Assignment 3

Jaedyn Damms
Department of Computer Engineering
University of Auckland
Auckland, New Zealand
Email: jdam534@aucklanduni.ac.nz

I. What is Object-Oriented Design

Firstly, to be able to analyze the quality of my objected oriented design we need to establish what object-oriented design is and what makes a 'good' object-oriented design. [1] States that object-oriented design is a structuring method that is applied to software development to create software decomposition and usability. Simply, software actions are performed on different objects of a certain type. Over time software developers have become aware that if you base their structure on the object types rather than their actions, you will create a more modular and reusable software system. This concept is commonly known to software developers as a class. Object-oriented design can be stripped down into it's smaller elements to judge and improve the quality of your object-oriented design. To analyze the quality of my object-oriented design I am going to review smaller concepts such as inheritance, polymorphism, class sizes and other 'good' object-oriented design requirements to see if they comply with industry standards. Mostly I will be referring to [2] to see how well I followed the first five object-orientated design principles which are:

- Single-responsibility principle
- Open-closed principle
- Liskov substitution principle
- Interface segregation principle
- Dependency inversion principle

II. My Object-Oriented Design

The first principle of object-oriented design (OOD) states *"A class should have one and only one reason to change, meaning that a class should have only one job"* [2]. If we take a look at my design I have five classes: kalah, board, building, house and store. Kalah's job is to interface the player's inputs with the game's logic. The board's job is to control the game's logic, that is making sure the rules of the game are being complied with. The building class is a parent class to house and store, it's job is to simply define and declare what actions houses and stores have in common, which is mainly for code re usability and cohesion. House and store together just define what actions are exclusive to their relative class.

The second OOD principle states *"Objects of entities should be open for extension, but closed for modification"* [2]. My OOD design follows this to an extent as my design is modulated, actions are pulled out into their own functions which are then called when it needs to be. This means my functions can be altered to account for extensions to my game. For example, if you wanted to add another rule that causes the game to end all you would have to do is add a case for your given rule inside my "checkGameStatus" function in my board class.

The third OOD principle states *"Every subclass/derived class should be substitutable for their base/parent class"* [2]. This is essentially the concept familiarly known as polymorphism and inheritance. I can see my design following the principle by looking at my building, house and store class. The building class is a pure base parent class for my houses and stores, thus my houses and stores inherit variables and seeds declared inside the parent class. If I were to change the type of my houses and stores to type "building" you can see that this will not break my code as they're inherited and substitutable.

The fourth OOD principle is not relate-able to my design as I did my abstractions through polymorphism and inheritance rather than through interfacing. However I can relate to the last OOD principle *"Entities must depend on abstractions not on concretions. It states that the high level module must not depend on the low level module, but they should depend on abstractions"* [2]. This OOD principle is something I can work on in the future. I have developed a lot of code that complies with this principle by relying on parameter inputs and using return values to then update values. An example is my "updateValues" function within my board class, which returns values which the caller then uses. However I do have areas of code which contradict this principle, for example my "getSeedsInHouse" function which stores a value for the number of seeds in a house directly to a variable that the board class depends on.

Overall I do believe my design has followed the first five principles of object-oriented design, however there are ares in which my OOD can be improved, in particular improving on the fifth principle.

References

[1] B. Meyer, Object-Orientated Software Construction, 2nd ed. Santa Barbrara: ISE Inc., 2018.

[2] "S.O.L.I.D: The First 5 Principles of Object Oriented Design", Scotch, 2018. [Online]. Available: https://scotch.io/bar-talk/s-o-l-i-d-the-first-five-principles-of-object-oriented-design. [Accessed: 01- May- 2018].