**Московский государственный технический университет**

**им. Н.Э. Баумана**

Кафедра ИУ–5

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

Выполнил:

студент гр. ИУ5-32Б

Деркачев А. В.

\_\_\_\_\_\_\_\_

Проверил:

к.т.н., доцент

Гапанюк Ю.Е.

\_\_\_\_\_\_\_\_

**Москва – 2014 г**

1. **Цель выполнения 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. Сохранить объекты в коллекцию. Отсортировать коллекцию. Вывести в цикле содержимое коллекции.

6. Модифицировать класс разреженной матрицы Matrix (представлен в разделе «Вспомогательные материалы для выполнения лабораторных работ») 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 (представлен в разделе 9 «Вспомогательные материалы для выполнения лабораторных работ»). 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.

1. **Листинг программы**

Program.cs

using System;

using System.Collections.Generic;

using System.Collections;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LR3FT

{ class Program

{ static void Main(string[] args)

{ pryamoug p1 = new pryamoug(2, 4);

krug k1 = new krug(2);

quad q1 = new quad(2);

ArrayList al = new ArrayList();

al.Add(111);

al.Add(12.13);

al.Add("aaa");

foreach (object obj in al)

{ Console.WriteLine(obj.ToString()); }

List<GeomFig> gList = new List<GeomFig>();

if (p1 is GeomFig)

{ gList.Add(p1); }

if (k1 is GeomFig)

{ gList.Add(k1); }

if (q1 is GeomFig)

{ gList.Add(q1); }

foreach (object o in gList)

{ if (o is GeomFig)

{Console.WriteLine(o.ToString());}}

gList.Sort();

foreach (object o in gList)

{if (o is GeomFig)

{Console.WriteLine(o.ToString());}}

Console.WriteLine("SimpleStack");

simpleStack<string> simplestack1= new simpleStack<string >();

simplestack1.Push("aaaaaa");

simplestack1.Push("bbbbbb");

simplestack1.Push("cccccc");

foreach (object o in simplestack1)

{ Console.WriteLine(o.ToString()); }

simplestack1.Pop(1);

Console.WriteLine(" ");

foreach (object o in simplestack1)

{ Console.WriteLine(o.ToString()); }

Console.WriteLine(" SimpleStack с GeomFig ");

simpleStack<GeomFig> smp2 = new simpleStack<GeomFig>();

smp2.Push(p1);

smp2.Push(q1);

smp2.Push(k1);

foreach (object o in smp2)

{ Console.WriteLine(o.ToString()); }

Console.ReadKey();}}}

GeomFig.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lr2FT

{ public abstract class GeomFig

{ public double s;

public virtual double S()

{ return s; }

public override string ToString()

{return base.ToString() + "";}}}

IPrint.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lr2FT

{ interface IPrint

{void print();}}

krug.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lr2FT

{class krug: GeomFig, IPrint

{private double r;

public double R

{ get { return r; } set { r = value; } }

public krug(double r)

{ this.r = r; }

public override double S()

{s = 3.14\*r\*r;

return base.S();}

public override string ToString()

{return base.ToString()+ "\n" + "Радиус круга: " + r + "\n"+ "Площадь круга" + S() +"\n";}

public void print()

{ Console.WriteLine(this.ToString()); }}}

pryamoug.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lr2FT

{class pryamoug:GeomFig,IPrint

{ public double shir;

public double dlin;

public pryamoug(double dlin, double shir)

{ this.dlin = dlin;

this.shir = shir;}

public override double S()

{ s = dlin \* shir;

return base.S();}

public override string ToString()

{return base.ToString() + "\n" + "Длина прямоугольника: " + dlin + "\n" + "Ширина прямоугольника: " + shir + "\n" + "Площадь прямоугольника: " + S() + "\n"; }

public void print()

{ Console.WriteLine(this.ToString()); }}}

quad.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lr2FT

{ class quad: pryamoug//, IPrint //pryamoug

{ public quad(double s): base(s, s)

{ }

public override string ToString()

{return base.ToString() + "\n" + "Длина квадрата: " + dlin + "\n" + "Площадь квадрата" + S() + "\n";}

public void print()

{ Console.WriteLine(this.ToString()); }}}

simpleStack.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LR3FT

{ class simpleStack<T>: List<T>

{ public void Push(T el)

{ Add(el); }

public void Pop(int n)

{ RemoveAt(n); }}}

Matrix.cs

using System;

using System.Collections.Generic;

using System.Runtime.Serialization.Formatters;

using System.Text;

using System.Linq;

namespace Lab3

{

public class Matrix3D<T>

{

Dictionary<string, T> \_matrix = new Dictionary<string, T>();

int maxX;

int maxY;

int maxZ;

IMatrixCheckEmpty<T> checkEmpty;

public Matrix3D(int px, int py, int pz, IMatrixCheckEmpty<T> checkEmptyParam)

{

this.maxX = px;

this.maxY = py;

this.maxZ = pz;

this.checkEmpty = checkEmptyParam;

}

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.checkEmpty.getEmptyElement();

}

}

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()

{

//Класс StringBuilder используется для построения длинных строк

//Это увеличивает производительность по сравнению с созданием и склеиванием

//большого количества обычных строк

StringBuilder b = new StringBuilder();

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

{

b.Append("\nZ: {k}\n");

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

{

b.Append("[");

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

{

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

if (checkEmpty.checkEmptyElement(this[i, j, k]) != true)

{

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

}

else

{

b.Append(" - ");

}

}

b.Append("]\n");

}

}

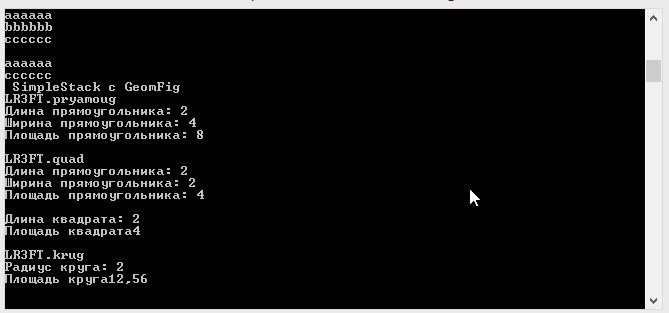
return b.ToString();

}

}

}

1. **Результаты работы программы**



1. **Диаграмма классов**

