Міністерство освіти і науки, молоді та спорту України
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«НАЦІОНАЛЬНИЙ ГІРНИЧИЙ УНІВЕРСИТЕТ»

Звіт
З лабораторної роботи №4
З дисципліни
«Об’єктно-орієнтоване програмування»
Варіант № 5

Виконав:
студент гр. :122-19-2
Вареник М. О

Перевірили:
доцент каф. ПЗКС
Приходченко С.Д

асистент каф. ПЗКС
Шевцова О.С

Дніпро

2020

Тема роботи: вивчення інтерфейсів, делегатів, та колекцій в мові C#.

Основне завдання:

1. Створити класову модель, яка включає в себе абстрактний клас CGraphicsObject, та два класи ,що його успадковують. Всі класи, за виключенням абстрактного, повинні містити конструктори, деструктори, відповідні set- и get-методы, а також метод *Show*, який може мати різну сигнатуру для різних класів и повинен виводити на екран в текстовому вигляді інформацию про об'єкт та його пращурів. Абстрактний клас повинен містити декларацію полей та методів.

2. Створити в класах-спадкоємцях по одному інтерфейсу та делегату.

3. Стовирити найпростійшу колекцію з екземплярів кожного з класіх-спадкоємців.

Код програми:

CGraphicsObject.cs

```
using System.Drawing;

namespace homework_6
{
    public abstract class CGraphicsObject
    {
        protected Point point;

        public virtual void Show()
        {

        }
    }
}
```

Circle.cs

```
using System;
using static System.Math;

namespace homework_6
{
    interface ICirclePerimeter
    {
        double Perimeter();
    }
    public class Circle : CGraphicsObject,ICirclePerimeter
    {
        private double radius;

        delegate double Operation(Circle operand1, Circle operand2);

        public Circle(double radius,int x,int y)
        {
            this.radius = radius;

            point.X = x;

            point.Y = y;
        }

        ~Circle()
        {
```

```
        }

        public double Radius // радиус окружности
        {
            get { return radius; }

            set { radius = value; }
        }

        public static void performOperationWithPerimeters(int operationType,Circle
operand1,Circle operand2)
        {
            Operation operation;

            if (operationType == 1)
            {
                operation = Sum;
                Console.WriteLine("Sum of perimeters:" + operation(operand1, operand2));
            }
            else
            {
                operation = Division;
                Console.WriteLine("Division of perimeters:" + operation(operand1,
operand2));
            }
        }

        private static double Sum(Circle operand1, Circle operand2)
        {
            return operand1.Perimeter() + operand2.Perimeter();
        }

        private static double Division(Circle operand1, Circle operand2)
        {
            return operand1.Perimeter() / operand2.Perimeter();
        }

        public double Perimeter()
        {
            return 2 * PI * Radius;
        }

        public override void Show()
        {
            Console.WriteLine("Perimeter=" + Perimeter() + ", R=" + Radius + "X=" +
point.X + ", Y=" + point.Y);
        }
    }
}
```

Ellipse.cs

```
using System;
using static System.Math;
namespace homework_6
{
    interface IEllipsePerimeter
    {
        double Perimeter();
    }

    public class Ellipse : CGraphicsObject,IEllipsePerimeter
    {
        private double a;//первая полуось эллипса
```

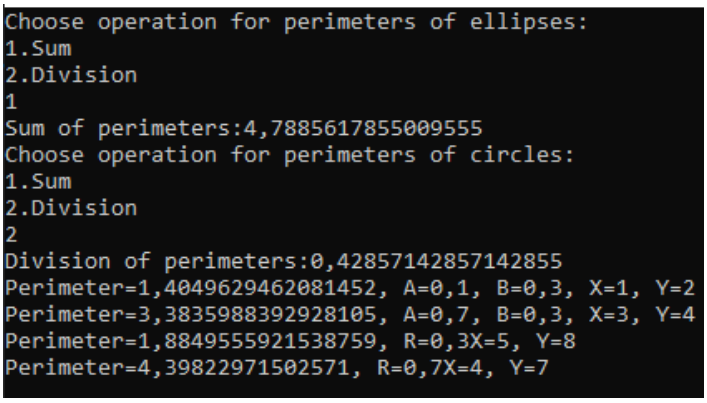
```
        private double b;//вторая полуось эллипса

        delegate double Operation(Ellipse operand1, Ellipse operand2);

        public Ellipse(double a,double b,int x,int y)
        {
            this.a = a;

            this.b = b;

            point.X = x;

            point.Y = y;
        }

        ~Ellipse()
        {

        }

        public double A
        {
            get { return a; }

            set { a = value; }
        }

        public double B
        {
            get { return b; }

            set { b = value; }
        }

        public double Perimeter()
        {
            return PI * Sqrt(2 * (Pow(A, 2) + Pow(B, 2)));
        }

        public static void performOperationWithPerimeters(int operationType, Ellipse
operand1, Ellipse operand2)
        {
            Operation operation;

            if(operationType == 1)
            {
                operation = Sum;

                Console.WriteLine("Sum of perimeters:" + operation(operand1, operand2));
            }
            else
            {
                operation = Division;
                Console.WriteLine("Division of perimeters:" + operation(operand1,
operand2));
            }
        }

        private static double Sum(Ellipse operand1, Ellipse operand2)
        {
            return operand1.Perimeter() + operand2.Perimeter();
        }

        private static double Division(Ellipse operand1, Ellipse operand2)
        {
```

```
            return operand1.Perimeter() / operand2.Perimeter();
        }

        public override void Show()
        {
            Console.WriteLine("Perimeter=" + Perimeter() + ", A=" + A + ", B=" + B + ",
X=" + point.X + ", Y=" + point.Y);
        }
    }
}
```

Program.cs

```
using System;
using System.Collections;

namespace homework_6
{
    class Program
    {
        static void Main(string[] args)
        {
            Ellipse ellipse1 = new Ellipse(0.1, 0.3, 1, 2);

            Ellipse ellipse2 = new Ellipse(0.7, 0.3, 3, 4);

            Console.WriteLine("Choose operation for perimeters of
ellipses:\n1.Sum\n2.Division");

            Ellipse.performOperationWithPerimeters(Convert.ToInt32(Console.ReadLine()),
ellipse1, ellipse2);

            Circle circle1 = new Circle(0.3, 5, 8);

            Circle circle2 = new Circle(0.7, 4, 7);

            Console.WriteLine("Choose operation for perimeters of
circles:\n1.Sum\n2.Division");

            Circle.performOperationWithPerimeters(Convert.ToInt32(Console.ReadLine()),
circle1, circle2);

            ArrayList list = new ArrayList() { ellipse1,ellipse2,circle1,circle2};

            foreach(CGraphicsObject graphicsObject in list)
            {
                graphicsObject.Show();
            }
        }
    }
}
```

Результат:



Висновки: засвоєно делегати та прості колекції мови C#. Створено класову модель із використанням інтерфейсів та делегатів.