Московский государственный технический университет
им. Н.Э. Баумана

Кафедра
“Системы обработки информации и управления”
(ИУ – 5)

Лабораторная работа по №3 по дисциплине «Базовые компоненты интернет-технологий»

Выполнил:
студент гр. ИУ5 - 31
Саадуев А.С.

“27” декабря 2017 г.

Москва – 2017 г.

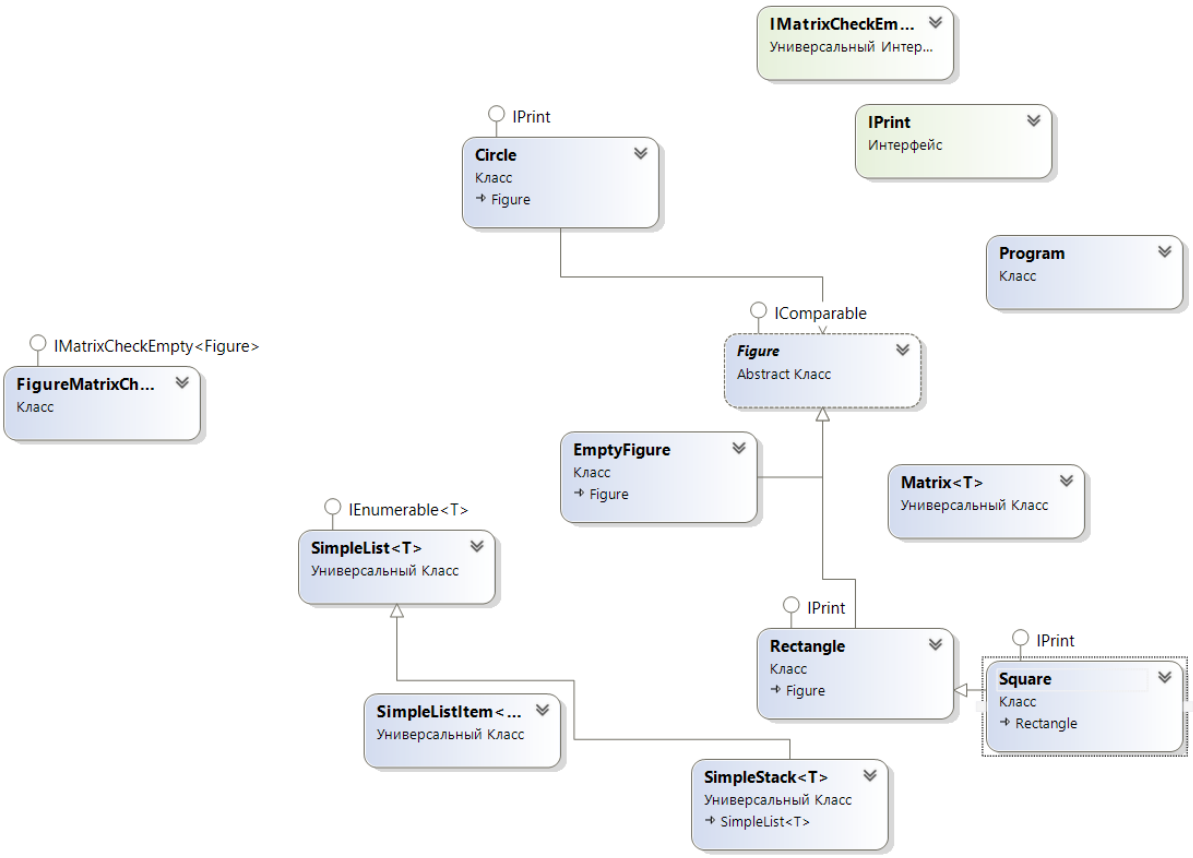
Задание:

Разработать программу для решения квадратного уравнения.

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:



Код:

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

namespace FigureCollections
{
    class EmptyFigure : Figure
    {
        public override double Area()
        {
            return 0;
        }
    }
    class Circle : Figure, IPrint
    {
```

```
    /// <summary>
    /// Ширина
    /// </summary>
    double radius;

    /// <summary>
    /// Основной конструктор
    /// </summary>
    /// <param name="ph">Высота</param>
    /// <param name="pw">Ширина</param>
    public Circle(double pr)
    {
        this.radius = pr;
        this.Type = "Круг";
    }

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

    public void Print()
    {
        Console.WriteLine(this.ToString());
    }
}
abstract class Figure : IComparable
{
    /// <summary>
    /// Тип фигуры
    /// </summary>
    public string Type
    {
        get
```

```
    {
        return this._Type;
    }
    protected set
    {
        this._Type = value;
    }
}
string _Type;

/// <summary>
/// Вычисление площади
/// </summary>
/// <returns></returns>
public abstract double Area();

/// <summary>
/// Приведение к строке, переопределение метода Object
/// </summary>
/// <returns></returns>
public override string ToString()
{
    return this.Type + " площадью " + this.Area().ToString();
}

/// <summary>
/// Сравнение элементов (для сортировки)
/// this - левый параметр сравнения
/// </summary>
/// <param name="obj">правый параметр сравнения</param>
/// <returns>
/// -1 - если левый параметр меньше правого
/// 0 - параметры равны
/// 1 - правый параметр меньше левого
/// </returns>
```

```
    public int CompareTo(object obj)
    {
        //Приведение параметра к типу "фигура"
        Figure p = (Figure)obj;
        //Сравнение
        if (this.Area() < p.Area()) return -1;
        else if (this.Area() == p.Area()) return 0;
        else return 1; //(this.Area() > p.Area())
    }
}
public interface IMatrixCheckEmpty<T>
{
    /// <summary>
    /// Возвращает пустой элемент
    /// </summary>
    T getEmptyElement();

    /// <summary>
    /// Проверка что элемент является пустым
    /// </summary>
    bool checkEmptyElement(T element);
}
interface IPrint
{
    void Print();
}
class FigureMatrixCheckEmpty : IMatrixCheckEmpty<Figure>
{
    /// <summary>
    /// В качестве пустого элемента возвращается null
    /// </summary>
    public Figure getEmptyElement()
    {
        return null;
    }
```

```
    /// <summary>
    /// Проверка что переданный параметр равен null
    /// </summary>
    public bool checkEmptyElement(Figure element)
    {
        bool Result = false;
        if (element == null)
        {
            Result = true;
        }
        return Result;
    }
}
public class Matrix<T>
{
    /// <summary>
    /// Словарь для хранения значений
    /// </summary>
    Dictionary<string, T> _matrix = new Dictionary<string, T>();

    /// <summary>
    /// Количество элементов по горизонтали (максимальное количество столбцов)
    /// </summary>
    int maxX;

    /// <summary>
    /// Количество элементов по вертикали (максимальное количество строк)
    /// </summary>
    int maxY;

    int maxZ;
    /// <summary>
    /// Реализация интерфейса для проверки пустого элемента
    /// </summary>
```

```
IMatrixCheckEmpty<T> checkEmpty;

/// <summary>
/// Конструктор
/// </summary>
public Matrix(int px, int py, int pz, IMatrixCheckEmpty<T> checkEmptyParam)
{
    this.maxX = px;
    this.maxY = py;
    this.maxZ = pz;
    this.checkEmpty = checkEmptyParam;
}

/// <summary>
/// Индексатор для доступа к данных
/// </summary>
public T this[int x, int y, int z]
{
    set
    {
        CheckBounds(x, y, z);
        string key = DictKey(x, y, z);
        this._matrix.Add(key, value);
    }
    get
    {
        CheckBounds(x, y, z);
        string key = DictKey(x, y, z);
        if (this._matrix.ContainsKey(key))
        {
            return this._matrix[key];
        }
        else
        {
            return this.checkEmpty.getEmptyElement();
```

```
                }
            }
        }

        /// <summary>
        /// Проверка границ
        /// </summary>
        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");
            }
        }

        /// <summary>
        /// Формирование ключа
        /// </summary>
        string DictKey(int x, int y, int z)
        {
            return x.ToString() + "_" + y.ToString() + "_" + z.ToString();
        }

        /// <summary>
        /// Приведение к строке
```

```
        /// </summary>
        /// <returns></returns>
        public override string ToString()
        {
            //Класс StringBuilder используется для построения длинных строк
            //Это увеличивает производительность по сравнению с созданием и
склеиванием
            //большого количества обычных строк

            StringBuilder b = new StringBuilder();



            for (int k = 0; k < this.maxZ; k++)
            {
                for (int j = 0; j < this.maxY; j++)
                {
                    b.Append("[");
                    for (int i = 0; i < this.maxX; i++)
                    {

                        //Добавление разделителя-табуляции
                        if (i > 0)
                        {
                            b.Append("\t");
                        }
                        //Если текущий элемент не пустой
                        if (!this.checkEmpty.checkEmptyElement(this[i, j, k]))
                        {
                            //Добавить приведенный к строке текущий элемент
                            b.Append(this[i, j, k].ToString());
                        }
                        else
                        {
                            //Иначе добавить признак пустого значения
                            b.Append(" - ");
                        }
```

```
                }
                b.Append("]\n");
            }
            b.Append("\n\n");
        }
        return b.ToString();
    }
}
class Rectangle : Figure, IPrint
{
    /// <summary>
    /// Высота
    /// </summary>
    double height;

    /// <summary>
    /// Ширина
    /// </summary>
    double width;

    /// <summary>
    /// Основной конструктор
    /// </summary>
    /// <param name="ph">Высота</param>
    /// <param name="pw">Ширина</param>
    public Rectangle(double ph, double pw)
    {
        this.height = ph;
        this.width = pw;
        this.Type = "Прямоугольник";
    }

    /// <summary>
    /// Вычисление площади
    /// </summary>
```

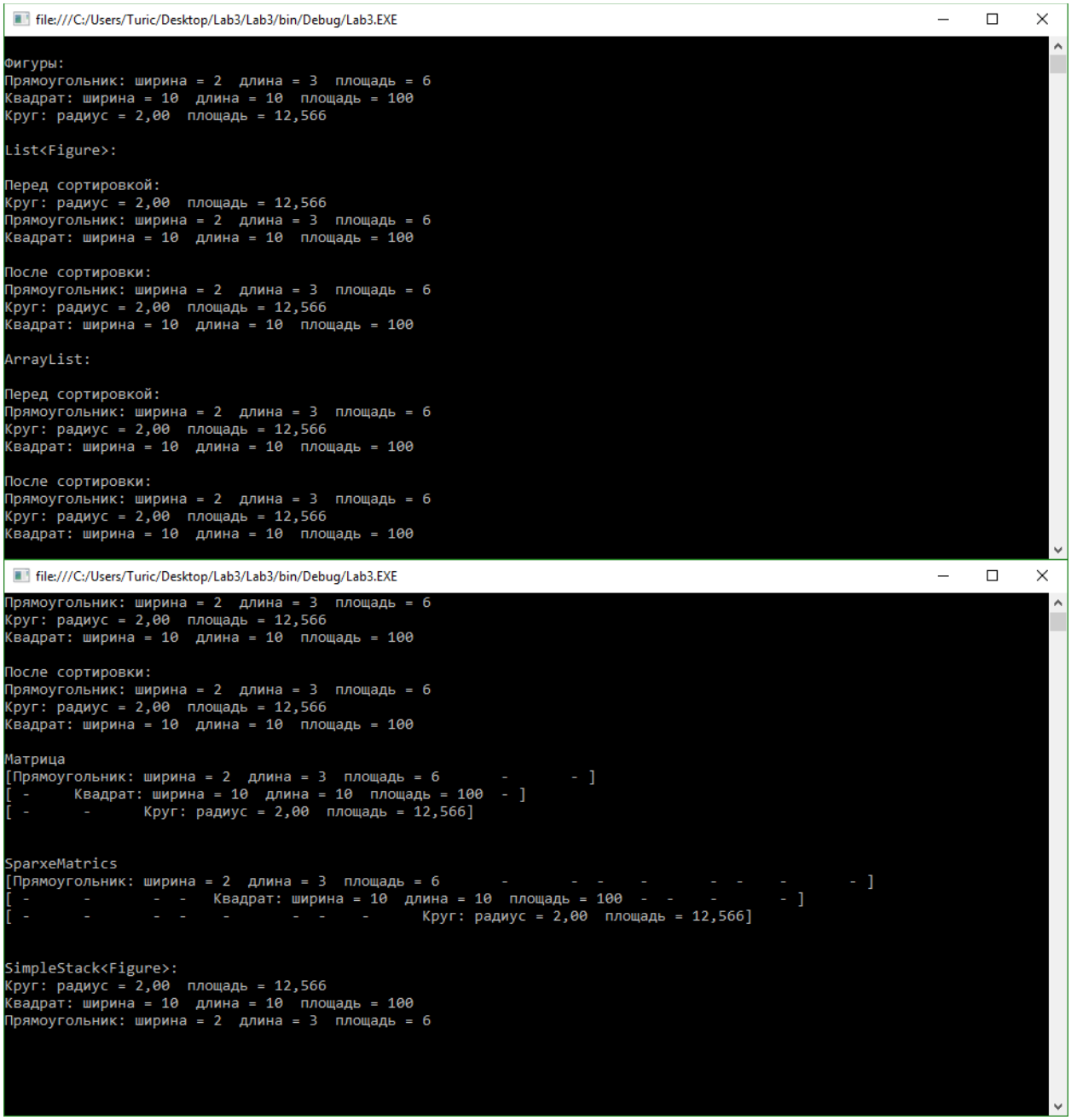
```
    public override double Area()
    {
        double Result = this.width * this.height;
        return Result;
    }

    public void Print()
    {
        Console.WriteLine(this.ToString());
    }
}
class Square : Rectangle, IPrint
{
    public Square(double size)
        : base(size, size)
    {
        this.Type = "Квадрат";
    }
}
class Program
{
    static void Main(string[] args)
    {
        //++++++++++++++++++++++++++++++++++++++++++++++++
        //Создание объектов классов фигур:
        Rectangle rect = new Rectangle(5, 4);
        Square square = new Square(5);
        Circle circle = new Circle(5);

        //++++++++++++++++++++++++++++++++++++++++++++++++
        Console.WriteLine("\nArrayList");
        ArrayList al = new ArrayList();
        al.Add(circle);
        al.Add(rect);
        al.Add(square);
```

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

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

            //++++++++++++++++++++++++++++++++++++++++++++++++
            Console.WriteLine("\nList<Figure>");
            List<Figure> fl = new List<Figure>();
            fl.Add(circle);
            fl.Add(rect);
            fl.Add(square);

            Console.WriteLine("\nПеред сортировкой:");
            foreach (var x in fl) Console.WriteLine(x);

            //сортировка
            fl.Sort();

            Console.WriteLine("\nПосле сортировки:");
            foreach (var x in fl) Console.WriteLine(x);

            //++++++++++++++++++++++++++++++++++++++++++++++++
            Console.WriteLine("\nМатрица");
            Matrix<Figure> matrix = new Matrix<Figure>(3, 3,3, new
FigureMatrixCheckEmpty());
            matrix[0, 0,0] = rect;
            matrix[1, 1,1] = square;
            matrix[2, 2,2] = circle;
            Console.WriteLine(matrix.ToString());

            //Выход за границы индекса и обработка исключения
            try
            {
                Figure temp = matrix[123, 123,123];
```

```
}
catch (ArgumentOutOfRangeException e)
{
    Console.WriteLine(e.Message);
}

//++++++++++++++++++++++++++++++++++++++++++++++++
Console.WriteLine("\nСписок");
SimpleList<Figure> list = new SimpleList<Figure>();
list.Add(circle);
list.Add(rect);
list.Add(square);
Console.WriteLine("\nПеред сортировкой:");
foreach (var x in list) Console.WriteLine(x);
//сортировка
list.Sort();
Console.WriteLine("\nПосле сортировки:");
foreach (var x in list) Console.WriteLine(x);

//++++++++++++++++++++++++++++++++++++++++++++++++
Console.WriteLine("\nСтек");

SimpleStack<Figure> stack = new SimpleStack<Figure>();
//добавление данных в стек
stack.Push(rect);
stack.Push(square);
stack.Push(circle);
//чтение данных из стека
while (stack.Count > 0)
{
    Figure f = stack.Pop();
    Console.WriteLine(f);
}

Console.ReadLine();
```

```
        }
    }
}
```

Скриншот:


file:///C:/Users/Turic/Desktop/Lab3/Lab3/bin/Debug/Lab3.EXE
Фигуры:
Прямоугольник: ширина = 2 длина = 3 площадь = 6
Квадрат: ширина = 10 длина = 10 площадь = 100
Круг: радиус = 2,00 площадь = 12,566
List<Figure>:
Перед сортировкой:
Круг: радиус = 2,00 площадь = 12,566
Прямоугольник: ширина = 2 длина = 3 площадь = 6
Квадрат: ширина = 10 длина = 10 площадь = 100
После сортировки:
Прямоугольник: ширина = 2 длина = 3 площадь = 6
Круг: радиус = 2,00 площадь = 12,566
Квадрат: ширина = 10 длина = 10 площадь = 100
ArrayList:
Перед сортировкой:
Прямоугольник: ширина = 2 длина = 3 площадь = 6
Круг: радиус = 2,00 площадь = 12,566
Квадрат: ширина = 10 длина = 10 площадь = 100
После сортировки:
Прямоугольник: ширина = 2 длина = 3 площадь = 6
Круг: радиус = 2,00 площадь = 12,566
Квадрат: ширина = 10 длина = 10 площадь = 100
file:///C:/Users/Turic/Desktop/Lab3/Lab3/bin/Debug/Lab3.EXE
Прямоугольник: ширина = 2 длина = 3 площадь = 6
Круг: радиус = 2,00 площадь = 12,566
Квадрат: ширина = 10 длина = 10 площадь = 100
После сортировки:
Прямоугольник: ширина = 2 длина = 3 площадь = 6
Круг: радиус = 2,00 площадь = 12,566
Квадрат: ширина = 10 длина = 10 площадь = 100
Матрица
[Прямоугольник: ширина = 2 длина = 3 площадь = 6
[- Квадрат: ширина = 10 длина = 10 площадь = 100 -]
[- - Круг: радиус = 2,00 площадь = 12,566]
SparxeMatrics
[Прямоугольник: ширина = 2 длина = 3 площадь = 6
[- - - - Квадрат: ширина = 10 длина = 10 площадь = 100 - - - -]
[- - - - - - - - Круг: радиус = 2,00 площадь = 12,566]
SimpleStack<Figure>:
Круг: радиус = 2,00 площадь = 12,566
Квадрат: ширина = 10 длина = 10 площадь = 100
Прямоугольник: ширина = 2 длина = 3 площадь = 6