Plan of Attack

Project: Watopoly

Members: Felix Zong(f2zong), Wang Qing(q473wang), Chen Tianhao(t345chen)

1.Breakdown of project

There will be four stages for our project.

The first stage is setting up the work environment and designing the structure of the project. The estimated completion date is July 15, contributed by all members. We have come up with a class breakdown listed as following: TextDisplay class, Controller class, Player class, Building class, Board class, Observer class and Subject class.

The second stage is implementation of codes. We will construct all the classes listed above and build the structure in the main function. It is agreed that Felix will take charge of Controller, Board, Observer and Subject class; Wang Qing will take charge of TextDisplay and Building class; Tianhao will take charge of Player class and the main function. The estimated completion date is July 21. We will meet up 2 or 3 times during the stage to discuss our progress and help one another if we encounter difficulties.

The third stage is debugging and improving which we will use test suites and Make files to debug our program and fix the bugs. The estimated completion date is July 24, contributed by all members.

The last stage is writing the final design document and the updated UML. The estimated completion date is July 26 and will be done by all members.

2.Answer to questions

Question: After reading this subsection, would the Observer Pattern be a good pattern to use when implementing a game board? Why or why not?

Answer: Yes, similar to assignment 4 question 3, an observer pattern can notify the board class only when there are changes inside building or player classes that will result changes in text displayed.

Question: Suppose that we wanted to model SLC and Needles Hall more closely to Chance and Community Chest cards. Is there a suitable design pattern you could use? How would you use it?

Answer: I don't find a proper design pattern that are suitable for model SLC and Needles Hall. In my opinion, these two buildings can be implemented by placing a random number generator

inside, by which the number generated each time will determine the behaviour of players on that building.

Question: Is the Decorator Pattern a good pattern to use when implementing Improvements? Why or why not?

Answer: Decorator Pattern might not be a good pattern to use for improvement. Since each improvement level will return different addition on tuition which is also differ from building to building, and there is a limit of 5 improvements maximum, if we implement it as a decorator, we need to implement too many decorators or too complicated logic inside decorators. Instead, an integer that saves the number of improvements of a building will be good enough to represent and easy to implement the tuition function.