МГТУ им. Баумана  
Факультет «Информатика и системы управления»  
Кафедра «Автоматизированные системы обработки информации и управления»  
Дисциплина «Базовые компоненты интернет-технологий»

Отчёт по лабораторной работе №3



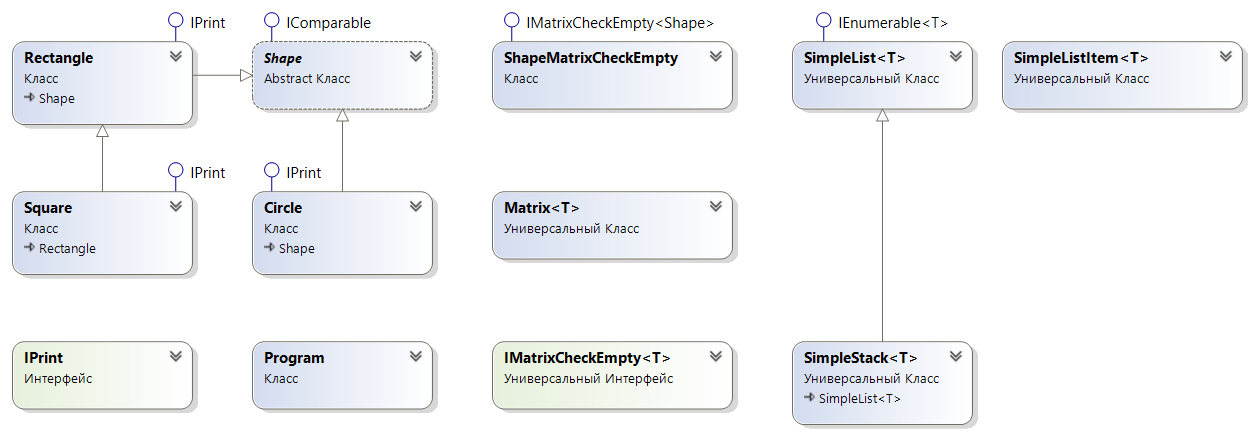
Описание задания:

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:
   * 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.

Диаграмма классов:



Текст программы:

Класс Shape:

using System;

namespace GeometryCollections {

public abstract class Shape: IComparable {

public abstract double Area();

public int CompareTo(object obj) {

Shape s = (Shape)obj;

if (Area() > s.Area()) {

return 1;

}

else if (Area() == s.Area()) {

return 0;

}

else {

return -1;

}

}

}

}

Класс Circle:

using System;

namespace GeometryCollections {

class Circle: Shape, IPrint {

public int Radius { get; set; }

public Circle(int a) {

Radius = a;

}

public override double Area() {

return Math.PI \* Radius \* Radius;

}

public override string ToString() {

return "Круг радиуса " + Radius.ToString() + " и площадью " + Area().ToString();

}

public void Print() {

Console.WriteLine(ToString());

}

}

}

Класс Rectangle:

using System;

namespace GeometryCollections {

class Rectangle : Shape, IPrint {

public int Width { get; set; }

public int Height { get; set; }

public Rectangle(int w, int h) {

Width = w;

Height = h;

}

public override double Area() {

return Width \* Height;

}

public override string ToString() {

return "Прямоугольник с шириной " + Width.ToString() + ", высотой " + Height.ToString() + " и площадью " + Area().ToString();

}

public void Print() {

Console.WriteLine(ToString());

}

}

}

Класс Square:

using System;

namespace GeometryCollections {

class Square: Rectangle, IPrint {

public Square(int side) : base(side, side) {

}

public override string ToString() {

return "Квадрат со стороной " + Width.ToString() + " и площадью " + Area().ToString();

}

public new void Print() {

Console.WriteLine(ToString());

}

}

}

Интерфейс IPrint:

namespace Geometry {

interface IPrint {

void Print();

}

}

Класс SimpleList<T>:

using System;

using System.Collections;

using System.Collections.Generic;

namespace GeometryCollections {

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) {

SimpleListItem<T> NewItem = new SimpleListItem<T>(elem);

Count++;

if (Last == null) {

First = NewItem;

Last = NewItem;

}

else {

Last.Next = NewItem;

Last = NewItem;

}

}

public SimpleListItem<T> GetItem(int index) {

if ((index < 0) || (index >= Count)) {

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;

Current = Current.Next;

}

}

IEnumerator IEnumerable.GetEnumerator() {

return GetEnumerator();

}

public void Sort() {

Sort(0, Count - 1);

}

private void Sort(int low, int high) {

int i = low, j = high;

T x = Get((low + high) / 2);

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;

}

}

}

Класс SimpleListItem<T>:

namespace GeometryCollections {

public class SimpleListItem<T> {

public T Data { get; set; }

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

public SimpleListItem(T param) {

Data = param;

}

}

}

Класс Matrix:

using System;

using System.Collections.Generic;

using System.Text;

namespace GeometryCollections {

class Matrix<T> {

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

int maxX, maxY, maxZ;

IMatrixCheckEmpty<T> CheckEmpty;

public Matrix(int xx, int yy, int zz, IMatrixCheckEmpty<T> CheckEmptyParam) {

maxX = xx;

maxY = yy;

maxZ = zz;

CheckEmpty = CheckEmptyParam;

}

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

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

}

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

if (x < 0 || x >= maxX) {

throw new ArgumentOutOfRangeException("x", "x=" + x.ToString() + " выходит за границы.");

}

if (y < 0 || y >= maxY) {

throw new ArgumentOutOfRangeException("y", "y=" + y.ToString() + " выходит за границы.");

}

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

throw new ArgumentOutOfRangeException("z", "z=" + z.ToString() + " выходит за границы.");

}

}

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

set {

CheckBounds(x, y, z);

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

\_matrix.Add(key, value);

}

get {

CheckBounds(x, y, z);

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

if (\_matrix.ContainsKey(key)) {

return \_matrix[key];

}

else {

return CheckEmpty.GetEmptyElement();

}

}

}

public override string ToString() {

StringBuilder tab = new StringBuilder();

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

tab.Append("Z = " + k.ToString() + "\n\n");

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

tab.Append("[");

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

if (i > 0) {

tab.Append('\t');

}

if (!CheckEmpty.CheckEmptyElement(this[i, j, k])) {

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

}

else {

tab.Append(" . ");

}

}

tab.Append("]\n");

}

if (k < maxZ - 1) {

tab.Append("\n==========================================================================\n\n");

}

}

return tab.ToString();

}

}

}

Интерфейс IMatrixCheckEmpty<T>:

namespace GeometryCollections {

interface IMatrixCheckEmpty<T> {

T GetEmptyElement();

bool CheckEmptyElement(T element);

}

}

Класс ShapeMatrixCheckEmpty:

namespace GeometryCollections {

class ShapeMatrixCheckEmpty: IMatrixCheckEmpty<Shape> {

public Shape GetEmptyElement() {

return null;

}

public bool CheckEmptyElement(Shape element) {

if (element == null) {

return true;

}

else {

return false;

}

}

}

}

Класс SimpleStack<T>:

using System;

namespace GeometryCollections {

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

public void Push(T elem) {

Add(elem);

}

public T Pop() {

T Result = default(T);

if (Count == 0) {

return Result;

}

if (Count == 1) {

Result = First.Data;

First = null;

Last = null;

}

else {

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

Result = NewLast.Next.Data;

Last = NewLast;

NewLast.Next = null;

}

Count--;

return Result;

}

}

}

Класс Program: using System;

using System.Collections.Generic;

using System.Collections;

namespace GeometryCollections {

class Program {

static void Main(string[] args) {

Shape c1 = new Circle(10), c2 = new Circle(7);

Shape r1 = new Rectangle(4, 15), r2 = new Rectangle(20, 5);

Shape s1 = new Square(8), s2 = new Square(6);

Console.WriteLine("ArrayList\n");

ArrayList arr = new ArrayList {

c1, r1, s1, c2, r2, s2

};

arr.Sort();

foreach (Shape s in arr) {

Console.WriteLine(s.ToString());

}

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

List<Shape> list = new List<Shape> {

c1, r1, s1, c2, r2, s2

};

list.Sort();

foreach (Shape s in list) {

Console.WriteLine(s.ToString());

}

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

Matrix<Shape> matr = new Matrix<Shape>(3, 3, 2, new ShapeMatrixCheckEmpty());

matr[0, 0, 0] = c1;

matr[1, 1, 0] = s1;

matr[2, 2, 0] = r1;

matr[0, 0, 1] = c2;

matr[1, 1, 1] = s2;

matr[2, 2, 1] = r2;

Console.WriteLine(matr.ToString());

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

SimpleList<Shape> slist = new SimpleList<Shape> {

c1, s1, r1, c2, s2, r2

};

slist.Sort();

foreach (var s in slist) {

Console.WriteLine(s.ToString());

}

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

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

stack.Push(s1);

stack.Push(c1);

stack.Push(c2);

stack.Push(r2);

stack.Push(r1);

foreach (var s in stack) {

Console.WriteLine(s.ToString());

}

Shape popped = stack.Pop();

Console.WriteLine("\nИз стека удалён " + popped.ToString() + "\n");

foreach (var s in stack) {

Console.WriteLine(s.ToString());

}

Console.ReadLine();

}

}

}

Экранные формы:

