**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

«Работа с коллекциями»

|  |  |  |
| --- | --- | --- |
| Выполнила: |  | Проверил: |
| студентка группы ИУ5-33 |  | преподаватель каф. ИУ5 |
| Беспалова У.А. |  | Гапанюк Ю.Е. |
| Подпись и дата:  15.10.2018 |  | Подпись и дата: |

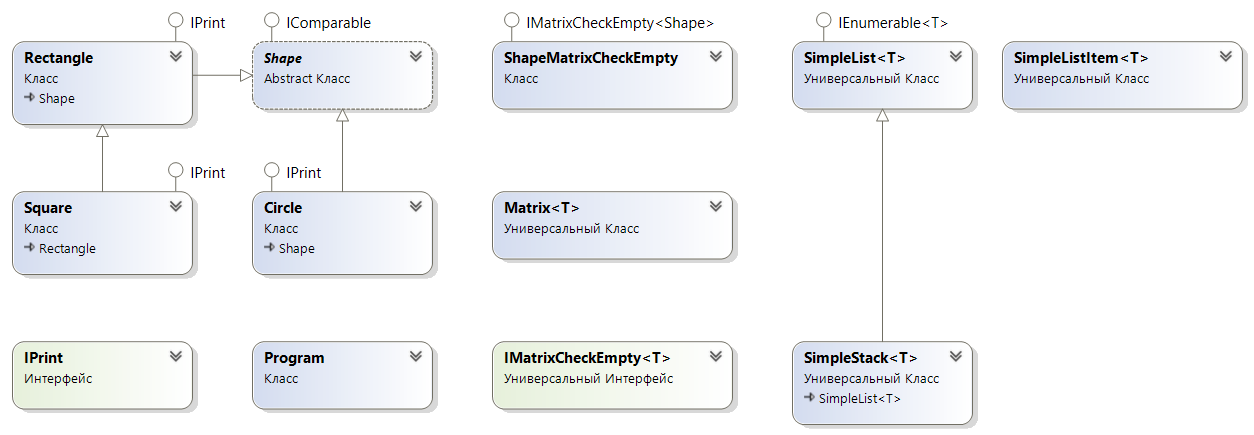
Москва, 2018 г.

1. Содержание задания

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 (разобранного в пособии). Необходимо добавить в класс методы:
   1. public void Push(T element) – добавление в стек;
   2. public T Pop() – чтение с удалением из стека.
8. Пример работы класса SimpleStack реализовать на основе геометрических фигур.

1. Диаграмма классов



1. Текст программы

Класс Program

using System;

using System.Collections;

using System.Collections.Generic;

using System.Text;

using System.Linq;

using System.Reflection;

namespace лаба3

{

class Program

{

static void Main(string[] args)

{

Rectangle rect = new Rectangle(3, 4); //создали объект класса Прямоугольник размером 3на4 и так далее

Square sq = new Square(4);

Circle circle = new Circle(3);

rect.Print(); //вывели ответ

sq.Print();

circle.Print();

ArrayList arr = new ArrayList(); //пустая коллекция

arr.Add(rect);

arr.Add(circle);

arr.Add(sq);

arr.Sort(); //отсортировали

Console.WriteLine("\n\nОтсортированный ArrayList:\n");

foreach (Figure ivan in arr) //все элементы имеют тип Figure

{

Console.WriteLine(ivan);

}

List<Figure> list = new List<Figure>();

list.Add(rect);

list.Add(circle);

list.Add(sq);

list.Sort();

Console.WriteLine("\n\nОтсортированный List:\n");

foreach (Figure ivan in list)

{

Console.WriteLine(ivan);

}

Console.WriteLine("\n\nМатрица:\n");

Matrix<Figure> cube = new Matrix<Figure>(2, 2, 2, null); //создали кубик с ребром 2 ячейки

cube[0, 0, 1] = rect;

cube[0, 1, 1] = sq;

cube[1, 1, 1] = circle;

Console.WriteLine(cube.ToString());

SimpleList<Figure> sl = new SimpleList<Figure>

{

rect, circle, sq

};

sl.Sort();

Console.WriteLine("\nОтсортированный SimpleList\n");

foreach (var i in sl)

{

Console.WriteLine(i.ToString());

}

SimpleStack<Figure> st = new SimpleStack<Figure>();

st.Push(rect);

st.Push(circle);

st.Push(sq);

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

foreach (var s in st)

{

Console.WriteLine(s.ToString());

}

Figure P = st.Pop();

Console.WriteLine("\nУдалено " + P.ToString() + "\n");

foreach (var s in st)

{

Console.WriteLine(s.ToString());

}

Console.ReadLine();

}

}

}

Класс Figure

abstract class Figure : IComparable //геометрическая фигура (+реализует интерфейс сравнения)

{

string \_Type;//название типа

public string Type //свойство

{

get

{

return this.\_Type;

}

protected set

{

this.\_Type = value;

}

}

public virtual double Area() { return 0; } //заготовка для площади

public override string ToString() //вывод ответа

{

return ("Площадь " + this.Type + "а = " + this.Area().ToString());

}

interface IPrint //заготовка для интерфейса

{

void Print();

}

public int CompareTo(object o)

{

Figure c = (Figure)o;//приводим тип

if (c != null)

{

return this.Area().CompareTo(c.Area());

}

else

throw new Exception("Не удалось сравнить два объекта"); //ругаемся, если не удалось привести тип

}

Класс Rectangle

class Rectangle : Figure//прямоугольник наследуется от фигуры (логично)

{

double A, B; //длина и высота

public Rectangle(double a, double b)//конструктор.

{

this.A = a;

this.B = b;

this.Type = "Прямоугольник";

}

public override double Area()//переопределенная функция суммы

{

return this.A \* this.B;

}

public void Print()//реализация интерфейса

{

Console.WriteLine(this.ToString());

}

}

class Square : Rectangle

{

public Square(double size) //присвоили значения сторонам с использованием базового конструктора

: base(size, size)//использовали конструктор базового класса

{

this.Type = "Квадрат";

}

Класс Circle

class Circle : Figure

{

double r;

public Circle(double R)

{

this.r = R;

this.Type = "Круг";

}

public override double Area()

{

return Math.PI \* this.r \* this.r;

}

public void Print()

{

Console.WriteLine(this.ToString());

}

Класс List

public class SimpleList<T> : IEnumerable<T> where T : IComparable

{

protected SimpleListItem<T> First = null;

protected SimpleListItem<T> Last = null;

public int Count { get; protected set; }

public void Add(T elem)//добавление элемента

{

SimpleListItem<T> NewItem = new SimpleListItem<T>(elem); //создали новый элемент

Count++;

if (Last == null) //если ничего не было

{

First = NewItem;

Last = NewItem;

}

else //если был хотя бы один

{

Last.Next = NewItem;

Last = NewItem;

}

}

public SimpleListItem<T> GetItem(int index)

{

if ((index < 0) || (index >= Count)) //проверка границ

{

throw new Exception("Выход за границы списка.");

}

SimpleListItem<T> Current = First;

int i = 0;

while (i < index)

{

Current = Current.Next;

i++;

}

return Current;

}

public T Get(int index)

{

return GetItem(index).Data;

}

public IEnumerator<T> GetEnumerator()

{

SimpleListItem<T> Current = First;

while (Current != null)

{

yield return Current.Data;//вывели все элементы

Current = Current.Next;

}

}

IEnumerator IEnumerable.GetEnumerator()

{

return GetEnumerator();

}

public void Sort()

{

Sort(0, Count - 1);

}

private void Sort(int low, int high) //функция сортировки

{

int i = low, j = high;

T x = Get((low + high) / 2);//би

while (i <= j)

{

while (Get(i).CompareTo(x) < 0) ++i;

while (Get(j).CompareTo(x) > 0) --j;

if (i <= j)

{

Swap(i, j);

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;

}

}

public class SimpleListItem<T>

{

public T Data { get; set; }

public SimpleListItem<T> Next { get; set; }

public SimpleListItem(T param)

{

Data = param;

}

Класс Matrix

public class Matrix<T>

{

Dictionary<string, T> \_matrix = new Dictionary<string, T>(); //словарь для хранения значений

int maxX; //максимальное колво столбцов

int maxY; //максимальное колво строк

int maxZ; //максимальное колво ..понятно чего

T nullEl;

public Matrix(int px, int py, int pz, T param)

{

this.maxX = px;

this.maxY = py;

this.maxZ = pz;

this.nullEl = param;

}

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

{

get

{

CheckBounds(x, y, z);

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

if (this.\_matrix.ContainsKey(key))

{

return this.\_matrix[key];

}

else

{

return this.nullEl;

}

}

set

{

CheckBounds(x, y, z);

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

this.\_matrix.Add(key, value);

}

}

void CheckBounds(int x, int y, int z)//проверка выхода за границы

{

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 || z >= this.maxZ) throw new Exception("z=" + z + " выходит за границы");

}

string FormKey(int x, int y, int z) //формирование ключа

{

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

}

public override string ToString()//приведение к строке

{

StringBuilder b = new StringBuilder();

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

{

b.Append("[");

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

{

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

{

if (i > 0)

b.Append("\t"); //пробел

if (this[i, j, k] != null) b.Append(this[i, j, k].ToString());

}

}

b.Append("]\n");

}

return b.ToString();

}

Класс Stack

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

{

public void Push(T elem)

{

Add(elem);

}

public T Pop()

{

T Result = default(T); //null или 0

if (Count == 0) //если элементов нет

{

return Result; //выводим 0

}

if (Count == 1) //если есть 1 элемент

{

Result = First.Data; //вывели

First = null;//занулили

Last = null;

}

else

{

SimpleListItem<T> NewLast = GetItem(Count - 2);

Result = NewLast.Next.Data;

Last = NewLast;

NewLast.Next = null;

}

Count--;

return Result;

}

1. Экранные формы с примерами выполнения программы

