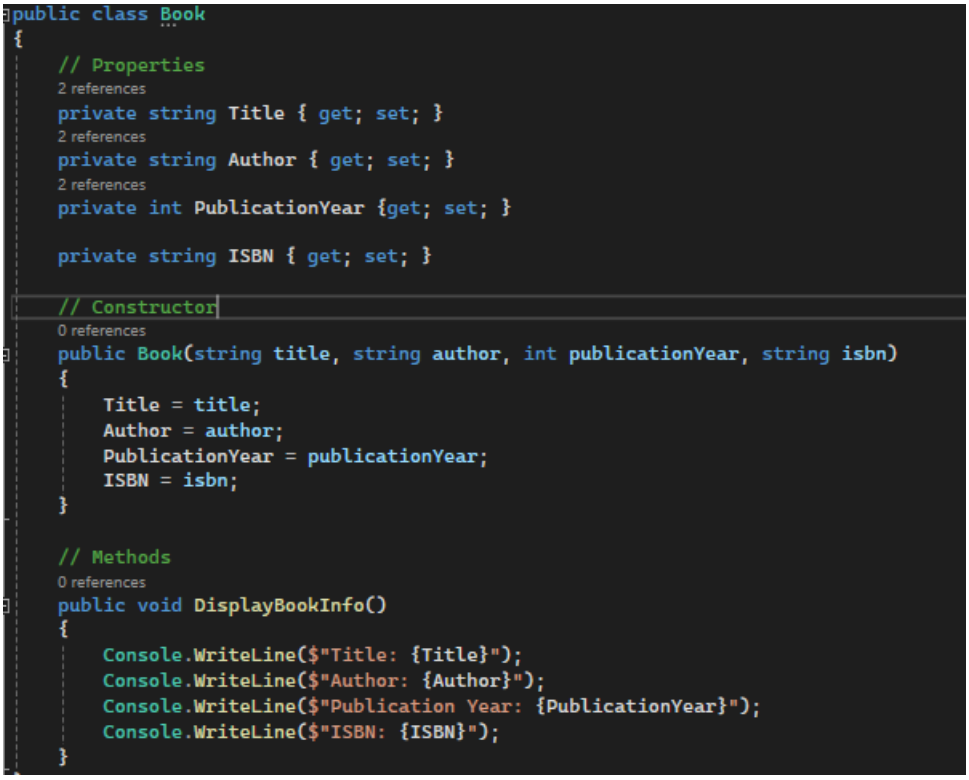
1.

- Polymorphism is a fundamental principle in object-oriented programming (OOP) where objects of different classes can be treated as objects of a common superclass, which allows the children classes can define their own definition and implementation from parent class's attributes.
- In task 1, I created a new class Thing(), which defined as parent class of Folder() and File().However, its children class, File and Folder respectively, do not calculate the size and print the content same way, so I set override attributes to Print() and Size() methods to make sure in print the different output based on different classes.

2. Yes, we need to separate the File and Folder objects, as I mentioned although they have the same parent class, they still have different logic. The folder can store a folder and file, folder can be empty, but the file can not be empty, and it does not have ability to store a file or a folder.
3. The class name "Thing" is too generic and lacks semantic meaning, making it unclear what it represents in the context of the file system. A better name for the class could be "FileSystemItem" or "FileSystemElement". These names better convey that instances of this class represent elements within the file system, such as files or folders. Using more descriptive names enhances code readability and maintainability.
4. The abstraction means it shows only the necessary attributes and hides the complex implementation details of a class. It allows us to focus on what something does rather than how it does.
   To design a class to represent a book using abstraction, we would identify the essential properties and behaviors of a book while abstracting away unnecessary details. Some essential properties of a book might include its title, author, publication year,... and e method to print all of that information.



```
public class Book
{
    // Properties
    2 references
    private string Title { get; set; }
    2 references
    private string Author { get; set; }
    2 references
    private int PublicationYear {get; set; }

    private string ISBN { get; set; }

    // Constructor
    0 references
    public Book(string title, string author, int publicationYear, string isbn)
    {
        Title = title;
        Author = author;
        PublicationYear = publicationYear;
        ISBN = isbn;
    }

    // Methods
    0 references
    public void DisplayBookInfo()
    {
        Console.WriteLine($"Title: {Title}");
        Console.WriteLine($"Author: {Author}");
        Console.WriteLine($"Publication Year: {PublicationYear}");
        Console.WriteLine($"ISBN: {ISBN}");
    }
```

This Book class hides how these internal information is stored, but provide method that easily print all these information. The abstraction allows users interact with the book without needing to know the implement details.