**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ» ИНЖЕНЕРНО- ТЕХНОЛОГИЧЕСКАЯ АКАДЕМИЯ**

**Институт компьютерных технологий и информационной безопасности**

**Кафедра математического обеспечения и применения ЭВМ**

**Отчет по лабораторной работе №5**

по курсу «Объектно ориентированное программирование»

**«Классы и объекты в С#»**

Выполнил: студент гр. КТбо2-7

Измайлов В.Н.

Таганрог 2020

**Оглавление**

[Техническое задание 3](#_TOC_250004)

[Выполнение задания 3](#_TOC_250003)

1. [Спецификация классов 3](#_TOC_250002)
2. [Диаграмма классов 4](#_TOC_250001)
3. [Листинг 5](#_TOC_250000)

# ТЕХНИЧЕСКОЕ ЗАДАНИЕ

# Создать класс Sorting (сортировка), и на его основе классы Choice (сортировка выбором) и Quick (быстрая сортировка). Размер сортируемых массивов задается при их создании, а элементы считываются из файла

# ВЫПОЛНЕНИЕ ЗАДАНИЯ

# Спецификация классов.

# Создаются 2 формы. SortForm - это основная форма, где содержится весь функционал программы. GenerationForm – форма в которой пользователь задает настройки для создания новой последовательности чисел.

# Класс FileParser имеет два метода для считывания и записи данных в файл.

# Структура классов построена по принципу MVP (model-view-presenter). Для каждой формы созданы интерфейсы Model и View, а также класс Presenter с названиями, соответствующими названиям классов (SortsView, GenerationView...)

# Каждая форма наследует свой интерфейс View и реализует его методы. Модель каждого класса наследует соответствующий ей интерфейс модели и реализует ее методы.

# Классы QuickSort и SelectionSort классы сортировок

# 2. Диаграмма классов.

# Листинг

SortForm.cs

using System;

using System.Diagnostics;

using System.Threading;

using System.Windows.Forms;

namespace JustSortings

{

public partial class SortForm : Form, ISortsView

{

public SortForm()

{

InitializeComponent();

\_sortsPresenter = new SortsPresenter(this, new SortsModel());

uploadToolTip = new ToolTip();

workTime.Visible = false;

uploadToolTip.SetToolTip(UploadButton, "Upload numbers in file");

}

public void ShowNumbers(int []numbers)

{

SortedNumbersTextBox.Clear();

for (int i = 0; i < numbers.Length; i++)

{

SortedNumbersTextBox.Text += Convert.ToString(numbers[i]) + "\n";

}

}

private void Button1\_Click(object sender, EventArgs e)

{

Stopwatch stopWatch = new Stopwatch();

stopWatch.Start();

if (QuickSortRadioButton.Checked)

{

\_sortsPresenter.SortNumbers("Quick");

}

if (SelectionSortRadioButton.Checked)

{

\_sortsPresenter.SortNumbers("Selection");

}

stopWatch.Stop();

// Get the elapsed time as a TimeSpan value.

TimeSpan ts = stopWatch.Elapsed;

// Format and display the TimeSpan value.

string elapsedTime = String.Format(

"{0:00}:{1:00}:{2:00}.{3:00}",

ts.Hours, ts.Minutes, ts.Seconds,

ts.Milliseconds / 10);

workTime.Text = "Время работы: " + elapsedTime;

workTime.Visible = true;

}

private void Button4\_Click(object sender, EventArgs e)

{

\_sortsPresenter.LoadNumbers();

}

private void Button2\_Click(object sender, EventArgs e)

{

GenerationForm generationForm = new GenerationForm(this);

generationForm.ShowDialog();

}

public SortsPresenter GetSortsPresenter()

{

return \_sortsPresenter;

}

public ISortsModel GetSortsModel()

{

return \_sortsPresenter.GetSortsModel();

}

private void RadioButton2\_CheckedChanged(object sender, EventArgs e)

{

}

private void SortForm\_Load(object sender, EventArgs e)

{

}

private void Button1\_Click\_1(object sender, EventArgs e)

{

\_sortsPresenter.UploadFile();

}

private void ToolTip1\_Popup(object sender, PopupEventArgs e)

{

}

}

}

SortModel

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace JustSortings

{

class SortsModel : ISortsModel

{

public int[] LoadNumbers()

{

FileParser parser = new FileParser();

numbers = parser.ParseFromFile();

return numbers;

}

public void UploadFile()

{

if (numbers.Length == 0)

{

throw new NullReferenceException();

}

FileParser parser = new FileParser();

parser.UploadInFile(numbers);

}

public int[] GetNumbers()

{

return numbers;

}

public void SetParamsOfSequence(ParamsOfSequence paramsOfSequence)

{

\_paramsOfSequence = paramsOfSequence;

}

public void GenerateNewNumbers()

{

int count = \_paramsOfSequence.count;

int start = \_paramsOfSequence.start;

int end = \_paramsOfSequence.end;

Random rand = new Random();

int[] new\_numbers = new int[count];

for (int i = 0; i < count; i++)

{

new\_numbers[i] = rand.Next(start, end);

}

numbers = new\_numbers;

}

public int[] QuickSort()

{

if (numbers.Length == 0)

{

throw new ArgumentException();

}

QuickSort sort = new QuickSort();

sort.Sort(ref numbers, 0, numbers.Length - 1);

return numbers;

}

public int[] SelectionSort()

{

if (numbers.Length == 0)

{

throw new ArgumentException();

}

SelectionSort sort = new SelectionSort();

sort.Sort(ref numbers);

return numbers;

}

private int[] numbers;

private ParamsOfSequence \_paramsOfSequence;

}

}

SortPresenter

using System;

using System.Windows.Forms;

namespace JustSortings

{

public class SortsPresenter

{

private readonly ISortsView view;

private readonly ISortsModel model;

public SortsPresenter(ISortsView view, ISortsModel model)

{

this.view = view;

this.model = model;

}

public void LoadNumbers()

{

var numbers = model.LoadNumbers();

view.ShowNumbers(numbers);

}

public void UploadFile()

{

try

{

model.UploadFile();

}

catch (Exception ex)

{

Console.WriteLine(ex.ToString());

MessageBox.Show("Storage is empty");

}

}

public void SortNumbers(string type)

{

try {

if (String.Compare(type, "Quick") == 0)

{

model.QuickSort();

}

else if (String.Compare(type, "Selection") == 0)

{

model.SelectionSort();

}

view.ShowNumbers(model.GetNumbers());

} catch(Exception ex)

{

Console.WriteLine(ex.ToString());

MessageBox.Show("Nothing to sort");

}

}

public ISortsModel GetSortsModel()

{

return model;

}

public void SetParamsOfSequence(ParamsOfSequence paramsOfSequence)

{

model.SetParamsOfSequence(paramsOfSequence);

model.GenerateNewNumbers();

view.ShowNumbers(model.GetNumbers());

}

}

}

GenerationForm

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace JustSortings

{

public partial class GenerationForm : Form, IGenerationView

{

public GenerationForm(SortForm sortForm)

{

\_sortForm = sortForm;

\_generationPresenter = new GenerationPresenter(this, new GenerationModel());

InitializeComponent();

}

public void CloseView()

{

this.Close();

}

public void GenerateNewNumbers()

{

throw new NotImplementedException();

}

private void Apply\_Click(object sender, EventArgs e)

{

try

{

\_generationPresenter.SetNewNumbersParams(

Convert.ToInt32(CountOfElementsTextBox.Text),

Convert.ToInt32(MinElementTextBox.Text),

Convert.ToInt32(MaxElementTextBox.Text),

\_sortForm

);

}catch(Exception ex)

{

Console.WriteLine(ex.ToString());

MessageBox.Show("Error. Try again");

}

}

private void Cancel\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

GenerationModel

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace JustSortings

{

public struct ParamsOfSequence

{

public int count;

public int start;

public int end;

}

class GenerationModel : IGenerationModel

{

public void SetNewNumbersParams(int count, int start, int end, SortForm sortForm)

{

\_paramsOfSequence.count = count;

\_paramsOfSequence.start = start;

\_paramsOfSequence.end = end;

sortForm.GetSortsPresenter().SetParamsOfSequence(\_paramsOfSequence);

}

public ParamsOfSequence GetParams()

{

return \_paramsOfSequence;

}

private ParamsOfSequence \_paramsOfSequence;

}

}

GenerationPresenter

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace JustSortings

{

class GenerationPresenter

{

private readonly IGenerationView view;

private readonly IGenerationModel model;

public GenerationPresenter(IGenerationView view, IGenerationModel model)

{

this.view = view;

this.model = model;

}

public void SetNewNumbersParams(int count, int start, int end, SortForm sortForm)

{

model.SetNewNumbersParams(count, start, end, sortForm);

view.CloseView();

}

}

}

IGenerationModel

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace JustSortings

{

interface IGenerationModel

{

void SetNewNumbersParams(int count, int start, int end, SortForm sortForm);

}

}

IGenerationView

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace JustSortings

{

interface IGenerationView

{

void GenerateNewNumbers();

void CloseView();

}

}

ISortsModel

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace JustSortings

{

public interface ISortsModel

{

int[] GetNumbers();

void UploadFile();

int[] LoadNumbers();

int[] QuickSort();

void SetParamsOfSequence(ParamsOfSequence paramsOfSequence);

int[] SelectionSort();

void GenerateNewNumbers();

}

}

ISortsView

namespace JustSortings

{

public interface ISortsView

{

void ShowNumbers(int[] numbers);

}

}

Program

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace JustSortings

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new SortForm());

}

}

}

QuickSort

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace JustSortings

{

class QuickSort

{

private int partition(int[] array, int start, int end)

{

int temp;

int marker = start;

for (int i = start; i < end; i++)

{

if (array[i] < array[end])

{

temp = array[marker];

array[marker] = array[i];

array[i] = temp;

marker += 1;

}

}

temp = array[marker];

array[marker] = array[end];

array[end] = temp;

return marker;

}

public void Sort(ref int[] array, int start, int end)

{

if (start >= end)

{

return;

}

int pivot = partition(array, start, end);

Sort(ref array, start, pivot - 1);

Sort(ref array, pivot + 1, end);

}

}

}

SelectionSort

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace JustSortings

{

class SelectionSort

{

public void Sort(ref int[] arr)

{

int min, temp;

int length = arr.Length;

for (int i = 0; i < length - 1; i++)

{

min = i;

for (int j = i + 1; j < length; j++)

{

if (arr[j] < arr[min])

{

min = j;

}

}

if (min != i)

{

temp = arr[i];

arr[i] = arr[min];

arr[min] = temp;

}

}

}

}

}

FileParser

using System;

using System.IO;

namespace JustSortings

{

class FileParser

{

public int[] ParseFromFile()

{

int[] numbers = new int[500];

string path = System.Environment.CurrentDirectory + @"\\Numbers.txt";

StreamReader sr = new StreamReader(path, System.Text.Encoding.Default);

string line;

int index = 0;

while ((line = sr.ReadLine()) != null)

{

//Перезапись в новый массив если послед. длинне выделенной памяти

if (index >= numbers.Length)

{

int[] new\_numbers = new int[numbers.Length \* 2];

for (int i = 0; i < numbers.Length; i++)

{

new\_numbers[i] = numbers[i];

}

numbers = new\_numbers;

}

numbers[index++] = Convert.ToInt32(line);

}

//Перезапись в отформатированном виде (убераем нули, если массив не полностью заполнен)ы

int[] correct\_numbers = new int[index];

for (int i = 0; i < index; i++)

{

correct\_numbers[i] = numbers[i];

}

sr.Close();

return correct\_numbers;

}

public void UploadInFile(int[] numbers)

{

string path = System.Environment.CurrentDirectory + @"\\Output.txt";

using (StreamWriter sw = new StreamWriter(path, false, System.Text.Encoding.Default))

{

for (int i = 0; i < numbers.Length; i++)

{

sw.WriteLine(numbers[i] + "\n");

}

}

}

}

}