Защищено:
Гапанюк Ю.Е.

"__"_____2017 г.

Демонстрация:
Гапанюк Ю.Е.

"__"_____2017 г.

Отчет по лабораторной работе № 3 по курсу Базовые компоненты интернет-технологий

12
(количество листов)

Студент группы ИУ5-33
Коционова Анна

Москва МГТУ 2017

СОДЕРЖАНИЕ

1. Описание задания лабораторной работы

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 (разобранного в пособии). 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.

2. Текст программы

Main

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Collections;

namespace Lab3_BKIT
{
    class Program
    {
        static void Main(string[] args)
        {
            Rectangle rect = new Rectangle(5, 4);
            Square square = new Square(5);
            Circle circle = new Circle(5);

            Console.WriteLine("\nArrayList");//4. коллекция класса ArrayList
            ArrayList al = new ArrayList();
            al.Add(circle);//добавление элов в необобщенный список
            al.Add(rect);
            al.Add(square);

            foreach (var x in al)//повторяет группу операторов для каждого эла в списке
                Console.WriteLine(x);// вывод элементов списка

            Console.WriteLine("\nArrayList - сортировка");
            al.Sort();
            foreach (var x in al) Console.WriteLine(x);

            Console.WriteLine("\nList<Figure>");//5. Коллекция класса List<Figure>
            List<Figure> fl = new List<Figure>();//из класса Figure тип
            fl.Add(circle);
            fl.Add(rect);
            fl.Add(square);

            foreach (var x in fl) Console.WriteLine(x);

            Console.WriteLine("\nList<Figure> - сортировка");
            fl.Sort();
            foreach (var x in fl) Console.WriteLine(x);

            Console.WriteLine("\nМатрица");
            Matrix3D<Figure> cube = new Matrix3D<Figure>(3, 3, 3, null);//размерность и
заполнение
            cube[0, 0, 0] = rect;
            cube[1, 1, 1] = square;
            cube[2, 2, 2] = circle;
            Console.WriteLine(cube.ToString());

            Console.WriteLine("\nСписок");
            SimpleList<Figure> list = new SimpleList<Figure>();
            list.Add(square);
            list.Add(rect);
            list.Add(circle);

            foreach (var x in list) Console.WriteLine(x);
```

```
            list.Sort();
            Console.WriteLine("\nСортировка списка");
            foreach (var x in list) Console.WriteLine(x);

            Console.WriteLine("\nСтек");
            SimpleStack<Figure> stack = new SimpleStack<Figure>();
            stack.Push(rect);
            stack.Push(square);
            stack.Push(circle);

            while (stack.Count > 0)
            {
                Figure f = stack.Pop();
                Console.WriteLine(f);
            }
            Console.ReadLine();
        }
    }
}
```

Абстрактный класс «Геометрическая фигура»

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab3_BKIT
{
   abstract class Figure : IComparable
    {
        public string Type
        {
            get
            {
                return this._Type;
            }
            protected set
            {
                this._Type = value;
            }
        }
        string _Type;

        public abstract double AreaCalc();

        public override string ToString()
        {
            return this.Type + " площадью " + this.AreaCalc().ToString();
        }

        public int CompareTo(object obj)
        {
            Figure p = (Figure)obj;

            if (this.AreaCalc() < p.AreaCalc()) return -1;
            else if (this.AreaCalc() == p.AreaCalc()) return 0;
            else return 1;
        }
    }
```

Класс «Прямоугольник»

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab3_BKIT
{
    class Rectangle : Figure, IPrint, IComparable
    {
        double height, width;

        public Rectangle(double high, double wide)//конструктор
        {
            this.height = high;
            this.width = wide;
            this.Type = "Прямоугольник";
        }

        public override double AreaCalc()// override отвечает за новую реализацию
базового метода
        {
            double Area = (this.width * this.height);
            return Area;
        }

        public void Print()
        {
            Console.WriteLine(this.ToString());//возвращает в виде строки основные
параметры фигуры и ее площадь
        }
    }
}
```

Класс «Квадрат»

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp2
{
    class Square:Rectangle
    {
        public Square(double size):base(size, size)
        {
            this.Type = "Квадрат";
        }
    }
}
```

Класс «Круг»

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

```
using System.Threading.Tasks;

namespace Lab3_BKIT
{
    class Circle : Figure, IPrint, IComparable
    {
        private double radius;

        public Circle(double rad)
        {
            this.radius = rad;
            this.Type = "Круг";
        }

        public override double AreaCalc()
        {
            double Area = (Math.PI) * (this.radius) * (this.radius);
            return Area;
        }

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

Интерфейс IPrint

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp2
{
    interface IPrint
    {
        void Print();
    }
}
```

Класс Matrix3D (класс разреженной матрицы для работы с тремя измерениями)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab3_BKIT
{
    public class Matrix3D<T>
    {
        Dictionary<string, T> _matrix = new Dictionary<string, T>();
```

```
        int maxX, maxY, maxZ; //кол-во элементов по гор и верт(макс кол-во столбцов и
строк)
        T nullElement;//0 эл возвр, если нет эла с координами

        public Matrix3D(int px, int py, int pz, T nullElementParam)//конструктор
        {
            this.maxX = px;
            this.maxY = py;
            this.maxZ = pz;
            this.nullElement = nullElementParam;
        }
        public T this[int x, int y, int z]//индексатор для доступа к данным
        {
            get
            {
                CheckBounds(x, y, z);
                string key = DictKey(x, y, z);
                if (this._matrix.ContainsKey(key))
                {
                    return this._matrix[key];
                }
                else
                {
                    return this.nullElement;
                }
            }
            set
            {
                CheckBounds(x, y, z);
                string key = DictKey(x, y, z);
                this._matrix.Add(key, value);
            }
        }

        void CheckBounds(int x, int y, int z)//проверка границ
        {
            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)//формирование ключа
        {
            return x.ToString() + "_" + y.ToString() + "_" + z.ToString();
        }

        public override string ToString() //приведение к строке
        {
            StringBuilder b = new StringBuilder();//b-экземпляр класса длинных строк

            for (int k = 0; k < this.maxZ; k++)
            {
                b.Append("{");
                for (int j = 0; j < this.maxY; j++)//this??
                {
                    if (j > 0) b.Append("\t");
                    b.Append("[");
                    for (int i = 0; i < this.maxX; i++)//this??
                    {
                        if (this[i, j, k] != null)
                            //this[i, j, k] = new ;
```

```
                            b.Append(this[i, j, k].ToString());//что-то null->
проинициализировать b
                        else
                            b.Append("0");
                        if (i != (maxX - 1)) b.Append(",");
                    }
                    b.Append("]");
                }
                b.Append("}\n");
            }
            return b.ToString();
        }
    }
}
```

Класс SimpleList

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab3_BKIT
{
    public class SimpleListItem<T>//элемент списка
    {
        public T data { get; set; }//Данные
        public SimpleListItem<T> next { get; set; }//Следующий элемент
        public SimpleListItem(T param)//конструктор
        {
            this.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);
            this.Count++;

            if (last == null)
            {
                this.first = newItem;
                this.last = newItem;
            }

            else
            {
```

```
            this.last.next = newItem;
            this.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;
        }
    }
}
```

Класс SimpleStack (наследуемый от класса SimpleList)

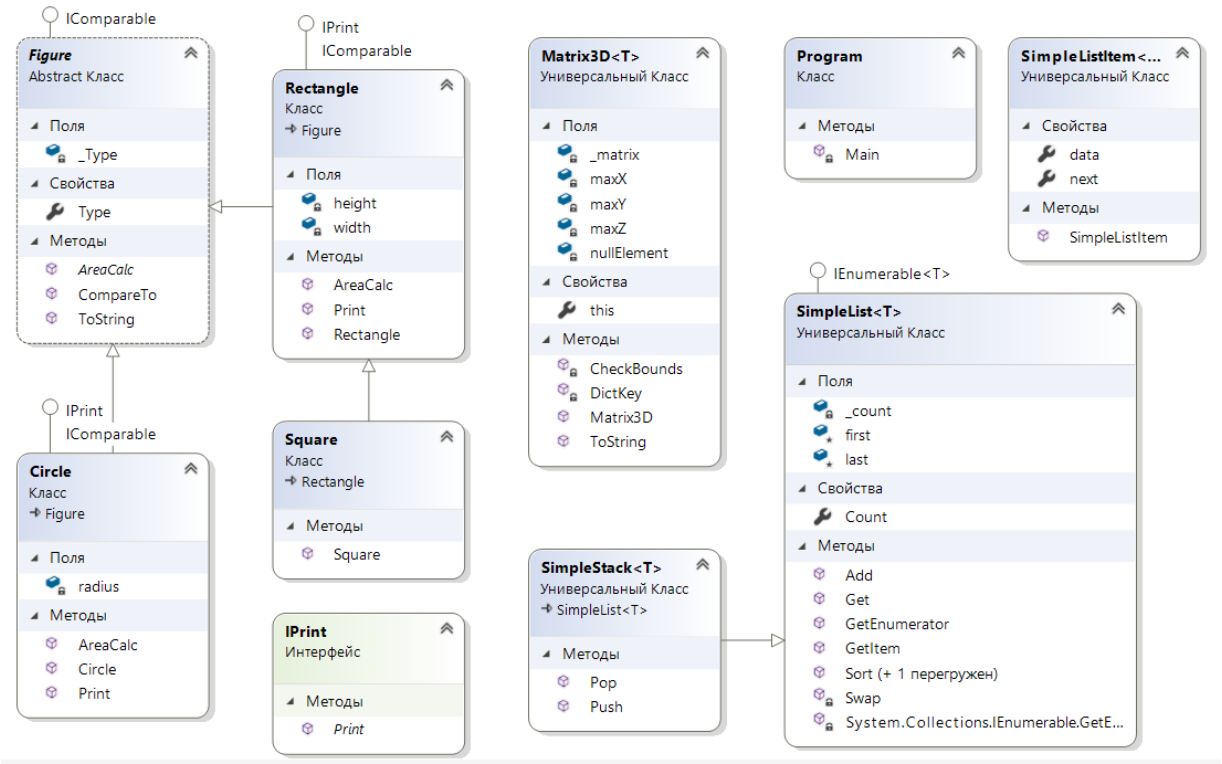
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab3_BKIT
{
    class SimpleStack<T>:SimpleList<T>
        where T:IComparable
    {
        public void Push(T element)
        {
            Add(element);
        }

        public T Pop()
        {
            T element = Get(Count - 1);

            SimpleListItem<T> listItem = GetItem(Count - 1);
            listItem = null;
            Count--;
            return element;
        }
    }
}
```

3. Диаграмма классов



4. Результат выполнения программы

```
C:\Users\kotsi\OneDrive\Документы\Visual Studio 2017\Projects\Lab3_BKIT\Lab3_BKIT\bin\Debug\Lab3_BKIT.exe

ArrayList
Круг площадью 78,5398163397448
Прямоугольник площадью 20
Квадрат площадью 25

ArrayList - сортировка
Прямоугольник площадью 20
Квадрат площадью 25
Круг площадью 78,5398163397448

List<Figure>
Круг площадью 78,5398163397448
Прямоугольник площадью 20
Квадрат площадью 25

List<Figure> - сортировка
Прямоугольник площадью 20
Квадрат площадью 25
Круг площадью 78,5398163397448

Матрица
{[Прямоугольник площадью 20,0,0][0,0,0][0,0,0]}
{[0,0,0][0,Квадрат площадью 25,0][0,0,0]}
{[0,0,0][0,0,0][0,0,Круг площадью 78,5398163397448]}


Список
Квадрат площадью 25
Прямоугольник площадью 20
Круг площадью 78,5398163397448

Сортировка списка
Прямоугольник площадью 20
Квадрат площадью 25
Круг площадью 78,5398163397448

Стек
Круг площадью 78,5398163397448
Квадрат площадью 25
Прямоугольник площадью 20
```