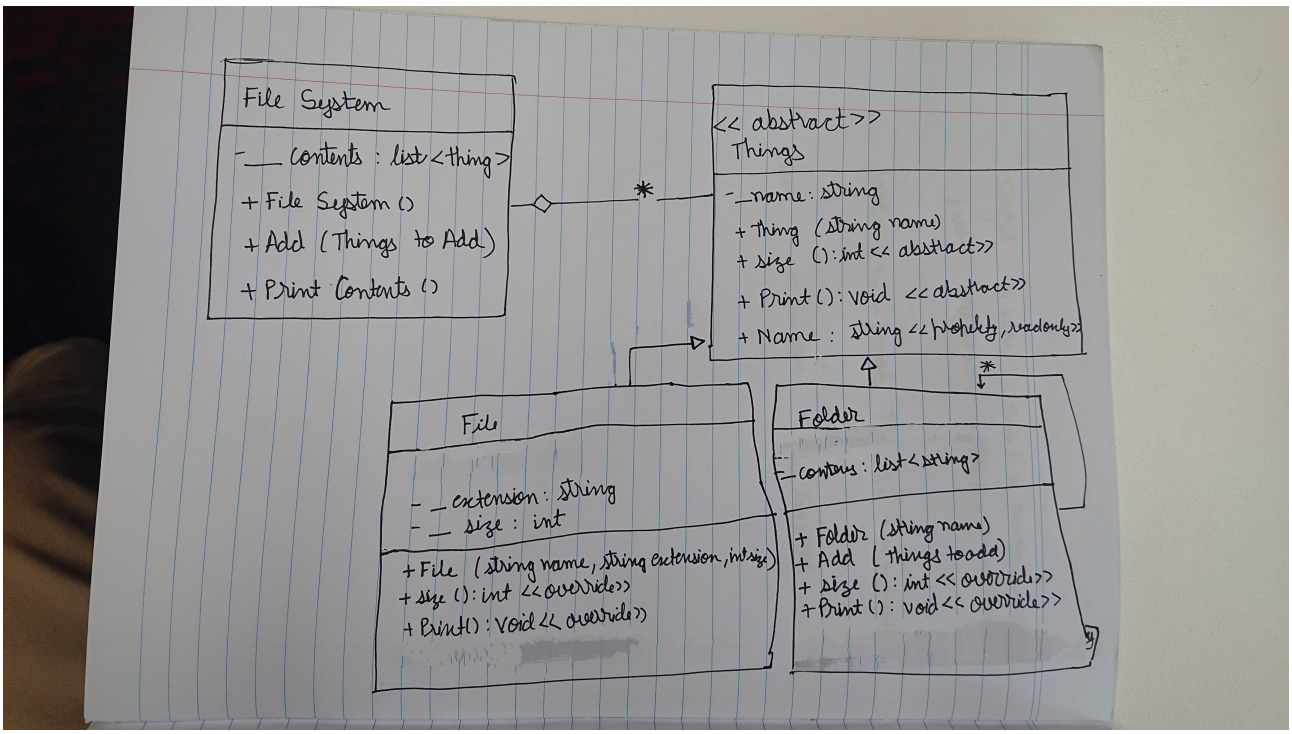
Swinburne University of Technology

COS20007 Object Oriented Programming

Semester test

PDF generated at 11:11 on Friday 13th October, 2023

Produced by OOP Submission Formatter


File System
- contents : list<thing>
+ File System ()
+ Add (Things to Add)
+ Print Contents ()
*
<< abstract >>
Things
- name : string
+ thing (string name)
+ size (): int << abstract >>
+ Print (): void << abstract >>
+ Name : string << property, readonly >>
File
- extension : string
- size : int
+ File (string name, string extension, int size)
+ size (): int << override >>
+ Print() : void << override >>
Folder
- contains : list<string>
+ Folder (string name)
+ Add (things to add)
+ size (): int << override >>
+ Print (): void << override >>

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

namespace semestertest
{
        public class Program
        {
            public static void Main(string[] args)
            {

                FileSystem fileSystem = new FileSystem();

                File newfile1 = new File("AnImage", "jpg", 5342);
                File newfile2 = new File("SomeFile", "txt", 832);
                fileSystem.Add(newfile1);
                fileSystem.Add(newfile2);

                Folder newfolder = new Folder("Photos");
                newfolder.Add(newfile1);
                fileSystem.Add(newfolder);

                Folder newfolder2 = new Folder("Folder within folder");
                Folder newfolder3 = new Folder("some folder");
                newfolder3.Add(newfile2);
                newfolder3.Add(newfile1);
                newfolder2.Add(newfolder3);
                fileSystem.Add(newfolder2);

                Folder folder4 = new Folder("Empty folder");
                fileSystem.Add(folder4);

                fileSystem.PrintContents();

                Console.ReadLine();

            }
        }
}
```

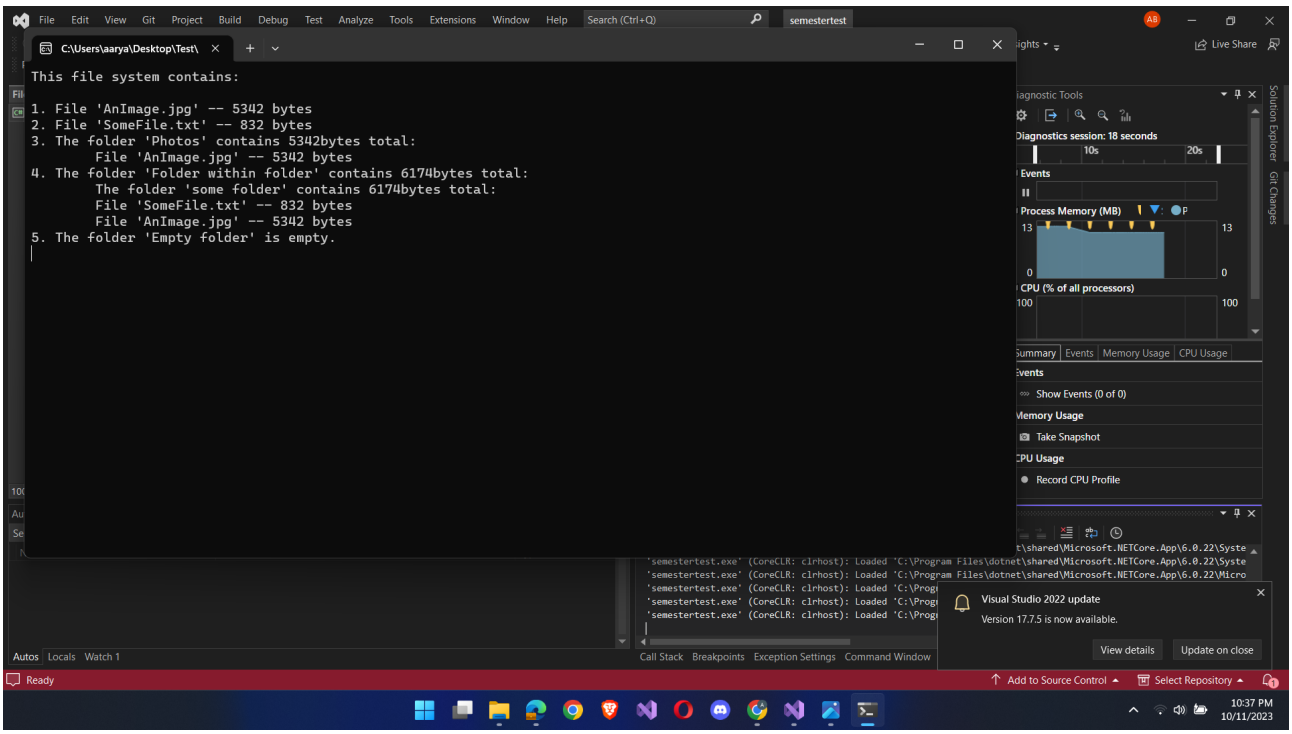
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace semestertest
{
    public class FileSystem
    {
        private List<Thing> _contents;
        public FileSystem()
        {
            _contents = new List<Thing>();
        }
        public void Add(Thing toAdd)
        {
            _contents.Add(toAdd);
        }
        public void PrintContents()
        {
            if (_contents.Count == 0)
            {
                Console.WriteLine($"This file system is empty!");
            }
            else
            {
                Console.WriteLine("This file system contains:\n");
                int i = 1;
                foreach (Thing thing in _contents)
                {
                    Console.Write($"{i}. ");
                    thing.Print();
                    i += 1;

                }
            }
        }
    }
}
```

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

namespace semestertest
{
    public abstract class Thing
    {
        private string _name;
        public Thing(string name)
        {
            _name = name;
        }
        public abstract int Size();
        public abstract void Print();
        public string Name
        {
            get { return _name; }
        }
    }
}
```

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

namespace semestertest
{
    public class Folder : Thing
    {
        private List<Thing> _contents;

        public Folder(string name) : base(name)
        {
            _contents = new List<Thing>();

        }
        public void Add(Thing toAdd)
        {
           _contents.Add(toAdd);
        }

        public override int Size()
        {
            int size = 0;
            foreach (Thing file in _contents)
            {
                size += file.Size();
            }
            return size;
        }
        public override void Print()
        {

            if (_contents.Count > 0)
            {
                Console.WriteLine($"The folder '{Name}' contains {Size()}bytes
↪  total:");

                foreach (Thing thing in _contents)
                {
                    Console.Write($"\t");
                    thing.Print();
                }
            }
            else
            {
                Console.WriteLine($"The folder '{Name}' is empty.");
            }
        }


    }
```

```
53 }
```

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

namespace semestertest
{
    public class File : Thing
    {
        private int _size;
        private string _extension;

        public File(string name, string extension, int size) : base(name)
        {

            _size = size;
            _extension = extension;

        }

        public override int Size()
        {
            return _size;
        }

        public override void Print()
        {

            Console.WriteLine($"File '{Name}.{_extension}' -- {_size} bytes");
        }



    }
}
```


C:\Users\aarya\Desktop\Test\
This file system contains:
1. File 'AnImage.jpg' -- 5342 bytes
2. File 'SomeFile.txt' -- 832 bytes
3. The folder 'Photos' contains 5342bytes total:
File 'AnImage.jpg' -- 5342 bytes
4. The folder 'Folder within folder' contains 6174bytes total:
The folder 'some folder' contains 6174bytes total:
File 'SomeFile.txt' -- 832 bytes
File 'AnImage.jpg' -- 5342 bytes
5. The folder 'Empty folder' is empty.
Visual Studio 2022 update
Version 17.7.5 is now available.
View details
Update on close

Task 2

1. Describe the principle of polymorphism and how it was used in Task 1

The word “polymorphism” stands for having multiple forms. Polymorphism, which enables objects of diverse types to be seen as belonging to the same base type, is a fundamental concept in object-oriented programming. This indicates that you don't need to be aware of any particular derived types of various object types in order to represent them using a base type and call methods on them.

Task 1 uses the Thing class to demonstrate polymorphism. This class is the parent of the Folder and File classes. Since Thing is the parent of both Folder and File, they must both implement its abstract methods Size() and Print().

When a FileSystem is formed and objects are added to it, both Folder and File objects can be added to the list of contents because they are both thought of as belonging to the Thing type. This is possible because of polymorphism.

2. Consider the FileSystem and Folder classes from the updated design in Task 1. Do we need both of these classes? Explain why or why not.

Yes, the revised design requires both the FileSystem and Folder class The 'FileSystem' class keeps a collection of 'Thing' objects and represents the full file system. It is necessary to provide operations at the system level and acts as a container for all the system's components. A folder or directory within the file system is represented by the "Folder" class. It can also be viewed as a "Thing" because it is an inheritor of the "Thing" class. It may include several 'Thing' objects (files or subfolders). This class is required to retain a collection of things inside a folder and represent the folder structure within the file system.

However, they might be combined into a single class with further abstraction. These classes can be combined into one class that can represent the file system and specific folders with careful design and abstraction. The shared properties and functions from both types, including name, size, and actions like adding or removing objects, should be included in this class. The 'FileSystem' and 'Folder' classes will subsequently derive from this unified class, enabling a simpler method of handling the file system and specific folders while keeping the necessary capabilities.

\

3. What is wrong with the class name Thing? Suggest a better name for the class, and explain the reasoning behind your answer

The class name "Thing" is without any clear definition or context regarding what it stands for within the file system and is rather broad. "FileSystemItem" might be a better name for the class since it makes it clear that the class represents a file system item. The class's general objective, to represent diverse objects like files and directories within the file system, is well suited to the descriptive nature of the object.

4. Define the principle of abstraction, and explain how you would use it to design a class to represent a Book.

Abstraction is a fundamental concept in object-oriented programming, which entails focusing on an object's essential characteristics while ignoring the specifics that are unimportant in the context at hand. It allows for the brief and straightforward representation of complex systems.

To design a book using abstraction --

To develop a class to represent a book, we would abstraction away the minor specifics and focus on the fundamental traits and behaviors that define it. This could consist of specifics about the book itself, as well as properties like the title, author, category, and publicationYear, as well as functions like getDetails(). We produce an abbreviated version of a book that developers can use and understand without getting mired down in unnecessary details.