Московский государственный технический

Университет им Н.Э.Баумана

Факультет «Информатика и системы управление»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №3

Выполнил:

Студент: Марянян А.А.

Группа: ИУ5-34Б

Подпись и дата:

Проверил:

Преподаватель каф. ИУ5

Нардид А.Н.

Подпись и дата:

Постановка задачи

Лабораторная работа №3

Разработать программу, реализующую работу с коллекциями.

1. Программа должна быть разработана в виде консольного приложения на языке C#.
2. Создать объекты классов «Прямоугольник», «Квадрат», «Круг».
3. Для реализации возможности сортировки геометрических фигур для класса «Геометрическая фигура» добавить реализацию интерфейса IComparable. Сортировка производится по площади фигуры.
4. Создать коллекцию класса ArrayList. Сохранить объекты в коллекцию. Отсортировать коллекцию. Вывести в цикле содержимое коллекции.
5. Создать коллекцию класса List<Figure>. Сохранить объекты в коллекцию. Отсортировать коллекцию. Вывести в цикле содержимое коллекции.
6. Модифицировать класс разреженной матрицы (проект SparseMatrix) для работы с тремя измерениями – x,y,z. Вывод элементов в методе ToString() осуществлять в том виде, который Вы считаете наиболее удобным. Разработать пример использования разреженной матрицы для геометрических фигур.
7. Реализовать класс «SimpleStack» на основе односвязного списка. Класс SimpleStack наследуется от класса SimpleList (проект SimpleListProject). Необходимо добавить в класс методы:
   - public void Push(T element) – добавление в стек;
   - public T Pop() – чтение с удалением из стека.
8. Пример работы класса SimpleStack реализовать на основе геометрических фигур.

Код

```
using System.Collections;

namespace Lab3.Additional
{
    public class SimpleStackItems<T>
    {
        public T? Element { get; set; }
        public SimpleStackItems<T>? Next { get; set; }
    }

    public class SimpleStack<T> : IEnumerable<T>
    {
        private SimpleStackItems<T>? _head;

        public bool IsEmpty => _head == null;
```

```
        public void Add(T value)
        {
            var newItem = new SimpleStackItems<T> { Element = value, Next =
_head };
            _head = newItem;
        }

        public T Pop()
        {
            if (IsEmpty) throw new InvalidOperationException("Сьек пуст");

            var result = _head!.Element;
            _head = _head.Next;

            return result ?? throw new InvalidOperationException();
        }

        public override string ToString()
        {
            string s = "";
            foreach (var item in this)
            {
                s += item + " ";
            }
            return s;
        }

        public IEnumerator<T> GetEnumerator()
        {
            var current = _head;
            while (current != null)
            {
                if (current.Element != null) yield return current.Element;
                current = current.Next;
            }
        }

        IEnumerator IEnumerable.GetEnumerator()
        {
            return GetEnumerator();
        }
    }
}


using Lab3.Interfaces;

namespace Lab3.Additional;

public class SparseMatrix<T> : IPrint
{
    public readonly Dictionary<(int, int, int), T> Matrix
        = new Dictionary<(int, int, int), T>();

    public void Set(int x, int y, int z, T value)
    {
        Matrix[(x, y, z)] = value;
    }

    public T Get(int x, int y, int z)
    {
        return Matrix[(x, y, z)];
    }
```

```
    public override string ToString()
    {
        string result = "";

        foreach (var item in Matrix)
        {
            result += $"Точка ({item.Key.Item1}, {item.Key.Item2},
{item.Key.Item3}) -> {item.Value?.ToString()}\n";
        }

        return result;
    }


    public void Print() => Console.WriteLine(ToString());
}



using Lab3.Interfaces;

namespace Lab3.Figures;

public class Circle : GeometricFigure, IPrint, IComparable
{
    public double Radius { get; set; }

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

    public override double? Result() => Math.PI*Math.Pow(Radius, 2);

    public override string ToString() => $"Круг радиусом: {Radius}\n" +
                                         $"Круг площадью: {Result()}\n";

    public override void Print() => Console.WriteLine(this.ToString());

    public override int CompareTo(object? obj)
    {
        if (obj is GeometricFigure figure)
        {
            return
Convert.ToDouble(Result()).CompareTo(Convert.ToDouble(figure.Result()));
        }
        throw new ArgumentException("Объект не является фигурой");
    }

}

using Lab3.Interfaces;

namespace Lab3.Figures;

public abstract class GeometricFigure : IPrint, IComparable
{
    public abstract double? Result();

    public abstract void Print();


    public abstract int CompareTo(object? obj);
}
```

```
using Lab3.Interfaces;

namespace Lab3.Figures;

public class Rectangle : GeometricFigure, IPrint, IComparable
{
    public double Width { get; set; }
    public double Height { get; set; }

    public Rectangle(double width, double height)
    {
        Width = width;
        Height = height;
    }

    public override double? Result() => Width * Height;

    public override string ToString() => $"Прямоугольник шириной: {Width}\n"
+
                                         $"Прямоугольник высотой: {Height}\n"
+
                                         $"Прямоугольник площадью:
{Result()}\n";

    public override void Print() => Console.WriteLine(ToString());


    public override int CompareTo(object? obj)
    {
        if (obj is GeometricFigure figure)
            return
Convert.ToDouble(Result()).CompareTo(Convert.ToDouble(figure.Result()));


        throw new ArgumentException("Объект не является фигурой");
    }

}


using Lab3.Interfaces;

namespace Lab3.Figures;

public class Square : GeometricFigure, IPrint, IComparable
{
    public double Side { get; set; }

    public Square(double side)
    {
        Side = side;
    }

    public override double? Result() => Side*Side;

    public override string ToString() => $"Квадрат стороной: {Side}\n" +
                                         $"Квадрат площадью: {Result()}\n";

    public override void Print() => Console.WriteLine(this.ToString());

    public override int CompareTo(object? obj)
```

```
    {
        if (obj is GeometricFigure figure)
        {
            return
Convert.ToDouble(Result()).CompareTo(Convert.ToDouble(figure.Result()));
        }
        throw new ArgumentException("Объект не является фигурой");
    }

}



namespace Lab3.Interfaces;

internal interface IPrint
{
    public void Print(){}
}


using System.Collections;
using Lab3.Additional;
using Lab3.Figures;

namespace Lab3;

class Program
{
    public static void Main()
    {
        ArrayList arrayList = new ArrayList();
        arrayList.Add(new Rectangle(12, 2));
        arrayList.Add(new Square(7));
        arrayList.Add(new Circle(3));

        foreach (var el in arrayList)
        {
            Console.WriteLine(el);
        }

        arrayList.Sort();
        foreach (var el in arrayList)
        {
            Console.WriteLine(el);
        }

        var figures = new List<GeometricFigure>
        {
            new Rectangle(12, 2),
            new Square(7),
            new Circle(3)
        };

        figures.Sort();
        foreach (var figure in figures)
        {
            figure.Print();
        }

        var matrix = new SparseMatrix<GeometricFigure>();

        matrix.Set(0,0,0, new Circle(8));
        matrix.Set(1,2,3, new Rectangle(4,2));
```

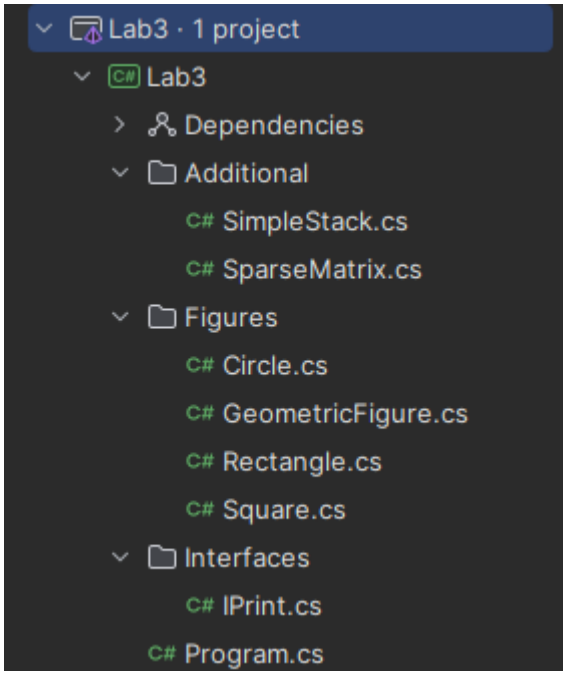
```
        matrix.Set(2,4,5, new Square(4));

        matrix.Get(0,0,0).Print();

        matrix.Print();

        var simpleStack = new SimpleStack<GeometricFigure>();
        simpleStack.Add(new Circle(2));
        simpleStack.Add(new Rectangle(12,3));
        simpleStack.Add(new Square(7));
        simpleStack.Pop();
        foreach (var eFigure in simpleStack)
        {
            Console.WriteLine(eFigure);
        }
    }
}
```

Построение проекта



Вывод

```
Прямоугольник шириной: 12
Прямоугольник высотой: 2
Прямоугольник площадью: 24

Квадрат стороной: 7
Квадрат площадью: 49

Круг радиусом: 3
Круг площадью: 28,274333882308138

Прямоугольник шириной: 12
Прямоугольник высотой: 2
Прямоугольник площадью: 24

Круг радиусом: 3
Круг площадью: 28,274333882308138

Квадрат стороной: 7
Квадрат площадью: 49

Прямоугольник шириной: 12
Прямоугольник высотой: 2
Прямоугольник площадью: 24
Круг радиусом: 3
Круг площадью: 28,274333882308138

Квадрат стороной: 7
Квадрат площадью: 49

Круг радиусом: 8
Круг площадью: 201,06192982974676

Точка (0, 0, 0) -> Круг радиусом: 8
Круг площадью: 201,06192982974676
```

```
Квадрат стороной: 7
Квадрат площадью: 49

Прямоугольник шириной: 12
Прямоугольник высотой: 2
Прямоугольник площадью: 24
Круг радиусом: 3
Круг площадью: 28,274333882308138

Квадрат стороной: 7
Квадрат площадью: 49

Круг радиусом: 8
Круг площадью: 201,06192982974676

Точка (0, 0, 0) -> Круг радиусом: 8
Круг площадью: 201,06192982974676

Точка (1, 2, 3) -> Прямоугольник шириной: 4
Прямоугольник высотой: 2
Прямоугольник площадью: 8

Точка (2, 4, 5) -> Квадрат стороной: 4
Квадрат площадью: 16


Прямоугольник шириной: 12
Прямоугольник высотой: 3
Прямоугольник площадью: 36

Круг радиусом: 2
Круг площадью: 12,566370614359172
```