Assignment3

Applying Design Pattern

Name: YaruLiu

Student ID: 855308491

Student Email: *yilu589@aucklanduni.ac.nz*

I. Command Pattern

In command pattern, in general, if we use command pattern, we need **client** that used to provide requirement for command, **command** that used to set different commands which have a method in implementation, **invoker** that used to receive command and transfer command, and **Receiver** that used to receive command from Invoker.

In Kalah that I developed, I create an abstract class, **command**, which has an abstract method **execute()** and a field in protected type, so that implementation class can implement this abstract method to perform different functions. In this project, there are a total of four commands that implement the command class, namely Command_LoadGame, Command_NewGame, Command_Quit, Command_SaveGame. These four classes correspond to the four operations that the user wants to select. In each class, it overrides the command class execute() method to achieve different functions. Next, I also created an **invoker** class to call different implementation classes of command according to requirements. In this class, there is only one field, command, and two methods setCommand() (set the corresponding command) and action() (execute Corresponding method in the corresponding command). In the main function, I only need to create an invoker object to call different commands to achieve different functions. The **client** is the main class that is kalah.java, the responsibility of it is to create some commands and define when we need to use which command. The **Receiver** is also the main class, because the main class define when and where the command will be invoked.

What I want to explain is that in the execute() method in the command and the action() method in the invoker, I set several parameters, io, two players and two caretakers (explained in the memento pattern later). Use these parameters passes the two players in the main function to change their values, and io is used to print some content.

The reason of why design is appropriate in this project is that there are different operations from users, each operation is a command released by the user. If the command mode is not used, the behavior request and the implementer are usually tightly coupled. In some cases, such as when the behavior needs to be recorded, undone, or redone, this tightly coupled design that cannot resist changes is not suitable. However, in command pattern, it can uncouple the relationship between the behavior request and the implementer, as a result, users can achieve undo, redone and other operations.

II. Memento Pattern

In Memento pattern, the intent of it is to capture the internal state of an object and save this state outside the object under the premise of not destroying the encapsulation. And there are three main class, **memento** (Contains the state of the object to be restored), originator (Create and store state in a Memento object), and caretaker (Restore the state of the object from Memento) respectively.

In Kalah I developed, I create Memento_SaveGame class in order to store the state of save by players, and only one field in it that is state with getter and setter methods. The originator is play in this project. In the player class, I add save_state field and createMemento(), restoreMemento() methods to save and get state to memento. As for Caretaker, there is one field, Memento_SaveGame and getter and setter methods. In the main class, I create two caretaker in order to store save states of both player and opponent respectively. When players input s, the memento can store the state of save that is true, through Command_SaveGame, change the save state into true. Therefore, in the main class, when save states of player and opponent are both false, printing No save game.

The reason of why design is appropriate is that memento pattern provides a recover machine. It can save the state of history, in this project, I just store the state that whether player save the game board or not in the history, if player want to load the previous board, the state also can be stored.