COMP47480 Practical 4: Object-Oriented Principles

Sukrat Kashyap (14200092)

3 April 2018

1 Work Done

The first task was to find the violation of the open-closed principle of PostageStamp and Square Stamp such that PostageStamp is closed for client change which is the Square and open for extensions. The violation found in the OCP.java was that the PostageStamp was tightly coupled with the Square class. This tight coupling closed the PostageStamp class for further extensions as changing the shape of stamp requires changing the PostageStamp class itself. To fix the issue, an interface named Shape is created which becomes the argument and shape of stamp for the PostageStamp. Now, other classes which can act as a shape to the PostageStamp class must implement the shape interface and can be passed without changing the PostageStamp and is extensible.

The second task requires finding the violation of Single responsibility principle. We know that the Hexapod is actually a dog and a human. Hence, we created two classes namely Dog and Human which dealt with task related to themselves only. These two objects were then created and combined by the Hexapod.

The third task, wanted us to find and correct the violation of the law of Demeter. We found that Customer was depended on Wallet. The Shopkeeper was depended on the Customer. But the Shopkeeper was accessing the Wallet class on which it was not directly dependent and hence violating the law of Demeter. To fix the issue, 2 new methods corresponding to the usage in Shopkeeper class were created in Customer and used in Shopkeeper class making dependency of Shopkeeper only up till Customer.

2 Reflections

The co-author of the Agile Manifesto Robert Cecil Martin is promoter of SOLID design principles. SOLID is mnemonic for five design principles for object oriented programming. It is intended to make software designs more understandable, flexible and maintainable.

The five design principles are as follows:

- Single responsibility principle: A class should have only a single responsibility. The responsibility should be entirely encapsulated by the class.

- Open-closed principle: Software entities must be open for extension but closed for modification.
- Liskov substitution principle: Objects in a program should be replaceable with the instances of their subtypes without altering the correctness of that program.
- Interface segregation principle: client-specific interfaces are better than one general-purpose interface.
- Dependency inversion principle: objects and specification should depend on abstractions and not on the concretions.

There were 2 other principles that were shown in the class. They were "No concrete superclasses" and "Law of demeter". No concrete superclasses recommends that all superclasses in a system be abstract and Law of demeter states that each unit should have only limited knowledge about other units that is only units closely related to current unit. It also states that each unit should only talk to its friends and not to strangers and can talk to their immediate friends.

In this, practical we worked on three design principles out of the 7 principles stated above namely: Single responsibility principle, Open-closed principle and law of demeter.