Міністерство освіти і науки України
Національний технічний університет України „КПІ”
Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки
інформації та управління

ЗВІТ

до лабораторної роботи № 2
з предмету:

„ Основи Web-програмування”

Виконав студент *ІП-61 Блануца Дмитро Сергійович, 2-й курс, ІП-6102*

(№ групи, прізвище, ім’я, по батькові, курс, номер залікової книжки)

Прийняв Ліщук К. І.

(посада, прізвище, ім’я, по батькові)

Київ 2018

ЗМІСТ

1. ПОСТАНОВКА ЗАДАЧІ

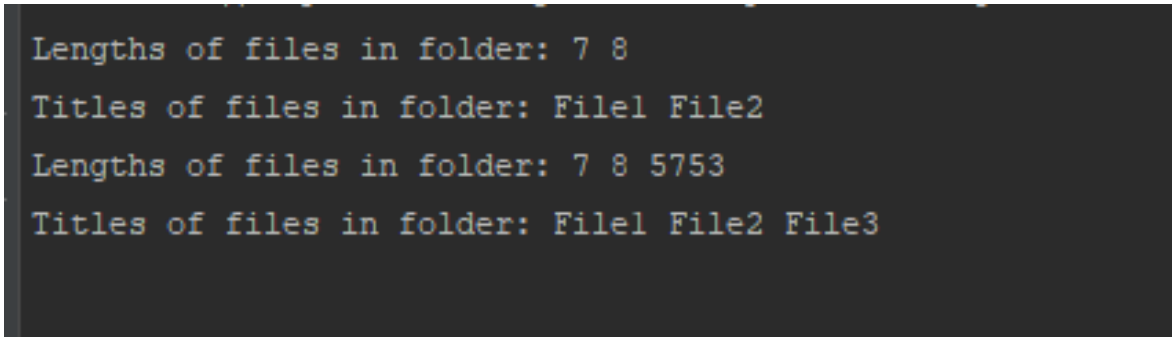
Вариант 1

Создать абстрактный класс File, инкапсулирующий в себе методы Open, Close, Seek, Read, Write, GetPosition и GetLength. Создать производные классы MyDataFile1 и MyDataFile2— файлы, содержащие в себе данные некоторого определенного типа MyData1 и MyData2, а также заголовки, облегчающие доступ к этим файлам.

Создать класс Folder, содержащий массив/параметризованную коллекцию объектов этих классов в динамической памяти.

Предусмотреть возможность вывода списка имен и длин файлов.

2. РЕЗУЛЬТАТ РОБОТИ ПРОГРАМИ



3. КОД ПРОГРАМИ

```
using System;
using System.Collections.Generic;
using System.IO;

namespace CSharp_Lab2
{
    public abstract class File
    {
        protected int position;
        protected string title;
        protected int length;

        protected File()
        {
        }

        public int GetPosition()
        {
            return position;
        }

        public string GetTitle()
        {
            return title;
        }


        public int GetLength()
        {
            return length;
        }

        public virtual void Open()
        {
            Console.WriteLine("File " + title + " is opened.");
        }

        public virtual void Close()
        {
            Console.WriteLine("File " + title + " is closed.");
        }

        public virtual void Seek(int n)
        {
            position = n;
        }

        public abstract void Write();


        public abstract void Read();


        private sealed class PositionTitleLengthEqualityComparer :
IEqualityComparer<File>
```

```
        {
            public bool Equals(File x, File y)
            {
                if (ReferenceEquals(x, y)) return true;
                if (ReferenceEquals(x, null)) return false;
                if (ReferenceEquals(y, null)) return false;
                if (x.GetType() != y.GetType()) return false;
                return x.position == y.position && x.title == y.title && x.length ==
y.length;
            }

            public int GetHashCode(File obj)
            {
                unchecked
                {
                    var hashCode = obj.position;
                    hashCode = (hashCode * 397) ^ obj.title.GetHashCode();
                    hashCode = (hashCode * 397) ^ obj.length;
                    return hashCode;
                }
            }
        }

        private static readonly IEqualityComparer<File>
PositionTitleLengthComparerInstance = new PositionTitleLengthEqualityComparer();

        public static IEqualityComparer<File> PositionTitleLengthComparer
        {
            get
            {
                if (PositionTitleLengthComparerInstance != null) return
PositionTitleLengthComparerInstance;
                return null;
            }
        }

        public override string ToString()
        {
            return "File{position: " + position + ", length: " + length + ", title :" +
title + " }";
        }

        public static bool operator ==(File file1, File file2)
        {
            return file1.Equals(file2);
        }

        public static bool operator !=(File file1, File file2)
        {
            return !(file1 == file2);
        }

        public abstract File DeepCopy();

    }
}
```

```
using System;

namespace CSharp_Lab2
{
    public class MyDataFile1<T> : File
    {

        private T MyData1;

        public MyDataFile1(string title, int position, int length)
        {
            this.title = title;
            this.position = position;
            this.length = length;
        }

        public T GetMyData()
        {
            return MyData1;
        }

        public override void Write()
        {
            Console.Write(MyData1);
        }

        public override void Read()
        {
            Console.WriteLine("Readin for " + title + " file.");
        }

        public override File DeepCopy()
        {
            return new MyDataFile1<T>(this.title, this.position, this.length);
        }
    }
}


using System;

namespace CSharp_Lab2
{
    public class MyDataFile2<F> : File
    {

        private F MyData2;

        public MyDataFile2(string title, int position, int length)
        {
            this.title = title;
            this.position = position;
            this.length = length;
        }

        public F GetMyData()
        {
            return MyData2;
```

```
        }

        public override void Write()
        {
            Console.Write(MyData2);
        }

        public override void Read()
        {
            Console.WriteLine("Readin for " + title + " file.");
        }

        public override File DeepCopy()
        {
            return new MyDataFile1<F>(this.title, this.position, this.length);
        }
    }
}


using System;
using System.Collections.Generic;

namespace CSharp_Lab2
{
    public class Folder
    {
        private List<File> files = new List<File>();

        public Folder()
        {
        }

        public Folder(List<File> files)
        {
            this.files = files;
        }

        public void Add(File file)
        {
            files.Add(file);
        }

        public void RemoveAt(int index)
        {
            files.RemoveAt(index);
        }

        public void PrintTitles()
        {
            Console.Write("Titles of files in folder: ");
            foreach (var file in files)
            {
                Console.Write(file.GetTitle() + " ");
            }
            Console.WriteLine();
        }
```

```
        public void PrintLengths()
        {
            Console.Write("Lengths of files in folder: ");
            foreach (var file in files)
            {
                Console.Write(file.GetLength() + " ");
            }
            Console.WriteLine();
        }

        public Folder DeepCopy()
        {
            var files = new List<File>();
            foreach (var file in this.files)
            {
                files.Add(file.DeepCopy());
            }
            return new Folder(files);
        }

        private sealed class FilesEqualityComparer : IEqualityComparer<Folder>
        {
            public bool Equals(Folder x, Folder y)
            {
                if (ReferenceEquals(x, y)) return true;
                if (ReferenceEquals(x, null)) return false;
                if (ReferenceEquals(y, null)) return false;
                if (x.GetType() != y.GetType()) return false;
                return Equals(x.files, y.files);
            }

            public int GetHashCode(Folder obj)
            {
                return (obj.files != null ? obj.files.GetHashCode() : 0);
            }
        }


    }
}



namespace CSharp_Lab2
{
    internal class Program
    {
        public static void Main(string[] args)
        {
            var folder1 = new Folder();
            folder1.Add(new MyDataFile1<string>("File1", 5, 7));
            folder1.Add(new MyDataFile1<string>("File2", 20, 8));
            folder1.PrintLengths();
            folder1.PrintTitles();

            var folder2 = folder1.DeepCopy();
            folder2.Add(new MyDataFile1<string>("File3", 753, 5753));
            folder2.PrintLengths();
```

```
            folder2.PrintTitles();

        }
    }
}
```

4. ВИСНОВОК

В даній лабораторній роботі я познайомився з ООП в мові програмування C# При написанні програми труднощів не виникло.