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

**Лабораторная работа № 5**

по курсу «Объектно-ориентированное программирование»

**«Классы, объекты, наследование в С#»**

**15 ВАРИАНТ**

Выполнил:

студенты гр. КТбо2-1

Самардак А.В.

Проверил:

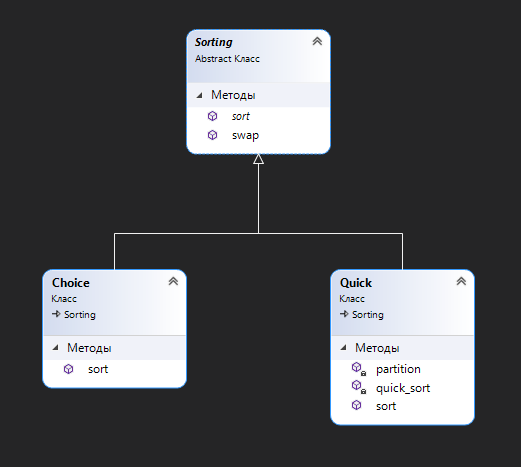
Тарасов С.А/ Быстров Э.А

**Таганрог 2020**

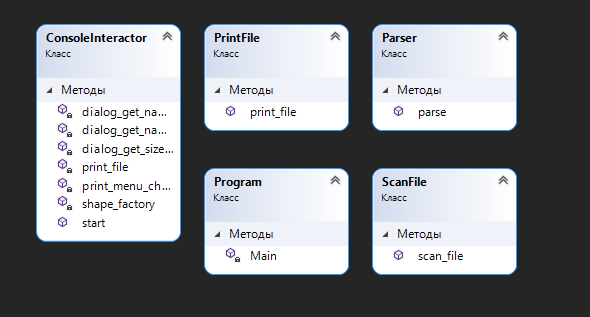
1. **Вариант задания**

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

1. **UML-диаграмма классов: Sorting, Quick, Choice**



1. **UML-диаграмма классов: ConsoleInt, PrintFile, ScanFile, main, Parser**



1. **Листинг программы на языке С#**
2. **Main.cs**

using System;

namespace OOP\_Laba5

{

class Program

{

static void Main(string[] args)

{

ConsoleInteractor console = new ConsoleInteractor();

console.start();

}

}

}

1. **ConsoleInteractor.cs**

using System;

namespace OOP\_Laba5

{

class ConsoleInteractor

{

public void start()

{

try

{

int check = dialog\_get\_name\_sort();

string name\_file = dialog\_get\_name\_file();

int size = dialog\_get\_size\_file();

ScanFile scan\_file = new ScanFile();

int[] arr = scan\_file.scan\_file(ref name\_file, size);

shape\_factory(check, ref arr);

print\_file(ref arr);

} catch (Exception ex)

{

Console.WriteLine(ex);

}

}

private int dialog\_get\_name\_sort()

{

int check;

do

{

print\_menu\_check();

check = Convert.ToInt32(Console.ReadLine());

Console.Clear();

} while (check < 1 || check > 2);

return check;

}

private string dialog\_get\_name\_file()

{

string name\_file;

do

{

Console.WriteLine("Название файла и его формат: ");

name\_file = Console.ReadLine();

Console.Clear();

} while (name\_file == "");

return name\_file;

}

private int dialog\_get\_size\_file()

{

int size;

do

{

Console.WriteLine("Сколько элементов считать с файла?:");

size = Convert.ToInt32(Console.ReadLine());

Console.Clear();

} while (size <= 0);

return size;

}

private void shape\_factory(int check, ref int[] arr)

{

Sorting sorting;

if (check == 1)

{

sorting = new Quick();

sorting.sort(ref arr, arr.Length);

}

else if (check == 2)

{

sorting = new Choice();

sorting.sort(ref arr, arr.Length);

}

}

private void print\_file(ref int[] arr)

{

PrintFile print\_file = new PrintFile();

print\_file.print\_file(ref arr);

Console.WriteLine("Успешно, файл будет выгружен в корневую папку проекта!");

}

private void print\_menu\_check()

{

Console.WriteLine("-------------------------------");

Console.WriteLine("Какую сортировку использовать?");

Console.WriteLine("1 - Quick");

Console.WriteLine("2 - Choice");

Console.WriteLine("-------------------------------");

}

}

}

1. **ScanFile.cs**

using System;

using System.IO;

using System.Text;

namespace OOP\_Laba5

{

class ScanFile

{

public int[] scan\_file(ref string name\_file, int size)

{

if (File.Exists(@name\_file))

{

string line;

int[] arr = new int[size];

using (StreamReader stream\_reader = new StreamReader(@name\_file))

{

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

{

if ((line = stream\_reader.ReadLine()) == null)

{

throw new Exception("Недопустимое количество элементов в файле!... Завершение работы");

}

else

{

arr[i] = Parser.parse(ref line);

}

}

}

return arr;

}

else

{

throw new Exception("Нет такого файла!... Завершение работы");

}

}

}

}

1. **Sorting.cs**

using System;

namespace OOP\_Laba5

{

abstract class Sorting

{

abstract public void sort(ref int[] arr, int size);

public void swap(ref int a, ref int b)

{

int temp = a;

a = b;

b = temp;

}

}

}

1. **Quick.cs**

using System;

namespace OOP\_Laba5

{

class Quick: Sorting

{

public override void sort(ref int[] arr, int size)

{

quick\_sort(ref arr, 0, size - 1);

}

private void quick\_sort(ref int[] arr, int minIndex, int maxIndex)

{

if (minIndex >= maxIndex)

{

return;

}

var pivotIndex = partition(ref arr, minIndex, maxIndex);

quick\_sort(ref arr, minIndex, pivotIndex - 1);

quick\_sort(ref arr, pivotIndex + 1, maxIndex);

}

private int partition(ref int[] arr, int minIndex, int maxIndex)

{

var pivot = minIndex - 1;

for (var i = minIndex; i < maxIndex; i++)

{

if (arr[i] < arr[maxIndex])

{

pivot++;

swap(ref arr[pivot], ref arr[i]);

}

}

pivot++;

swap(ref arr[pivot], ref arr[maxIndex]);

return pivot;

}

}

}

1. **Choice.cs**

using System;

namespace OOP\_Laba5

{

class Choice: Sorting

{

public override void sort(ref int[] arr, int size)

{

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

{

int i\_min = i;

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

{

if (arr[j] < arr[i\_min])

{

i\_min = j;

}

}

swap(ref arr[i], ref arr[i\_min]);

}

}

}

}

1. **PrinfFile.cs**

using System;

using System.IO;

using System.Text;

namespace OOP\_Laba5

{

class PrintFile

{

public void print\_file(ref int[] arr)

{

string path = @"result.txt";

using (StreamWriter fout = new StreamWriter(path, false))

{

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

{

fout.Write($"{arr[i]}\n");

}

}

}

}

}

1. **Parser.cs**

using System;

namespace OOP\_Laba5

{

class Parser

{

static public int parse(ref string data)

{

return Convert.ToInt32(data);

}

}

}