МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ Н.Э. БАУМАНА

Факультет Информатика и системы управления

Кафедра Системы обработки информации и управления (ИУ5)

Базовые компоненты интернет-технологий

Отчет по лабораторной работе №3

Выполнила: Бессонова Ксения Сергеевна

Группа: ИУ5-31Б

Преподаватель: Гапанюк Юрий Евгеньевич

Дата: 18.12.20

Подпись:

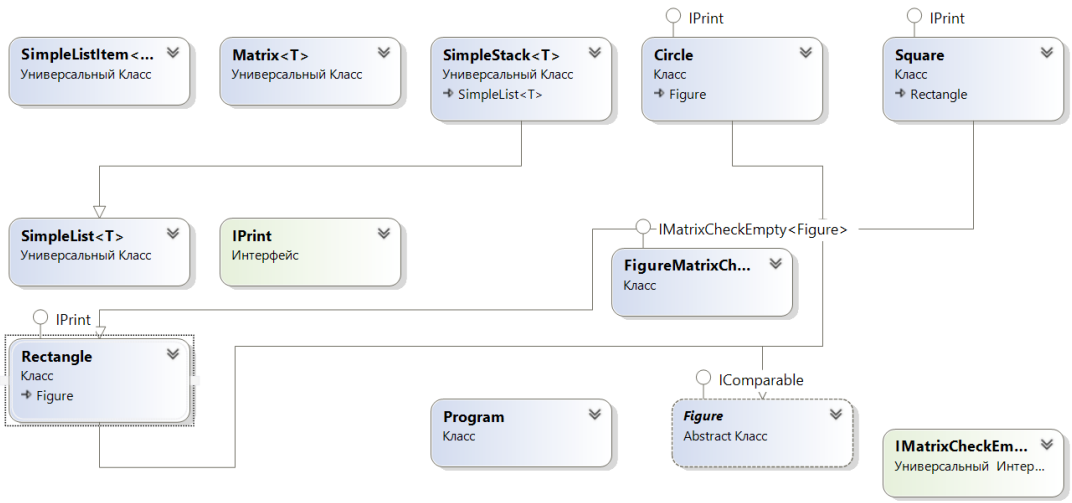
Москва, 2020 г.

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:

Circle.cs

```
using System;


namespace Lab3
{
    class Circle : Figure, IPrint
    {
        public Circle(double radius = 0)
        {
            Radius = radius;
        }
        public double Radius { get; set; }

        public override string FigureName => "Круг";

        public override double Area() => Math.PI * Radius * Radius;

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

        public override string ToString()
        {
            return $"{this.FigureName} с площадью {this.Area()} и радиусом {Radius}";
        }

    }
}
```

Figure.cs

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

namespace Lab3
{
    abstract class Figure : IComparable
    {
        public string Type { get; protected set; }
        public abstract string FigureName { get; }
        public abstract double Area();
        public int CompareTo(object obj)
        {
            Figure f = (Figure)obj;
            if (this.Area() < f.Area()) return -1;
            else if (this.Area() == f.Area()) return 0;
            else return 1;
        }

    }
}
```

IPrint.cs

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

namespace ConsoleApp3
{
    interface IPrint
    {
        void Print(); // выводит в консоль то, что переопределяется методом ToString()
```

```
    }
}
```

Rectangle.cs

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

namespace ConsoleApp3
{
    class Rectangle : Figure, IPrint
    {
        public Rectangle(double height = 0, double width = 0)
        {
            H = height;
            W = width;
        }
        public double H { get; set; }
        public double W { get; set; }

        public override string FigureName => "Прямоугольник";//виртуальный метод

        public override double Area()
        {
            return W * H;
        }

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

        public override string ToString()
        {
            return $"{this.FigureName} с площадью {this.Area()} и высотой {this.H}, а
шириной {this.W}";
        }
    }
}
```

Square.cs

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

namespace ConsoleApp3
{
    class Square : Rectangle, IPrint
    {
        public Square(double height = 0) : base(height, height) { }
        public override string FigureName => "Квадрат";
        public new void Print() => Console.WriteLine(this.ToString());

        public override string ToString()
        {
            return $"{this.FigureName} с площадью {this.Area()} и стороной {H}";
        }
    }
}
```

Matrix.cs

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

```
namespace Lab3
{
    class Matrix<T>
    {
        Dictionary<string, T> _matrix = new Dictionary<string, T>();
        int maxX;//столбцы
        int maxY;//строки
        int maxZ;
        IMatrixCheckEmpty<T> checkEmpty;//проверка пустого элемента
        public Matrix(int px, int py, int pz, IMatrixCheckEmpty<T>
checkEmptyParam)//констуктор

        {
            this.maxX = px;
            this.maxY = py;
            this.maxZ = pz;
            this.checkEmpty = checkEmptyParam;
        }

        public T this[int x, int y, int z]
        {
            set//запись элемента
            {
                CheckBounds(x, y, z);//проверка границ
                string key = DictKey(x, y, z);//вычисление ключа для записи в словарь
                this._matrix.Add(key, value);//запись
            }
            get
            {
                CheckBounds(x, y, z);
                string key = DictKey(x, y, z);
                if (this._matrix.ContainsKey(key)) { return this._matrix[key]; }//если
есть элемент с таким ключом, то возвращаем значение элемента
                else { return this.checkEmpty.getEmptyElement(); }
            }
        }
        //проверка границ
        void CheckBounds(int x, int y, int z)
        {

            if (x < 0 || x >= this.maxX) { throw new ArgumentOutOfRangeException("x",
"x=" + x + " выходит за границы"); }
            if (y < 0 || y >= this.maxY) { throw new ArgumentOutOfRangeException("y",
"y=" + y + " выходит за границы"); }
            if (z < 0 || z >= this.maxZ) { throw new ArgumentOutOfRangeException("z",
"z=" + z + " выходит за границы"); }
        }
        //формирование ключа
        string DictKey(int x, int y, int z)
        { return x.ToString() + "_" + y.ToString() + "_" + z.ToString();
        }
        // i- слой(номер таблицы), j - строка, k - столбец
        //приводит матрицу к строке
        public override string ToString()
        {
            StringBuilder b = new StringBuilder();
            for (int i = 0; i<this.maxZ; i++)
            {
                for (int j = 0; j < this.maxY; j++)
                {
                    b.Append("[");
                    for (int k = 0; k < this.maxX; k++)
                    {
                        if (k > 0) { b.Append("\t"); }//добавление разделителя-табуляции
```

```
                        if (!this.checkEmpty.checkEmptyElement(this[i, j, k])) {
b.Append(this[i, j, k].ToString()); }//текущий элемент не пустой, то добавить к строке
элемент
                        else { b.Append(" - "); }//добавить пустоту
                    }
                    b.Append("]\n");
                }
                b.Append("\n***\n\n");
            }
            return b.ToString();
        }
    }

}
```

SimpleList.cs

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

namespace Lab3
{
    class SimpleList<T> //реализует список
        where T : IComparable
    {
        protected SimpleListItem<T> first = null;//первый элемент
        protected SimpleListItem<T> last = null;//последний
        public int Count//кол-во элементов
        {
            get { return _count; }
            protected set { _count = value; }
        }
        int _count;
        public void Add(T element)//добавление элемента
        {
            SimpleListItem<T> newItem = new SimpleListItem<T>(element);
            this.Count++;
            if (last == null)//добавление первого элемента
            {
                this.first = newItem;
                this.last = newItem;
            }
            else//добавление следующих элементов
            {
                //обновление последнего
                this.last.next = newItem;//присоединенбие элемента к цепочке
                this.last = newItem;//новый элемент=последний
            }
        }
        // Чтение элемента с заданным номером
        public SimpleListItem<T> GetItem(int number)
        {
            if ((number < 0) || (number >= this.Count))
            { throw new Exception("Выход за границу индекса"); }//исключение
            SimpleListItem<T> current = this.first;
            int i = 0;
            while (i < number)//пропускаем нужное кол-во элементов
            {
                current = current.next;//переход к следующему
                i++;
            }
            return current;
        }
        // Чтение элемента с заданным номером
```

```
        public T Get(int number) { return GetItem(number).data; }
    }
}
```

SimpleStack.cs

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

namespace Lab3
{
    class SimpleStack<T> : SimpleList<T> where T : IComparable
    {
        public void Push(T element) { Add(element); }//добавить в конец
        public T Pop()//удаление и чтение из стека
        {
            //default(T) - значение по умолчанию T
            T Result = default(T);
            if (this.Count == 0) return Result;//возвращает знач-ие по умолчаниию
            if (this.Count == 1)//элемент единственный
            {
                Result = this.first.data;//читаем данные
                this.first = null;
                this.last = null;
            }
            else
            {
                SimpleListItem<T> newLast = this.GetItem(this.Count - 2);//предпоследний
элемент
                Result = newLast.next.data;//чтение из последнего
                this.last = newLast;//предпоследний=последний
                newLast.next = null;//удалим последний
            }
            this.Count--;
            return Result;
        }
    }

}
```

Program.cs

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

namespace Lab3
{
    class Program
    {
        static void Main(string[] args)
        {
            Circle cir = new Circle(100);
            Rectangle rec = new Rectangle(5, 7);
            Square sc = new Square(9);
            Circle cir2 = new Circle(100);

            //*** Test 1: ArrayList ***
            ArrayList al = new ArrayList();//массив фигур
            al.Add(cir);
            al.Add(rec);
            al.Add(sc);
```

```
            al.Add(cir2);

            Console.WriteLine("Test 1: ArrayList");
            Console.WriteLine("\nДо сортировки:");
            foreach (Figure f in al) Console.WriteLine(f);
            al.Sort();
            Console.WriteLine("\nПосле сортировки:");
            foreach (Figure f in al) Console.WriteLine(f);

            //*** Test 2: List<T> ***
            List<Figure> list = new List<Figure>();
            list.Add(sc);
            list.Add(rec);
            list.Add(cir);
            list.Add(cir2);

            Console.WriteLine("\n\nTest 2: List<T>");
            Console.WriteLine("\nДо сортировки:");
            foreach (Figure f in list) Console.WriteLine(f);
            list.Sort();
            Console.WriteLine("\nПосле сортировки:");
            foreach (Figure f in list) Console.WriteLine(f);

            //*** Test 3: Matrix<T> ***
            Matrix<Figure> matrix = new Matrix<Figure>(3, 3, 3, new
FigureMatrixCheckEmpty());
            matrix[0, 0, 0] = rec;
            matrix[1, 1, 1] = sc;
            matrix[2, 2, 2] = cir;
            matrix[2, 0, 2] = cir2;
            Console.WriteLine("\n\nTest 3: Matrix<T>\n");
            Console.WriteLine(matrix.ToString());



            //*** Test 4: SimpleStack<T> ***
            SimpleStack<Figure> stack = new SimpleStack<Figure>();//стек из фигур
            stack.Push(rec);
            stack.Push(sc);
            stack.Push(cir2);
            stack.Push(cir);
            Console.WriteLine("\nTest 4: SimpleStack<T>\n");
            while (stack.Count > 0)//печать
            {
                Figure f = stack.Pop();
                Console.WriteLine(f);
            }

            Console.ReadKey();
        }
    }
}
```

IMatrixCheckEmpty.cs:

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

namespace Lab3
{
    interface IMatrixCheckEmpty<T>
    {
        T getEmptyElement();//возвращает пустой элемент матрицы
        bool checkEmptyElement(T element);//проверка на пустой элемент
```

```
    }
}
```

FigureMatrixCheckEmpty.cs:

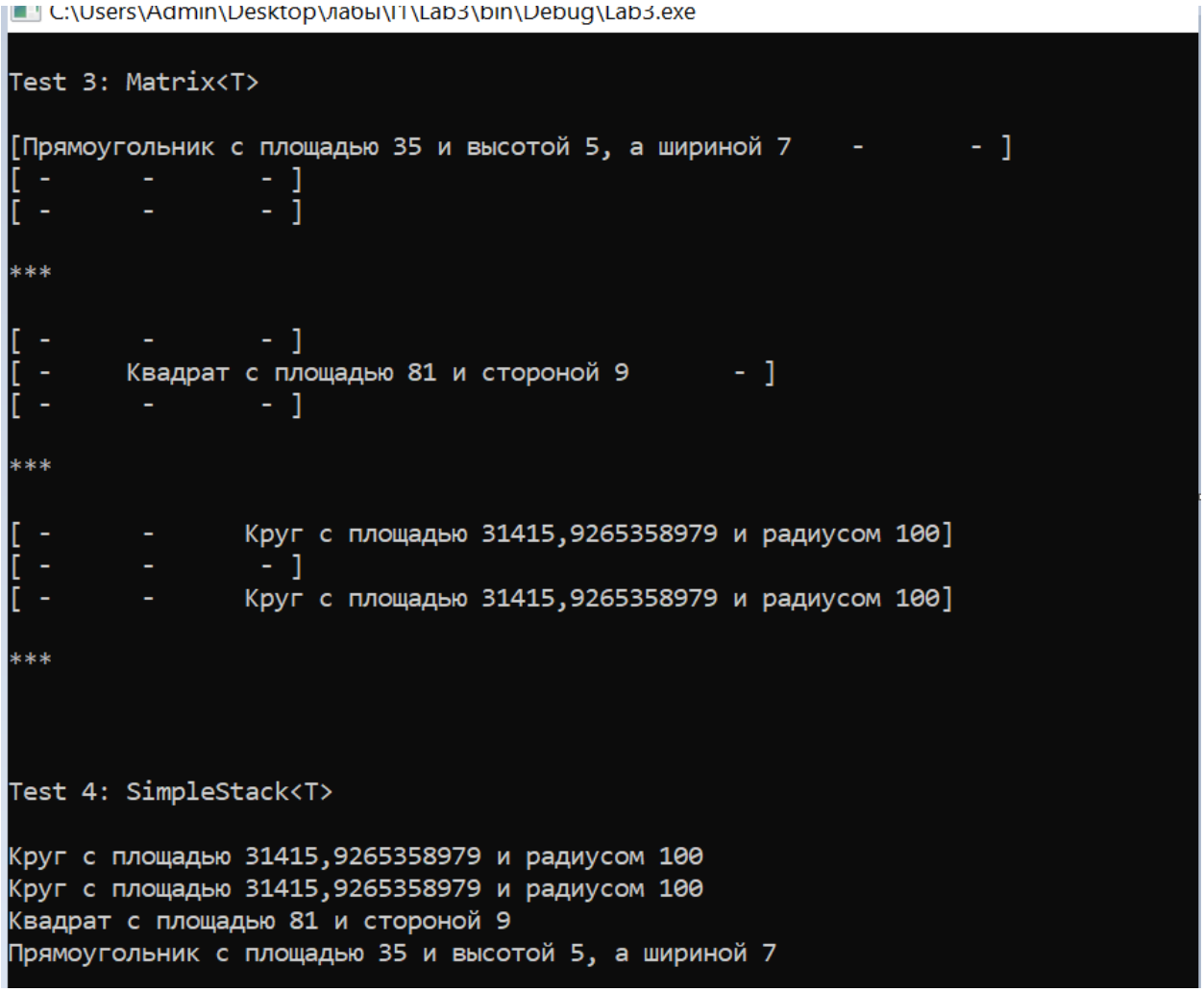
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab3
{
    class FigureMatrixCheckEmpty : IMatrixCheckEmpty<Figure>
    {
        public Figure getEmptyElement() { return null; }//возвращаем null в кач-ве
пустого элемента
        public bool checkEmptyElement(Figure element)//проверка, что переданный параметр
= null
        {
            if (element == null) { return true; }
            return false;
        }
    }
}
```

SimpleListItem.cs:

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

namespace Lab3
{
    //хранит данные и указывает на следующий элемент
    //элемент списка
    class SimpleListItem<T>//контейнер элемента списка
    {
        public T data { get; set; }//данные
        public SimpleListItem<T> next { get; set; }//следующий элемент
        public SimpleListItem(T param) { this.data = param; }//конструктор
    }
}
```

Результаты программы:

```
Test 1: ArrayList

До сортировки:
Круг с площадью 31415,9265358979 и радиусом 100
Прямоугольник с площадью 35 и высотой 5, а шириной 7
Квадрат с площадью 81 и стороной 9
Круг с площадью 31415,9265358979 и радиусом 100

После сортировки:
Прямоугольник с площадью 35 и высотой 5, а шириной 7
Квадрат с площадью 81 и стороной 9
Круг с площадью 31415,9265358979 и радиусом 100
Круг с площадью 31415,9265358979 и радиусом 100


Test 2: List<T>

До сортировки:
Квадрат с площадью 81 и стороной 9
Прямоугольник с площадью 35 и высотой 5, а шириной 7
Круг с площадью 31415,9265358979 и радиусом 100
Круг с площадью 31415,9265358979 и радиусом 100

После сортировки:
Прямоугольник с площадью 35 и высотой 5, а шириной 7
Квадрат с площадью 81 и стороной 9
Круг с площадью 31415,9265358979 и радиусом 100
Круг с площадью 31415,9265358979 и радиусом 100
```


C:\Users\Admin\Desktop\labы\ПТ\Lab3\bin\Debug\Lab3.exe
Test 3: Matrix<T>
[Прямоугольник с площадью 35 и высотой 5, а шириной 7 - -]
[- - -]
[- - -]
***
[- - -]
[- Квадрат с площадью 81 и стороной 9 -]
[- - -]
***
[- - Круг с площадью 31415,9265358979 и радиусом 100]
[- - -]
[- - Круг с площадью 31415,9265358979 и радиусом 100]
***
Test 4: SimpleStack<T>
Круг с площадью 31415,9265358979 и радиусом 100
Круг с площадью 31415,9265358979 и радиусом 100
Квадрат с площадью 81 и стороной 9
Прямоугольник с площадью 35 и высотой 5, а шириной 7