:

```
namespace HelloWorld
```

```
{
```

```
class Student
{
    private string firstName;
    private string lastName;
    private int age;
    public string group;
}
```

```
class Program
{
    static void Main (string [] args)
    {
        Student student1 = new Student ();
        Student student2 = new Student ();
    }
}
```

Доступ до членів об'єкта здійснюється за допомогою оператора точка «.»:

```
static void Main (string [] args)
{
    Student student1 = new Student ();
    Student student2 = new Student ();
    student1.group = "Group1";
    student2.group = "Group2";
```

```
    Console.WriteLine (student1.group); // Виводить на екран "Group1"
    Console.Write (student2.group);
    Console.ReadKey ();
}
```

Такі поля класу Student, як firstName, lastName і age вказані з модифікатором доступу private, тому доступ до них буде заборонений поза класом:

```
static void Main (string [] args)
{
    Student student1 = new Student ();
    student1.firstName = "Nikolay"; // помилка, немає доступу
}
```

Константи

Константа - це змінна, значень якої не можна змінити. Константа оголошується за допомогою ключового слова const. Приклад оголошення константи:

```
class Math
{
    private const double Pi = 3.14;
}
```

Метод - це невелика підпрограма, яка виконує, в ідеалі, тільки одну функцію. Методи дозволяють скоротити обсяг коду. Методи разом з полями, є основними членами класу.

Статичний метод - це метод, який не має доступу до полів об'єкта, і для виклику такого методу не потрібно створювати екземпляр (об'єкт) класу, в якому він оголошений.

Простий метод - це метод, який має доступ до даних об'єкта, і його виклик виконується через об'єкт. Прості методи служать для обробки внутрішніх даних об'єкта.

Клас Телевізор, у нього є поле switchedOn, яке відображає стан включений / виключений, і два методи - включення і виключення:

```
class TVSet
{
    private bool switchedOn;
    public void SwitchOn()
```

```

    {
        switchedOn = true;
    }

    public void SwitchOff()
    {
        switchedOn = false;
    }
}

class Program
{
    static void Main(string[] args)
    {
        TVSet myTV = new TVSet();
        myTV.SwitchOn(); // включаємо телевізор, switchedOn = true;
        myTV.SwitchOff(); // виключаємо телевізор, switchedOn = false;
    }
}

```

Щоб викликати простий метод, перед його ім'ям, вказується ім'я об'єкта. Для виклику статичного методу необхідно вказувати ім'я класу.

Статичні методи, зазвичай, виконують якусь глобальну, загальну функцію, обробляють «зовнішні дані». Наприклад, сортування масиву, обробка рядка, зведення числа в ступінь і інше.

Приклад статичного методу, який обрізає рядок до вказаної довжини, і додає крапки:

```

class StringHelper
{
    public static string TrimIt(string s, int max)
    {
        if (s == null)
            return string.Empty;
    }
}

```

```

    if (s.Length <= max)
        return s;

    return s.Substring(0, max) + "...";
}
}
class Program
{
    static void Main(string[] args)
    {
        string s = "Дуже довгий рядок, який необхідно обрізати до зазначеної
довжини і додати три крапки";
        Console.WriteLine(StringHelper.TrimIt(s, 20)); //"Дуже довгий рядок ..."
        Console.ReadLine();
    }
}

```

Статичний метод не має доступу до нестатичних полів класу:

```

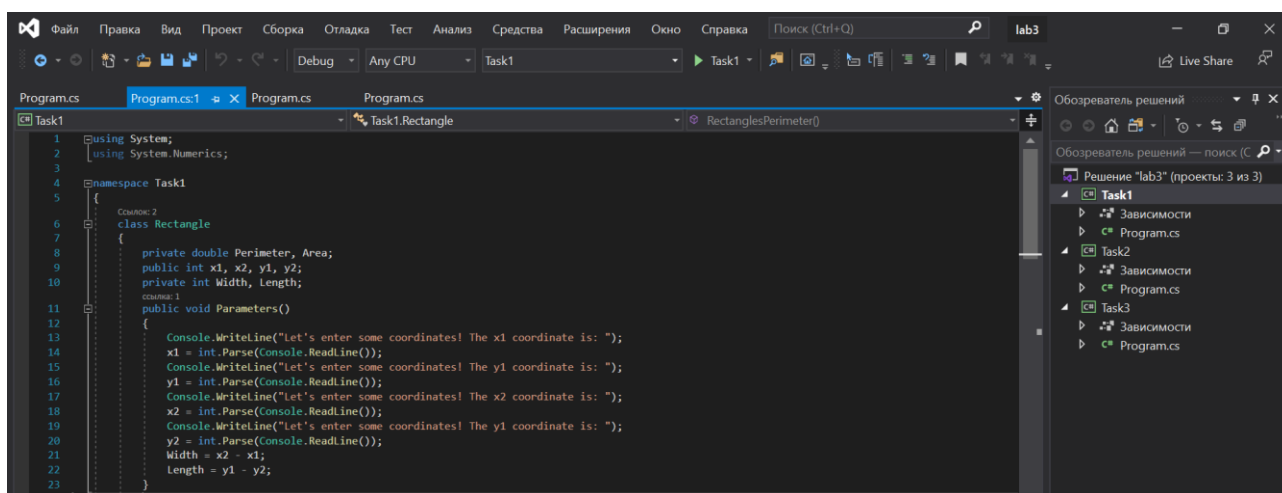
class SomeClass
{
    private int a;
    private static int b;
    public static void SomeMethod()
    {
        a=5; // помилка
        b=10; // допустимо
    }
}

```

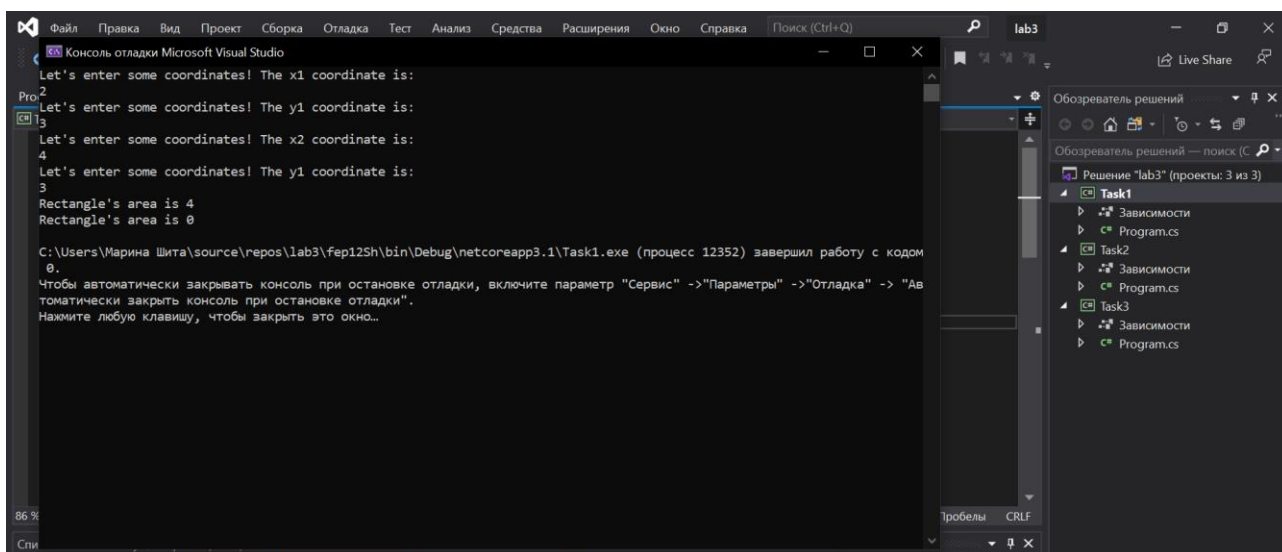

ПОРЯДОК ВИКОНАННЯ РОБОТИ:

1. Не використовуючи System.Math реалізувати клас Rectangle з методами, що дозволяють обрахувати периметр та площу прямокутника. Координати лівого верхнього та правого нижнього кута передаються параметрами в конструктор класу Rectangle.
2. Реалізувати завдання 1 використовуючи автоматично реалізовані властивості (AutoImplemented Properties) замість методів.
3. Не використовуючи System.Math реалізувати клас Circle з методами, що дозволяють обрахувати довжину кола та площу круга. Радіус передається параметром в відповідний метод. Константи для підрахунків повинні знаходитись в класі Circle.
4. Ознайомитись з теоретичними відомостями, написати висновок. В звіті привести приклади та зображення коду.

Метою першого завдання стало написання програми для ПЕОМ, котра змогла б не використовуючи System.Math реалізувати клас Rectangle з методами, що дозволяють обрахувати периметр та площу прямокутника. Додатковою умовою, було те, що координати лівого верхнього та правого нижнього кута передаються параметрами в конструктор класу.



```
1 using System;
2 using System.Numerics;
3
4 namespace Task1
5 {
6     class Rectangle
7     {
8         private double Perimeter, Area;
9         public int x1, x2, y1, y2;
10        private int Width, Length;
11        public void Parameters()
12        {
13            Console.WriteLine("Let's enter some coordinates! The x1 coordinate is: ");
14            x1 = int.Parse(Console.ReadLine());
15            Console.WriteLine("Let's enter some coordinates! The y1 coordinate is: ");
16            y1 = int.Parse(Console.ReadLine());
17            Console.WriteLine("Let's enter some coordinates! The x2 coordinate is: ");
18            x2 = int.Parse(Console.ReadLine());
19            Console.WriteLine("Let's enter some coordinates! The y2 coordinate is: ");
20            y2 = int.Parse(Console.ReadLine());
21            Width = x2 - x1;
22            Length = y2 - y1;
23        }
24    }
25 }
```



```
Консоль отладки Microsoft Visual Studio
Let's enter some coordinates! The x1 coordinate is:
Pro2 Let's enter some coordinates! The y1 coordinate is:
3 Let's enter some coordinates! The x2 coordinate is:
4 Let's enter some coordinates! The y2 coordinate is:
3
Rectangle's area is 4
Rectangle's area is 0
C:\Users\Марина Шига\source\repos\lab3\fepl2Sh\bin\Debug\netcoreapp3.1\Task1.exe (процесс 12352) 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...
86 %
Спи
```

зображення коду та виконання першого завдання

Приклад коду:

```
using System;
using System.Numerics;

namespace Task1
{
    class Rectangle
    {
        private double Perimeter, Area;
        public int x1, x2, y1, y2;
        private int Width, Length;
        public void Parameters()
        {
            Console.WriteLine("Let's enter some coordinates! The x1 coordinate is: ");
            x1 = int.Parse(Console.ReadLine());
            Console.WriteLine("Let's enter some coordinates! The y1 coordinate is: ");
            y1 = int.Parse(Console.ReadLine());
            Console.WriteLine("Let's enter some coordinates! The x2 coordinate is: ");
            x2 = int.Parse(Console.ReadLine());
            Console.WriteLine("Let's enter some coordinates! The y2 coordinate is: ");
            y2 = int.Parse(Console.ReadLine());
            Width = x2 - x1;
            Length = y2 - y1;
        }

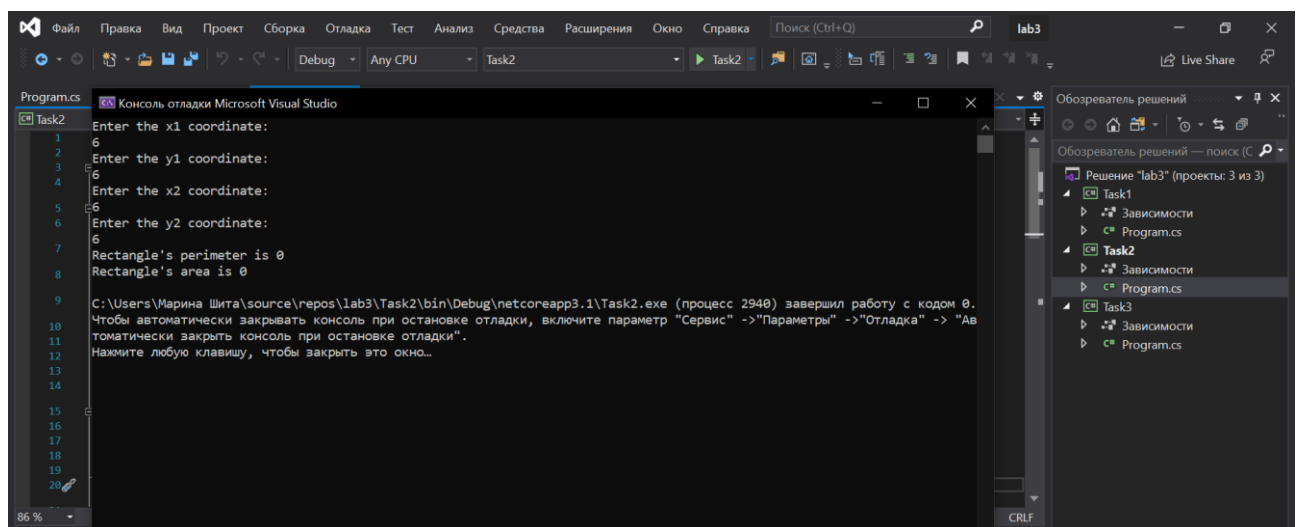
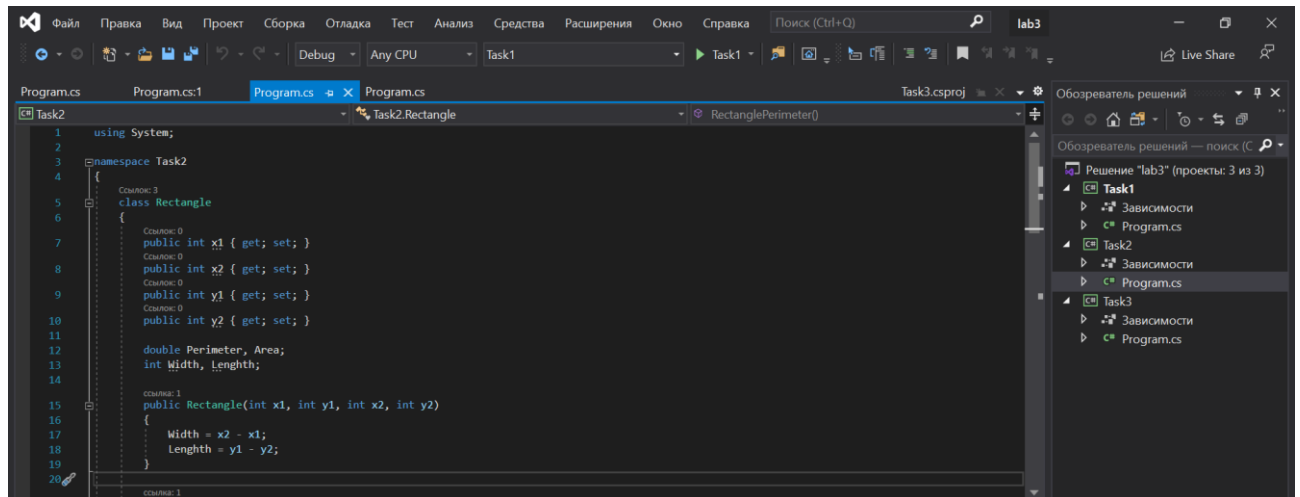
        public void RectanglesPerimeter()
        {
            Perimeter = 2 * (Width + Length);
            Console.WriteLine("Rectangle's area is {0}", Perimeter);
        }

        public void RectanglesArea()
        {
            Area = Width * Length;
            Console.WriteLine("Rectangle's area is {0}", Area);
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            Rectangle rectangle = new Rectangle();
            rectangle.Parameters();
            rectangle.RectanglesPerimeter();
        }
    }
}
```

```
        rectangle.RectanglesArea();  
    }  
}  
}
```

Завдання два полягало в тому, щоб написати першу програму, проте цього разу, використавши автоматично реалізовані властивості. А не методи.



зображення коду та виконання другого завдання

Приклад коду:

```
using System;

namespace Task2
{
    class Rectangle
    {
        public int x1 { get; set; }
        public int x2 { get; set; }
        public int y1 { get; set; }
        public int y2 { get; set; }

        double Perimeter, Area;
        int Width, Length;

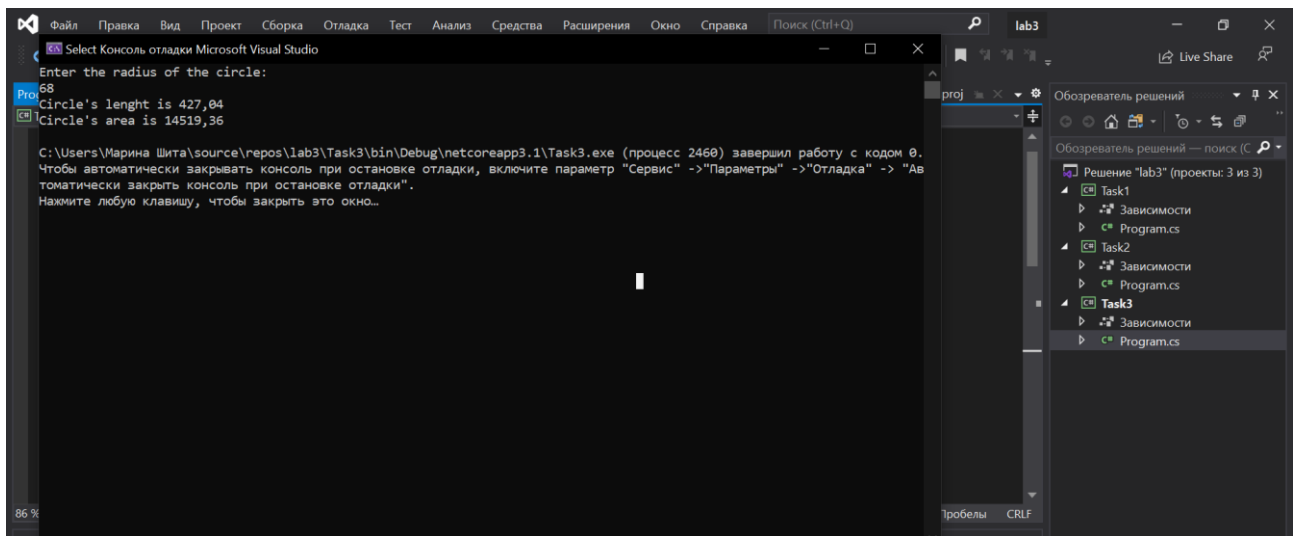
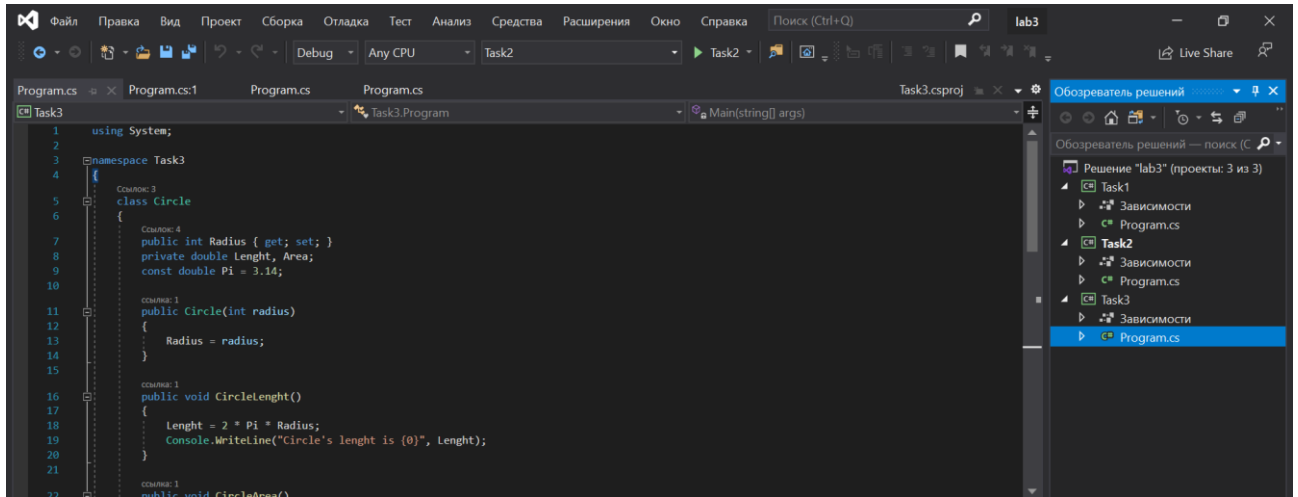
        public Rectangle(int x1, int y1, int x2, int y2)
        {
            Width = x2 - x1;
            Length = y1 - y2;
        }

        public void RectanglePerimeter()
        {
            Perimeter = 2 * (Width + Length);
            Console.WriteLine("Rectangle's perimeter is {0}", Perimeter);
        }

        public void RectangleArea()
        {
            Area = Width * Length;
            Console.WriteLine("Rectangle's area is {0}", Area);
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            int[] coordinates = new int[4];
            Console.WriteLine("Enter the x1 coordinate: ");
            coordinates[0] = int.Parse(Console.ReadLine());
            Console.WriteLine("Enter the y1 coordinate: ");
            coordinates[1] = int.Parse(Console.ReadLine());
            Console.WriteLine("Enter the x2 coordinate: ");
            coordinates[2] = int.Parse(Console.ReadLine());
            Console.WriteLine("Enter the y2 coordinate: ");
            coordinates[3] = int.Parse(Console.ReadLine());
        }
    }
}
```

```
    Rectangle rect = new Rectangle(coordinates[0], coordinates[1],
coordinates[2], coordinates[3]);
    rect.RectanglePerimeter();
    rect.RectangleArea();
}
}
```

Виконуючи третє завдання, я написала програму, що реалізує клас Circle з методами, які дозволяють обрахувати довжину кола та площу круга. Додатковими умовами було те, що радіус передається параметром в відповідний метод, а константи для підрахунків повинні знаходитись в класі Circle.



зображення коду та виконання третього завдання

Приклад коду:

```
namespace Task3
{
    class Circle
    {
        public int Radius { get; set; }
        private double Lenght, Area;
        const double Pi = 3.14;

        public Circle(int radius)
        {
            Radius = radius;
        }

        public void CircleLenght()
        {
            Lenght = 2 * Pi * Radius;
            Console.WriteLine("Circle's lenght is {0}", Lenght);
        }

        public void CircleArea()
        {
            Area = Pi * Radius * Radius;
            Console.WriteLine("Circle's area is {0}", Area);
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Enter the radius of the circle: ");
            int radius = int.Parse(Console.ReadLine());
            Circle circle = new Circle(radius);
            circle.CircleLenght();
            circle.CircleArea();
        }
    }
}
```

Висновок: на цій лабораторній роботі я ознайомилася з класами та методами мови програмування. А також написала 3 програми для ПЕОМ, що дали змогу краще ознайомитися з завданням класів в мові С#.