Name: Xuan Tuan Minh Nguyen
Student ID: 103819212

Semester Test Answer Sheet

- **Describe the principle of polymorphism and how it was used in Task 1**
    - The term polymorphism is one of the four essential terms in object-oriented programming. Polymorphism will let objects from certain classes to be treated like objects of a super class that those classes are inherited. Looking on Task 1, both *File* and *Folder* classes are inherited from the abstract class *Thing*, this allows both classes to be treated as *Thing* objects. Thus allowing the *FileSystem* to add both files and folders in a unified way and called to the common methods, such as *Print()* without exactly knows which child objects to be called.
- **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**
    - Yes. Although *FileSystem* and Folder classes could have multiples items (Files or Folders), both are served for different purposes. While *Folder* class represents individual directories that can contain other sub-folder or files, *FileSystem* acts as the entire file system where contain all folders and files. By separate the *FileSystem* and *Folder*, we can respectively distinct the behaviors and properties of the entire file system and individual folders.
- **What is wrong with the class name Thing? Suggest a better name for the class, and explain reasoning behind your answer**
    - The class name *Thing* is a very generic name, and it does not show any specific information or purposes of the objects. Thus, a new name of *FileSystemObject* or *FSObject* would be a more clarify name for *Thing* object as it represents that a file or a folder is a part of the file system.
- **Define the principle of abstraction, and explain how would you use it to design a class to represent a Book**
    - Beside from polymorphism, abstraction is another important concept of object-oriented programming. This concept allows users to interact with things (high-level interface) that they do not specifically know how it works. When implementing a class to represent a *Book*, abstraction would help to determine which important attributes and methods that a book should have. For example, a book should include the attributes of *title, author* and *number of pages* and methods could be *Open(), Read(), GetBookInfomation()* and *Close()*. The implementation inside these methods could be hidden from the user, allowing them to interact with the Book object without understanding all of the complex steps underlying.