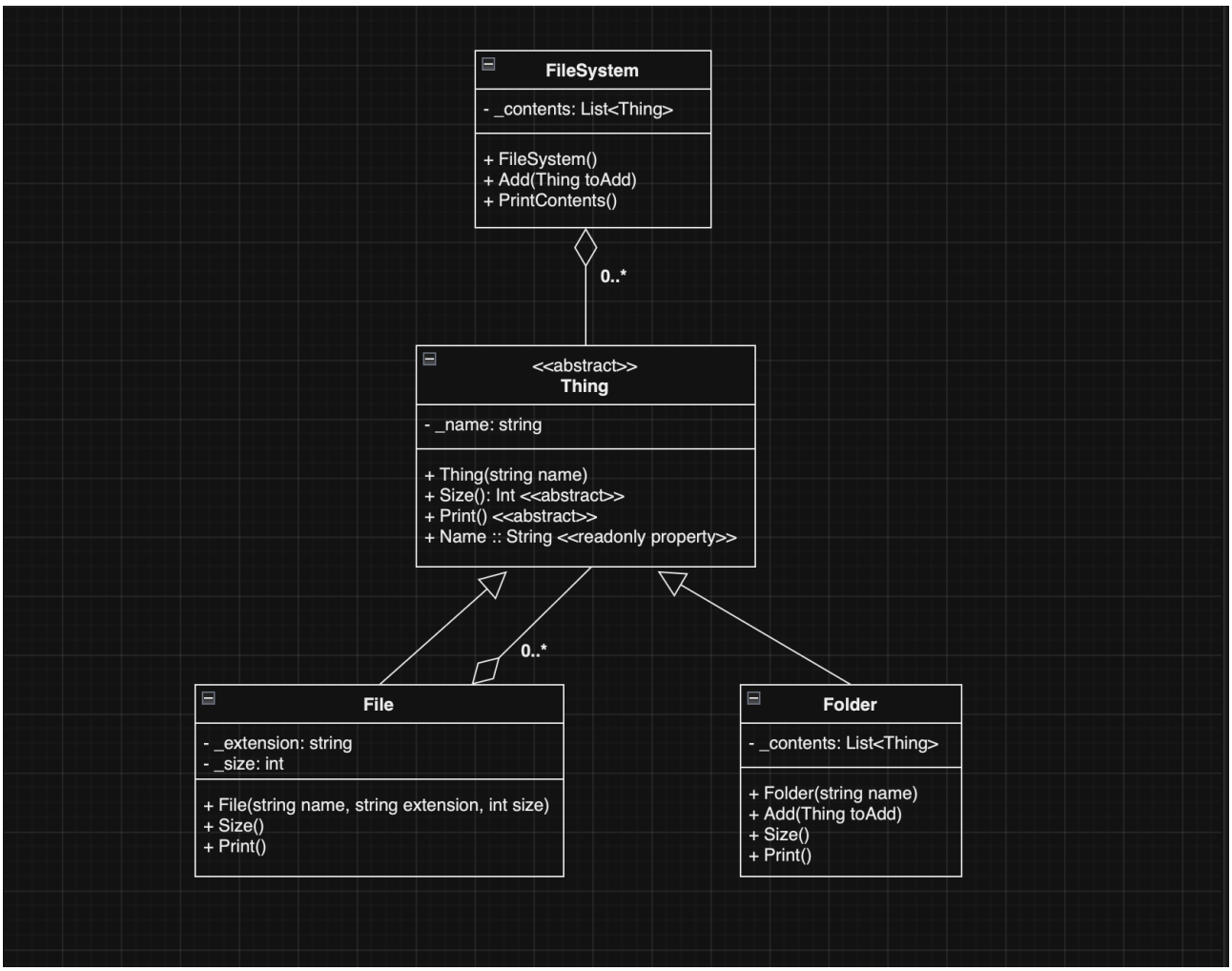
Swinburne University of Technology

COS20007 Object Oriented Programming

Semester test

PDF generated at 22:12 on Wednesday 27th September, 2023


FileSystem
- _contents: List<Thing>
+ FileSystem()
+ Add(Thing toAdd)
+ PrintContents()
0..*
<<abstract>>
Thing
- _name: string
+ Thing(string name)
+ Size(): Int <<abstract>>
+ Print() <<abstract>>
+ Name :: String <<readonly property>>
0..*
File
- _extension: string
- _size: int
+ File(string name, string extension, int size)
+ Size()
+ Print()
Folder
- _contents: List<Thing>
+ Folder(string name)
+ Add(Thing toAdd)
+ Size()
+ Print()

```
using System;

namespace SemesterTest
{
    public class Program
    {
        public static void Main(string[] args)
        {
            FileSystem fileSystem = new FileSystem();
            Folder semesterTest = new Folder("Semester Test");
            semesterTest.Add(new File("socket_server", "py", 1024));
            semesterTest.Add(new File("socket_client", "py", 512));
            fileSystem.Add(semesterTest);

            Folder nothing = new Folder("Nothing");
            fileSystem.Add(nothing);

            File wireShark = new File("wiresharkCapture", "pcap", 464);
            File headerJSON = new File("header", "json", 16384);
            fileSystem.Add(wireShark);
            fileSystem.Add(headerJSON);

            fileSystem.PrintContents();
        }
    }
}

```

```
using System;
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()
        {
            Console.WriteLine("This file system contains:");
            foreach(Thing? thing in _contents)
            {
                thing.Print();
            }
        }
    }
}

```

```
using System;
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 => _name;}
    }
}

```

```
using System;
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 totalSize = 0;
            foreach(File file in _contents)
            {
                totalSize += file.Size();
            }

            return totalSize;
        }

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

            foreach (File file in _contents)
            {
                file.Print();
            }
        }
    }
}

```

```
using System;
namespace SemesterTest
{
    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} bytes");
        }

    }
}

```

```
❯ dotnet run .
This file system contains:
The folder 'Semester Test' contains 1536 bytes total:
File 'socket_server.py' -- 1024 bytes
File 'socket_client.py' -- 512 bytes
The folder 'Nothing' is empty!
File 'wiresharkCapture.pcap' -- 464 bytes
File 'header.json' -- 16384 bytes
```

Name: Xuan Tuan Minh Nguyen
Student ID: 103819212

Semester Test Answer Sheet

- **Describe the principle of polymorphism and how it was used in Task 1**
    - The term polymorphism is one of the four essential terms in object-oriented programming. Polymorphism will let objects from certain classes to be treated like objects of a super class that those classes are inherited. Looking on Task 1, both *File* and *Folder* classes are inherited from the abstract class *Thing*, this allows both classes to be treated as *Thing* objects. Thus allowing the *FileSystem* to add both files and folders in a unified way and called to the common methods, such as *Print()* without exactly knows which child objects to be called.
- **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. Although *FileSystem* and Folder classes could have multiples items (Files or Folders), both are served for different purposes. While *Folder* class represents individual directories that can contain other sub-folder or files, *FileSystem* acts as the entire file system where contain all folders and files. By separate the *FileSystem* and *Folder*, we can respectively distinct the behaviors and properties of the entire file system and individual folders.
- **What is wrong with the class name Thing? Suggest a better name for the class, and explain reasoning behind your answer**
    - The class name *Thing* is a very generic name, and it does not show any specific information or purposes of the objects. Thus, a new name of *FileSystemObject* or *FSObject* would be a more clarify name for *Thing* object as it represents that a file or a folder is a part of the file system.
- **Define the principle of abstraction, and explain how would you use it to design a class to represent a Book**
    - Beside from polymorphism, abstraction is another important concept of object-oriented programming. This concept allows users to interact with things (high-level interface) that they do not specifically know how it works. When implementing a class to represent a *Book*, abstraction would help to determine which important attributes and methods that a book should have. For example, a book should include the attributes of *title, author* and *number of pages* and methods could be *Open(), Read(), GetBookInfomation()* and *Close()*. The implementation inside these methods could be hidden from the user, allowing them to interact with the Book object without understanding all of the complex steps underlying.