King Fahd University of Petroleum & Minerals
Information and Computer Science Department

SWE 316: Software Design and Construction (Term 231)

Homework # 2

Date of submission

12/10/2023

Ayed al qahtani

202023400

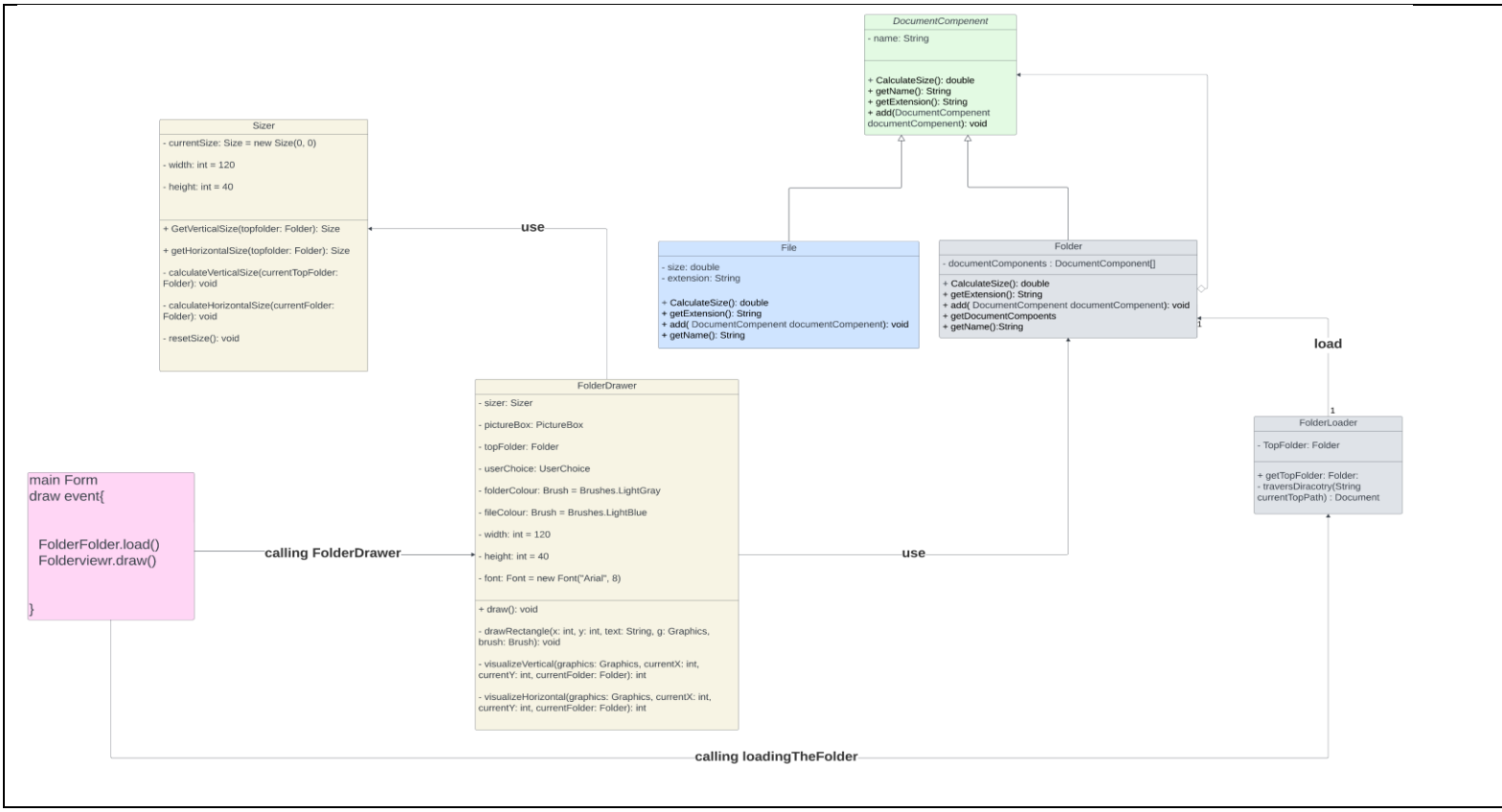
Task	Grade	Your Grade	Comments
Task # 1: Class Diagram	10		
Task # 2: implementation	50		
Task # 3: Class Diagram (Strategy Pattern)	10		
Check list and penalties No Cover page with grade table File name (report) Not in PDF format			-10 ☐ -5 ☐ -10 ☐
Total	70		

Task # 1: Composition Design Pattern

File/Folder combination is a typical example of the composite design pattern. A file has a name, size, extension. A folder has similar attribute (without an extension) plus a list of files or other folders. You are required to write a demonstration application that traverses files and folders in a selected directory.

Class diagram

Design a class diagram showing the above-mentioned structure using the composite design pattern. You have to show all components including the Application class.



Note: the main form represents the main application class

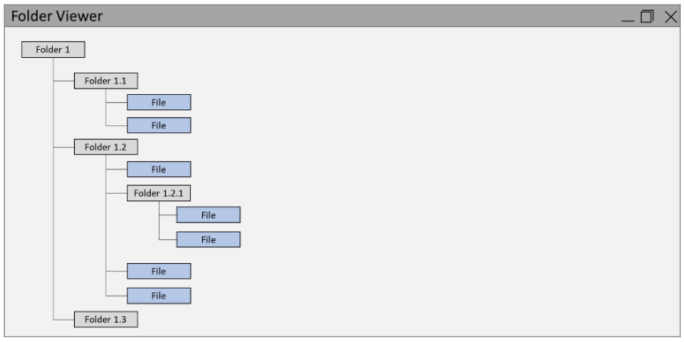
Application

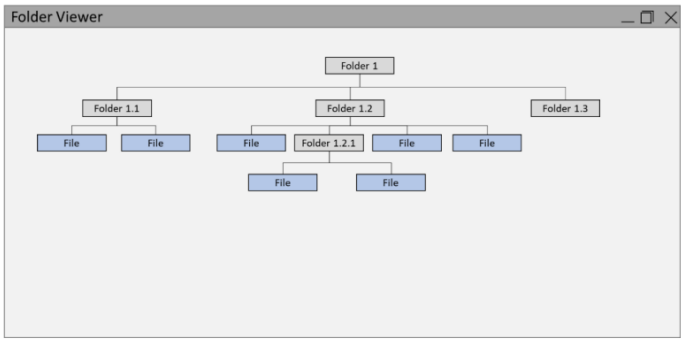
Implement a .Net desktop application (C# or VB) by which you can choose a certain folder when the program starts. Once you select a folder, you should recursively traverse all of its contents (files and folders) and **fill the required information as follows**:

- Folder: only name
- File: name, size, extension

After traversing, your application should traverse the created structure (your structure) again and calculate the size **of all folders by single line call (x.CalculateSize())** where x represent the top most folder.

After calculating the sizes of all folders and subfolders, you should **visualize** the folder and its contents as shown in the sample below. You should show the file or folder size besides its name. This should be accomplished using a single line **(x.visualize())** where x represent the top most folder. You should support visualizing the folder either vertically or horizontally as shown in the samples below.

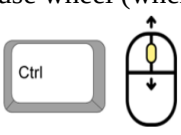




Requirements

Develop your program to fulfill the following requirements:

1- When executed, it should display a button and give the user the freedom of choosing the folder to visualize.
   - For your testing purposes, you can hardcode the folder while you are testing.
2- Once the user selects a folder, you should display the visualization on a panel inside your main form.
   - The visualization should be done by code (You can't use any ready components such as Treeview)
   - The panel should be able to respond to the changes in the size of the form (i.e, bigger or smaller)
3- If the visualization is getting bigger than the panel, you should display scrollbars.
4- You should allow the user to change visualization from vertical to horizontal and vice versa.
5- Zooming: you should allow the user to zoom in and out using:
   - Mouse wheel (when pressing Control button)



   - Pressing

Task # 2: Implementation

Loading date method

```
public Folder TraverseDirectory(string currentTopPath)
        {

            DirectoryInfo directoryInfo = new DirectoryInfo(currentTopPath);
            Folder currentTopFolder = new Folder(directoryInfo.Name);
            // normal case
            string[] files = Directory.GetFiles(currentTopPath);


            foreach (string file in files)
            {
                FileInfo fileInfo = new FileInfo(file);
                DocumentComponent theFile = new File(fileInfo.Name, fileInfo.Length, fileInfo.Extension);
                currentTopFolder.add(theFile);
            }
            // recursive case
            string[] subdirectories = Directory.GetDirectories(currentTopPath);
            foreach (string subdirectory in subdirectories)
            {
                DocumentComponent nextTopFolder = TraverseDirectory(subdirectory);
                currentTopFolder.add(nextTopFolder);
            }

            return currentTopFolder;

        }
```

Here it will go through all the files in the current folder only.

Here it will go the next folder and load its content and go the main call to be stored.

File/Folder

```
public class Folder : DocumentComponent
    {
        private List<DocumentComponent> documentComponents;
        public Folder(string name) : base(name) {
```

```
            documentComponents = new List<DocumentComponent>();
        }
        public override void add(DocumentComponent documentComponent)
        {
            documentComponents.Add(documentComponent);
        }

        public override double calculateSize()
        {
            double size = 0;
            foreach(DocumentComponent documentComponent in documentComponents)
            {
                size += documentComponent.calculateSize();
            }
            return size;
        }

        public override string getExtension()
        {
            throw new Exception();
        }

        public List<DocumentComponent> getDocuments()
        {
            return documentComponents;
        }

        public override string ToString()
        {
            return getName();
        }


    }

public class File : DocumentComponent

    {

        private double size;
        private string extension;
        public File(string name, double size, string extension) : base(name) {

            this.size = size;
            this.extension = extension;
        }


        public override void add(DocumentComponent documentComponent)
        {
            throw new Exception();
        }

        public override double calculateSize()
        {
            return size;
        }

        public override string getExtension()
        {
            return extension;
```

Here we are dealing with folder and file with the same interface. We are calling the same method with a recursive call to calculate the size.

ClaclauteSize()

In folder we do not have extension, so we just want the name.

It is considered as the base case for method.

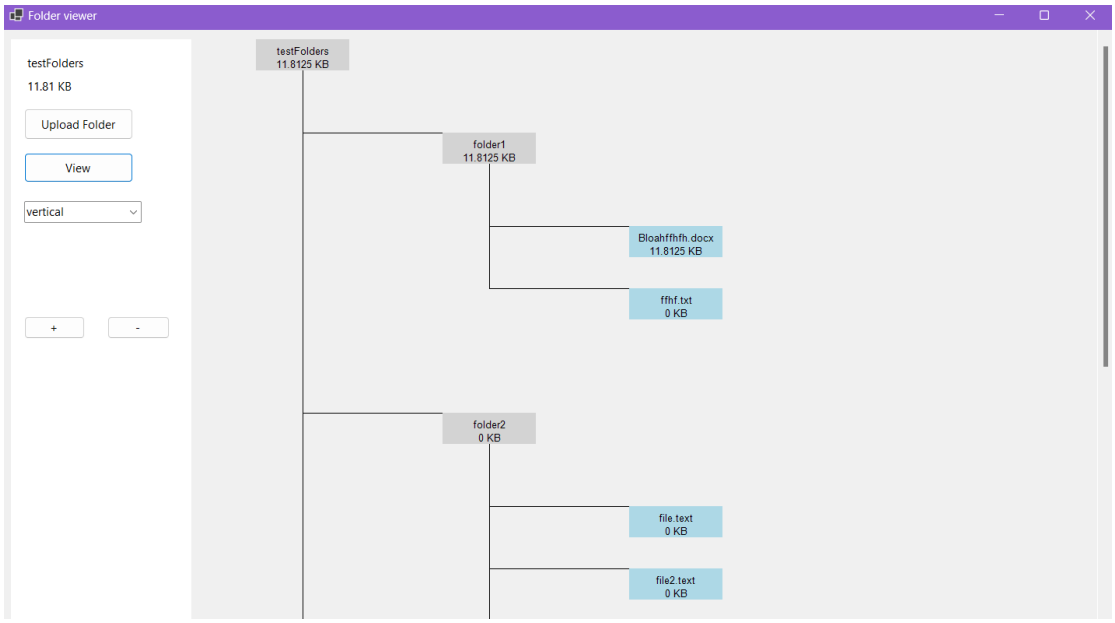
```
        }

        public override string ToString()
        {
            return getName()+"."+extension;
        }
    }
```

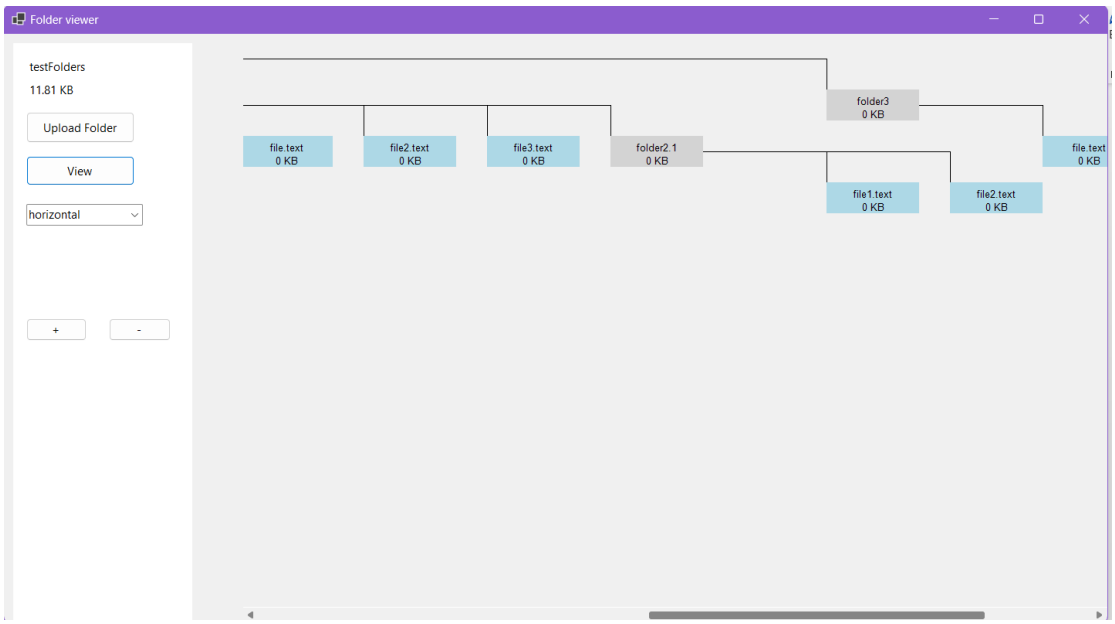
In file, we want the name and the extension.

Result

A sample of vertical view



A sample of horizontal view



Note: there are 3 ways to zoom by Ctrl + mouse, buttons on the interface, and (+, -) keys on the keyboard

Task # 3

Strategy design pattern

Drawing the folders in two different ways represents a good case for the Strategy Design Pattern. Draw a class.

Result