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 ИУ5-31

Бондаренко Иван

Подпись и дата:

Проверил:

преподаватель каф. ИУ5

Гапанюк Ю. Е.

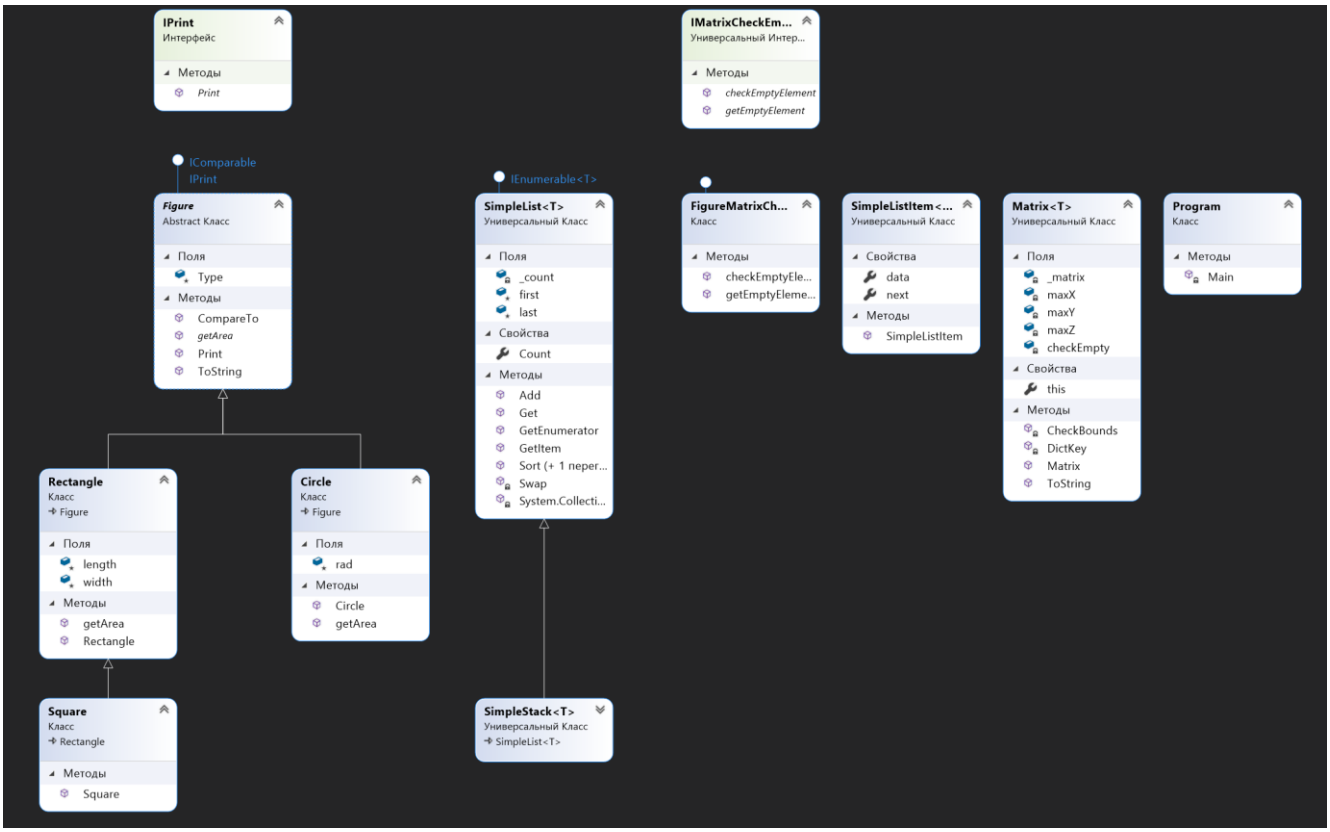
Подпись и дата:

Москва, 2020 г.

Описание задания:

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:

1. Program.cs

```
1) using System;
2) using System.Collections;
3) using System.Collections.Generic;
4)
5) namespace Lab3
6) {
7)     class Program
8)     {
9)         static void Main(string[] args)
10)             {
11)                 // Объекты фигур
12)                 Rectangle rec = new Rectangle(4.5, 2);
13)                 Square sq = new Square(3.14);
14)                 Circle circle = new Circle(1);
15)
16)                 // Проверка через ArrayList
17)                 Console.ForegroundColor = ConsoleColor.Green;
18)                 Console.WriteLine("\t\tArrayList");
19)                 Console.ResetColor();
20)                 ArrayList alist = new ArrayList();
21)
22)                 alist.Add(4);
23)                 alist.Add(sq);
24)                 alist.Add(rec);
25)                 alist.Add(circle);
26)
27)                 foreach (object item in alist)
28)                     Console.WriteLine(item);
29)
30)                 try
31)                 {
32)                     alist.Sort();
33)                 }
34)                 catch (InvalidOperationException e)
35)                 {
36)                     Console.ForegroundColor = ConsoleColor.Red;
37)                     Console.WriteLine('\n'+e.Message);
38)                     Console.ResetColor();
39)                 }
40)
41)                 Console.ForegroundColor = ConsoleColor.DarkYellow;
42)                 Console.WriteLine("\nПосле сортировки:\n");
43)                 Console.ResetColor();
44)
45)                 foreach (object item in alist)
46)                     Console.WriteLine(item);
47)
48)                 alist.Remove(4);
49)
50)                 Console.ForegroundColor = ConsoleColor.DarkYellow;
51)                 Console.WriteLine("\n\nУдаление мешающего элемента\n");
52)                 Console.ResetColor();
53)
54)                 foreach (object item in alist)
55)                     Console.WriteLine(item);
56)
57)                 alist.Sort();
58)
59)                 Console.ForegroundColor = ConsoleColor.DarkYellow;
60)                 Console.WriteLine("\nПосле сортировки:\n");
61)                 Console.ResetColor();
62)
63)                 foreach (object item in alist)
64)                     Console.WriteLine(item);
```

```
65)
66)                     // Проверка через List<>
67)                     Console.ForegroundColor = ConsoleColor.Green;
68)                     Console.WriteLine("\n\t\tList<Figure>");
69)                     Console.ResetColor();
70)                     List<Figure> list = new List<Figure>();
71)                     list.Add(rec);
72)                     list.Add(sq);
73)                     list.Add(circle);
74)                     foreach (Figure item in list)
75)                         Console.WriteLine(item);
76)                     list.Sort();
77)
78)                     Console.ForegroundColor = ConsoleColor.DarkYellow;
79)                     Console.WriteLine("\nПосле сортировки:\n");
80)                     Console.ResetColor();
81)
82)                     foreach (Figure item in list)
83)                         Console.WriteLine(item);
84)
85)                     // Проверка доработанной трехмерной матрицы
86)                     Console.ForegroundColor = ConsoleColor.Green;
87)                     Console.WriteLine("\n\t\tМатрица");
88)                     Console.ForegroundColor = ConsoleColor.Gray;
89)                     Matrix<Figure> matrix = new Matrix<Figure>(new FigureMatrixCheckEmpty(), 3, 3, 3);

90)                     matrix[0, 0, 0] = rec;
91)                     matrix[1, 0, 1] = sq;
92)                     matrix[1, 1, 1] = sq;
93)                     matrix[2, 2, 0] = circle;
94)                     Console.WriteLine(matrix.ToString());
95)
96)                     // Проверка SimpleStack
97)                     Console.ForegroundColor = ConsoleColor.Green;
98)                     Console.WriteLine("\n\t\tSimpleStack<Figure>");
99)                     Console.ForegroundColor = ConsoleColor.Gray;
100)                    SimpleStack<Figure> slist = new SimpleStack<Figure>();
101)                    slist.Push(rec);
102)                    slist.Push(sq);
103)                    slist.Push(circle);
104)                    while(slist.Count != 0)
105)                        slist.Pop().Print();
106)                    slist.Push(rec);
107)                    slist.Push(sq);
108)                    slist.Push(circle);
109)                    slist.Sort();
110)
111)                    Console.ForegroundColor = ConsoleColor.DarkYellow;
112)                    Console.WriteLine("\nПосле сортировки:\n");
113)                    Console.ResetColor();
114)
115)                    foreach (Figure item in slist)
116)                        Console.WriteLine(item);
117)
118)                    Console.WriteLine("\n");
119)                }
120)            }
121)        }
```

2. Фигуры\\Figure.cs

```
1. using System;
2.
3. namespace Lab3
4. {
5.     abstract class Figure : IComparable, IPrint
6.     {
7.         protected string Type;
```

```
        public abstract double getArea();
        public void Print()
                {
                    Console.WriteLine(this.ToString());
                }
                public int CompareTo(object obj)
                {
                    Figure robj = obj as Figure;
                    if (robj != null)
                    {
                        return this.getArea().CompareTo(robj.getArea());
                    }
                    else
                        throw new InvalidOperationException(message:"Несравнимые объекты");
                }
                public override string ToString()
                {
                    return $"[{this.Type} площадью {this.getArea()}]";
                }
            }
        }
```

3. Rectangle.cs

```
namespace Lab3
{
    class Rectangle : Figure
    {
        protected double length, width;

        public Rectangle(double length, double width)
        {
            this.length = length;
                    this.width = width;
                    Type = "Прямоугольник";
                }
                public override double getArea()
                {
                    return length * width;
                }
            }
        }
```

4. Square.cs

```
namespace Lab3
{
    class Square : Rectangle
    {
        public Square(double length) : base(length, length) { Type = "Квадрат"; }
    }
}
```

5. Circle.cs

```
using System;
namespace Lab3
{
    class Circle : Figure
    {
        protected double rad;

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

```
                }
                public override double getArea()
                {
                    return rad * rad * Math.PI;
                }
            }
        }
```

6. IPrint.cs

```
namespace Lab3
{
    interface IPrint
    {
        void Print();
    }
}
```

7. FigareMatrixCheckEmpty.cs

```
namespace Lab3
{
    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;
                }
            }
        }
```

8. IMatrixCheckEmpty.cs

```
namespace Lab3
{
    /// <summary>
    /// Проверка пустого элемента матрицы
    /// </summary>
    public interface IMatrixCheckEmpty<T>
    {
        /// <summary>
        /// Возвращает пустой элемент
                /// </summary>
                T getEmptyElement();

                /// <summary>
                /// Проверка что элемент является пустым
                /// </summary>
                bool checkEmptyElement(T element);
            }
```

```
18.       }
```

9. Matrix.cs

```
1. using System;
2. using System.Collections.Generic;
3. using System.Linq;
4. using System.Text;
5.
6. namespace Lab3
7. {
8.     public class Matrix<T>
9.     {
10.                /// <summary>
11.                /// Словарь для хранения значений
12.                /// </summary>
13.                Dictionary<string, T> _matrix = new Dictionary<string, T>();
14.
15.                /// <summary>
16.                /// Количество элементов по горизонтали (максимальное количество столбцов)
17.                /// </summary>
18.                int maxX;
19.
20.                /// <summary>
21.                /// Количество элементов по вертикали (максимальное количество строк)
22.                /// </summary>
23.                int maxY;
24.
25.
26.                /// <summary>
27.                /// Количество элементов в глубину (максимальное количество строк)
28.                /// </summary>
29.                int maxZ;
30.
31.
32.                /// <summary>
33.                /// Реализация интерфейса для проверки пустого элемента
34.                /// </summary>
35.                IMatrixCheckEmpty<T> checkEmpty;
36.
37.                /// <summary>
38.                /// Конструктор
39.                /// </summary>
40.                public Matrix(IMatrixCheckEmpty<T> checkEmptyParam, int px, int py, int pz = 0)
41.                {
42.                    this.maxX = px;
43.                    this.maxY = py;
44.                    this.maxZ = pz;
45.                    this.checkEmpty = checkEmptyParam;
46.                }
47.
48.                /// <summary>
49.                /// Индексатор для доступа к данных
50.                /// </summary>
51.                public T this[int x, int y, int z]
52.                {
53.                    set
54.                    {
55.                        CheckBounds(x, y, z);
56.                        string key = DictKey(x, y, z);
57.                        this._matrix.Add(key, value);
58.                    }
59.                    get
60.                    {
61.                        CheckBounds(x, y, z);
62.                        string key = DictKey(x, y, z);
63.                        if (this._matrix.ContainsKey(key))
64.                        {
```

```
65.                         return this._matrix[key];
66.                     }
67.                     else
68.                     {
69.                         return this.checkEmpty.getEmptyElement();
70.                     }
71.                 }
72.             }
73.
74.             /// <summary>
75.             /// Проверка границ
76.             /// </summary>
77.             void CheckBounds(int x, int y, int z)
78.             {
79.                 if (x < 0 || x >= this.maxX)
80.                 {
81.                     throw new ArgumentOutOfRangeException("x", "x=" + x + " выходит за границы");

82.                 }
83.                 if (y < 0 || y >= this.maxY)
84.                 {
85.                     throw new ArgumentOutOfRangeException("y", "y=" + y + " выходит за границы");

86.                 }
87.                 if (z < 0 || z >= this.maxZ)
88.                 {
89.                     throw new ArgumentOutOfRangeException("z", "z=" + y + " выходит за границы");

90.                 }
91.             }
92.
93.             /// <summary>
94.             /// Формирование ключа
95.             /// </summary>
96.             string DictKey(int x, int y, int z)
97.             {
98.                 return x.ToString() + "_" + y.ToString() + "_" + z.ToString();
99.             }
100.
101.            /// <summary>
102.            /// Приведение к строке
103.            /// </summary>
104.            /// <returns></returns>
105.            public override string ToString()
106.            {
107.                //Класс StringBuilder используется для построения длинных строк
108.                //Это увеличивает производительность по сравнению с созданием и склеиванием
109.                //большого количества обычных строк
110.
111.                StringBuilder b = new StringBuilder();
112.
113.                for (int k = 0; k < this.maxZ; k++)
114.                {
115.                    b.Append($"Глубина\t{k}:\n");
116.                    for (int j = 0; j < this.maxY; j++)
117.                    {
118.                        b.Append("|");
119.                        for (int i = 0; i < this.maxX; i++)
120.                        {
121.                            //Добавление разделителя-табуляции
122.                            if (i > 0)
123.                            {
124.                                b.Append("\t");
125.                            }
126.                            //Если текущий элемент не пустой
127.                            if (!this.checkEmpty.checkEmptyElement(this[i, j, k]))
128.                            {
129.                                //Добавить приведенный к строке текущий элемент
130.                                b.Append(this[i, j, k].ToString());
```

```
                                }
                                else
                                {
                                    //Иначе добавить признак пустого значения
                                    b.Append(" NULL ");
                                }
                            }
                            b.Append("|\n");
                        }
                        b.Append("\n\n");
                    }
                    return b.ToString();
                }
            }
        }
```

10. SimpleList.cs

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

namespace Lab3
{
    /// <summary>
    /// Список
    /// </summary>
    public class SimpleList<T> : IEnumerable<T>
                where T : IComparable
            {
                /// <summary>
                /// Первый элемент списка
                /// </summary>
                protected SimpleListItem<T> first = null;

                /// <summary>
                /// Последний элемент списка
                /// </summary>
                protected SimpleListItem<T> last = null;

                /// <summary>
                /// Количество элементов
                /// </summary>
                public int Count
                {
                    get { return _count; }
                    protected set { _count = value; }
                }
                int _count;

                /// <summary>
                /// Добавление элемента
                /// </summary>
                public void Add(T element)
                {
                    SimpleListItem<T> newItem = new SimpleListItem<T>(element);
                    this.Count++;

                    //Добавление первого элемента
                    if (last == null)
                    {
                        this.first = newItem;
                        this.last = newItem;
                    }
                    //Добавление следующих элементов
                    else
                    {
                        //Присоединение элемента к цепочке
                        this.last.next = newItem;
```

```
51.                         //Просоединенный элемент считается последним
52.                         this.last = newItem;
53.                     }
54.                 }
55.
56.                 /// <summary>
57.                 /// Чтение контейнера с заданным номером
58.                 /// </summary>
59.                 public SimpleListItem<T> GetItem(int number)
60.                 {
61.                     if ((number < 0) || (number >= this.Count))
62.                     {
63.                         //Можно создать собственный класс исключения
64.                         throw new Exception("Выход за границу индекса");
65.                     }
66.
67.                     SimpleListItem<T> current = this.first;
68.                     int i = 0;
69.
70.                     //Пропускаем нужное количество элементов
71.                     while (i < number)
72.                     {
73.                         //Переход к следующему элементу
74.                         current = current.next;
75.                         //Увеличение счетчика
76.                         i++;
77.                     }
78.
79.                     return current;
80.                 }
81.
82.                 /// <summary>
83.                 /// Чтение элемента с заданным номером
84.                 /// </summary>
85.                 public T Get(int number)
86.                 {
87.                     return GetItem(number).data;
88.                 }
89.
90.                 /// <summary>
91.                 /// Для перебора коллекции
92.                 /// </summary>
93.                 public IEnumerator<T> GetEnumerator()
94.                 {
95.                     SimpleListItem<T> current = this.first;
96.
97.                     //Перебор элементов
98.                     while (current != null)
99.                     {
100.                        //Возврат текущего значения
101.                        yield return current.data;
102.                        //Переход к следующему элементу
103.                        current = current.next;
104.                    }
105.                }
106.
107.                //Реализация обобщенного IEnumerator<T> требует реализации необобщенного интерфейса
108.                //Данный метод добавляется автоматически при реализации интерфейса
109.                System.Collections.IEnumerator System.Collections.IEnumerable.GetEnumerator()
110.                {
111.                    return GetEnumerator();
112.                }
113.
114.                /// <summary>
115.                /// Сортировка
116.                /// </summary>
117.                public void Sort()
118.                {
119.                    Sort(0, this.Count - 1);
```

```
120.             }
121.
122.             /// <summary>
123.             /// Алгоритм быстрой сортировки
124.             /// </summary>
125.             private void Sort(int low, int high)
126.             {
127.                 int i = low;
128.                 int j = high;
129.                 T x = Get((low + high) / 2);
130.                 do
131.                 {
132.                     while (Get(i).CompareTo(x) < 0) ++i;
133.                     while (Get(j).CompareTo(x) > 0) --j;
134.                     if (i <= j)
135.                     {
136.                         Swap(i, j);
137.                         i++; j--;
138.                     }
139.                 } while (i <= j);
140.
141.                 if (low < j) Sort(low, j);
142.                 if (i < high) Sort(i, high);
143.             }
144.
145.             /// <summary>
146.             /// Вспомогательный метод для обмена элементов при сортировке
147.             /// </summary>
148.             private void Swap(int i, int j)
149.             {
150.                 SimpleListItem<T> ci = GetItem(i);
151.                 SimpleListItem<T> cj = GetItem(j);
152.                 T temp = ci.data;
153.                 ci.data = cj.data;
154.                 cj.data = temp;
155.             }
156.         }
157.     }
```

11. SimpleListItem.cs

```
1. namespace Lab3
2. {
3.     /// <summary>
4.     /// Элемент списка
5.     /// </summary>
6.     public class SimpleListItem<T>
7.     {
8.         /// <summary>
9.         /// Данные
10.             /// </summary>
11.             public T data { get; set; }
12.
13.             /// <summary>
14.             /// Следующий элемент
15.             /// </summary>
16.             public SimpleListItem<T> next { get; set; }
17.
18.             ///конструктор
19.             public SimpleListItem(T param)
20.             {
21.                 this.data = param;
22.             }
23.         }
24.     }
```

12. SimpleStack.cs

```
using System;

namespace Lab3
{
    class SimpleStack<T> : SimpleList<T> where T : IComparable
    {
        public void Push(T obj)
        {
            Add(obj);
               }
               public T Pop()
               {

                   if (this.Count == 0) return default(T);
                   if (this.Count == 1)
                   {
                       this.first = null;
                       T res = this.last.data;
                       this.last = null;
                       this.Count--;
                       return res;
                   }
                   else
                   {
                       T res = this.last.data;
                       SimpleListItem<T> newLast = this.GetItem(this.Count - 2);
                       this.last = newLast;
                       newLast.next = null;
                       this.Count--;
                       return res;
                   }
               }
           }
       }
```

Пример выполнения программы:

```
                ArrayList
4
[Квадрат площадью 9,8596]
[Прямоугольник площадью 9]
[Круг площадью 3,14159265358979]

Сбой при сравнении двух элементов массива.

После сортировки:

4
[Квадрат площадью 9,8596]
[Прямоугольник площадью 9]
[Круг площадью 3,14159265358979]


Удаление мешающего элемента

[Квадрат площадью 9,8596]
[Прямоугольник площадью 9]
[Круг площадью 3,14159265358979]

После сортировки:

[Круг площадью 3,14159265358979]
[Прямоугольник площадью 9]
[Квадрат площадью 9,8596]

                List<Figure>
[Прямоугольник площадью 9]
[Квадрат площадью 9,8596]
[Круг площадью 3,14159265358979]

После сортировки:

[Круг площадью 3,14159265358979]
[Прямоугольник площадью 9]
[Квадрат площадью 9,8596]

                Матрица
Глубина 0:
|[Прямоугольник площадью 9]      NULL    NULL |
| NULL   NULL    NULL |
| NULL   NULL   [Круг площадью 3,14159265358979]|


Глубина 1:
| NULL  [Квадрат площадью 9,8596]        NULL |
| NULL  [Квадрат площадью 9,8596]        NULL |
| NULL   NULL    NULL |


Глубина 2:
| NULL   NULL    NULL |
| NULL   NULL    NULL |
| NULL   NULL    NULL |




                SimpleStack<Figure>
[Круг площадью 3,14159265358979]
[Квадрат площадью 9,8596]
[Прямоугольник площадью 9]

После сортировки:

[Круг площадью 3,14159265358979]
[Прямоугольник площадью 9]
[Квадрат площадью 9,8596]
```