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

Факультет «Информатика и управление»

Кафедра «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчет по лабораторной работе №3
«Разработать программу реализующую работу с коллекциями»

Выполнил:
студент группы ИУ5-31Б
Гапчук Людмила
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Осликов С.П.
Подпись и дата:

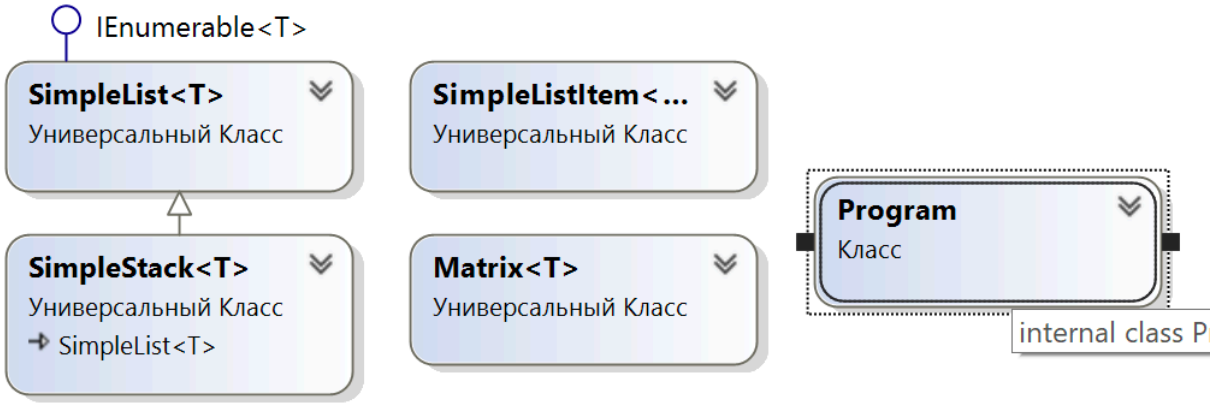
г. Москва, 2019 г.

Задание лабораторной работы

Разработать программу (отдельный проект в рамках решения включающего проекты ЛР1 и ЛР2), реализующую работу с коллекциями.

1. Программа должна быть разработана в виде консольного приложения на языке C#.
2. Объекты классов «Прямоугольник», «Квадрат», «Круг» использовать из проекта лабораторной работы №2.
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 (типы) Прямоугольник, Квадрат, Круг, в коллекцию. Вывести в цикле содержимое площади элементов 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

SimpleList.cs

```
using System;
using System.Collections.Generic;
namespace Lab3
{
   public class SimpleListItem<T>
   {
      public T data { get; set; }
      public SimpleListItem<T> next { get; set; }
      public SimpleListItem(T param)
      {
         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;
      }
   }
}
```

SimpleStack.cs

```
using System;
namespace Lab3
{
   public class SimpleStack<T> : SimpleList<T> where T : IComparable
   {
      public void Push(T element)
      {
         Add(element);
      }
      public T Pop()
      {
         T current = default(T);
         if (Count == 0) return current;
         if (Count == 1)
         {
            current = first.data;
            first = null;
            last = null;
         }
         else
         {
            SimpleListItem<T> newLast = GetItem(Count - 2);
            current = newLast.next.data;
            last = newLast;
            newLast.next = null;
         }
         Count--;
         return current;
      }
   }
}
```

SparseMatrix.cs

```
using System;
using System.Collections.Generic;
using System.Text;
namespace Lab3
{
   public class Matrix<T>
   {
```

```
Dictionary<string, T> _matrix = new Dictionary<string, T>();
int maxX;
int maxY;
int maxZ;
T nullElement;
public Matrix(int px, int py, int pz, T nullElementParam)
{
   this.maxX = px;
   this.maxY = py;
   this.maxZ = pz;
   this.nullElement = nullElementParam;
}
public T this[int x, int y, int z]
{
   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)
{
   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 || y >= this.maxZ) throw new Exception("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($"{k + 1}-ая плоскость матрицы \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[i, j, k] == default) b.Append("-");
            else b.Append(this[i, j, k].ToString());
         }
         b.Append("]\n");
      }
      b.Append("\n");
   }
```

```
            return b.ToString();
        }

    }
}
```

Program.cs

```
using Lab2;
using System;
using System.Collections;
using System.Collections.Generic;
using System.Text;
namespace Lab3
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Title = "Алехин Сергей ИУ5-31Б";
            Console.OutputEncoding = Encoding.UTF8;
            Console.WriteLine("Начальные данные:");
            double dWidth = 6;
            double dHeight = 3;
            double dSide = 4;
            double dRadius = 5;
            Console.WriteLine($"Прямоугольник: ширина = {dWidth}, длина = {dHeight}");
            var rectangle = new Rectangle(dWidth, dHeight);
            Console.WriteLine($"Квадрат: сторона = {dSide}");
            var square = new Square(dSide);
            Console.WriteLine($"Круг: радиус = {dRadius}");
            var circle = new Circle(dRadius);
            Console.WriteLine();
            Console.WriteLine("ArrayList:");
            var geometricFiguresArrayList = new ArrayList
            {
                rectangle, square, circle
            };
            foreach (var currentGeometricFiguresArrayList in geometricFiguresArrayList)
            {
                Console.WriteLine(currentGeometricFiguresArrayList.ToString());
            }
            Console.WriteLine();
            Console.WriteLine("Sort List:");
            var geometricFiguresList = new List<GeometricFigure>
            {
                rectangle, square, circle
            };
            geometricFiguresList.Sort();
            foreach (var currentGeometricFiguresList in geometricFiguresList)
            {
                Console.WriteLine(currentGeometricFiguresList.ToString());
            }
            Console.WriteLine();
            Matrix<GeometricFigure> cube = new Matrix<GeometricFigure>(3, 3, 3, null);
            cube[1, 1, 1] = rectangle;
            cube[2, 2, 2] = square;
            cube[0, 0, 0] = circle;
            Console.WriteLine(cube.ToString());
            Console.WriteLine();
            Console.WriteLine("SimpleStack:");
            var geometricFiguresSimpleStack = new SimpleStack<GeometricFigure>();
```

```
            geometricFiguresSimpleStack.Push(rectangle);
            geometricFiguresSimpleStack.Push(square);
            geometricFiguresSimpleStack.Push(circle);
            while (geometricFiguresSimpleStack.Count > 0)
            {
                Console.WriteLine(geometricFiguresSimpleStack.Pop());
            }
            Console.WriteLine();

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

Примеры работы программы

```
Начальные данные:
Прямоугольник: ширина = 6, длина = 3
Квадрат: сторона = 4
Круг: радиус = 5

ArrayList:
Площадь Прямоугольник = 18
Площадь Квадрат = 16
Площадь Круг = 78,53981633974483

Sort List:
Площадь Квадрат = 16
Площадь Прямоугольник = 18
Площадь Круг = 78,53981633974483

1-ая плоскость матрицы
[Площадь Круг = 78,53981633974483       -       -]
[-      -       -]
[-      -       -]

2-ая плоскость матрицы
[-      -       -]
[-      Площадь Прямоугольник = 18      -]
[-      -       -]

3-ая плоскость матрицы
[-      -       -]
[-      -       -]
[-      -       Площадь Квадрат = 16]



SimpleStack:
Площадь Круг = 78,53981633974483
Площадь Квадрат = 16
Площадь Прямоугольник = 18
```