University of Toronto Mississauga

CSC207 Fall 2022

Design Document

AUTHORS

Andy Han
Richie Hsieh
Jinx Jin
Neal Wang

GITHUB TEAM: **CTRL**
GITHUB URL: **https://github.com/An10000/Chinese-chess.git**

11 13 2022

Contents

2 Project Identification

Welcome to Chinese chess! This is a Chinese chess board game, which contains three different modes to play. The game is pretty similar to international chess, which allows two players to fight against each other using different types of chess on the board. When playing, the players would click on the chess on the board and move it to other places on the board to invade and finally kill the general on the other side.

3 User Stories

The following table includes demands for Users. After classification and sorting, it is divided into three overall stages and ranked from high to low priority.

Name	ID	Owner	Description	Implementation Details	Priority	Effort
logic of Board	1.1	Neal	As a user, I want to have a formal Chinese chessboard so I can put chess on the board.	Using two dimension Array to represent a chessboard, each position may or may not contain chess depending on game status. Print the chessboard on the "Run" window.	5	2
logic of Chess	1.2	Jinx	As a user, I want to move chess on the board.	we need to create an abstract chess class or interface and complete corresponding actions for different chess pieces, for example killing the opposite player's chess, moving chess, and other actions. In this phase, achieve control of chess only by keyboard input.	5	3
Player1 V.s Player2	1.3	Neal and Jinx	As a user, we want at least and most 2 players to play this game.	Create a Player class that has two factions respectively for Player1 and Player2. so we can easily identify who is moving chess and so on.	5	2
Visualization of Chessboard and chess	2.1	Richie and Andy	As a user, we should be able to see the game in a larger window with a real chessboard and chess image.	Use JavaFX to create a window to show the board and the chess on it. And also chess image changes position as the game progresses.	4	4

Different Language	2.2	Jinx	As a user, I want both Chinese and English speakers could play this game for purpose of accessibility.	Based on the input of Player1 and Player2, we will receive what language they prefer (Only Chinese or English). Then, we could print and show the message to Players in the corresponding language.	3	2
Move chess by using the mouse	2.3	Richie	As a user, I want to use the mouse to control chess and that is faster and more convenient than the keyboard.	import JavaFX, create buttons on each chess, and use mouse events to connect to the method in Chess class.	3	4
Score System	3.1	Andy	As a user, I would like to see a scoring system in the game, so we can see who won and who lost when the game is over	Create a HashMap used to store scores for a specific player. And create a method that can add a score to the player.	2	2
Timing System	3.2	Neal	As a user, we want to have a timing system so we can train our speed in playing the game. 4	Create a Timer class in the View package that controls the time each player uses and stops the player from playing when time is out. Use JavaFX to show a textbox on the game window beside the chessboard. Refresh the textbox each second so that we can imitate the behavior of a countdown clock.	2	5

4 Software Design

This section should describe the design patterns you will use to realize your user stories. For each design pattern, provide a draft of UML diagrams to show how you'll implement the pattern as well as a written description that details how it will be used in the context of your user stories. An example of a pattern description is provided below. Note that you can adjust these patterns as you progress with your implementation, as this is necessary.

4.1 Design Pattern 1: Strategy Pattern

Overview: This pattern will be used to implement User Story 1.2 .

UML Diagram: Refer to Figure 1, below.

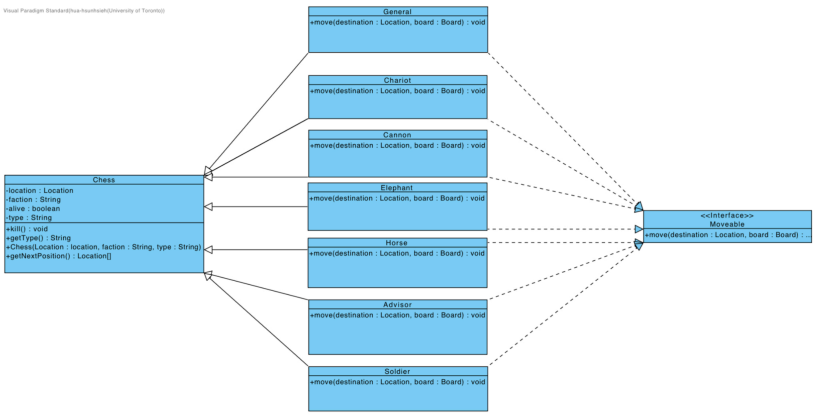

Figure 1: The strategy Pattern

Implementation Details: The UML diagram outlines these main components:

1. The Movable interface, which only includes one method: move.
2. Different child classes of the chess class implement this interface and overwrite the move method.

We use a strategy pattern. Because different sorts of chess have a unique way of moving so each child class should overwrite the Move method. The move method needs two parameters: destination and board.

4.2 Design Pattern 2: Singleton Pattern

Overview: This pattern will be used to implement User Story 1.1.

UML Diagram: Refer to Figure 2, below.

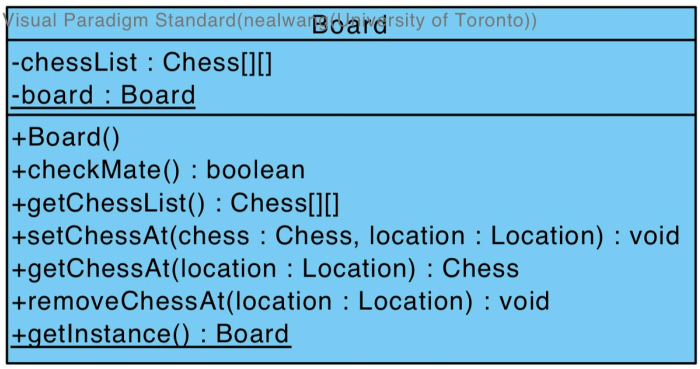

Figure 2: The Singleton Pattern

Implementation Details: The UML diagram outlines these main components:

1. Getinstance Method will let us get the instance of the board.

When running the whole program we will only create one instance of the board and we will only change the data in the board. We use a singleton pattern so every time we call the board, we use Getinstance to acquire the same board instead of creating a board frequently.

4.3 Design Pattern 3: Builder Pattern

UML Diagram: Refer to Figure 3, below.

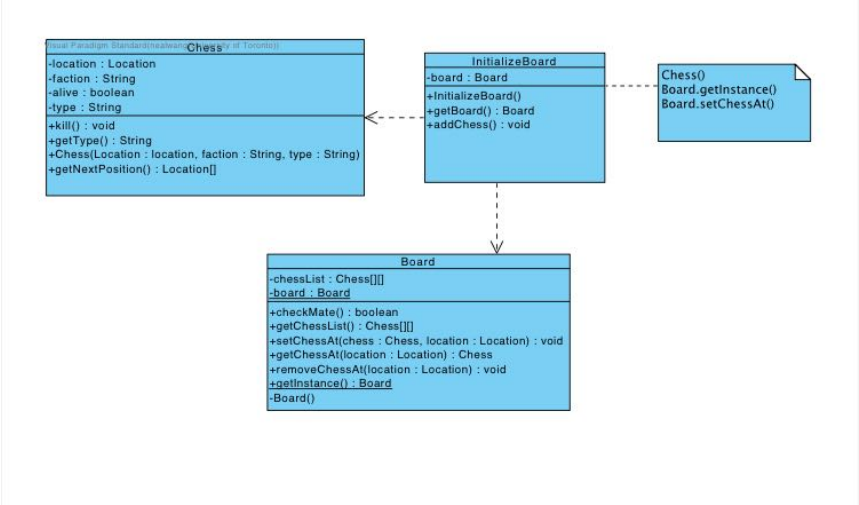

Figure 3: The Builder Pattern

Overview: This pattern will be used to implement User Story 2.1.
Implementation Details: The UML diagram outlines these main components:

1. InitializeBoard is a class that initializes Board and chess in advance.

2. Methods in Chess and Board Class will be used to initialize the game.

We use the Builder Pattern to allocate different methods to the corresponding class. And this helps us organize structures. And these three classes are combined as a complete and interactive process.

4.4 Design Pattern 4: MVC Pattern

UML Diagram: Refer to Figure 4, below.

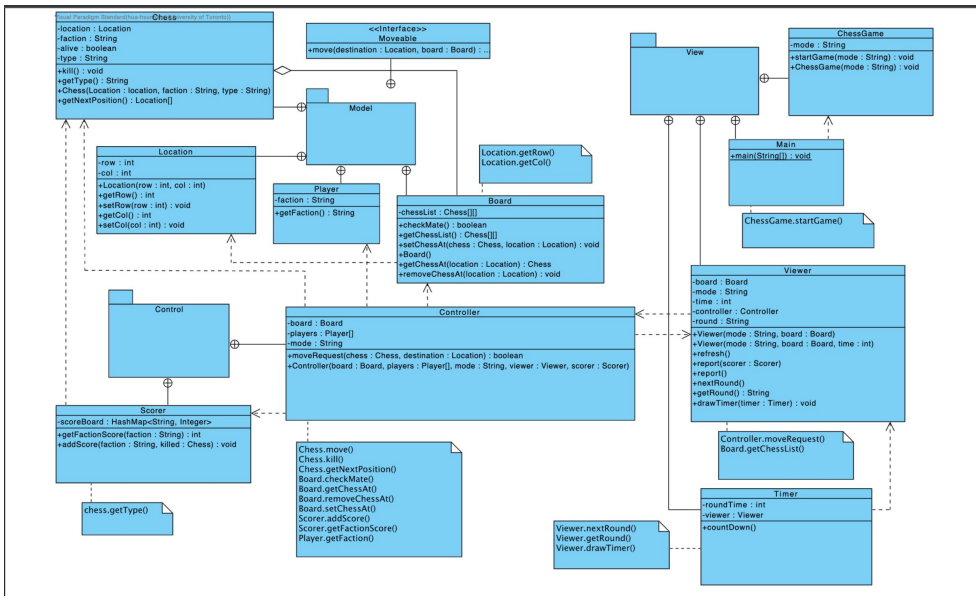

Figure 4: The MVC Pattern

Overview: This pattern will be used to implement the whole User Stories.
Implementation Details: The UML diagram outlines these main components:

1. View will receive the user's input such as a mouse event.
2. Classes in Control package are designed to understand these inputs so it could control the model.
3. Classes in the Model package are the game itself such as chessboard, chess.
4. Classes in the View package are designed to give user feedback and then the user will make the next judgment and give input to the game again. Timer, board, and chess images will all be processed through the view package.

We use an MVC design pattern and each package we identified will process and provide input and feedback from the users. This pattern is a cycle when the program is running.

5 Expected Timeline

CSC207 Project TImeline

Timeline		
11/19-11/20	Phase 1	
	1.1	Logic board
	1.2	Logic chess
11/21-11/23	1.3	Player1 vs player2
FINISH PHASE1	Able to play by keyboard	
11/24-11/29	Phase 2	
	2.1	Visualization of chessboard and chess
	2.1	Different language
	2.3	Move chess by mouse
FINISH PHASE2	Able to play game by mouse in a seperate window	
11/30-12/03	Phase 3	
	3.1	Score system
	3.2	Timing system
FINISH PHASE3		
12/04-12/06	Debug Phase	
2022/12/07	Due	