1. **Describe the principle of polymorphism and how it was used in Task 1**
   Polymorphism is like the ability of one thing to take on many forms. Think of it as having one action or command, but different things interpret and react to that command in their own unique ways.

   For example, I have a Drawing Program which has a Rectangle class and a Circle class, they are both Shapes, and both are Objects. Object can call Shape to draw itself, and Shape can ask Rectangle and Circle to draw themselves without know how to do it.

   In the case of Task 1, it is like the Drawing Program above:

   I have created a general class named “Thing”. Think of Thing as a general concept, saying "If you are a Thing, you should know how to calculate your Size and how to Print yourself".

   Now, there are two specific classes in the program, “File” and “Folder”, which are like specific types of Things. They both agree to the Thing concept. So, a “File” says, "I’m a Thing, and here is how I calculate my Size and how I Print myself", and a Folder says something similar.

   When the “FileSystem” wants to know the Size of a “Thing” or wants a “Thing” to Print itself, it does not care whether it is dealing with a “File” or a “Folder”. It just communicates with them as Things, asking them to execute their Size or Print methods. The “FileSystem” does not need to know the details of how “Files” and “Folders” calculate their Sizes or how they Print themselves, it just knows that everything knows how to perform these actions.

   So, thanks to polymorphism, I can interact with both Files and Folders in a consistent way through the Thing class. Each File or Folder knows how to calculate its Size and how to Print itself without the FileSystem needing to manage the details. This approach simplifies the design and makes the code more modular and easier to manage.

2. **Consider the FileSystem and Folder classes from the updated design in Task 1. Do we need both classes? Explain why or why not.**
   The answer is yes. Both the FileSystem and Folder classes are necessary for my design.

   FileSystem does system-wide management, Folder manages its contents. Each has special functions and attributes the other doesn’t, making both necessary for efficient file management.

   Key Functional Differences:
   - FileSystem has attributes and methods that Folder doesn’t. For example, FileSystem might store and handle global metadata and configurations. It has a wider scope.

   - FileSystem handles things like space allocation, and formatting, which are beyond Folder’s responsibility. It’s designed for bigger, system-wide tasks.

- FileSystem sets and enforces permissions on a large scale, whereas Folder's permissions are localized.

- There's usually one FileSystem, but there can be many Folders. FileSystem is the entry point; Folders are subdivisions.

3. **What is wrong with the class name Thing? Suggest a better name for the class, and explain the reasoning behind your answer.**
   "Thing" is a bit too vague; it does not tell much about what it is supposed to represent in the code. A better name would be "FileSysItem" (Item of the FileSystem). Calling it "FileSystemItem" makes it instantly clear what the class is all about: it is an item in a file system, which could be a file, a folder. It will be simpler and clearer than the name "Thing".

4. **Define the principle of abstraction and explain how you would use it to design a class to represent a Book.**
   Abstraction is about simplifying complex reality by modeling classes appropriate to the problem and working at the most relevant level of inheritance for a particular aspect of the problem. In simpler words, it means focusing on the essential features of an object while ignoring the unimportant details.

   When it comes to Book, abstraction means I only focus on the important stuff and ignore the rest. It is like looking at a book and seeing its title, author, and content but not its weight or smell because those do not matter when we are reading.

   For a Book class, the essential classes might be title, author, published year, page, and content, and I will leave the rest (such as color, feedback or other stuffs which are not necessary).