Syarif Hidayatullah  
Unibas-Sweng

Exercise 1:

* Examples of separately defined Interfaces in GanttProject:

There are a lot of separately defined interfaces in GanttProject. I picked two Interfaces as examples. The first Interface is the UIFacade Interface which is implemented –among others– in UIFacadeImpl class. The second Interface is the Document interface which is implanted –among others– in FileDocument class.

* Short description of how this separation works

As explained in the Java doc, interface form a contract between the class and the outside world. If a class implemented an Interface, it is enforced to implements all methods defined in that interface. It means that your code will not be compiled successfully otherwise.

Example: The method getFileName is defined in Document Interface. The class FileDocument implemented this Interface. Therefore all the method defined in the Document Interface are implemented in this class, including the getFileName method. The implementation method is marked by the @override annotation (not necessary).

To use the getFileName method, we need a reference of the implementing class. Either as an Object which is declared globally like in ProxyDocument class or as a Paramter like in open method in GanttProject class.

* Possible reasons why it is useful to do so

Using an Interface

Exercise 2:

* Pros & cons of using Optional
* Pros & cons of using Exception
* When to use which method
* Which method is better in this particular case

Exercise 3:

* What is hidden in ADT

The detail implementation of the functionalities are encapsulated.

* State management of each implementation

In OOP the state of the stack is managed by the attribute of the stack class.

* Why is the OO implementation looks so natural for us?

Because it is easier for us to think of a Stack as an Object.

* Advantages of OOP vs functional programming and vice verse

OOP is good when you have a fixed set of functionalities on an Object. You can add new Object by implement existing functionality and add some special functionality to the new Object, without changing the existing Objects.

The functional approach is good if you have a fixed set of Objects. When your application grows, you can add functionalities to an existing object, without changing the existing functionalities of this object.

Exercise 4:

* Which problems are solved by the new module system of java 9
* What input is expected in requires clause
* What input is expected in export clause
* When is it useful to specify use and provides clauses