|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**  **«Московский государственный технический университет имени Н.Э. Баумана**  **(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ                              Информатика и системы управления

КАФЕДРА                  Системы обработки информации и управления

**Лабораторная работа №3**

**По курсу «Базовые компоненты интернет-технологий»**

Подготовила:

Студентка группы ИУ5-35Б.

Очеретная С.В.

18.11.2020

Проверил:

Преподаватель кафедры ИУ5  
Гапанюк Ю.Е.

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.

Текст программы

Program.cs

using System;

using System.Linq;

using System.Text; //для StringBuilder

using System.Collections;

using System.Collections.Generic;

namespace LABA3\_

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Hello World!");

Rectangle rect = new Rectangle(3, 4);

Square square = new Square(3);

Round round = new Round(10);

string printrect = rect.ToString();

Console.Write("{0} \n", rect.Area());

Console.Write("{0} \n", printrect);

Console.Write("{0} \n", square.ToString());

Console.Write("{0} \n", round.ToString());

rect.Print();

square.Print();

round.Print();

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

figure.Add(rect);

figure.Add(square);

figure.Add(round);

Console.Write("\nИсходный массив: \n");

foreach (var i in figure) Console.Write("{0} \n", i);

Console.Write("Отсортированный массив: \n");

figure.Sort();

foreach (var i in figure) Console.Write("{0} \n", i);

ArrayList list = new ArrayList();

list.Add(rect);

list.Add(square);

list.Add(round);

Console.Write("\nИсходная коллекция: \n");

foreach (var i in list) Console.Write("{0} \n", i);

Console.Write("Отсортированная коллекция: \n");

figure.Sort();

foreach (var i in list) Console.Write("{0} \n", i);

List<GeomFigure> list2 = new List<GeomFigure>();

list2.Add(rect);

list2.Add(square);

list2.Add(round);

Console.Write("\nИсходная коллекция класса Геометрическая фигура: \n");

foreach (var i in list2) Console.Write("{0} \n", i);

Console.Write("Отсортированная коллекция класса Геометрическая фигура: \n");

figure.Sort();

foreach (var i in list2) Console.Write("{0} \n", i);

Matrix<GeomFigure> matrix = new Matrix<GeomFigure>(3, 3, 2, new FigureMatrixCheckEmpty());

for (int i = 0; i < 3; i++)

{

for (int j = 0; j < 2; j++)

{

for (int k = 0; k < 2; k++)

{

Random rand = new Random();

int val = rand.Next(1, 5);

Square square1 = new Square(val);

matrix[i, j, k] = square1;

}

}

}

Console.WriteLine(matrix.ToString());

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

stack.Push(rect);

stack.Push(square);

stack.Push(round);

while (stack.Count() > 0)

{

GeomFigure fig = stack.Pop();

Console.Write("{0} \n", fig);

}

}

}

}

Classes.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace LABA3\_

{

interface IPrint

{

void Print();

}

public interface IMatrixCheckEmpty<T>

{

T getEmptyElement();

bool checkEmptyElement(T element);

}

class FigureMatrixCheckEmpty : IMatrixCheckEmpty<GeomFigure>

{

public GeomFigure getEmptyElement() { return null; }

public bool checkEmptyElement(GeomFigure element) { return element == null; }

}

abstract class GeomFigure : IPrint, IComparable

{

public abstract double Area();

public string Type

{

get { return this.type; }

protected set { this.type = value; }

}

public override string ToString()

{

string param = this.type + " площадью: " + this.Area();

return param;

}

string type;

public void Print()

{

Console.Write("{0} \n", this.ToString());

}

public int CompareTo(object obj)

{

GeomFigure f = (GeomFigure)obj;

if (f.Area() < this.Area()) return 1;

else if (f.Area() == this.Area()) return 0;

else return -1;

}

}

class Rectangle : GeomFigure

{

private double width, height;

public double GetWidth() { return this.width; }

public Rectangle(double width, double height)

{

this.width = width;

this.height = height;

this.Type = "Прямоугольник";

}

public override double Area()

{

return this.width \* this.height;

}

}

class Square : Rectangle

{

public Square(double side) : base(side, side) { this.Type = "Квадрат"; }

}

class Round : GeomFigure

{

private double radius;

public Round(double radius)

{

this.radius = radius;

this.Type = "Круг";

}

public override double Area()

{

return Math.Round(this.radius \* this.radius \* Math.PI, 3);

}

}

}

Matrix.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace LABA3\_

{

public class Matrix<T>

{

int x, y, z; //индексы в матрице

Dictionary<string, T> matrix = new Dictionary<string, T>(); //ключ + значение

IMatrixCheckEmpty<T> checkEmpty;

public Matrix(int px, int py, int pz, IMatrixCheckEmpty<T> EmptyParam)

{

this.x = px;

this.y = py;

this.z = pz;

this.checkEmpty = EmptyParam;

}

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

{

set

{

CheckBounds(x, y, z);

this.matrix.Add(DictKey(x, y, z), value);

}

get

{

CheckBounds(x, y, z);

if (this.matrix.ContainsKey(DictKey(x, y, z))) return this.matrix[DictKey(x, y, z)];

else return this.checkEmpty.getEmptyElement();

}

}

void CheckBounds(int px, int py, int pz)

{

if (px < 0 || px >= this.x)

throw new ArgumentOutOfRangeException("Выход координаты х = " + x + " за границы матрицы");

if (py < 0 || py >= this.y)

throw new ArgumentOutOfRangeException("Выход координаты y = " + y + " за границы матрицы");

if (pz < 0 || pz >= this.z)

throw new ArgumentOutOfRangeException("Выход координаты z = " + z + " за границы матрицы");

}

string DictKey(int x, int y, int z) { return "[" + x.ToString() + "][" + y.ToString() + "][" + z.ToString() + "]"; }

public override string ToString()

{

StringBuilder str = new StringBuilder(); //применяем этот класс, чтобы при контакенации не создавать новый объект, а добавлять к старому

for (int i = 0; i < this.x; i++)

{

str.Append("\n");

for (int j = 0; j < this.y; j++)

{

str.Append("\n");

for (int k = 0; k < this.z; k++)

{

if (!this.checkEmpty.checkEmptyElement(this[i, j, k])) str.Append(this[i, j, k].ToString());

else str.Append("\t\_\t");

str.Append(" ");

}

}

}

return str.ToString();

}

}

}

Stack.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace LABA3\_

{

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

{

public void Push(T elem) { Add(elem); }

public T Pop()

{

if (this.Count == 0) return default(T);

if (this.Count == 1)

{

T result = this.first.data;

this.first = null;

this.last = null;

this.Count--;

return result;

}

else

{

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

T result = preLast.next.data;

this.last = preLast;

preLast.next = null;

this.Count--;

return result;

}

}

}

}

Результаты работы программы

