Software 316 – Homework #2 Report

King Fahd University of Petroleum and Minerals

Information of Computer Science Department

SWE316: Software Design and Construction

Term 251

Due Date: Dec 6, 2025

Student Name: Joud Aljabri

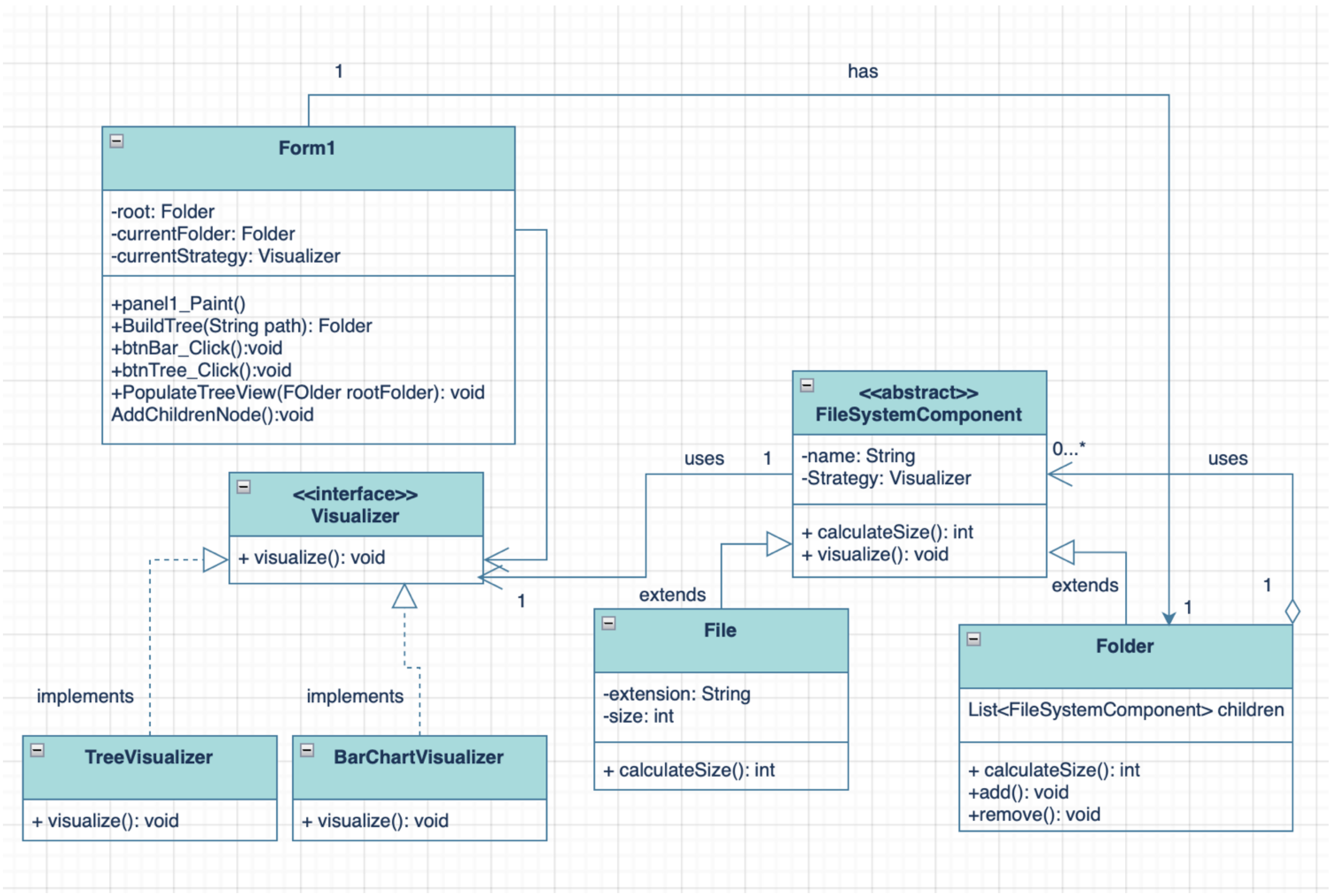
Student ID: 202277140

Instructor: Khalid Aljasser

Task	Grade	Your Grade	Comments
Task # 1: Class Diagram	20		
Task # 2: Implementation	60		
Check list and penalties			
No Cover page with grade table			-10 ☐
File name (report)			-5 ☐
Not in PDF format			-10 ☐
Total	80		

1. Class diagram [20 marks]

Design a class diagram showing the above-mentioned structure using the **composite design pattern**. You have to show all components **including the Application class**. Drawing the folders in two different ways represents a good case for the **Strategy Design Pattern**.



2. Application [60 marks]

Implement a desktop application (Java, C#, or VB.NET) 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 :onlyname
- 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.

Code Implementation

1. FileSystemComponent Class

The FileSystemComponent class is the component class for the composite design pattern. It defined the calculate Size methods which is too be implemented by both file and folder

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace xAcademics
{
    internal abstract class FileSystemComponent
    {
        protected String name;

        public Visualizer Strategy { get; set; } // here
        protected FileSystemComponent(String name) {
            this.name = name;
        }

        public String getName()
        {
            return name;
        }

        public void visualize(Graphics g, int x, int y)
        {
```

The FileSystemComponent class uses Visualizer class to draw the required illustration. The method check which strategy the user choses and depending on it draws the needed shape.

```
            Strategy?.visualize(this, g, x, y); // here
        }

        public abstract int calculateSize();
    }
}
```

Code Implementation

2. Folder Class

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace xAcademics
{
    internal class Folder : FileSystemComponent
    {

        public List<FileSystemComponent> children = new List<FileSystemComponent>();

        public Folder(string name) :
            base(name)
        { }

        public void add(FileSystemComponent compnenet)
        {
            children.Add(compnenet);
        }

        public void remove(FileSystemComponent compnenet)
        {
            children.Remove(compnenet);
        }

        public override int calculateSize()
        {
            int totalSize = 0;

            for (int i = 0; i < children.Count; i++)
            {
                totalSize += children[i].calculateSize();

            }
            return totalSize;
        }
    }
}
```

The Folder class is the composite class that defines the behavior for all components having children

These methods are not required for the HW but they should always be added for the composite class for the list of components

Using this method here to traverse all the components (children) of the folders and calculating their sized

Code Implementation

3. File Class

The File class is the leaf class which defines the individual component having no children

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace xAcademics
{
    internal class File1 : FileSystemComponent{  // this is h                    mns in
C#

        private int size;
        private String extension;

        public File1(string name, int size, string extension)
              : base(name)
        {
            this.size = size;
            this.extension = extension;
        }

        public override int calculateSize()
        {
            return size / 1024; // convert to bytes
        }
    }
}
```

Defined the size attribute here so that the folder size will basically be the sum of sizes of their components and each component is in the end a file

Code Implementation

4. Visualizer

The Visualizer interface defined the strategy for visualizing a FileSystemComponent. Different visualization behaviors (tree, barchart, etc) implement this interface to allow the program switch styles at runtime

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace xAcademics
{
    internal interface Visualizer
    {
        void visualize(FileSystemComponent componenet, Graphics G, int x, int y);

    }
}
```

Code Implementation

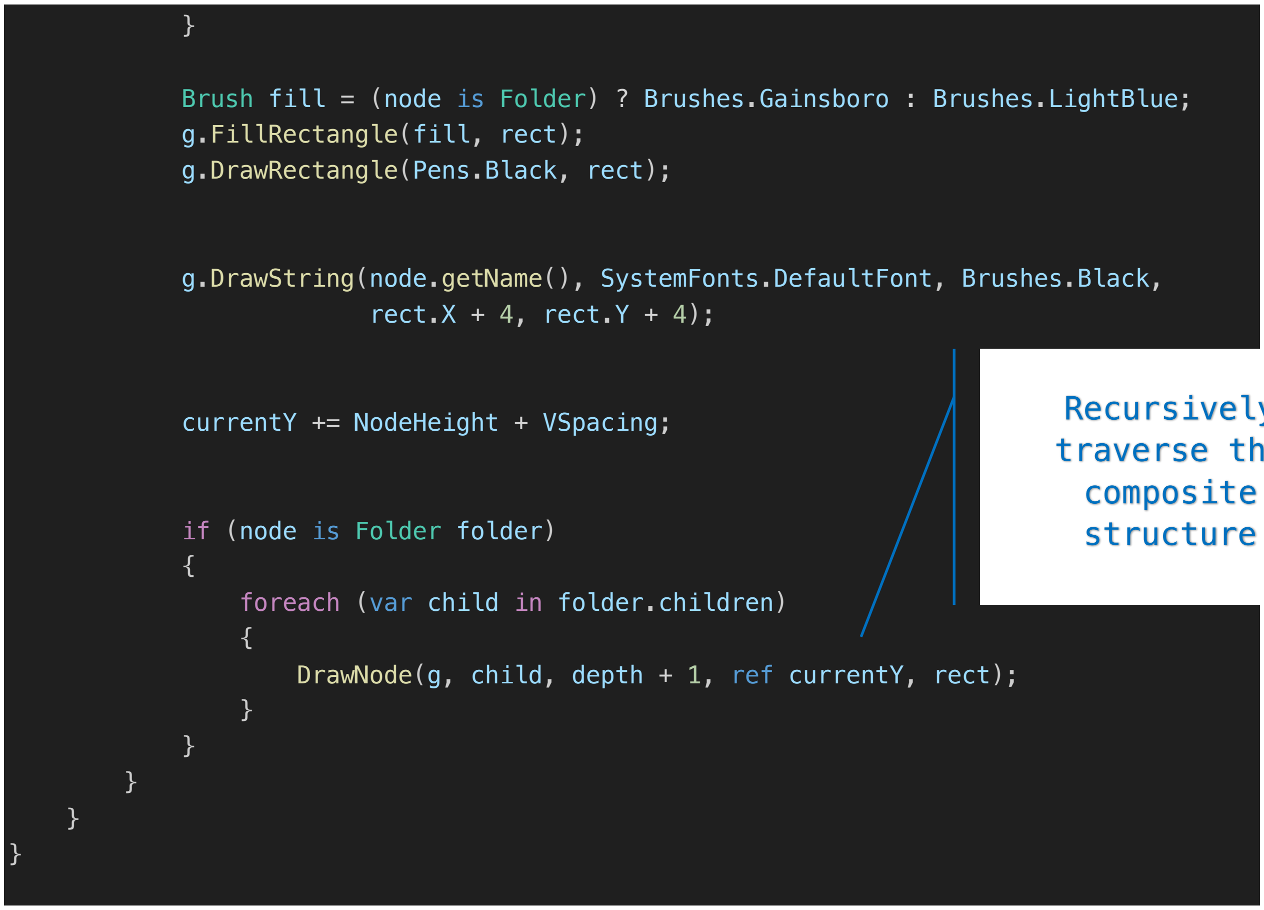
5. TreeVisualizer Class

This class implements the Visualizer interface and provides a node-based tree drawing

```
using System.Drawing;

namespace xAcademics
{
    internal class TreeVisualizer : Visualizer
    {
        private const int NodeWidth = 130;
        private const int NodeHeight = 26;
        private const int HSpacing = 80;
        private const int VSpacing = 25;

        public void visualize(FileSystemComponent root, Graphics g, int x, int y)
        {
            g.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.AntiAlias;
            g.Clear(Color.White);

            int currentY = 20;
            DrawNode(g, root, depth: 0, currentY: ref currentY, parentRect: null);
        }

        private void DrawNode(Graphics g,
                              FileSystemComponent node,
                              int depth,
                              ref int currentY,
                              Rectangle? parentRect)
        {
            int logicalX = 20 + depth * HSpacing;
            int logicalY = currentY;

            Rectangle rect = new Rectangle(logicalX, logicalY, NodeWidth, NodeHeight);

            if (parentRect.HasValue)
            {
                Point parentCenter = new Point(
                    parentRect.Value.Right,
                    parentRect.Value.Top + NodeHeight / 2
                );

                Point childCenter = new Point(
                    rect.Left,
                    rect.Top + NodeHeight / 2
                );

                g.DrawLine(Pens.Gray, parentCenter, childCenter);
```

This method is to draw the node of each tree

To draw a connecter from the parent to child



```
            }

            Brush fill = (node is Folder) ? Brushes.Gainsboro : Brushes.LightBlue;
            g.FillRectangle(fill, rect);
            g.DrawRectangle(Pens.Black, rect);


            g.DrawString(node.getName(), SystemFonts.DefaultFont, Brushes.Black,
                         rect.X + 4, rect.Y + 4);


            currentY += NodeHeight + VSpacing;


            if (node is Folder folder)
            {
                foreach (var child in folder.children)
                {
                    DrawNode(g, child, depth + 1, ref currentY, rect);
                }
            }
        }
    }
}
```

Code Implementation

5. BarChartVisualizer Class

This class visualizes folder contents as a horizontal bar chart

```
using System.Drawing;

namespace xAcademics
{
    internal class BarChartVisualizer : Visualizer
    {
        public void visualize(FileSystemComponent comp, Graphics g, int x, int y)
        {
            if (comp is Folder rootFolder)
            {
                DrawBars(rootFolder.children, g, x, y);
            }
        }

        private void DrawBars(List<FileSystemComponent> items, Graphics g, int x, int
startY)
        {
            int maxSize = items.Max(item => item.calculateSize());
            float scale = 300f / maxSize;

            int y = startY;

            foreach (var item in items)
            {
                int size = item.calculateSize();
                int width = (int)(size * scale);
                if (width < 40) width = 40;

                Brush color = (item is Folder)
                    ? new SolidBrush(Color.FromArgb(150, 120, 170, 255))
                    : new SolidBrush(Color.FromArgb(150, 255, 150, 120));

                g.FillRectangle(color, x, y, width, 25);
                g.DrawRectangle(Pens.Black, x, y, width, 25);

                string label = $"({size} KB) {item.getName()}";
                g.DrawString(label, SystemFonts.DefaultFont, Brushes.Black, x + 5, y +
5);


                y += 40;
            }
        }
    }
```

It displays bar chart only for the content of the root folder just like it shows in the HW document

DrawBars method is to traverse through the components of the root folder, calculate their size by calling the calculateSize() method and display them as bars

```
}
```