

**Московский государственный технический
университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра ИУ5.

Курс «Базовые компоненты интернет-технологий»

Отчет по лабораторной работе №3

Выполнил:

студент группы ИУ5-31Б

Федоров Иван

Подпись и дата:

Проверил:

преподаватель каф. ИУ5

Гапанюк Юрий Евгеньевич

Подпись и дата:

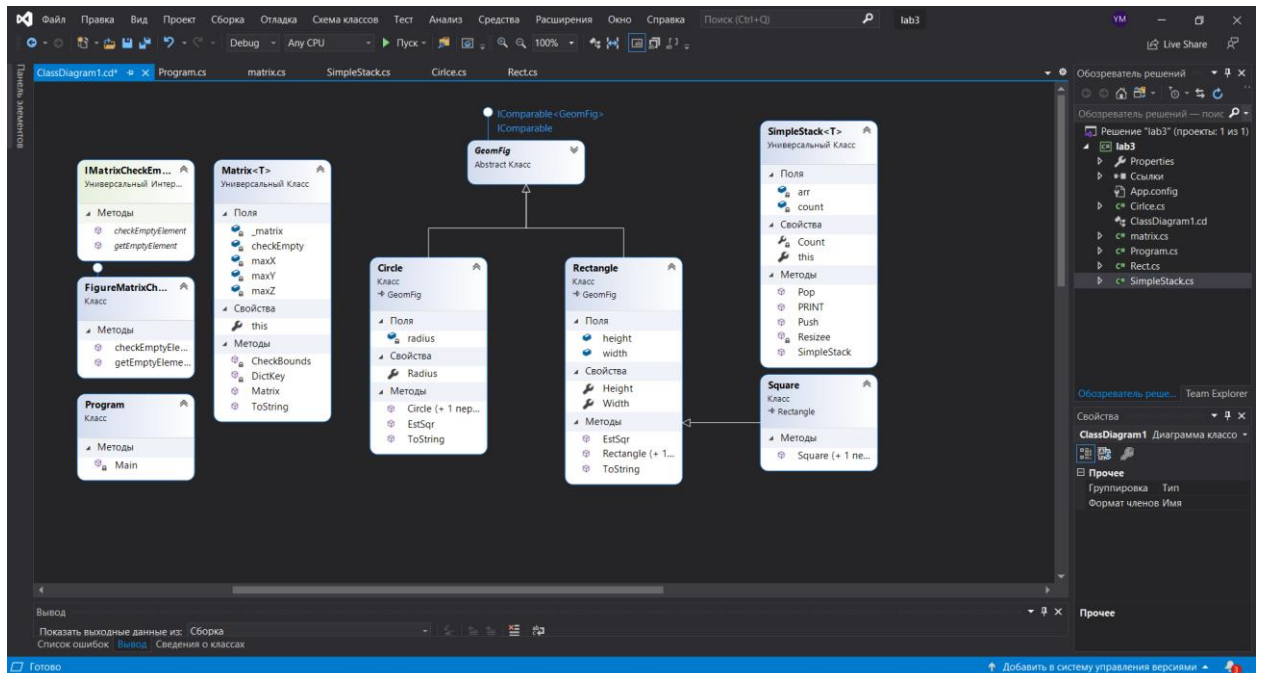
г. Москва, 2020 г.

Постановка задачи

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 `Comparable`. 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.

Разработка интерфейса класса



Листинг программы

Program.cs

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

namespace lab3
{

    abstract class GeomFig : IComparable<GeomFig>, IComparable
    {

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

        public int CompareTo(GeomFig a)
        {
            if (a.s < this.s) return 1;
            if (a.s > this.s) return -1;
            return 0;
        }
    }
}
```

```

public int CompareTo( object temp)
{
    GeomFig o = temp as GeomFig;
    if (o != null)
        return this.s.CompareTo(o.s);
    else
        throw new Exception("Невозможно сравнить два объекта");
}
protected double s;

public virtual double EstSqr()
{ return 1; }
}

```

```

class Program
{
    static void Main(string[] args)
    {
        Square Example = new Square(20);
        Circle one = new Circle(10);
        Rectangle two = new Rectangle(30, 5);
        Square three = new Square(20);

        //ARRlist

        //
        Console.WriteLine("\n ARRAYLIST \n");

        ArrayList figlist = new ArrayList() { Example, one, two, three };

        figlist.Sort();

        foreach (GeomFig temp in figlist)
        {
            Console.WriteLine(temp.ToString());
        }

        //
        Console.WriteLine("\n LIST \n");

        List<GeomFig> listof = new List<GeomFig>() { Example, one, two, three };

        listof.Sort();

        foreach (GeomFig temp in listof)
        {
            Console.WriteLine(temp.ToString());
        }
        //
        Console.WriteLine("\n Односвязный список \n");

        SimpleStack<GeomFig> SSS = new SimpleStack<GeomFig>(5);
        SSS.Push(Example);
        SSS.Push(one);
        SSS.Push(two);
        SSS.Push(three);
        SSS.Pop();
        Console.WriteLine("\n Односвязный список \n");
    }
}

```

```

        SSS.PRINT();

        //Разреженная матрица
        Console.ForegroundColor = ConsoleColor.Green;
        Console.WriteLine("\nРазреженная матрица");
        Console.ResetColor();
        Matrix<GeomFig> matrix = new Matrix<GeomFig>(4, 4, 3, new
FigureMatrixCheckEmpty());
        matrix[1, 1, 0] = one;
        matrix[1, 2, 1] = two;
        matrix[3, 3, 2] = three;
        Console.WriteLine(matrix.ToString());
    }
}

```

Matrix.cs

```

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

namespace lab3
{
    public interface IMatrixCheckEmpty<T>
    {
        T getEmptyElement();
        bool checkEmptyElement(T element);
    }
    class FigureMatrixCheckEmpty : IMatrixCheckEmpty<GeomFig>
    {
        public GeomFig getEmptyElement()
        {
            return null;
        }
        public bool checkEmptyElement(GeomFig element)
        {
            bool Result = false;
            if (element == null)
                Result = true;
            return Result;
        }
    }
    public class Matrix<T>
    {
        Dictionary<string, T> _matrix = new Dictionary<string, T>();
        int maxX;
        int maxY;
        int maxZ;

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

```

```

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();
    }
}
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");
}

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 < this.maxZ; k++)
    {
        b.Append("{\n");
        for (int j = 0; j < this.maxY; j++)
        {
            b.Append("[");
            for (int i = 0; i < this.maxX; i++)
            {
                if (i > 0)
                    b.Append("\t");
                if (!this.checkEmpty.checkEmptyElement(this[i, j, k]))
                {
                    b.Append(this[i, j, k].ToString());
                }
                else b.Append(" - ");
            }
            b.Append("]\n");
        }
        b.Append("}\n\n");
    }
    return b.ToString();
}
}
}

```

SimpleStack.cs

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

namespace lab3
{
    class SimpleStack<T>
    {
        T[] arr;
        int count;

        int Count
        {
            get { return count; }
            set { }
        }

        public SimpleStack(int size)
        {
            if (size < 1) size = 2;

            arr = new T[size];
        }

        public T this[int index] // перегрузка
        {
            get
            {
                if (index < 0 || index >= count)
                    throw new Exception("index is incorrect");

                return arr[index];
            }
            set
            {
                if (index < 0 || index >= count)
                    throw new Exception("index is incorrect");

                arr[index] = value;
            }
        }
        public void Push(T value)
        {
            if (count == arr.Length)
                Resize();

            arr[count] = value;
            count++;
        }

        public void Pop()
        {
            PRINT();
        }
    }
}
```

```

        Console.WriteLine("Выберите номер элемента, который вы хотите удалить");
        string entered = Console.ReadLine();
        int i = Convert.ToInt32(entered);
        if (i > arr.Length || i < 0) throw new Exception("Вы ввели неправильный
индекс массива");
        for (int j = i; arr[j] != null; j++)
        { arr[j] = arr[j + 1]; }
    }

    private void Resizee()
    {
        int newCapacity = arr.Length * 2;
        T[] newArray = new T[newCapacity];
        for (int i = 0; i < count; i++)
            newArray[i] = arr[i];

        arr = newArray;
    }

    public void PRINT()
    {
        for (int i = 0; arr[i] != null; i++)
        {
            Console.WriteLine(i + " " + arr[i].ToString());
        }
    }
}
}

```

Circle.cs

```

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

namespace lab3
{
    class Circle : GeomFig
    {
        double radius;
        public Circle() { radius = 1; }
        public Circle(double R) { radius = R; }
        public double Radius
        {
            get { return radius; }
            set { if (value < 0) value *= (-1); radius = value; }
        }

        public override double EstSqr()
        {
            s = 3.14 * radius * radius;
            return s;
        }
        public override string ToString()
        {
            return "Площадь круга = " + EstSqr().ToString() + " Радиус: " +
radius.ToString();
        }
    }
}

```

```
}  
}
```

Анализ результатов

```
C:\Windows\system32\cmd.exe  
ARRAYLIST  
Площадь фигуры: 400 Стороны: 20, 20  
Площадь круга = 314 Радиус: 10  
Площадь фигуры: 150 Стороны: 30, 5  
Площадь фигуры: 400 Стороны: 20, 20  
LIST  
Площадь фигуры: 150 Стороны: 30, 5  
Площадь круга = 314 Радиус: 10  
Площадь фигуры: 400 Стороны: 20, 20  
Площадь фигуры: 400 Стороны: 20, 20  
Односвязный список  
0 Площадь фигуры: 400 Стороны: 20, 20  
1 Площадь круга = 314 Радиус: 10  
2 Площадь фигуры: 150 Стороны: 30, 5  
3 Площадь фигуры: 400 Стороны: 20, 20  
Выберите номер элемента, который вы хотите удалить  
2  
Односвязный список  
0 Площадь фигуры: 400 Стороны: 20, 20  
1 Площадь круга = 314 Радиус: 10  
2 Площадь фигуры: 400 Стороны: 20, 20  
Разреженная матрица  
{  
- - - - ]  
- Площадь круга = 314 Радиус: 10 - - ]  
- - - - ]  
- - - - ]  
}  
  
{  
- - - - ]  
- - - - ]  
- Площадь фигуры: 150 Стороны: 30, 5 - - ]  
- - - - ]  
}  
  
{  
- - - - ]  
- - - - ]  
- - - - ]  
- - - - Площадь фигуры: 400 Стороны: 20, 20 ]  
}
```