Київський національний університет імені Тараса Шевченка
Факультет інформаційних технологій

Кафедра програмних систем і технологій

ЗВІТ

з лабораторної роботи № 4

Тема: “Автоматизована система підземного паркування”

Дисципліна «Об’єктно-орієнтоване конструювання програм»

Підготував:

студент гр. ІПЗ-23(1)

Польнюк Руслан Дмитрович

Дата : _____

Перевірила:

доц. Зубик Л.В.

Тема: Породжуючі паттерни.

Мета роботи: застосувати на практиці один з породжуючих паттернів наданих в роботі.

Умова:

Розробити класи завдань за допомогою фабричних патернів. Протестувати роботу класів в консольному застосунку.

Завдання: Створити клас **Площа геометричної фігури**, який обраховує площі (трикутника, прямокутника, трапеції, круга, ромба, квадрата, паралелограма). Перевіряти на коректність введених даних.

Тема проекту: “Автоматизована система підземного паркування”.

Хід роботи:

Для реалізації даної лабораторної роботи з запропонованих паттернів я обрав паттерн “Фабричний метод”, тому що він цілком підходить під мій варіант, ми повинні створити базовий клас фабрика мов, та створювати нові об’єкти вже в спадкоємцях цього класу, згодом можна розширювати застосунок додаванням нових геометричних фігур.

Перший крок який ми повинні зробити, це створити базовий інтерфейс, який будуть реалізовувати класи спадкоємці, в моєму випадку - це інтерфейс фігур.

```
Ссылок: 0
internal interface IPolygon
{
    Ссылок: 0
    double Triangle(double a, double b);
    Ссылок: 0
    double Rectangle(double a, double b);
    Ссылок: 0
    double Circle(double r);
    Ссылок: 0
    double Square(double a);
    Ссылок: 0
    double Trapezium(double a, double b, double h);
    Ссылок: 0
    double Rhomb(double a, double h);
    Ссылок: 0
    double Parallelogram(double a, double h);
}
```

```
Ссылок: 1
public double Circle(double r)
{
    return Math.PI * Math.Pow(r, 2);
}

Ссылок: 1
public double Parallelogram(double a, double h)
{
    return a * h;
}

Ссылок: 1
public double Rectangle(double a, double b)
{
    return a * b;
}

Ссылок: 1
public double Rhomb(double a, double h)
{
    return a * h;
}
```

```
Ссылок: 1
public double Square(double a)
{
    return Math.Pow(a, 2);
}

Ссылок: 1
public double Trapezium(double a, double b, double h)
{
    return ((a + b) / 2) * h;
}

Ссылок: 1
public double Triangle(double a, double b)
{
    return 1 / 2 * (a * b);
}
```

Другий крок який ми потрібні зробити для реалізації патерну, це створити базовий клас нашої фабрики, та його спадкоємців які будуть відповідати за створення об’єктів, в моєму випадку фігур. Базовий клас повинен бути абстрактним, щоб перезвичначити метод, який буде відповідати за створення екземплярів.

```
Ссылок: 0
public abstract class PolygonFactory
{
    Ссылок: 1
    protected abstract IPolygon CalculatePolygon();

    Ссылок: 0
    public IPolygon MakeCalculate()
    {
        return this.CalculatePolygon();
    }
}
```

```
Ссылок: 0
public class FigureFactory : PolygonFactory
{
    Ссылок: 2
    protected override IPolygon CalculatePolygon()
    {
        IPolygon polygon = new PolygonService();
        return polygon;
    }
}
```

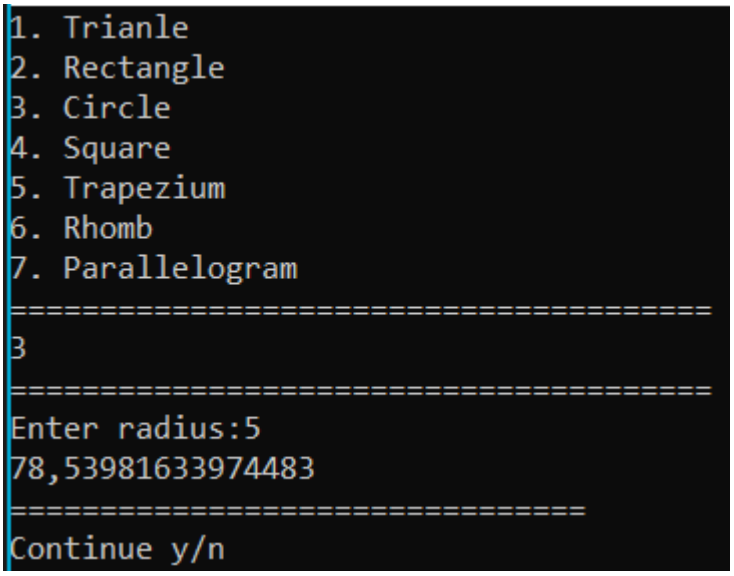
Після виконання цих кроків, можно створювати користувацький інтерфейс, в моєму випадку це консольний застосунок, згідно умов лабораторної роботи.

```
IPolygon polygon = new FigureFactory().MakeCalculate();
char key;
char answer;
do
{
    Console.WriteLine("1. Trianle");
    Console.WriteLine("2. Rectangle");
    Console.WriteLine("3. Circle");
    Console.WriteLine("4. Square");
    Console.WriteLine("5. Trapezium");
    Console.WriteLine("6. Rhomb");
    Console.WriteLine("7. Parallelogram");
    Console.WriteLine("=====================================");
    key = (char)Console.Read();
    Console.ReadLine();
    Console.WriteLine("=====================================");
```

```
case '1':
    Console.Write("Enter a:");
    double triangleA = Convert.ToDouble(Console.ReadLine());
    Console.Write("Enter b:");
    double triangleB = Convert.ToDouble(Console.ReadLine());
    Console.WriteLine(polygon.Triangle(triangleA,triangleB));
    break;
case '2':
    Console.Write("Enter a:");
    double rectangleA = Convert.ToDouble(Console.ReadLine());
    Console.Write("Enter b:");
    double rectangleB = Convert.ToDouble(Console.ReadLine());
    Console.WriteLine(polygon.Rectangle(rectangleA, rectangleB));
    break;
case '3':
    Console.Write("Enter radius:");
    double r = Convert.ToDouble(Console.ReadLine());
    Console.WriteLine(polygon.Circle(r));
    break;
case '4':
    Console.Write("Enter a:");
    double squareA = Convert.ToDouble(Console.ReadLine());
    Console.WriteLine(polygon.Square(squareA));
    break;
```

```
    break;
case '5':
    Console.Write("Enter a:");
    double trapeziumA = Convert.ToDouble(Console.ReadLine());
    Console.Write("Enter b:");
    double trapeziumB = Convert.ToDouble(Console.ReadLine());
    Console.Write("Enter h:");
    double trapeziumh = Convert.ToDouble(Console.ReadLine());
    Console.WriteLine(polygon.Trapezium(trapeziumA, trapeziumB, trapeziumh));
    break;
case '6':
    Console.Write("Enter a:");
    double rhombA = Convert.ToDouble(Console.ReadLine());
    Console.Write("Enter h:");
    double rhombh = Convert.ToDouble(Console.ReadLine());
    Console.WriteLine(polygon.Rhomb(rhombA, rhombh));
    break;
case '7':
    Console.Write("Enter a:");
    double parallelogramA = Convert.ToDouble(Console.ReadLine());
    Console.Write("Enter h:");
    double parallelogramh = Convert.ToDouble(Console.ReadLine());
    Console.WriteLine(polygon.Parallelogram(parallelogramA, parallelogramh));
    break;
```

Скріншот результатів:



Висновки:

Під час виконання цією лабораторної роботи я ознайомився з породжуючими патернами проектування, які входять в Gof, та допомагають створювати більш гнучкі та великі системи. Для реалізації завдання я обрав патерн “Фабричний метод”, тому що вважаю він більш доцільно підходить під це завдання.