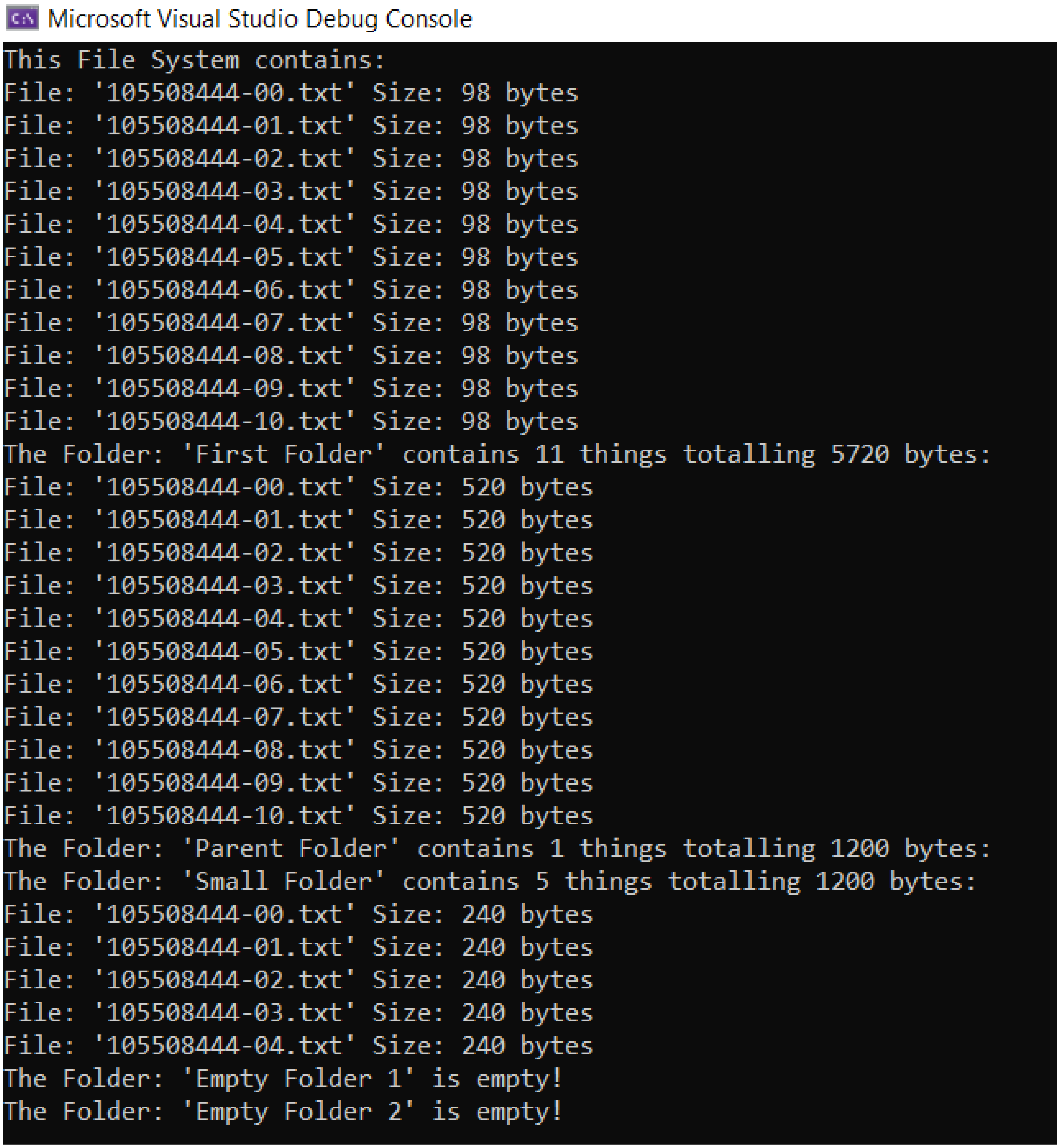
Week_8

(Hurdle_Test)

A. Task I

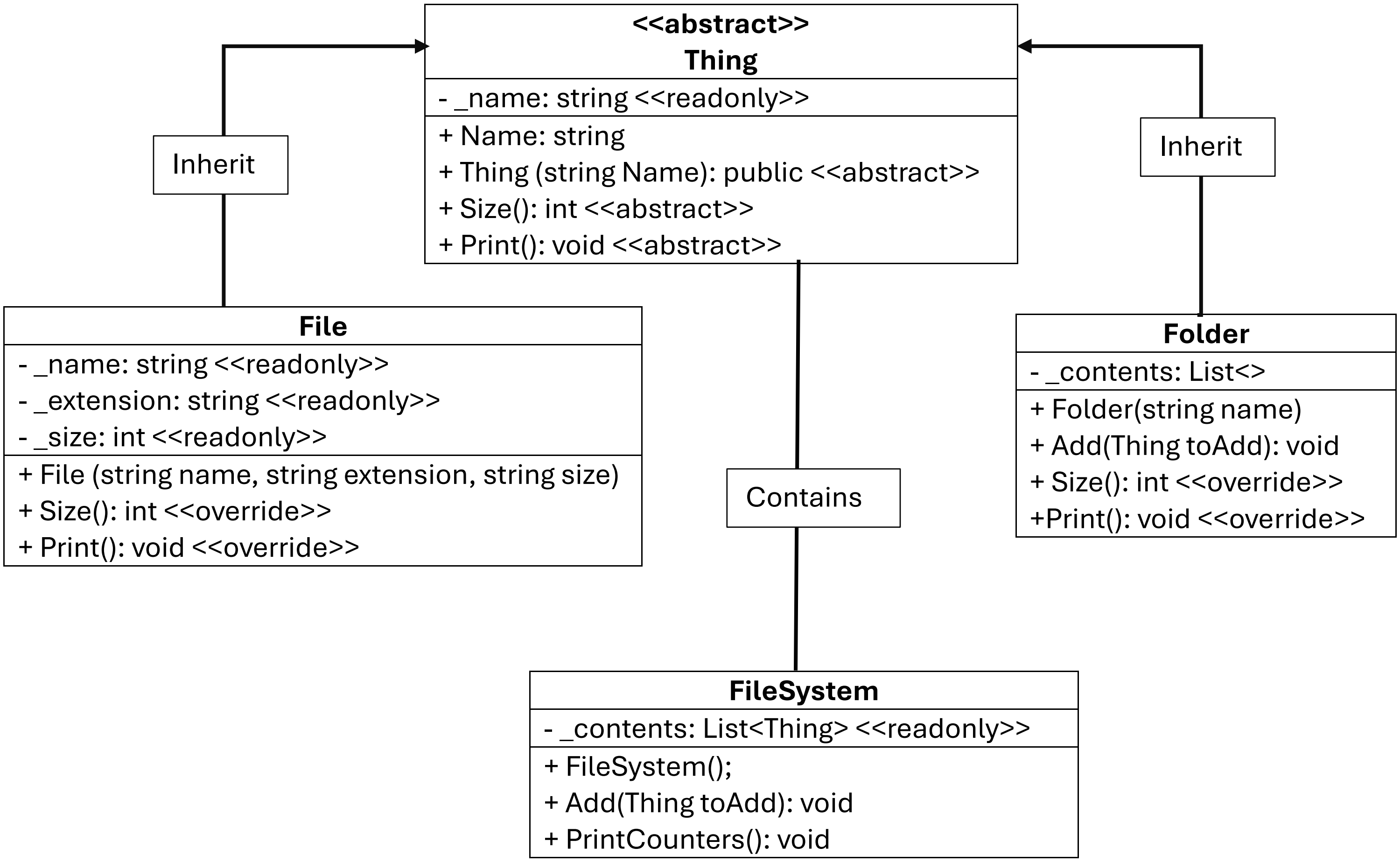
I. **Program files**: PDF files

II. **Output**



```
This File System contains:
File: '105508444-00.txt' Size: 98 bytes
File: '105508444-01.txt' Size: 98 bytes
File: '105508444-02.txt' Size: 98 bytes
File: '105508444-03.txt' Size: 98 bytes
File: '105508444-04.txt' Size: 98 bytes
File: '105508444-05.txt' Size: 98 bytes
File: '105508444-06.txt' Size: 98 bytes
File: '105508444-07.txt' Size: 98 bytes
File: '105508444-08.txt' Size: 98 bytes
File: '105508444-09.txt' Size: 98 bytes
File: '105508444-10.txt' Size: 98 bytes
The Folder: 'First Folder' contains 11 things totalling 5720 bytes:
File: '105508444-00.txt' Size: 520 bytes
File: '105508444-01.txt' Size: 520 bytes
File: '105508444-02.txt' Size: 520 bytes
File: '105508444-03.txt' Size: 520 bytes
File: '105508444-04.txt' Size: 520 bytes
File: '105508444-05.txt' Size: 520 bytes
File: '105508444-06.txt' Size: 520 bytes
File: '105508444-07.txt' Size: 520 bytes
File: '105508444-08.txt' Size: 520 bytes
File: '105508444-09.txt' Size: 520 bytes
File: '105508444-10.txt' Size: 520 bytes
The Folder: 'Parent Folder' contains 1 things totalling 1200 bytes:
The Folder: 'Small Folder' contains 5 things totalling 1200 bytes:
File: '105508444-00.txt' Size: 240 bytes
File: '105508444-01.txt' Size: 240 bytes
File: '105508444-02.txt' Size: 240 bytes
File: '105508444-03.txt' Size: 240 bytes
File: '105508444-04.txt' Size: 240 bytes
The Folder: 'Empty Folder 1' is empty!
The Folder: 'Empty Folder 2' is empty!
```

III. UML Design



<<abstract>>
Thing
- _name: string <<readonly>>
+ Name: string
+ Thing (string Name): public <<abstract>>
+ Size(): int <<abstract>>
+ Print(): void <<abstract>>
Inherit
Inherit
File
- _name: string <<readonly>>
- _extension: string <<readonly>>
- _size: int <<readonly>>
+ File (string name, string extension, string size)
+ Size(): int <<override>>
+ Print(): void <<override>>
Folder
- _contents: List<>
+ Folder(string name)
+ Add(Thing toAdd): void
+ Size(): int <<override>>
+Print(): void <<override>>
Contains
FileSystem
- _contents: List<Thing> <<readonly>>
+ FileSystem();
+ Add(Thing toAdd): void
+ PrintCounters(): void

B. Task II

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

From my perspective, “Polymorphism” means ‘a bunch of form” or “many forms which means that let different objects, share same method but have their own way of doing it. In the test, the File and Folder classe, they share Size() and Print() methods but have their own unique way of doing it (calculates the size and print their description in different ways)

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.

From my point of view, I think that FileSystem and Folder clasess are crucial because the FileSystem serves as the main container for everything at the top level and Folder is a specific container that holds other items, including other folders, inside it. We need both to create a realistic file system where you can have folders inside of other folders.

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

   The name ***Thing*** is too vague. It doesn't tell you what the class is for. A better name would be FileSystemEntry or IFileSystemComponent because it's the base for anything that can be in the file system. This makes the code easier to understand for everyone.

4. **Define the principle of abstraction, and explain how you would use it to design a class to repre sent the FileSystem and Folder classes.**

   In my opinion, Abstraction is about focusing on the big picture and hiding the complicated details. We used it by creating the **Thing** abstract class. This class tells us that all file system items have a name, a size, and can print a description. But it doesn't say *how* they do it, that part is left to the File and Folder classes to figure out on their own.

5. **Which Pass (and Credit) tasks you have submitted to Canvas utilize both the principles of polymorphism and abstraction? You can list two examples.**

   Two examples Pass (and Credit) tasks that I have submitted to Canvas which utilize both the principles of polymorphism and abstraction is ShapeDrawing and SwinAdventure tasks.