Московский государственный технический университет им. Н.Э. Баумана

Факультет «Радиотехнический»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет технологий»
Отчет по лабораторной работе №3

Выполнил:
студент группы РТ5-31Б
Сысоев Александр

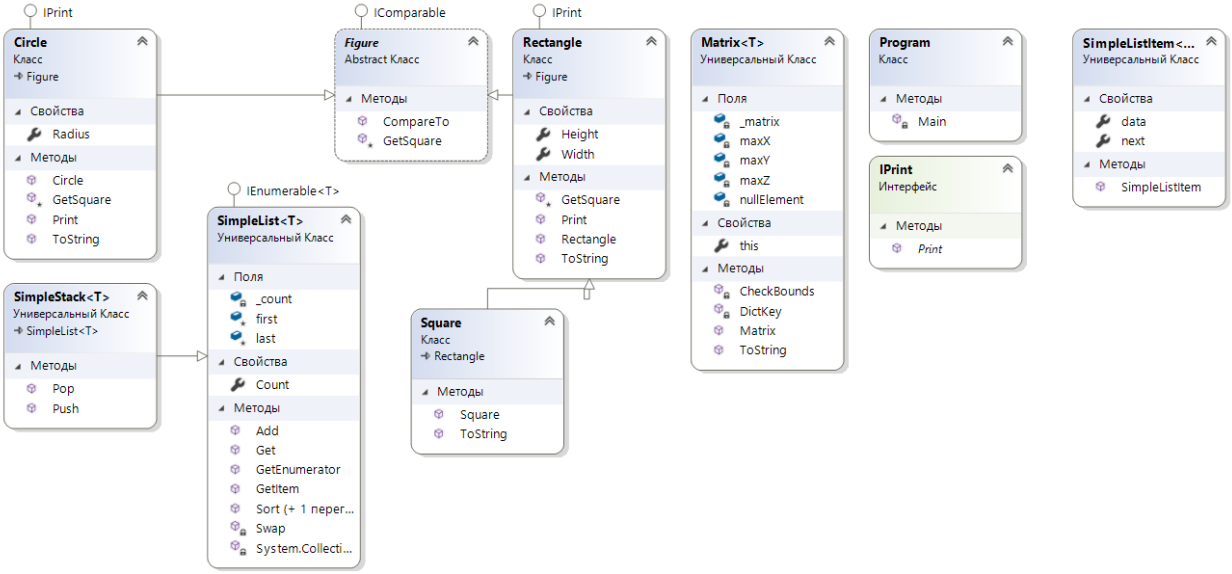
Проверил:
доцент каф. ИУ5
Гапанюк Ю.Е.

Описание 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<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.

Диаграмма классов



Текст программы

Файл SimpleStack.cs

```
using System;

namespace Lab3
{
    class SimpleStack<T> : SimpleList<T>
        where T : IComparable
    {
        public void Push(T element)
        {
            SimpleListItem<T> item = new SimpleListItem<T>(element);
            if (last == null)
                item.next = null;
            else
                item.next = last;
            last = item;
            Count++;
        }

        public T Pop()
        {
            T temp = last.data;
            last = last.next;
            return temp;
        }
    }
}
```

Файл SimpleList.cs

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

namespace Lab3
{
    public class SimpleListItem<T>
    {
        public T data { get; set; }
        public SimpleListItem<T> next { get; set; }
        public SimpleListItem(T param)
        {
            data = param;
        }
    }

    public class SimpleList<T> : IEnumerable<T>
        where T : IComparable
    {
        protected SimpleListItem<T> first = null;
        protected SimpleListItem<T> last = null;

        public int Count
        {
            get { return _count; }
            protected set { _count = value; }
        }

        int _count;

        public void Add(T element)
        {
            SimpleListItem<T> newItem = new SimpleListItem<T>(element);
            Count++;
            if (last == null)
            {
                this.first = newItem;
                this.last = newItem;
            }
            else
            {
                last.next = newItem;
                last = newItem;
            }
        }

        public SimpleListItem<T> GetItem(int number)
        {
            if ((number < 0) || (number >= this.Count))
            {
                throw new Exception("Выход за границу индекса");
            }
            SimpleListItem<T> current = this.first;
            int i = 0;
            while (i < number)
            {
                current = current.next;
                i++;
            }
            return current;
        }

        public T Get(int number)
        {
```

```
            return GetItem(number).data;
        }

        public IEnumerator<T> GetEnumerator()
        {
            SimpleListItem<T> current = this.first;
            while (current != null)
            {
                yield return current.data;
                current = current.next;
            }
        }

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

        public void Sort()
        {
            Sort(0, this.Count - 1);
        }

        private void Sort(int low, int high)
        {
            int i = low;
            int j = high;
            T x = Get((low + high) / 2);
            do
            {
                while (Get(i).CompareTo(x) < 0) ++i;
                while (Get(j).CompareTo(x) > 0) --j;
                if (i <= j)
                {
                    Swap(i, j);
                    i++; j--;
                }
            } while (i <= j);

            if (low < j) Sort(low, j);
            if (i < high) Sort(i, high);
        }

        private void Swap(int i, int j)
        {
            SimpleListItem<T> ci = GetItem(i);
            SimpleListItem<T> cj = GetItem(j);
            T temp = ci.data;
            ci.data = cj.data;
            cj.data = temp;
        }
    }
}
```

Файл Program.cs

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

namespace Lab3
{
    abstract class Figure : IComparable
    {
        protected virtual double GetSquare()
        {
            return 0.0;
        }

        public int CompareTo(object o)
        {
            Figure figure = o as Figure;
            return GetSquare().CompareTo(figure.GetSquare());
        }
    }

    interface IPrint
    {
        void Print();
    }

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

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

        protected override double GetSquare()
        {
            return Width * Height;
        }

        public override string ToString()
        {
            return "Ширина = " + Width + ", Высота = " + Height + ", Площадь = " +
GetSquare();
        }

        public void Print()
        {
            Console.WriteLine(ToString());
        }
    }

    class Square : Rectangle
    {
        public Square(double length) : base(length, length) { }

        public override string ToString()
        {
            return "Длина = " + Width + ", Площадь = " + GetSquare();
        }
    }
```

```
    class Circle : Figure, IPrint
    {
        public double Radius { get; set; }

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

        protected override double GetSquare()
        {
            return Math.PI * Radius * Radius;
        }

        public override string ToString()
        {
            return "Радиус = " + Radius + ", Площадь = " + GetSquare();
        }

        public void Print()
        {
            Console.WriteLine(ToString());
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            Rectangle rectangle = new Rectangle(3, 5);
            Square square1 = new Square(5), square2 = new Square(3), square3 = new
Square(4);
            Circle circle = new Circle(5);
            ArrayList figures1 = new ArrayList() { circle, square1, rectangle };
            figures1.Sort();
            foreach (Figure figure in figures1)
                Console.WriteLine(figure.ToString());
            Console.WriteLine();
            List<Figure> figures2 = new List<Figure>() { circle, square2, rectangle };
            figures2.Sort();
            foreach (Figure figure in figures2)
                Console.WriteLine(figure.ToString());
            Console.WriteLine();
            Matrix<Square> matrix = new Matrix<Square>(3, 3, 3, new Square(0));
            matrix[0, 0, 0] = square1;
            matrix[1, 1, 1] = square2;
            matrix[2, 2, 2] = square3;
            Console.WriteLine(matrix);
            Console.WriteLine();
            SimpleStack<Figure> figures3 = new SimpleStack<Figure>();
            figures3.Push(square1);
            figures3.Push(rectangle);
            figures3.Push(circle);
            for (int i = 0; i < figures3.Count; i++)
                Console.WriteLine(figures3.Pop().ToString());
            Console.ReadKey();
        }
    }
}
```

Файл Matrix.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Lab3
{
    public class Matrix<T>
    {
        Dictionary<string, T> _matrix = new Dictionary<string, T>();
        int maxX;
        int maxY;
        int maxZ;

        T nullElement;

        public Matrix(int px, int py, int pz, T nullElementParam)
        {
            maxX = px;
            maxY = py;
            maxZ = pz;
            nullElement = nullElementParam;
        }

        public T this[int x, int y, int z]
        {
            get
            {
                CheckBounds(x, y, z);
                string key = DictKey(x, y, z);
                if (_matrix.ContainsKey(key))
                {
                    return _matrix[key];
                }
                else
                {
                    return this.nullElement;
                }
            }
            set
            {
                CheckBounds(x, y, z);
                string key = DictKey(x, y, z);
                _matrix.Add(key, value);
            }
        }

        void CheckBounds(int x, int y, int z)
        {
            if (x < 0 || x >= this.maxX) throw new Exception("x=" + x + " выходит за
границы");
            if (y < 0 || y >= this.maxY) throw new Exception("y=" + y + " выходит за
границы");
            if (z < 0 || z >= this.maxZ) throw new Exception("z=" + z + " выходит за
границы");
        }

        string DictKey(int x, int y, int z)
        {
            return x.ToString() + "_" + y.ToString() + "_" + z.ToString();
        }

        public override string ToString()
        {
```

```
            StringBuilder b = new StringBuilder();
            for (int k = 0; k < maxZ; k++)
            {
                for (int j = 0; j < this.maxY; j++)
                {
                    b.Append("[");
                    for (int i = 0; i < this.maxX; i++)
                    {
                        if (i > 0) b.Append(" | ");
                        b.Append(this[i, j, k].ToString());
                    }
                    b.Append("]\n");
                }
                b.Append("\n");
            }
            return b.ToString();
        }

    }
}
```

Результат работы программы