Swinburne University of Technology

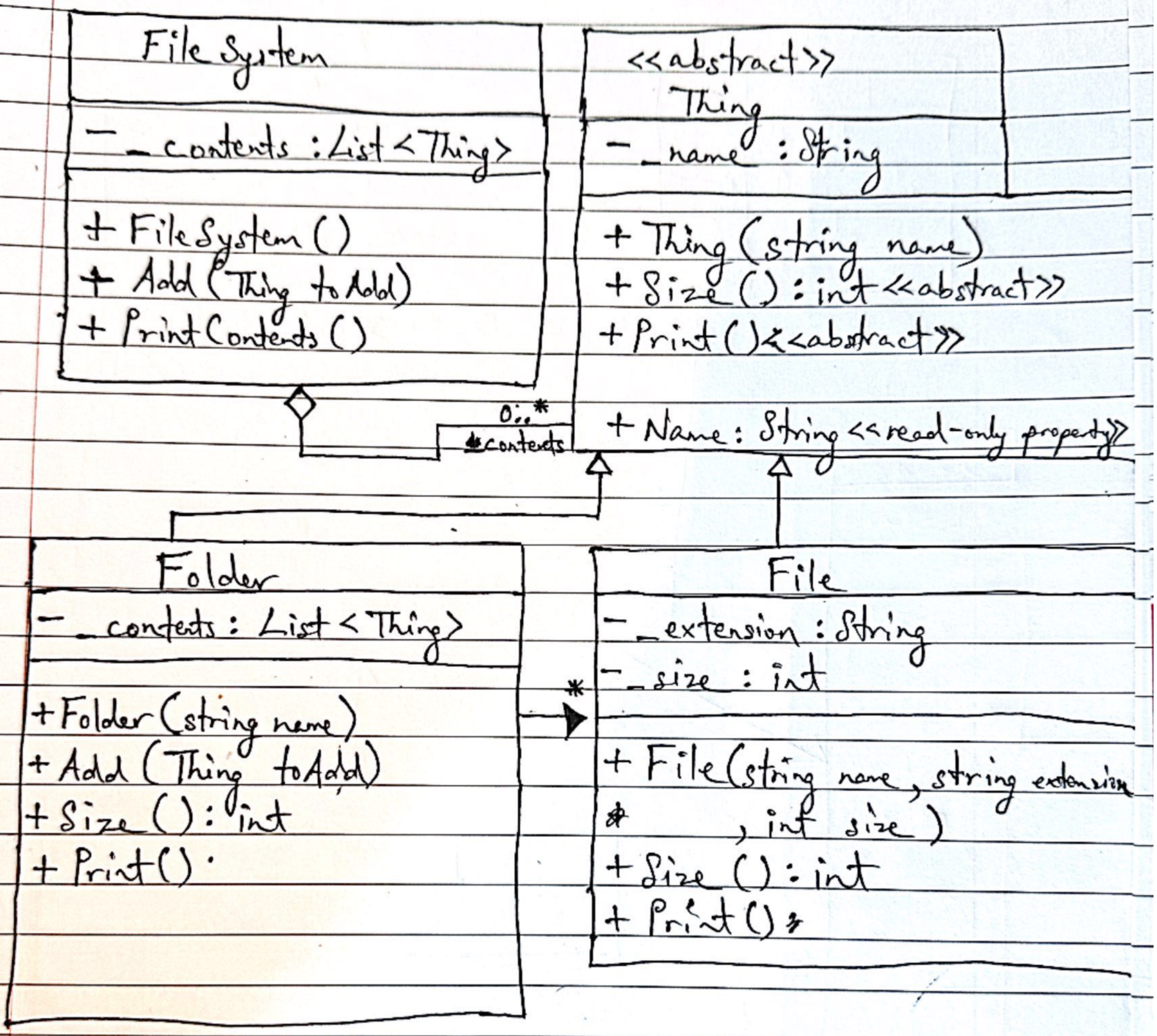
COS20007 Object Oriented Programming

Semester test

PDF generated at 07:36 on Friday 29th September, 2023

Semester Test

UML



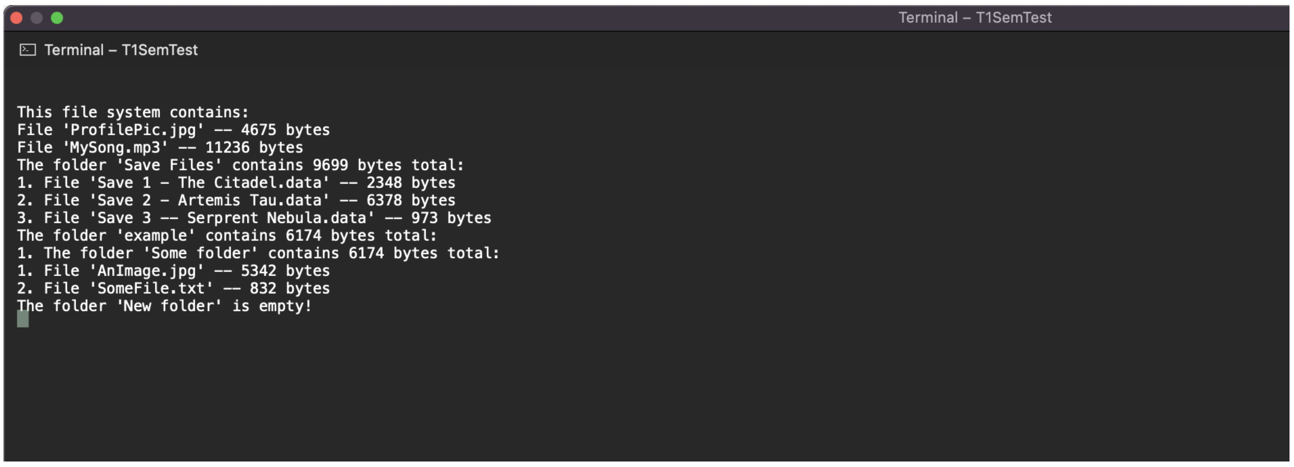
```csharp
using System;

namespace T1SemTest
{
    public class Program
    {
        public static void Main()
        {
            FileSystem fileSystem = new FileSystem();

            File file1 = new File("Save 1 - The Citadel", ".data", 2348);
            File file2 = new File("Save 2 - Artemis Tau", ".data", 6378);
            File file3 = new File("Save 3 -- Serprent Nebula", ".data", 973);
            Folder folder1 = new Folder("Save Files");
            folder1.Add(file1);
            folder1.Add(file2);
            folder1.Add(file3);

            File file4 = new File("AnImage", ".jpg", 5342);
            File file5 = new File("SomeFile", ".txt", 832);
            Folder folder2 = new Folder("Some folder");
            folder2.Add(file4);
            folder2.Add(file5);
            Folder folder3 = new Folder("example");
            folder3.Add(folder2);

            Folder folder4 = new Folder("New folder");

            File file6 = new File("ProfilePic", ".jpg", 4675);
            File file7 = new File("MySong", ".mp3", 11236);

            fileSystem.Add(file6);
            fileSystem.Add(file7);
            fileSystem.Add(folder1);
            fileSystem.Add(folder3);
            fileSystem.Add(folder4);

            fileSystem.PrintContents();


            Console.ReadLine();
        }
    }
}
```

```
using System;

namespace T1SemTest
{
    public class FileSystem
    {
        private List<Thing> _contents;

        public FileSystem()
        {
            _contents = new List<Thing>();
        }

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

        public void PrintContents()
        {
            Console.WriteLine("This file system contains:");
            foreach(Thing i in _contents)
            {
                i.Print();
            }
        }
    }
}

```

```
using System;

namespace T1SemTest
{
    public abstract class Thing
    {
        private string _name;

        public Thing(string name)
        {
            _name = name;
        }

        public string Name
        {
            get { return _name; }
        }

        public abstract int Size();
        public abstract void Print();
    }
}

```

```csharp
using System;

namespace T1SemTest
{
    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 i in _contents)
            {
                size += i.Size();
            }

            return size;
        }

        public override void Print()
        {
            if(_contents.Count() <= 0)
            {
                Console.WriteLine("The folder '" + Name + "' is empty!");
            }
            else
            {
                Console.WriteLine("The folder '" + Name + "' contains " +
 ↪  Size().ToString() + " bytes total:");

                int index = 0;
                foreach(Thing itm in _contents)
                {
                    index += 1;
                    Console.Write(index.ToString() + ". ");
                    itm.Print();
                }
            }
        }
    }
}
```

```
using System;

namespace T1SemTest
{
    public class File : Thing
    {
        private string _extension;
        private int _size;

        public File(string name, string extension, int size) : base(name)
        {
            _extension = extension;
            _size = size;
        }

        public override int Size()
        {
            return _size;
        }

        public override void Print()
        {
            Console.WriteLine("File '" + Name + _extension + "' -- " +
 ↪  Size().ToString() + " bytes");
        }
    }
}

```



```
This file system contains:
File 'ProfilePic.jpg' -- 4675 bytes
File 'MySong.mp3' -- 11236 bytes
The folder 'Save Files' contains 9699 bytes total:
1. File 'Save 1 - The Citadel.data' -- 2348 bytes
2. File 'Save 2 - Artemis Tau.data' -- 6378 bytes
3. File 'Save 3 -- Serprent Nebula.data' -- 973 bytes
The folder 'example' contains 6174 bytes total:
1. The folder 'Some folder' contains 6174 bytes total:
1. File 'AnImage.jpg' -- 5342 bytes
2. File 'SomeFile.txt' -- 832 bytes
The folder 'New folder' is empty!
```

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

Polymorphism in OOP is a principle that allows objects of different classes to be treated as objects of a common class. This is done using inheritance where different classes inherit from the same parent class, inheriting the parents' fields and methods. This also allows the objects created from the children classes to be of more than one class type.

In Task 1, the Folder and File classes are polymorphic as they inherit from the Thing class. This means they inherit the _name field and the Name, Size and Print methods from the Thing class. Also, objects created from either of these classes belong to both the child and parent class i.e. an object created from the File class would be of type File and of type Thing.

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:

We do need both of these classes because even though they are quite similar, they don't represent the same thing. If we apply the principle of abstraction, in this program an abstraction of a file system component and an abstraction of a folder component are both necessary. If we were to remove one of these classes and adjust the other to act as both of these components, it would result in the class being cluttered and confusing.

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 quite vague and doesn't represent its child classes accurately enough. A better class name would be "Item" as the term items are more commonly associated with lists and directories.

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

In OOP, abstraction is the principle of constructing the ideas and blueprints of each component in a program before the code is written. If we were to create a Book class using abstraction, at first, we would try to determine how many components there were and what their responsibilities/characteristics were:

- A Book has a title, an author, a date published.
- A book has many pages.
- You should be able to turn to the next and previous page.
- A page has text on it.
- A page is supposed to be read.

Using these key ideas of what our programs characteristics/responsibilities are, we can start constructing our abstractions.

There should be two classes, a Page class, and a Book class. The Page class should contain a text field that contains the writing of the page and a method that displays the text. The Book class should contain fields and properties for its title, author and publish date. It should also contain a List field containing Page objects which represent each page of the book and it should have a method that allows the user to turn to the next or previous Page item in the list.