Mid semester test

COS20007

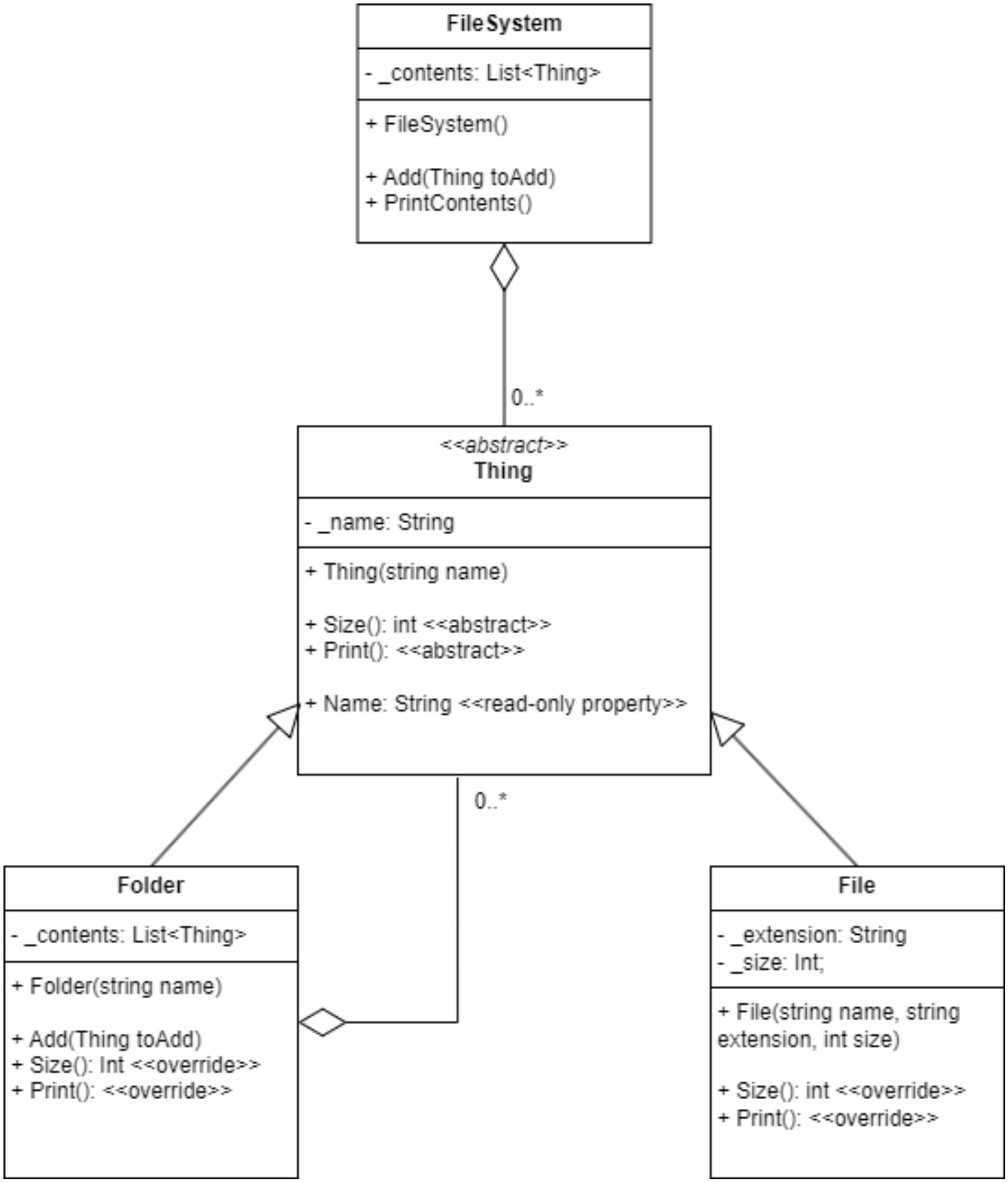
Nguyen Duc Thang

104776473

# Revision Summary:

In this resubmission i have changed the UML diagram structure by replacing the old connection arrow connected between "FileSystem" class and "Thing" class with the aggregation connection arrows. Also I have delete the connection between the "File" and "Folder" class and replace it with another aggregation connection arrow between the "Thing" class and "Folder" class to show the recursive relationships where the folder can contains folders and files.

Task 1:


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

```
1  namespace Mid_sem_test
2  {
3      public class Program
4      {
5          static void Main(string[] args)
6          {
7              int[] A = new int[10] { 2, 3, 5, 7, 11, 13, 17, 19, 23, 29 };
8              int[] B = new int[4] { A[6], A[4], A[7], A[3] };
9
10             FileSystem my_file_system = new FileSystem();
11
12
13             for(int i=0; i < B[0]; i++)
14             {
15                 if(i<10)
16                 {
17                     my_file_system.Add(new File("104776473-0" + $"{i}",
                   ".txt", 473));
18                 }
19                 else
20                 {
21                     my_file_system.Add(new File("104776473-" + $"{i}",
                   ".txt", 473));
22                 }
23
24             }
25
26             Folder my_folder_1 = new Folder("Folder have files part c");
27             for(int i=0; i < B[1]; i++)
28             {
29                 if (i < 10)
30                 {
31                     my_folder_1.Add(new File("104776473-0" + $"{i}",
                   ".txt", 473));
32                 }
33                 else
34                 {
35                     my_folder_1.Add(new File("104776473-" + $"{i}", ".txt",
                       473));
36                 }
37             }
38             my_file_system.Add(my_folder_1);
39
40             Folder my_folder_2 = new Folder("Folder have folder contains
                 files");
41             Folder my_folder_3 = new Folder("Folder have files part d");
42             for (int i = 0; i < B[2]; i++)
43             {
44                 if (i < 10)
```

```
45                  {
46                      my_folder_3.Add(new File("104776473-0" + $"{i}",
                    ".txt", 473));
47                  }
48                  else
49                  {
50                      my_folder_3.Add(new File("104776473-" + $"{i}", ".txt",
                    473));
51                  }
52              }
53              my_folder_2.Add(my_folder_3);
54              my_file_system.Add(my_folder_2);
55
56              for (int i = 0; i < B[3]; i++)
57              {
58                  my_file_system.Add(new Folder("Empty folder"));
59              }
60
61              my_file_system.PrintContents();
62          }
63      }
64  }
65
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Mid_sem_test
8  {
9      public class FileSystem
10     {
11         private List<Thing> _contents;
12
13         public FileSystem()
14         {
15             _contents = new List<Thing>();
16         }
17
18         public void Add(Thing toAdd)
19         {
20             _contents.Add(toAdd);
21         }
22
23         public void PrintContents()
24         {
25             Console.WriteLine("This File System contains:");
26             foreach (Thing? t in _contents)
27             {
28                 t.Print();
29             }
30         }
31     }
32 }
33
```

```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.Linq;
 4  using System.Text;
 5  using System.Threading.Tasks;
 6
 7  namespace Mid_sem_test
 8  {
 9      public abstract class Thing
10      {
11          private string _name;
12
13          public Thing(string name)
14          {
15              _name = name;
16          }
17
18          public abstract int Size();
19
20          public abstract void Print();
21
22          public string Name
23          {
24              get
25              {
26                  return _name;
27              }
28          }
29      }
30  }
31
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Mid_sem_test
8  {
9      public class File:Thing
10     {
11         private string _extension;
12         private int _size;
13
14         public File(string name, string extension, int size) : base(name)
15         {
16             _extension = extension;
17             _size = size;
18         }
19
20         public override int Size()
21         {
22             return _size;
23         }
24
25         public override void Print()
26         {
27             Console.WriteLine($"File '{this.Name}{_extension}' Size:
                 {_size} bytes");
28         }
29     }
30 }
31
```

```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.Linq;
 4  using System.Text;
 5  using System.Threading.Tasks;
 6
 7  namespace Mid_sem_test
 8  {
 9      public class Folder:Thing
10      {
11          private List<Thing> _contents;
12
13          public Folder(string name) : base(name)
14          {
15              _contents = new List<Thing>();
16          }
17
18          public void Add(Thing toAdd)
19          {
20              _contents.Add(toAdd);
21          }
22
23          public override int Size()
24          {
25              int cnt = 0;
26              foreach (Thing t in _contents)
27              {
28                  cnt += t.Size();
29              }
30              return cnt;
31          }
32
33          public override void Print()
34          {
35              if(_contents.Count == 0)
36              {
37                  Console.WriteLine($"The Folder: '{this.Name}' is empty!");
38              }
39              else
40              {
41                  Console.WriteLine($"The Folder: '{this.Name}' contains
                      {_contents.Count} files totalling {Size()} bytes:");
42              }
43
44              foreach (Thing f in _contents)
45              {
46                  f.Print();
47              }
48          }
```



```
49      }
50  }
51
```

```
This File System contains:
File '104776473-00.txt' Size: 473 bytes
File '104776473-01.txt' Size: 473 bytes
File '104776473-02.txt' Size: 473 bytes
File '104776473-03.txt' Size: 473 bytes
File '104776473-04.txt' Size: 473 bytes
File '104776473-05.txt' Size: 473 bytes
File '104776473-06.txt' Size: 473 bytes
File '104776473-07.txt' Size: 473 bytes
File '104776473-08.txt' Size: 473 bytes
File '104776473-09.txt' Size: 473 bytes
File '104776473-10.txt' Size: 473 bytes
File '104776473-11.txt' Size: 473 bytes
File '104776473-12.txt' Size: 473 bytes
File '104776473-13.txt' Size: 473 bytes
File '104776473-14.txt' Size: 473 bytes
File '104776473-15.txt' Size: 473 bytes
File '104776473-16.txt' Size: 473 bytes
The Folder: 'Folder have files part c' contains 11 files totalling 5203 bytes:
File '104776473-00.txt' Size: 473 bytes
File '104776473-01.txt' Size: 473 bytes
File '104776473-02.txt' Size: 473 bytes
File '104776473-03.txt' Size: 473 bytes
File '104776473-04.txt' Size: 473 bytes
File '104776473-05.txt' Size: 473 bytes
File '104776473-06.txt' Size: 473 bytes
File '104776473-07.txt' Size: 473 bytes
File '104776473-08.txt' Size: 473 bytes
File '104776473-09.txt' Size: 473 bytes
File '104776473-10.txt' Size: 473 bytes
The Folder: 'Folder have folder contains files' contains 1 files totalling 8987 bytes:
The Folder: 'Folder have files part d' contains 19 files totalling 8987 bytes:
File '104776473-00.txt' Size: 473 bytes
File '104776473-01.txt' Size: 473 bytes
File '104776473-02.txt' Size: 473 bytes
File '104776473-03.txt' Size: 473 bytes
File '104776473-04.txt' Size: 473 bytes
File '104776473-05.txt' Size: 473 bytes
File '104776473-06.txt' Size: 473 bytes
File '104776473-07.txt' Size: 473 bytes
File '104776473-08.txt' Size: 473 bytes
File '104776473-09.txt' Size: 473 bytes
File '104776473-10.txt' Size: 473 bytes
File '104776473-11.txt' Size: 473 bytes
File '104776473-12.txt' Size: 473 bytes
File '104776473-13.txt' Size: 473 bytes
File '104776473-14.txt' Size: 473 bytes
File '104776473-15.txt' Size: 473 bytes
File '104776473-16.txt' Size: 473 bytes
File '104776473-17.txt' Size: 473 bytes
File '104776473-18.txt' Size: 473 bytes
The Folder: 'Empty folder' is empty!
The Folder: 'Empty folder' is empty!
The Folder: 'Empty folder' is empty!
The Folder: 'Empty folder' is empty!
The Folder: 'Empty folder' is empty!
The Folder: 'Empty folder' is empty!
The Folder: 'Empty folder' is empty!
```

Task 2:

1, Polymorphism is an OOP concept that will give you the flexibility of treating an object, in other word polymorphism allow you to treat a child class as the parent class, In task 1 I have used the "abstract" and "override" syntax for the Size and Print method where the "abstract" syntax is for the parent class Thing.cs and "override" syntax for the child class Folder.cs and File.cs to change the content of the method depend on the class.

2, In the updated UML diagram in task 1, we still need both "FileSystem" and "Folder" class because each class serves different roles where the "FilesSystem" class is a global container that manage the content inside it which are the files and folders, "Folder" class is a more localize component that can store the files and itself. Separating this two class will also be helpful if in the future you want to add more type of file system.

3, The class name Thing is too general which don't really explain the functionality of the class fully, we can change this class name into "Contents" class because it have informations of a folder or a file which are the name and the size of that folder or file.

4, Abstraction is an OOP concept where the programmer provide enough necessary information for the user to use all the features that the class have In order to apply abstraction into designing a class represent "FileSystem" and "Folder" class i will have to implement methods such as Add(), PrintContents() and atributes for them like _name, _size but i have to hide the implementation inside the method for example how they work and interact with the attributes. This way the user will be able to use the method inside the class directly without the need to learn about how the method work.

5, During the task is submitted in canvas, task 4.2P and task 5.3C utilize polymorphism and abstraction:
- 4.2P you have to implement different full description output for different class using the "FullDescription" method.
- 5.3C you have to save and load different shape informations, for example the circle information saved conatins the radius of the circle while for the rectangle you need the width and height of the shape using the Save() and Load() method.