CS 246 Final Project Plan

Project chosen: BB7K

Worked by Chenran Xing and Zefang Zhu

1. UML

UML is submitted separately.

2. Breakdown

We break our work into 2 demos before the final game. We will work on main.cc together until the final copy. The other part will be work by one of us or together as following:

Subject	Date	Person in Charge
TextDisplay Class	July 15th - 17th	Chenran Xing
GameBoard Class	July 15th - 16th (basic) By July 24st (Fully Developed)	Chenran Xing
Command Line	July 18th - 19th	Together
Testing Mode	July 17th & 18th	Chenran Xing
(Human) Player Class	July 15th - 17th	Zefang Zhu
Square Class	July 16th - 17th	Zefang Zhu
1st DEMO[1]	By July 19th	Together
Building Squares	July 17th - 20th	Zefang Zhu
Non-property Squares and Cards	July 18th - 20th	Chenran Xing
Player – Bankruptcy & Game Ending	July 20th - 21st	Together
2nd DEMO[2]	By July 21st	Together
Building - improvements	July 21st	Zefang Zhu
Player – Auctions	July 22nd & 23rd	Zefang Zhu
Mortgages Function	July 21st & 22nd	Chenran Xing
Saving & Loading Function	July 23th	Chenran Xing
Testing & Debugging	July 24th & 25th	Together
Basic Game	By July 24th (before testing) By July 25th (ready-to-submit version)	Together
DLC – A.I. player[3]	July 25th – July 27th	Undecided
DLC - House rules	July 25th – July 27th	Chenran Xing
Final Game	By July 27th	Together

[1] The first DEMO comes with a game board that players can roll and move. Some Command line and Testing mode is included.

[2] The second DEMO will provide a basic monopoly experience such as rolling, player movement, buying property and paying rent. We will try to add as much non-property square as possible.

[3] "DLC" will be available only if there are extra time like our planning. We are considering A.I. player and House rules, but we may do differ/other DLCs.

3. Answers to Provided Question

Question. *After reading this subsection, would the Observer Pattern be a good pattern to use when implementing a gameboard? Why or why not?*

Yes, it is a good choice for the gameboard. The gameboard will be the subject that collect and modify data, and all player will be the observers. Player will be notified of the result/changes to the board each time a player moves, so it would always have the newest data ready. (Or let squares be observers if the player position info is stored in squares, and notify the gameboard every update when fileds in square is changed. Both choices are good.)

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?*

I would choose Singleton Patterns for this scenario. Since Chance and Community Chest cards each has exactly one pile shared by all players, and the used card from top will be put on the bottom. I would choose to make each of SLC and Needles Hall singleton, and provide a method to "draw" an outcome from a shuffled queue. All player that arrive in SLC and Needle's will call this method, and the selected outcome will be executed, then this event will be pushed to the bottom of queue.

Question. *What could you do to ensure there are never more than 4 Roll Up the Rim cups?*

Use an array of 4 player class pointers to track the ownership of Roll Up the Rim cup. (Initialized to null, which indicates it has no owner.) When someone wins the cup, check if each pointers are points to a player object or null, and do not assign new ownership if there is no space. When someone in Tim's line up, check if he has the ownership. If so, free him from line up and reset his space to null.

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

Decorator Pattern can be used here, however, it may not be a very suitable choice. All Improvements has only minor difference in tuition, and they are identical beside that. Since there is no different behaviors between improvements, there is no need to use decorator patterns, which are used to add multiple differ add-ons. Using an array to track tuition with differ number of improvements would be more clear and efficient.