|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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)** |

Базовые компоненты интернет технологий

Отчет по лабораторной работе №3

Студент: Рябкин А. В.

Группа: ИУ5Ц-51Б

Преподаватель: Гапанюк Ю. Е.

2020 г.

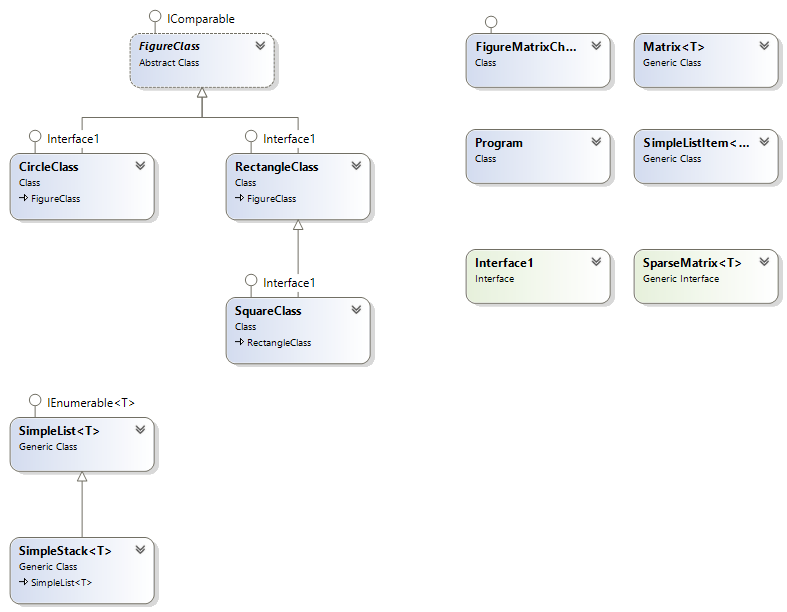
**Лабораторная работа №3**

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.

Диаграмма классов:



Текст программы:

Файл CircleClass.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_3lab

{

class CircleClass : FigureClass, Interface1

{

private double \_Radius;

public double Radius

{

get { return \_Radius; }

set { this.\_Radius = value; }

}

public CircleClass(double radius)

{

this.Radius = radius;

this.Type = "Круг";

}

public override double Area()

{

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

}

public void Print()

{

Console.WriteLine(this.ToString());

Console.WriteLine("Радиус: " + this.Radius);

}

}

}

Файл: FigureClass.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_3lab

{

abstract partial class FigureClass : IComparable

{

private string \_Type;

public string Type

{

get { return this.\_Type; }

set { this.\_Type = value; }

}

abstract public double Area();

public override string ToString()

{

//Console.WriteLine(this.Type + ":");

return this.Type + " с площадью " + this.Area().ToString();

}

public int CompareTo(object obj)

{

FigureClass p = (FigureClass)obj;

if (this.Area() > p.Area())

{

return 1;

}

else if (this.Area() < p.Area())

{

return -1;

}

else if (this.Area() == p.Area())

{

return 0;

}

else

{

throw new NotImplementedException();

}

}

}

}

Файл: FigureMatrixCheckEmpty.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_3lab

{

partial class FigureMatrixCheckEmpty : SparseMatrix<FigureClass>

{

public FigureClass getEmptyElement()

{

return null;

}

public bool CheckEmptyElement(FigureClass element)

{

bool Result = false;

if (element == null)

{

Result = true;

}

return Result;

}

}

}

Файл: Interface1.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_3lab

{

interface Interface1

{

void Print();

}

}

Файл: Matrix.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_3lab

{

partial class Matrix<T>

{

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

int MaxX;

int MaxY;

SparseMatrix<T> checkEmpty;

public Matrix(int x, int y, SparseMatrix<T> param)

{

this.MaxX = x;

this.MaxY = y;

this.checkEmpty = param;

}

///<summary>

///Индексатор для доступа к данным

/// </summary>

public T this[int x, int y]

{

set

{

CheckBounds(x, y);

string key = DictKey(x, y);

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

}

get

{

CheckBounds(x, y);

string key = DictKey(x, y);

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

{

return this.\_matrix[key];

}

else

{

return this.checkEmpty.getEmptyElement();

}

}

}

void CheckBounds(int x, int y)

{

if (x < 0 || x > this.MaxX)

{

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

}

if (y < 0 || y > this.MaxY)

{

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

}

}

string DictKey(int x, int y)

{

return x.ToString() + " " + y.ToString();

}

public override string ToString()

{

StringBuilder b = new StringBuilder();

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

{

b.Append("[");

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

{

if (i > 0)

{

b.Append("\t");

}

//если элемент не пустой

if (!this.checkEmpty.CheckEmptyElement(this[i, j]))

{

//добавить этот элемент, преобразованный в строку

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

}

//иначе добавить "пусто"

else

{

b.Append(" - ");

}

}

b.Append("]\n");

}

return b.ToString();

}

}

}

Файл: Prorgam.cs

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_3lab

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Лабораторная работа №3");

Console.Title = "Рябкин Александр ИУ5Ц-51Б";

RectangleClass rect = new RectangleClass(2, 4);

rect.Print();

SquareClass square = new SquareClass(5);

square.Print();

CircleClass circle = new CircleClass(3);

circle.Print();

ArrayList figures = new ArrayList();

figures.Add(circle);

figures.Add(rect);

figures.Add(square);

Console.WriteLine("\nДо сортировки для ArrayList");

foreach (var i in figures)

{

Console.WriteLine(i);

}

figures.Sort();

Console.WriteLine("\nПосле сортировки для ArrayList");

foreach (var i in figures)

{

Console.WriteLine(i);

}

List<FigureClass> figures1 = new List<FigureClass>();

figures1.Add(circle);

figures1.Add(rect);

figures1.Add(square);

Console.WriteLine("\n\nДо сортировки для List<Figure>:");

foreach (var i in figures1)

{

Console.WriteLine(i);

}

Console.WriteLine("\nПосле сортировки для List<Figure>:");

figures1.Sort();

foreach (var i in figures1)

{

Console.WriteLine(i);

}

Console.WriteLine("\n\nМатрица:");

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

matrix[0, 0] = rect;

matrix[1, 1] = square;

matrix[2, 2] = circle;

Console.WriteLine(matrix.ToString());

SimpleList<FigureClass> list = new SimpleList<FigureClass>();

list.Add(circle);

list.Add(rect);

list.Add(square);

Console.WriteLine("\n\nПеред сортировкой (SimpleList):");

foreach (var a in list)

{

Console.WriteLine(a);

}

list.Sort();

Console.WriteLine("\n\nПосле сортировки (SimpleList):");

foreach (var a in list)

{

Console.WriteLine(a);

}

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

stack.Push(circle);

stack.Push(rect);

stack.Push(square);

Console.WriteLine("\n\nИспользование стека:");

while (stack.Count > 0)

{

FigureClass f = stack.Pop();

Console.WriteLine(f);

}

Console.ReadKey();

}

}

}

Файл: RectangleClass.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_3lab

{

class RectangleClass : FigureClass, Interface1

{

protected double Width;

protected double Height;

public RectangleClass(double width, double height)

{

this.Width = width;

this.Height = height;

this.Type = "Прямоугольник";

}

public override double Area()

{

return Width \* Height;

}

public void Print()

{

Console.WriteLine(this.ToString());

Console.WriteLine("Высота: " + this.Height);

Console.WriteLine("Ширина: " + this.Width);

}

}

}

Файл: SimpleList.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_3lab

{

partial class SimpleListItem<T>

{

///<summary>

///Данные

///</summary>

public T data { get; set; }

///<summary>

///Следующий элемент

/// </summary>

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

/// <summary>

/// конструктор

/// </summary>

public SimpleListItem(T param)

{

this.data = param;

}

}

}

Файл: SimpleListItem.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_3lab

{

///<summary>

///Список

/// </summary>

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

{

///<summary>

///Первый элемент списка

/// </summary>

protected SimpleListItem<T> first = null;

///<summary>

///Послдений элемент списка

/// </summary>

protected SimpleListItem<T> last = null;

///<summary>

///Количество элементов

/// </summary>

public int Count

{

get { return \_count; }

protected set { \_count = value; }

}

int \_count;

///<summary>

///Добавление элемента

///</summary>

public void Add(T element)

{

SimpleListItem<T> newItem = new SimpleListItem<T>(element);

this.Count++;

//добавление первого элемента

if (last == null)

{

this.first = newItem;

this.last = newItem;

}

//добавление следующих элементов

else

{

//присоединение элемента к цепочке

this.last.next = newItem;

//присоединенный элемент считается последним

this.last = newItem;

}

}

/// <summary>

/// Чтение контейнера с заданным номером

/// </summary>

public SimpleListItem<T> GetItem(int number)

{

if ((number < 0) || (number >= this.Count))

{

//Можно создать собственный класс исключения

throw new Exception("Выход за границу индекса");

}

SimpleListItem<T> current = this.first; int i = 0;

//Пропускаем нужное количество элементов

while (i < number)

{

//Переход к следующему элементу

current = current.next;

//Увеличение счетчика

i++;

}

return current;

}

/// <summary>

/// Чтение элемента с заданным номером

/// </summary>

public T Get(int number)

{

return GetItem(number).data;

}

/// <summary>

/// Для перебора коллекции

/// </summary>

public IEnumerator<T> GetEnumerator()

{

SimpleListItem<T> current = this.first;

//Перебор элементов

while (current != null)

{

yield return current.data;

current = current.next;

}

}

//Реализация обобщенного IEnumerator<T> требует реализации необобщенного интерфейса

//Данный метод добавляется автоматически при реализации интерфейса

System.Collections.IEnumerator System.Collections.IEnumerable.GetEnumerator()

{

return GetEnumerator();

}

public void Sort()

{

Sort(0, this.Count - 1);

}

/// <summary>

/// Алгоритм быстрой сортировки

/// </summary>

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;

}

}

}

Файл: SimpleStack.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_3lab

{

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

{

public void Push(T element)

{

Add(element);

}

/// <summary>

/// удаление и чтение из стека

/// </summary>

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

{

//поиск предпоследнего элемента

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

//чтение из последнего элемента

Result = newLast.next.data;

//предпоследний элемент считается последним

this.last = newLast;

//последний элемент удаляется

newLast.next = null;

}

//уменьшение количества элементов в списке

this.Count--;

return Result;

}

}

}

Файл: SparseMatrix.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_3lab

{

public interface SparseMatrix<T>

{

//возвращает пустой элемент

T getEmptyElement();

//проверка, что элемент является пустым

bool CheckEmptyElement(T element);

}

}

Файл: SquareClass.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_3lab

{

class SquareClass : RectangleClass, Interface1

{

public SquareClass(double width) :

base(width, width)

{ this.Type = "Квадрат"; }

}

}

Экранные формы с примерами выполнения программы:

Изображение выглядит как текст

Автоматически созданное описаниеИзображение выглядит как текст

Автоматически созданное описание