**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: |  | Проверил: |
| студент группы ИУ5-35б |  | преподаватель каф. ИУ5 |
| Ширшов Артём |  |  |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2020 г.

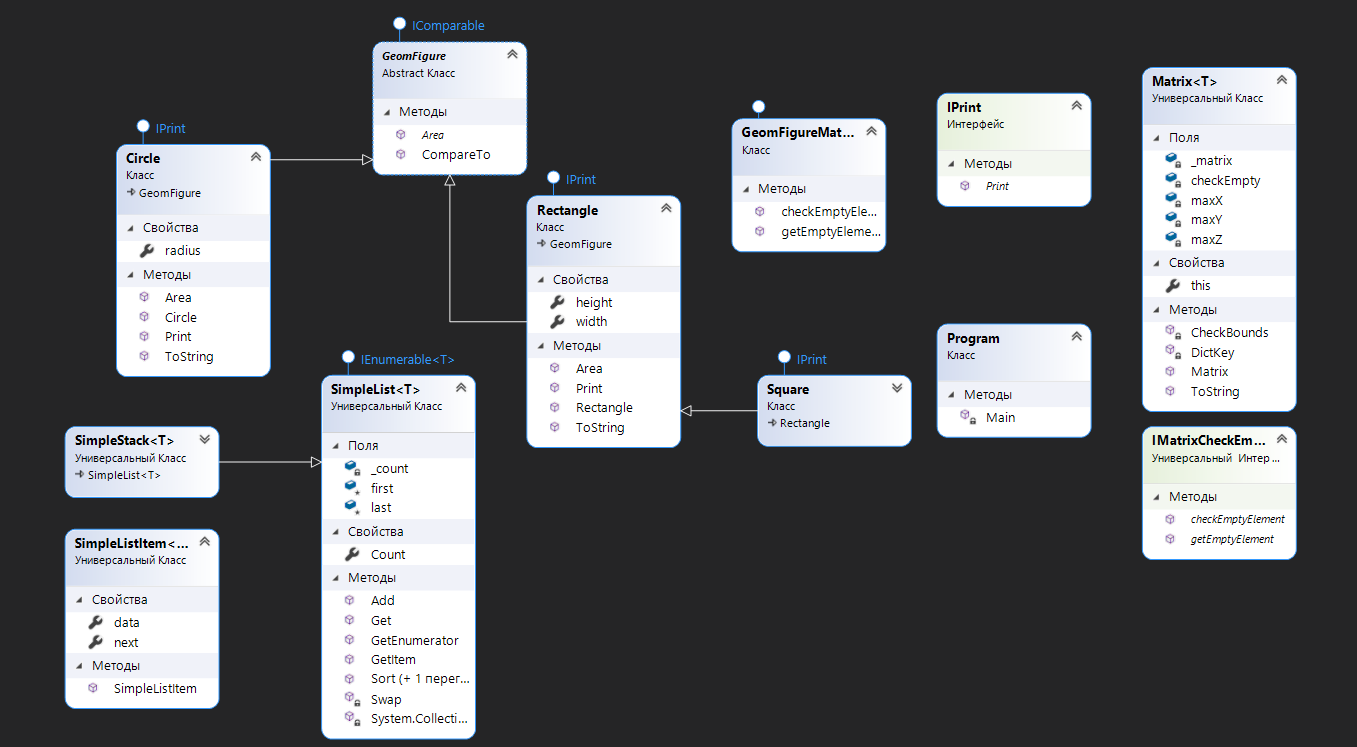
# Задание.

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.

Пример работы класса SimpleStack реализовать на основе геометрических фигур.

# Диаграмма классов



# Текст программы

GeomFigures.cs

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace lab3

{

interface IPrint

{

void Print();

}

public abstract class GeomFigure: IComparable

{

public abstract double Area();

public int CompareTo(object o)

{

GeomFigure p= o as GeomFigure;

if(this.Area()>p.Area())

return 1;

else if (this.Area()==p.Area())

return 0;

else

return -1;

}

}

class Rectangle : GeomFigure, IPrint

{

public double width { get; set; }

public double height { get; set; }

public Rectangle(double width, double height)

{

this.width = width;

this.height = height;

}

public override double Area()

{

return width \* height;

}

public override string ToString()

{

return $"Ширина:{width} Высота:{height} Площадь прямоугольника:{Area()}";

}

public void Print()

{

Console.WriteLine(this.ToString());

}

}

class Square : Rectangle, IPrint

{

public Square(double width) : base(width, width)

{

}

public override double Area()

{

return width \* width;

}

public override string ToString()

{

return $"Длина стороны:{width} Площадь квадрата:{Area()}";

}

new public void Print()

{

Console.WriteLine(this.ToString());

}

}

class Circle : GeomFigure, IPrint

{

public double radius { get; set; }

public Circle(double radius)

{

this.radius = radius;

}

public override double Area()

{

return Math.PI \* radius \* radius;

}

public override string ToString()

{

return $"Радиус:{radius} Площадь круга:{Area()}";

}

public void Print()

{

Console.WriteLine(this.ToString());

}

}

}

SimpleListItem.cs

using System;

using System.Collections;

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;

}

}

}

SimpleList.cs

using System;

using System.Collections;

using System.Collections.Generic;

namespace lab3{

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>Count)

{

throw new Exception ("Выход за границу массива");

}

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;

}

System.Collections.IEnumerator System.Collections.IEnumerable.GetEnumerator()

{

return GetEnumerator();

}

public IEnumerator<T> GetEnumerator()

{

SimpleListItem<T> current =this.first;

while (current!= null){

yield return current.data;

current=current.next;

}

}

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;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace lab3{

public class SimpleStack<T>:SimpleList<T> where T: IComparable{

public void push(T element)

{

this.Add(element);

}

public T pop(){

T result =default(T);

if(this.Count==0) return result;

if(this.Count==1)

{

result=this.first.data;

this.first=null;

this.last=null;

}

else

{

SimpleListItem<T> newLast = this.GetItem(this.Count - 2);

result = newLast.next.data;

this.last=newLast;

newLast.next=null;

}

this.Count--;

return result;

}

}

}

Program.cs

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace lab3

{

class Program

{

static void Main(string[] args)

{

Rectangle rectangle=new Rectangle(2,5);

Square square=new Square(4);

Circle circle =new Circle(4);

System.Console.WriteLine("\nArrayList");

ArrayList arrayList= new ArrayList();

arrayList.Add(circle);

arrayList.Add(rectangle);

arrayList.Add(square);

foreach (GeomFigure o in arrayList){

Console.WriteLine(o.ToString());

}

System.Console.WriteLine("\nСорт.ArrayList");

arrayList.Sort();

foreach (GeomFigure o in arrayList){

Console.WriteLine(o.ToString());

}

System.Console.WriteLine("\nЛист");

List<GeomFigure> FiguresList=new List<GeomFigure>();

FiguresList.Add(circle);

FiguresList.Add(rectangle);

FiguresList.Add(square);

foreach (GeomFigure o in FiguresList){

Console.WriteLine(o.ToString());

}

System.Console.WriteLine("\nСорт.Лист");

FiguresList.Sort();

foreach (GeomFigure o in FiguresList){

Console.WriteLine(o.ToString());

}

System.Console.WriteLine("\nМатрица");

Matrix <GeomFigure> matrix = new Matrix<GeomFigure>(3,3,3,new GeomFigureMatrixCheckEmpty());

matrix[0, 0,0] = rectangle;

matrix[1,1,1] = square;

matrix[2,2,2]=circle;

System.Console.WriteLine(matrix.ToString());

System.Console.WriteLine( "\nSimpleList");

SimpleList<GeomFigure> simpleList = new SimpleList<GeomFigure>();

simpleList.Add(circle);

simpleList.Add(rectangle);

simpleList.Add(square);

foreach(var x in simpleList)

{

System.Console.WriteLine(x);

}

System.Console.WriteLine("\nСорт.SimpleList");

simpleList.Sort();

foreach(var x in simpleList)

{

System.Console.WriteLine(x);

}

System.Console.WriteLine("\nSimpleStack");

SimpleStack<GeomFigure> simplestack = new SimpleStack<GeomFigure>();

simplestack.push(circle);

simplestack.push(rectangle);

simplestack.push(square);

foreach(var x in simplestack)

{

System.Console.WriteLine(x);

}

System.Console.WriteLine("\nSimpleStack popping out");

while (simplestack.Count > 0)

{

GeomFigure f = simplestack.pop();

Console.WriteLine(f);

}

}

}

}

# Результаты

