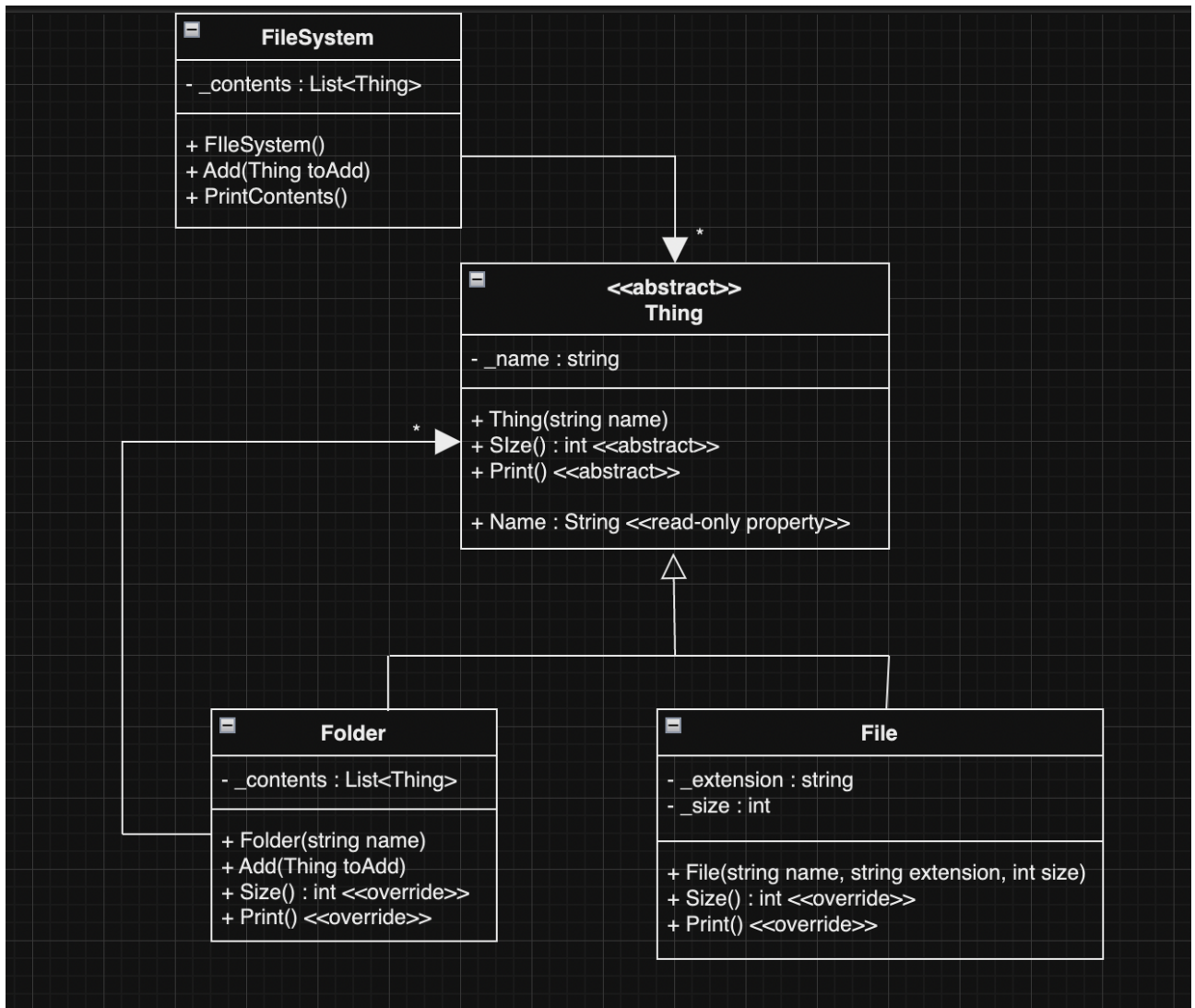


SWINBURNE UNIVERSITY OF TECHNOLOGY

COS20007 OBJECT ORIENTED PROGRAMMING

Semester test

PDF generated at 22:17 on Sunday 29th October, 2023



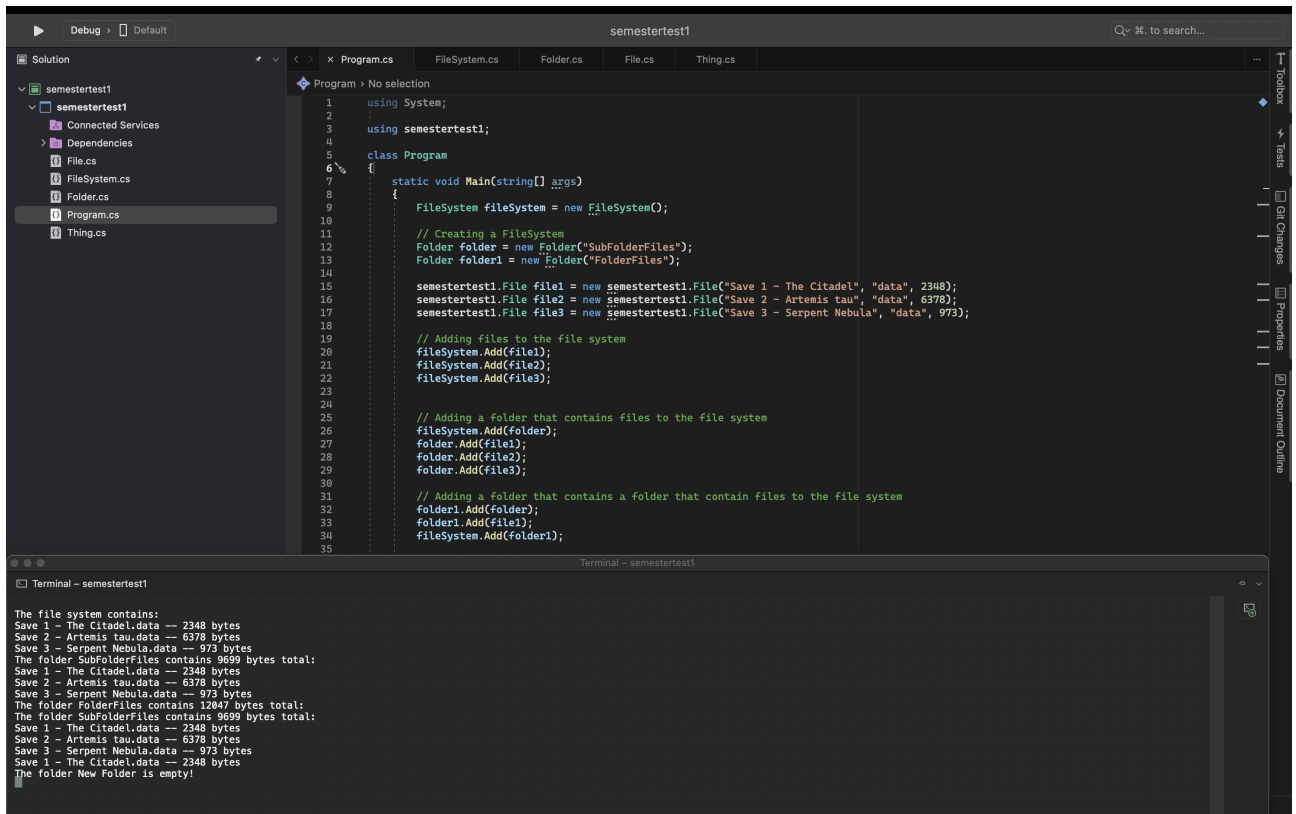
```
1  using System;
2
3  using semestertest1;
4
5  class Program
6  {
7      static void Main(string[] args)
8      {
9          FileSystem fileSystem = new FileSystem();
10
11         // Creating a FileSystem
12         Folder folder = new Folder("SubFolderFiles");
13         Folder folder1 = new Folder("FolderFiles");
14
15         semestertest1.File file1 = new semestertest1.File("Save 1 - The Citadel",
↵ "data", 2348);
16         semestertest1.File file2 = new semestertest1.File("Save 2 - Artemis tau",
↵ "data", 6378);
17         semestertest1.File file3 = new semestertest1.File("Save 3 - Serpent Nebula",
↵ "data", 973);
18
19         // Adding files to the file system
20         fileSystem.Add(file1);
21         fileSystem.Add(file2);
22         fileSystem.Add(file3);
23
24
25         // Adding a folder that contains files to the file system
26         fileSystem.Add(folder);
27         folder.Add(file1);
28         folder.Add(file2);
29         folder.Add(file3);
30
31         // Adding a folder that contains a folder that contain files to the file
↵ system
32         folder1.Add(folder);
33         folder1.Add(file1);
34         fileSystem.Add(folder1);
35
36
37         // Adding an empty folder to the file system
38         Folder emptyFolder = new Folder("New Folder");
39         fileSystem.Add(emptyFolder);
40
41         // Calling the PrintContents method
42         fileSystem.PrintContents();
43     }
44 }
```

```
1  using System;
2  using System.Collections.Generic;
3
4  public class FileSystem
5  {
6      private List<Thing> _contents;
7
8      public FileSystem()
9      {
10         _contents = new List<Thing>();
11     }
12
13     public void Add(Thing toAdd)
14     {
15         _contents.Add(toAdd);
16     }
17
18     public void PrintContents()
19     {
20         Console.WriteLine("The file system contains:");
21         foreach (Thing thing in _contents)
22         {
23             thing.Print();
24         }
25     }
26 }
```

```
1  using System;
2  using System.Collections.Generic;
3
4  public abstract class Thing
5  {
6      private string _name;
7
8      public Thing (string name)
9      {
10         _name = name;
11     }
12
13     public abstract int Size();
14     public abstract void Print();
15
16
17     public string Name
18     {
19         get
20         {
21             return _name;
22         }
23     }
24 }
```

```
1  using System;
2
3  namespace semestertest1
4  {
5      public class Folder : Thing
6      {
7          private List<Thing> _contents;
8
9          public Folder(string name) : base(name)
10         {
11             _contents = new List<Thing>();
12         }
13
14         public void Add(Thing toAdd)
15         {
16             _contents.Add(toAdd);
17         }
18
19         public override int Size()
20         {
21             int size = 0;
22             foreach (var thing in _contents)
23             {
24                 size += thing.Size();
25             }
26             return size;
27         }
28
29         public override void Print()
30         {
31             if (_contents.Count == 0)
32             {
33                 Console.WriteLine($"The folder {Name} is empty!");
34             }
35
36             else
37             {
38                 Console.WriteLine($"The folder {Name} contains {Size()} bytes total:
39 ↵ ");
40                 foreach (Thing thing in _contents)
41                 {
42                     thing.Print();
43                 }
44             }
45         }
46     }
47 }
48
49
```

```
1
2 using System;
3 using System.Collections.Generic;
4
5 namespace semestertest1
6 {
7     public class File : Thing
8     {
9         private string _extension;
10        private int _size;
11
12        public File(string name, string extension, int size) : base(name)
13        {
14            _extension = extension;
15            _size = size;
16        }
17
18        public override int Size()
19        {
20            return _size;
21        }
22
23        public override void Print()
24        {
25            Console.WriteLine($"{Name}._extension} -- {_size} bytes");
26        }
27    }
28 }
29
```



The screenshot shows the Visual Studio IDE with a C# project named 'semestertest1'. The 'Program.cs' file is open, displaying the following code:

```
1 using System;
2
3 using semestertest1;
4
5 class Program
6 {
7     static void Main(string[] args)
8     {
9         FileSystem fileSystem = new FileSystem();
10
11         // Creating a FileSystem
12         Folder folder = new Folder("SubFolderFiles");
13         Folder folder1 = new Folder("FolderFiles");
14
15         semestertest1.File file1 = new semestertest1.File("Save 1 - The Citadel", "data", 2348);
16         semestertest1.File file2 = new semestertest1.File("Save 2 - Artemis tau", "data", 6378);
17         semestertest1.File file3 = new semestertest1.File("Save 3 - Serpent Nebula", "data", 973);
18
19         // Adding files to the file system
20         fileSystem.Add(file1);
21         fileSystem.Add(file2);
22         fileSystem.Add(file3);
23
24         // Adding a folder that contains files to the file system
25         fileSystem.Add(folder);
26         folder.Add(file1);
27         folder.Add(file2);
28         folder.Add(file3);
29
30         // Adding a folder that contains a folder that contain files to the file system
31         folder1.Add(folder);
32         folder1.Add(file1);
33         fileSystem.Add(folder1);
34
35     }
36 }
```

The 'Terminal - semestertest1' window at the bottom shows the output of the program:

```
The file system contains:
Save 1 - The Citadel.data --- 2348 bytes
Save 2 - Artemis tau.data --- 6378 bytes
Save 3 - Serpent Nebula.data --- 973 bytes
The folder SubFolderFiles contains 9699 bytes total:
Save 1 - The Citadel.data --- 2348 bytes
Save 2 - Artemis tau.data --- 6378 bytes
Save 3 - Serpent Nebula.data --- 973 bytes
The folder FolderFiles contains 12047 bytes total:
Save 1 - The Citadel.data --- 2348 bytes
Save 2 - Artemis tau.data --- 6378 bytes
Save 3 - Serpent Nebula.data --- 973 bytes
The folder New Folder is empty!
```


- 1) Describe the principle of polymorphism and how it was used in Task 1?

Polymorphism means many forms.

It allows objects of different classes to be treated as objects of a common base class. It allows for method overriding, where a subclass can override a specific implementation of a method defined in its base class, and dynamic method binding, which means that the appropriate method is determined at runtime based on the actual type of the object.

In this task the **Folder**, **FileSystem** classes demonstrate polymorphism.

Both **FileSystem** and **Folder** are working with objects of type **Thing**. This means that you can add both folders and other items (files) to the **FileSystem** because they share a common base class.

```
public override int Size()
{
    int size = 0;
    foreach (var thing in _contents)
    {
        size += thing.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()} bytes total: ");
        foreach (Thing thing in _contents)
        {
            thing.Print();
        }
    }
}
```

- 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.

We do not need both of them. The FileSystem class function mainly as the Folder class, the FileSystem class can do the same work.

- 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 not very descriptive and lacks clarity. "Thing" is excessively generic and vague, failing to communicate the intended role or characteristics of the class effectively.

FileSystemItem: This name is more generic and indicates that the class represents items within the file system

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

Abstraction involves defining the roles and responsibilities of classes, specifying their properties, actions, and interactions with other classes. This allows for clear and reusable definition of classes within a program. Additionally, abstraction helps classify objects appropriately, ensuring they are handled correctly.

When designing a class like "Book," the concept of simplification involves focusing on the essential attributes that distinguish a book and needed for the users, such as the Title, Author, and YearPublished. These attributes capture the core information required to identify and understand a book.

Besides, unnecessary attributes such as Pages and PaperQuality could be omitted. While these details are relevant in a more detailed representation of a book, they are not considered essential for the general concept of a book. By leaving out these unnecessary attributes, we strive to create a simplified and generalized representation that captures the fundamental characteristics of a book.