**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчет по лабораторной работе №3

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-33Б |  | преподаватель каф. ИУ5 |
| Ветошкин Артём |  | Гапанюк Ю.Е. |
| Подпись и дата: |  | Подпись и дата: |

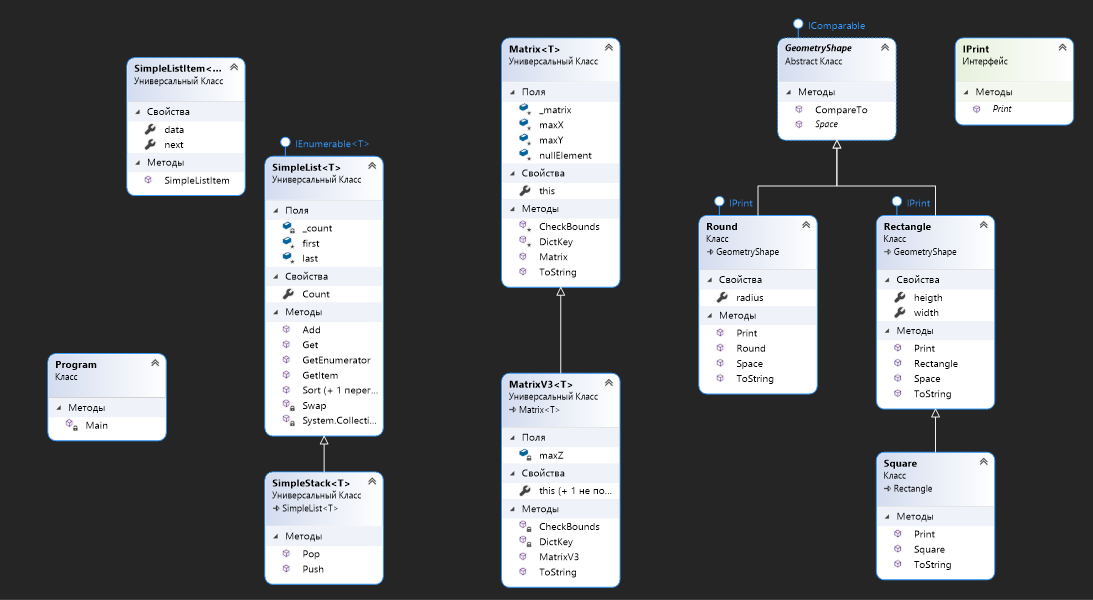
Москва, 2020 г.

**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 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**

Файл Program.cs:

using System;

using System.Collections;

using System.Collections.Generic;

using SimpleStackProject;

using Geometry;

using System.Security.Cryptography.X509Certificates;

namespace IT2 {

class Program {

static void Main(string[] args) {

Console.WriteLine("Hello World!");

Rectangle rectangle = new Rectangle(23, 48.0);

Square square = new Square(95);

Round round = new Round(2);

rectangle.Print();

square.Print();

round.Print();

ArrayList arr = new ArrayList { rectangle, square, round };

arr.Sort();

Console.WriteLine("\nArray:");

foreach(var el in arr) {

Console.WriteLine(el.ToString());

}

List<GeometryShape> list = new List<GeometryShape>

{ rectangle, square, round };

list.Sort();

Console.WriteLine("\nList:");

foreach (var el in list) {

Console.WriteLine(el.ToString());

}

SimpleStack<GeometryShape> stack = new SimpleStack<GeometryShape>

{ rectangle, square, round };

stack.Push(rectangle);

stack.Push(round);

stack.Pop();

Console.WriteLine("\nStack:");

foreach (var el in stack) {

Console.WriteLine(el.ToString());

}

SparseMatrix.MatrixV3<GeometryShape> matrix

= new SparseMatrix.MatrixV3<GeometryShape>(3, 3, 3);

matrix[0, 0, 0] = rectangle;

matrix[1, 0, 0] = square;

matrix[2, 0, 0] = round;

Console.WriteLine("\nMatrix:");

Console.WriteLine(matrix.ToString());

}

}

}

Файл Geometry.cs:

using System;

namespace Geometry {

abstract class GeometryShape : IComparable {

public abstract double Space();

public int CompareTo(object? obj) {

if (obj == null) return 1;

GeometryShape shape = obj as GeometryShape;

if (shape != null) {

return Space().CompareTo(shape.Space());

}

throw new ArgumentException("Object is not a GeometryShapee");

}

}

interface IPrint {

public void Print();

}

class Rectangle : GeometryShape, IPrint {

public Rectangle(double a\_width, double a\_height) {

width = a\_width;

heigth = a\_height;

}

public override double Space() {

return width \* heigth;

}

public void Print() {

Console.WriteLine("Restangle: " + ToString());

}

public override string ToString() {

return "widht = " + width.ToString() + " height = "

+ heigth.ToString() + " space = " + Space().ToString();

}

public double width { get; set; }

public double heigth { get; set; }

}

class Square : Rectangle {

public Square(double a)

: base(a, a) { }

public new void Print() {

Console.WriteLine("Square: " + ToString());

}

public override string ToString() {

return "line = " + width.ToString() + " space = " + Space().ToString();

}

}

class Round : GeometryShape, IPrint {

public Round(double a\_radius) {

radius = a\_radius;

}

public override double Space() {

return Math.PI \* radius \* radius;

}

public void Print() {

Console.WriteLine("Round: " + ToString());

}

public override string ToString() {

return "radius = " + radius.ToString() + " space = " + Space().ToString();

}

public double radius { get; set; }

}

}

Файл SparseMatrix.cs:

using System;

using System.Runtime.CompilerServices;

using System.Text;

namespace SparseMatrix {

public class MatrixV3<T>: Matrix<T> {

int maxZ;

public MatrixV3(int px, int py, int pz, T nullElementParam = default)

: base(px, py, nullElementParam) {

maxZ = pz;

}

void CheckBounds(int x, int y, int z) {

CheckBounds(x, y);

if (z < 0 || z >= maxZ)

throw new Exception("z=" + z + " выходит за границы");

}

string DictKey(int x, int y, int z) {

return DictKey(x, y) + "\_" + z.ToString();

}

public T this[int x, int y, int z] {

get {

CheckBounds(x, y, z);

string key = DictKey(x, y, z);

if (\_matrix.ContainsKey(key)) {

return \_matrix[key];

}

else {

return nullElement;

}

}

set {

CheckBounds(x, y, z);

string key = DictKey(x, y, z);

\_matrix.Add(key, value);

}

}

public new T this[int x, int y] {

get {

throw new NotSupportedException("Метод не поддерживается.");

}

set {

throw new NotSupportedException("Метод не поддерживается.");

}

}

public override string ToString() {

StringBuilder b = new StringBuilder();

for (int k = 0; k < maxZ; k++) {

b.Append("[");

for (int j = 0; j < maxY; j++) {

if (j > 0) b.Append(",\t");

b.Append("[");

for (int i = 0; i < maxX; i++) {

if (i > 0) b.Append(";\t");

b.Append(this[i, j, k]?.ToString() ?? "null");

}

b.Append("]");

}

b.Append("]\n");

}

return b.ToString();

}

}

}

Файл SimpleStack.cs:

using System;

using System.Security.Cryptography;

using SimpleListProject;

namespace SimpleStackProject {

class SimpleStack<T> : SimpleList<T>

where T : IComparable {

public void Push(T el) {

Add(el);

}

public object Pop() {

var tmp = first;

while (tmp != null && tmp.next != last) tmp = tmp.next;

last = tmp;

T data = last.next.data;

last.next = null;

Count--;

return data;

}

}

}

**Анализ результатов**

