

## Task 2

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

-Polymorphism in object-oriented programming allows different objects to respond to the same method or message based on their individual types. In Task 1, it's used by having the File and Folder classes override the Print and Size methods from the common base class Thing. This enables the FileSystem class to work with a collection of mixed File and Folder objects, treating them uniformly through the common interface provided by Thing. This promotes code flexibility and reusability.

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.

-In the updated design from Task 1, both the FileSystem and Folder classes are necessary:

FileSystem Class: Represents the entire file system, providing a top-level container for organizing and managing all elements within it.

Folder Class: Represents folders or directories within the file system, enabling hierarchical organization of files and subfolders. Folders are essential for structuring and navigating the file system.

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 insufficiently descriptive and fails to convey its role within the file system context. To enhance clarity and comprehension, a more appropriate name like "FileSystemItem" or "FileSystemElement" should be used. These names explicitly indicate that the class represents an element or item within the file system, such as files, folders, or related entities. This increased specificity aids developers and maintainers in quickly grasping the class's intended function and usage, improving code readability and long-term maintainability. By opting for a more descriptive name, you contribute to a more comprehensible and maintainable codebase.

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

-The principle of abstraction in software design involves simplifying complex systems by modeling them using a set of abstract and simplified representations. It allows developers to focus on essential features and hide unnecessary details. Abstraction helps manage complexity, improves understandability, and enhances code maintainability.

- Here's a simplified example of how I could design a Book class:

```

0 references
public class Class1
{
    1 reference
    public class Book
    {
        2 references
        public string Title { get; set; }
        2 references
        public string Author { get; set; }
        2 references
        public int YearPublished { get; set; }
        2 references
        public string ISBN { get; set; }

        0 references
        public Book(string title, string author, int yearPublished, string isbn)
        {
            Title = title;
            Author = author;
            YearPublished = yearPublished;
            ISBN = isbn;
        }
        0 references
        public void DisplayInfo()
        {
            Console.WriteLine($"Title: {Title}");
            Console.WriteLine($"Author: {Author}");
            Console.WriteLine($"Year Published: {YearPublished}");
            Console.WriteLine($"ISBN: {ISBN}");
        }
    }
}

```

In this design:

-Abstraction of Attributes: We abstract the essential attributes of a book, such as title, author, year published, and ISBN, as properties of the Book class. These attributes provide a high-level view of what's important about a book without exposing unnecessary implementation details.

-Constructor: The constructor allows us to initialize a Book object with its essential attributes when it's created. This encapsulates the process of creating a book and ensures that it's always in a valid state.

-Abstraction of Behavior: The "DisplayInfo" method abstracts the behavior of displaying information about the book. Instead of exposing individual attributes, this method provides a simple way to retrieve and display all the essential book details in one go. This encapsulates the details of how the book information is presented.