TASK 2

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

- Polymorphism enables objects of various classes to be handled as instances of a shared superclass, allowing a single action to be executed in multiple ways. In Task 1, polymorphism was achieved by using abstract class “Thing”, then there are two classes inherited from class “Thing”, which are “File” and “Folder”. The “Thing” class has two abstract methods “Size()” and “Print()”, which must be overridden by two classes “File” and “Folder”.

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 context of updating the design from Task 1, I think that we need both of the “FileSystem” and “Folder” classes because of their distinct roles and responsibilities within the program. The “FileSystem” acts as the main manager of the file structure; it provides methods for adding and printing the contents of files and folders at root level. Contrary to this, the Folder class presents a class that can contain files and other folders. This separation can ensure a clear and easy-to-maintain design, with “FileSystem” handling top level management and “Folder” solving inner structures.

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 quite general, and it does not provide much information about what the class represents in the context of a file system. The file system might store Folders and Files instead of “Thing”. Therefore, using a more descriptive name can make the code more readable and understandable.
- The suggested name for the class should be “FileSystemContent”. This name shows that instances of these classes will be the content within a file system, which could be either a Folder or File. This name is more descriptive than “Thing”, and it is easier for people to read the code and understand what this class does.

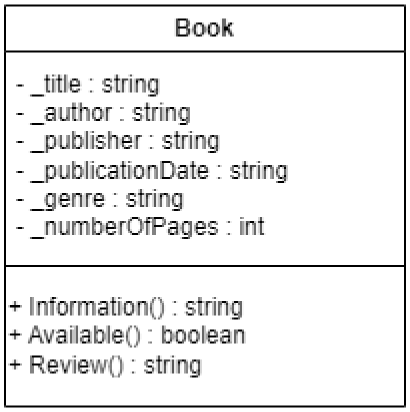
4. **Define the principle of Abstraction and explain how you would use it to design a class to represent a Book.**

- Abstraction is the process of hiding irrelevant details and only showing the essential features of the object. It allows you to focus on what it does instead of how it does it. Abstraction will provide simplicity, reusability and flexibility for the code; it will make the code easier to read and understand.

- In designing a Book:
  - First, we have to look at the essential characteristics and behaviors of a book.

+ The characteristics (Properties) could include title, author, publisher, publicationDate, genre, numberOfPages.
+ The behaviors (Methods) could include information(), available(), review().

- Then we will create a class name Book and declare the private variables as described above. The methods information() will return all information about the books; the available() method will check if the book is available in the library; the review() method will give the reviews of other readers about the book.



⇨ In this class Book, abstraction will help us to hide the implementation of the methods, such as how to check the availability of the book, how to get the book information and how to get the reviews of the book from others. We just know that there are those methods associated with the book and we just use it.