Московский государственный технический университет им. Н.Э. Баумана

Факультет «Радиотехнический»
Кафедра «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №3 «Реализация работы с классами на языке C#»

Выполнил:
студент группы РТ5-31Б:
Паншин М.В.

Проверил:
преподаватель кафедры ИУ5
Гапанюк Ю.Е.

Москва, 2024 г.

Текст задания

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. 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.

Текст программы

```
using System;
using System.Numerics;
using System.Collections.Generic;
using System.Collections;



abstract class Figure : IComparable<Figure>,IPrint
{
    public abstract double Area();

    public int CompareTo(Figure other)
    {
        if (other == null)
        {
            return 1;
        }
        return this.Area().CompareTo(other.Area());
    }
    public abstract void Print();
}


class Check
{
   static public double InputCheck(string input)
    {
        double result;
        while (true)
        {
            if (double.TryParse(input, out result))
            {
                return result;
            }
```

```
            else
            {
                Console.WriteLine("Вы ввели неверное значение");
            }
        }
    }
}

class Rectangle : Figure
{

    public override double Area()
    {
        return height * width;
    }
    public Rectangle()
    {
        this.height = 0;
        this.width = 0;
    }
    public Rectangle(double height, double width)
    {
        this.height = height;
        this.width = width;
    }

    public double height { get; set; }
    public double width { get; set; }

    public override string ToString()
    {
        return "Высота:" + this.height.ToString() + " Ширина: " +
this.width.ToString() + " Площадь: " + this.Area().ToString();
    }
    override public void Print()
    {
        Console.WriteLine("Прямоугольник\n"+this.ToString());
    }

}
class Square : Rectangle
{

    public Square(double side): base(side,side)
    {

    }
    public Square() { }

    public override string ToString()
    {
        return "Сторона: " + this.height.ToString() + " Площадь: " +
this.Area().ToString();
    }
    override public void Print()
    {
        Console.WriteLine("Квадрат\n" + this.ToString());
    }
}
class Circle : Figure
{


    public Circle() { }
    public Circle(double radius)
    {
        this.radius = radius;
```

```
    }
    public double radius { get; set; }

    public override double Area() {
        return Math.PI * radius * radius;
    }
    public override string ToString()
    {
        return "Радиус: " + this.radius.ToString() + " Площадь: " +
this.Area().ToString();
    }
    public override void Print()
    {
        Console.WriteLine("Круг\n" + this.ToString());
    }

}

interface IPrint
{
    public void Print();
}

class Program
{

    static int Main()
    {

        Figure r1 = new Rectangle(5, 10);
        Figure s1 = new Square(2);
        Figure s2 = new Square(9);
        Figure c1 = new Circle(3);
        Figure r2 = new Rectangle(7, 2);

        List<Figure> figures = new List<Figure>(){ r1, s1, r2, s2, c1 };
        ArrayList list = new ArrayList() {r1,s1,r2,s2,c1};


        list.Sort(new FigureComparer());

        figures.Sort();
        foreach(Figure item in list)
        {
            item.Print();
        }

        Cube cube = new Cube(6);
        cube.PrintMatrix();

        SparseMatrix sparseMatrix = new SparseMatrix();
        sparseMatrix.SetValue(2, 4, 1, 1);
        sparseMatrix.SetValue(2, 1, 1, 1);

        Console.WriteLine("Разреженная матрица:");
        Console.WriteLine(sparseMatrix.ToString());

        SimpleStack<Figure> stack = new SimpleStack<Figure>();
        stack.Push(r1);
        stack.Push(s1);
        stack.Push(c1);
        stack.Pop();
        Console.ReadLine();
        return 0;

    }
```

```
}

public class FigureComparer : IComparer
{
    public int Compare(object x, object y)
    {
        Figure figure1 = x as Figure;
        Figure figure2 = y as Figure;

        if (figure1 == null || figure2 == null)
            throw new ArgumentException();

        return figure1.Area().CompareTo(figure2.Area());
    }
}

class SparseMatrix
{
    private Dictionary<(int x, int y, int z), double> matrix;

    public SparseMatrix()
    {
        matrix = new Dictionary<(int, int, int), double>();
    }

    public void SetValue(int x, int y, int z, double value)
    {
        if (value != 0)
        {
            matrix[(x, y, z)] = value;
        }
        else
        {
            matrix.Remove((x, y, z));
        }
    }

    public double GetValue(int x, int y, int z)
    {
        matrix.TryGetValue((x, y, z), out double value);
        return value;
    }
    public override string ToString()
    {
        if (matrix.Count == 0)
            return "Разреженная матрица пуста.";

        var result = "\n";
        foreach (var item in matrix)
        {
            result += $"Координаты: ({item.Key.x}, {item.Key.y}, {item.Key.z}),
Значение: {item.Value}\n";
        }
        return result;
    }
}

public class Cube
{
    public int Size { get; set; }
    private SparseMatrix matrix;

    public Cube(int size)
    {
        Size = size;
        matrix = new SparseMatrix();
```

```
            InitMatrix();
        }


        private void InitMatrix()
        {
            matrix.SetValue(0, 0, 0, 1.0);
            matrix.SetValue(Size, 0, 0, 1.0);
            matrix.SetValue(Size, Size, 0, 1.0);
            matrix.SetValue(0, Size, 0, 1.0);
            matrix.SetValue(0, 0, Size, 1.0);
            matrix.SetValue(Size, 0, Size, 1.0);
            matrix.SetValue(Size, Size, Size, 1.0);
            matrix.SetValue(0, Size, Size, 1.0);
        }


        public void PrintMatrix()
        {
            Console.WriteLine("\nВершины куба:");
            Console.WriteLine(matrix.ToString());
        }
    }

    class SimpleStack<T>
    {
        private LinkedList<T> list = new LinkedList<T>();

        public T Pop()
        {
            if (list.Count == 0)
            {
                throw new InvalidOperationException("пустой стек"); //проверка на
    пустой стек
            }

            T element = list.First.Value;
            list.RemoveFirst();
            return element;
        }

        public void Push(T element)
        {
            list.AddFirst(element);
        }
      };
```

Результат выполнения программы

```
Квадрат
Сторона: 2 Площадь: 4
Прямоугольник
Высота:7 Ширина: 2 Площадь: 14
Круг
Радиус: 3 Площадь: 28,274333882308138
Прямоугольник
Высота:5 Ширина: 10 Площадь: 50
Квадрат
Сторона: 9 Площадь: 81

Вершины куба:

Координаты: (0, 0, 0), Значение: 1
Координаты: (6, 0, 0), Значение: 1
Координаты: (6, 6, 0), Значение: 1
Координаты: (0, 6, 0), Значение: 1
Координаты: (0, 0, 6), Значение: 1
Координаты: (6, 0, 6), Значение: 1
Координаты: (6, 6, 6), Значение: 1
Координаты: (0, 6, 6), Значение: 1

Разреженная матрица:

Координаты: (2, 4, 1), Значение: 1
Координаты: (2, 1, 1), Значение: 1
```