**Отчёт по лабораторной работе №3  
 по курсу  
 «Базовые компоненты интернет-технологий»**

Выполнил студент:

Бакулин Глеб Георгиевич группа: РТ5-31

Проверил:

к.т.н., доцент

Гапанюк Юрий Евгеньевич

Москва, 2017

**Описание задания лабораторной работы**

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.

**Исходный код программы:**

**Класс GeoFig:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApplication3

{

abstract class GeoFig : IComparable

{

public string Tip

{

get

{

return this.\_Tip;

}

set

{

this.\_Tip = value;

}

}

string \_Tip;

public abstract double Ploshad();

public override string ToString()

{

return this.Tip + " площадью " + this.Ploshad().ToString();

}

public int CompareTo(object obj)

{

GeoFig g = (GeoFig)obj;

if (this.Ploshad() < g.Ploshad()) return -1;

else if (this.Ploshad() == g.Ploshad()) return 0;

else return 1;

}

}

class Pryamougolnik : GeoFig, VivodData

{

double visota;

double shirina;

public Pryamougolnik(double vi, double sh)

{

this.visota = vi;

this.shirina = sh;

this.Tip = "Прямоугольник";

}

public override double Ploshad()

{

double Otvet = this.visota \* this.shirina;

return Otvet;

}

public void Print()

{

Console.WriteLine(this.ToString());

}

}

class Kvadrat : Pryamougolnik

{

public Kvadrat(double dlina)

: base(dlina, dlina)

{

this.Tip = "Квадрат";

}

}

class krug : GeoFig, VivodData

{

public double Radius;

public krug(double Ra)

{

this.Radius = Ra;

this.Tip = "Круг";

}

public override double Ploshad()

{

double Otvet = Math.PI \* (Math.Pow(this.Radius, 2));

return Otvet;

}

public void Print()

{

Console.WriteLine(this.ToString());

}

}

}

**Класс LiILiSt:**

using System;

using System.Collections.Generic;

using System.Collections;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApplication3

{

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 || z >= 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.maxY; 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 (this[i, j, k] != null)

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

else

b.Append("0");

if (i != (maxX - 1)) b.Append(", ");

}

b.Append("]");

}

b.Append("]\n");

}

return b.ToString();

}

}

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; //возврат тек. знач.

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;

}

}

public class SimpleStack<T> : SimpleList<T>

where T : IComparable

{

public void Push(T element)

{

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;

}

}

}

**Класс Program:**

using System;

using System.Collections.Generic;

using System.Collections;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApplication3

{

class Program

{

static void Main(string[] args)

{

Pryamougolnik pryam = new Pryamougolnik(5, 6);

Kvadrat kvad = new Kvadrat(4);

krug kru = new krug(7);

ArrayList al = new ArrayList();

al.Add(kru);

al.Add(pryam);

al.Add(kvad);

Console.WriteLine("Коллекция:");

foreach (var x in al) Console.WriteLine(x);

Console.WriteLine();

Console.WriteLine("Сортировка:");

al.Sort();

foreach (var x in al) Console.WriteLine(x);

Console.WriteLine();

Console.WriteLine("Коллекция класса List<Figure>:");

List<GeoFig> fl = new List<GeoFig>();

fl.Add(kru);

fl.Add(pryam);

fl.Add(kvad);

foreach (var x in fl) Console.WriteLine(x);

Console.WriteLine();

Console.WriteLine("Сортировка:"); fl.Sort();

foreach (var x in fl) Console.WriteLine(x);

Console.WriteLine();

Console.WriteLine("Матрица:");

Matrix<GeoFig> cube = new Matrix<GeoFig>(3, 3, 3, null);

cube[0, 0, 0] = pryam;

cube[1, 1, 1] = kvad;

cube[2, 2, 2] = kru;

Console.WriteLine(cube.ToString());

Console.WriteLine("Стек:");

SimpleStack<GeoFig> Stack1 = new SimpleStack<GeoFig>();

Stack1.Push(pryam);

Stack1.Push(kru);

Stack1.Push(kvad);

while (Stack1.Count > 0)

{

GeoFig fig = Stack1.Pop();

Console.WriteLine(fig);

}

Console.ReadKey();

}

}

}

**Интерфейс VivodData:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApplication3

{

interface VivodData

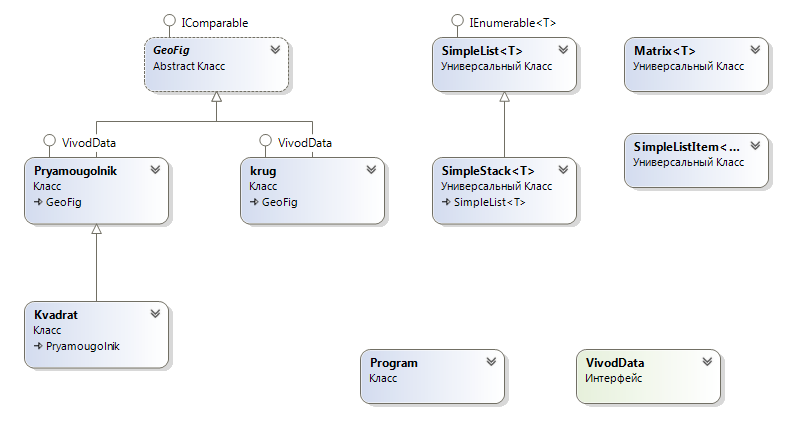
{

void Print();

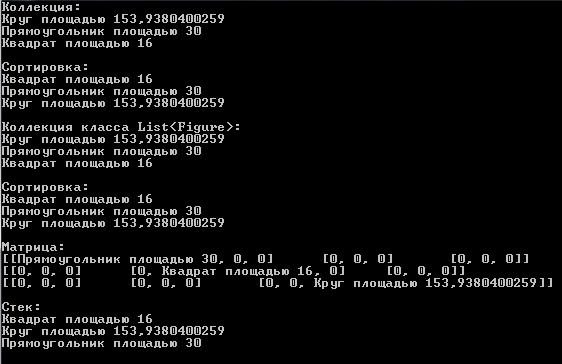
}

}

**Диаграмма классов**

****

**Результаты выполнения программы**

****