UNIVERSITY OF APPLIED SCIENCES

Advanced Programming Structure Project

Cross Wise

Name	:	Chien-Hsun Chao
Field of study	:	IT Engineering
Matriculation number	:	ITE104218
Semester of study	:	SS2022
Administrative semester:		6

Contents

List of Figures

List of Tables

Chapter 1

User manual

1.1 Requirements

A Jar file named "pp-CrossWise-Chao-1.0-SNAPSHOT" is stored in the folder "final-binaries".

1.2 Program installation/start

The section here describes how the program start, save and load:

- Game Start
  The initialization of the program is simply launching the jar file which is in the folder "final-binaries".
- Save
  With the saved setting, the user must provide a file name in order to avoid the error of the file is not chosen.
- Load
  With the load setting, the user must click on and choose a specific file to open it.

1.3 Operating instructions

This section primarily describes how the game works, including the approach for calculating scores, all the necessary elements this game involves, and also the different modes of players.

1.3.1 The game description

CrossWise is a simple 6X6 board game that has a maximum of four players in two teams, a vertical team and a horizontal team. Two teams always play against each other. On the side of the vertical team, all vertical lines on the game board are always noticed. And on the horizontal side, they always focus on the horizontal board lines.

With regard to players' hands, every player has four tokens as a hand from the beginning. In each round, they have a chance to place a token on the chess board or to use a functional token to reach their purpose.

Furthermore, there are more following sections that describe the detail of the game rules, the game element involved, and the condition of winning a game.

1.3.1.1 Token

This game is played with six different symbol tokens which have unique shapes and colors, and four different action tokens have their own functionality. This increases game variability and difficulties. The following description is about the utilization of symbol token and action token:

1. **Symbol Token**
   The symbol tokens consist of six different shapes, for example, cross, pentagon, square, star, sun, and triangle. As you can see in the Figure 1. Each symbol token exists seven times in a round of a game. The main functionality of those symbol tokens is that player selects one of them as a placing token onto the chess board to occupy a position on the board.

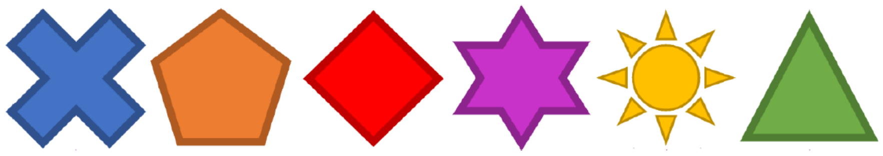


Figure 1: All symbol tokens

2. **Action Token**
   The action tokens consist of four different patterns, e.g., Remove, Shifter, Exchange, and Replace. Each of them exists three times in a round of a game. The description of those action tokens is shown below based on the order of the figure 2

   - Remove
     The player moves a token of his choice on the board to another empty square of his choice (the square does not have to be adjacent).
   - Shifter
     The player removes a token of his choice on the board. He takes the removed stone into his hand.
   - Exchange
     The player swaps two tokens of his choice on the board with each other.
   - Replace
     The player replaces a token of his choice on the board with one of his own tokens. He takes the replaced token into his hand.

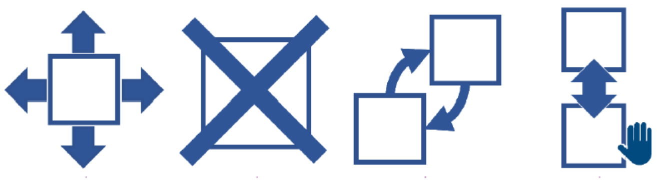


Figure 2: Action tokens

1.3.1.2 Scoring Points

In essence, scoring points has six combinations in this game. As you can see those combinations are under the table 1. In each line, some combinations can appear simultaneously. Thus, the scoring points of a line depending on the combinations achieved in the respective line.

Combinations	Point
Six different symbols	6 Points
Two same symbols	1 Points
Three same symbols	3 Points
Four same symbols	5 Points
Five same symbols	7 Points

Table 1: A scoring points table.

1.3.1.3 Player Mode

This game has different player modes that can be only two players or four players to start a game. In addition, there are free combinations for each player that can be all human players, a mixture of human players and computer players, or all computer players.

1.3.1.4 The condition of winning a game

There are two different approaches to reach winning a game. Generally speaking, the game should be ended when the game board is wholly occupied. However, if there is one team has completed six same tokens on a line on the chess board, then the game can be ended in advance and the winner appears.

- Six same tokens
  A team has arranged six same tokens on a board, then this team becomes the winner and the game is ended.

- High Score
  If there is no team that reaches six same tokens during the game, then scoring both teams at the end of the game. Which team has the higher score, the team is the winner.

1.3.2 Game Interface

1.3.2.1 Main Interface

The main interface consists of several parts, a chess board, the area of players' hands, the display of the scoring combinations, the functional explanation of all action tokens, the demonstration of two team score points, and the label of the current player.

1. **Menu bar**
   In the main interface, the menu bar is apart into two separate sections. The first bar is called "setting", which obtains four menu items for the fundamental setting of the game, as Figure 3 below.

   - Setting

     - New Game
       The aims of the Menu item of New Game are used for creating a new game.
     - Save
       The aim of Menu item Save is used to save an incomplete game. The saving file is using GSON to translate the whole game system, including every player's state, the current board, the remained action tokens, and the current player.
     - Load
       The aim of Menu item Load is used to load a given game. A given file is provided with JSON and it can be analysed by using GSON to the whole game system, including every player's state, the current board, the remained action tokens, and the current player.
     - Close
       The function of Menu item Close is closing the game.

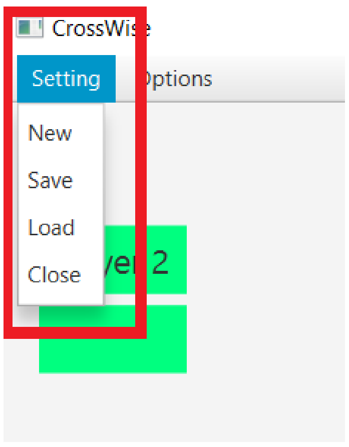


Figure 3: The setting menu item

- Options
  The options menu Figure 4 and explanation are shown below:
  - Duration
    The menu item Duration contains three subdivisions, short, medium, and long, which are for dominating the duration of highlighting a cell on the chess board.
  - Row/Column Score Display
    The menu item Display Row/Column Score controls the visibility of seeing the score of each row and column. The default setting is invisible for all rows and columns.
  - Current Team Points
    The menu item Current Points controls the visibility of seeing the current total points of two teams. The default setting is invisible for the current team point.
  - Computer Hands
    The menu item Computer Hands is utilized to manipulate the exhibition of the computer hands token. As there are bots participating in a game, the menu item could be executed for deciding whether their hands token should be visible or not.
  - Stop AI playing
    The menu item Stop AI Playing can be used to interrupt and cease the game when all players are a bot. The default setting is that this menu item is able implemented when all players are bots.

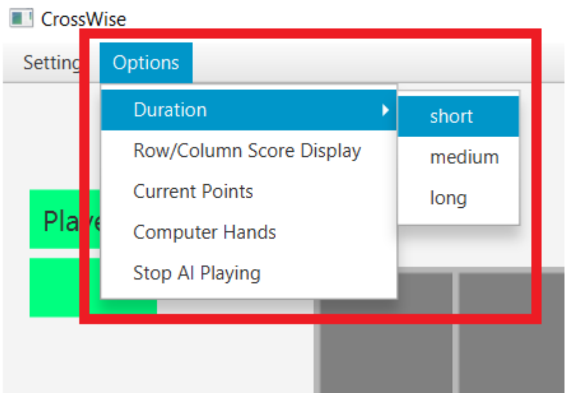


Figure 4: The option menu

2. **Chess board**
   The chessboard is made of a 7x7-table, which is used for placing symbol tokens and displaying the score points of each line. Among, the 6x6-table where symbol tokens can be settled in each cell, is entirely occupied with a symbol token called None. Additionally, the endmost row and column remain empty in order to show the scoring point of each line. The graphic is shown below Figure 5 .

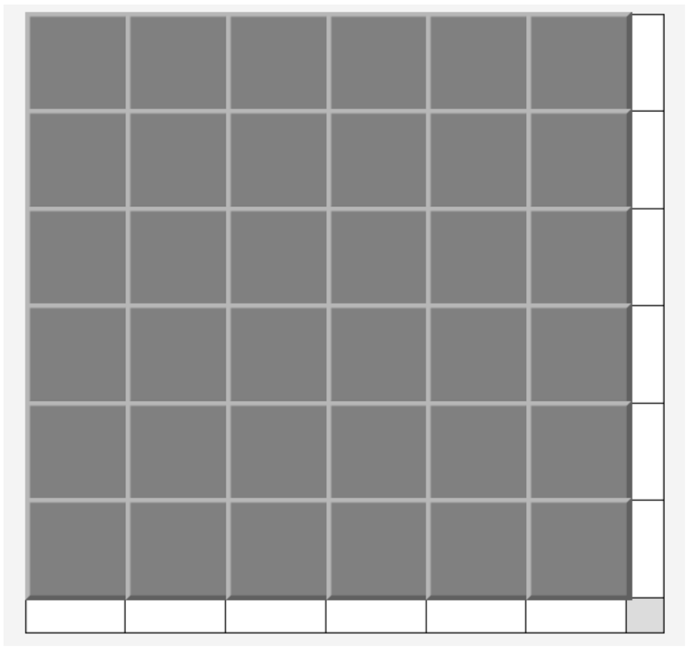


Figure 5: The chess board

3. **Players' hands**
   As shown in the Figure 6 below, there are four red marking areas that surround the chess board, those areas are the players' hand tokens. If there are only two players involved in the game, then the exclusive area which is for player 1 and player 2 is visible..

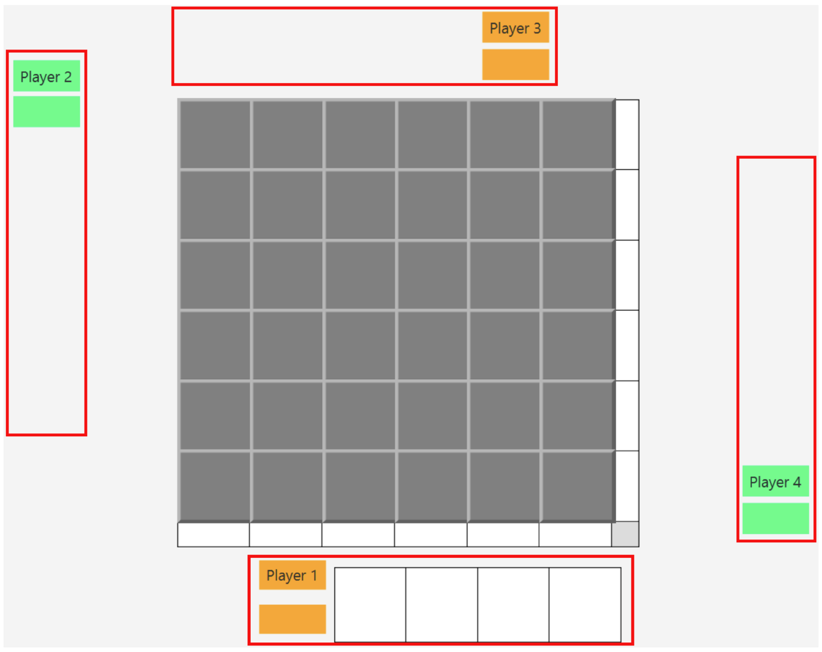
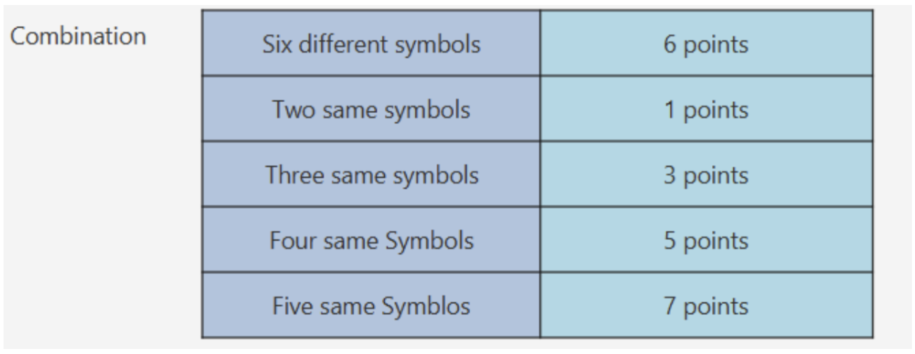


Figure 6: Players' hands

4. **The combination of the scoring points**
   On the right side of the main interface, a zone is for displaying all attainable combinations of every score. While the player is in the progress of the game, it is possibly forgotten how is the point being calculated. Hence, this region can remind players of the scoring logic. The demonstration is shown below in Figure 7.



Combination

Six different symbols	6 points
Two same symbols	1 points
Three same symbols	3 points
Four same Symbols	5 points
Five same Symblos	7 points

Figure 7: The combinations of scoring points

5. **The demonstration of two team score**
   Undeniably, if the player can regularly inspect their current achievement, the experience of the game, and the convenience of the exploring game are increased undoubtedly. Therefore, possessing an area that demonstrates the achievement of two teams is a necessary arrangement. The demonstration is shown below in Figure 8.

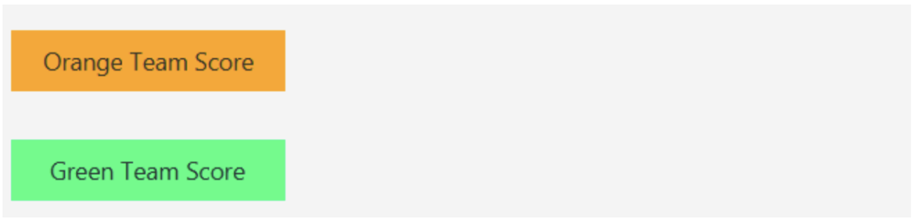


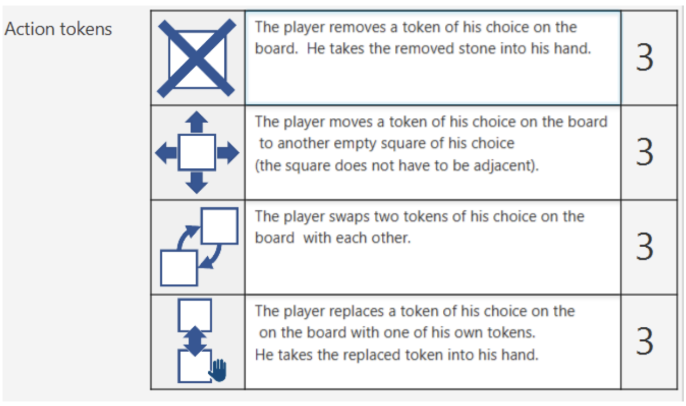
Figure 8: The display of team scores

6. **The label of the current player**
   As with the display of two team scores, labelling a current player is also an excellent option to enable participants to realize who turns. Meanwhile, the background colour of this label can be varied as the team belongs to the current player. The label is shown below in Figure 9.

Figure 9: The label of current players

7. **The explanation of all action tokens**
   As with the same purpose of exhibiting combinations of the scoring points, having a description for action tokens provide considerable benefits for the game. Players can be reminded of the function of those four action tokens.

   Moreover, on the right side of the figure 10 shown below, a list of the remaining number of action tokens is exhibited. Once one of these action tokens has been utilized, the list of the remaining number of action tokens will be updated automatically.



Action tokens

	The player removes a token of his choice on the board. He takes the removed stone into his hand.	3
	The player moves a token of his choice on the board to another empty square of his choice (the square does not have to be adjacent).	3
	The player swaps two tokens of his choice on the board with each other.	3
	The player replaces a token of his choice on the on the board with one of his own tokens. He takes the replaced token into his hand.	3

Figure 10: The demonstration of action tokens

1.3.2.2 Secondary Interface

1. **Menu bar**
   The red marked area of the Figure 11 represent the options menu items for deciding participants' number in a game. The default setting for initiating a game is four players.

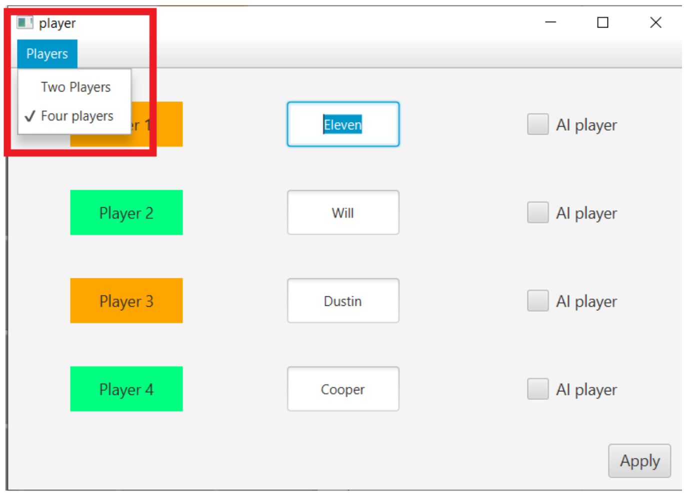


Figure 11: The menu bar in the second window

2. **The second window**
   The Figure 12 below represent an interface for initializing the state of players, involved in establishing player amounts, inputting players' name, and determining the state of the player in human or bot. As you can see the top-left corner has a menu item that can set up the number of players in a game. And in the middle of this scene which is circled as blue, there are four labels that can be entered players' names. At last, the right side of the scene which is marked as red provides four check boxes that enable the user to determine which players are manipulated by AI or humans.

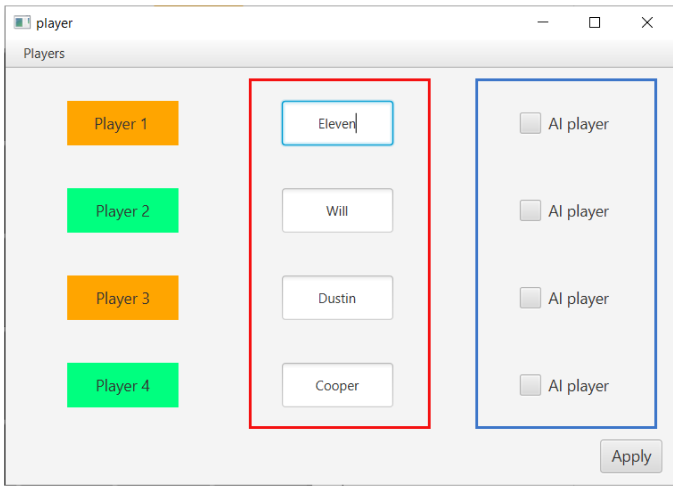


Figure 12: The second interface - player scene

1.3.3 The Operation

This paragraph is branched into three sections, before the beginning of the game, the game start, and the end of the game.

1.3.3.1 Before beginning of the game

When the game is activated, a primary window is created and appeared, but the current interface cannot be clicked or manipulated by any action. Therefore, there are a few steps that can begin and activate a new game, which is shown below:

1. **The menu item ”New Game”**
   The user should press the Menu Item ”New Game” on the top-left corner to generate a new game.
2. **Number of participants**
   Since the new game has been generated, a secondary scene which is for initiating player state arises. On the top-left corner of this secondary scene, the user can select the check menu Item to determine how many players should exist in a game.
3. **Inserting participants name and status**
   According to the selection of the number of participants, the user can insert every participant’s name and decide which player is a bot or human.
4. **Apply**
   At last, press the button, which is on the bottom-right corner to execute the game.

1.3.3.2 Game start

1. **A player’s turn**
   During the game playing, one of the designs should be noticed that the game is being operated based on the turn of the current player. The hand token can be manipulated for the current player, nevertheless, other hand tokens are not allowed to give any movement or action.
2. **Utilising tokens**
   Here have two different manipulative approaches for symbol tokens and action tokens. The instruction for two types of tokens is below:
   - Symbol token
     The manipulation behaviour of the symbol token is using drag and drop. The user needs to move their mouse on the symbol token of his or her hand token, which is chosen to move. And hold the left click and drag it to the chess board and release it on the desired cell of the chess board.
   - Action token
     The manipulative approach for using action tokens is clicking. Four action tokens use clicking action to activate the function of the action token. Firstly, as the user decides to use the action token, which is in his or her hand, it is simply a single click on the action token. However, the

important and necessary noticed gist is that once the action token has been clicked, the user is not allowed to have other tokens chosen.

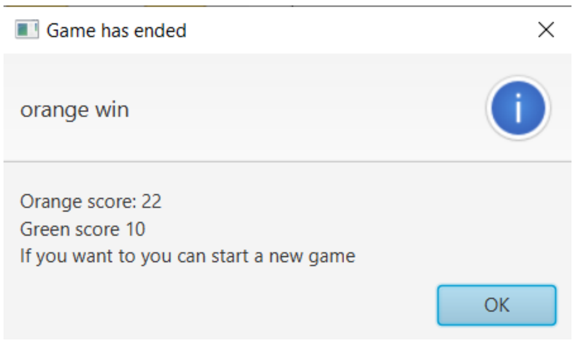
   - Remove
     The user should directly click on the desired cell of the chess board. Indeed, it is vital to realize that Remove is only allowing to click on the cell which has already occupied a token.
   - Shifting
     Based on the instruction of the shifting function, the user should select two positions on the board after clicking on the shifting token on their token hand. One is the position that contains a symbol token on a cell of the chess board and another should be an empty position which means no symbol token is placed.
   - Exchange
     Regarding the function of the exchange token, the user has to choose two tokens on the chess board. After two tokens are selected by clicking, the animation of exchanging tokens is shown up to swap these two chosen tokens.
   - Replace
     Replace token has a unique function among these four action tokens. As the replace token has been activated, the user needs to select one symbol token on his or her hand before choosing another board token, which will be replaced. Hence, if there is no other symbol token on the player's hand, then the Replace token cannot be executed.

3. **Highlighting tokens**
   One design for raising up the game experience for players is that highlight every token, which is being dropped, clicked, or swapped. Indeed, the token can be immediately and apparently noticed by highlighting after they are being changed position.

1.3.3.3 End the game

Along with the game approaching the end and the winner is determined and appears, an alert message has emerged with a window like a Figure 13 below. The message shows which team is winning the game and also the reason for winning the game.

However, once the message is read and closed, the main interface becomes disabled to access or manipulate until the user selects a new game on the menu item.


Game has ended

orange win

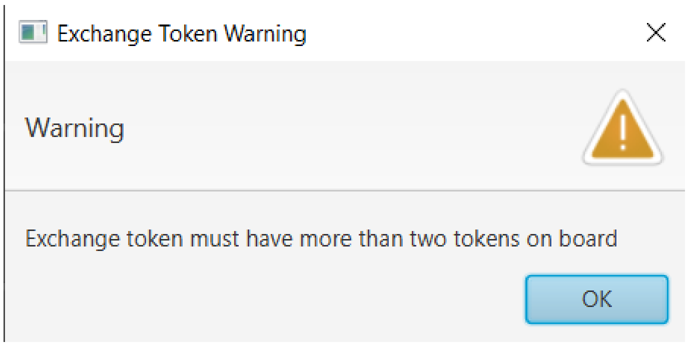
Orange score: 22
Green score 10
If you want to you can start a new game

Figure 13: The winning message

1.4 Error messages

1.4.1 During the game

error message	cause	corrective action
Exchange token cannot be used	Player may not use Exchange Token when there are less than two tokens on the board	Reselect an available token on his hand


Exchange Token Warning

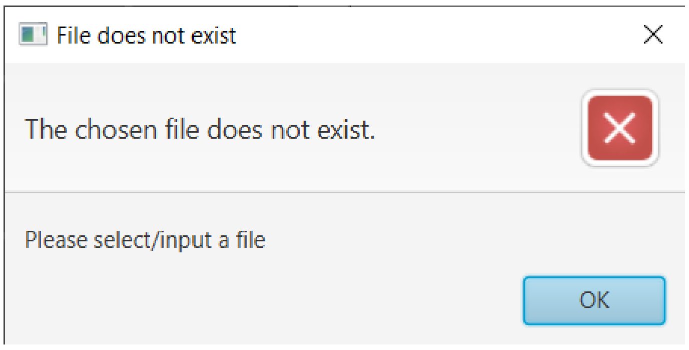
Warning

Exchange token must have more than two tokens on board

Figure 14: Exchange Token Warning

1.4.2 Load and Save

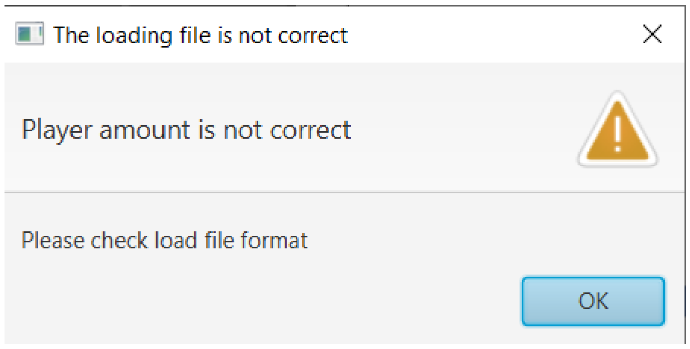
error message	cause	corrective action
The chosen file does not exist	There is no file selected to load and the file for saving does not input the file name	Press Load/Save Menu Item again and re-selected a file
The player amount is not correct	If the input file contains an invalid player amount, it causes the incorrect execution of the game.	Check the format of the input file and reload again.


File does not exist

The chosen file does not exist.

Please select/input a file

Figure 15: The chosen file is not exist


The loading file is not correct

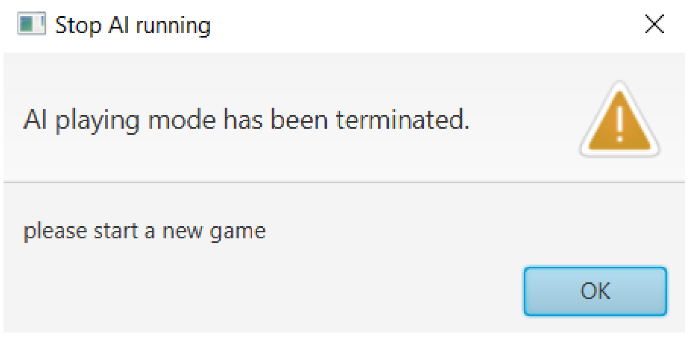
Player amount is not correct

Please check load file format

Figure 16: The alert of wrong player amount

1.4.3 Stop AI running

error message	cause	corrective action
Stop AI Running	A radio button is pressed and it can terminate the JavaFX Program	Following the instruction given in the window. Press the New game button to start a new game


Stop AI running

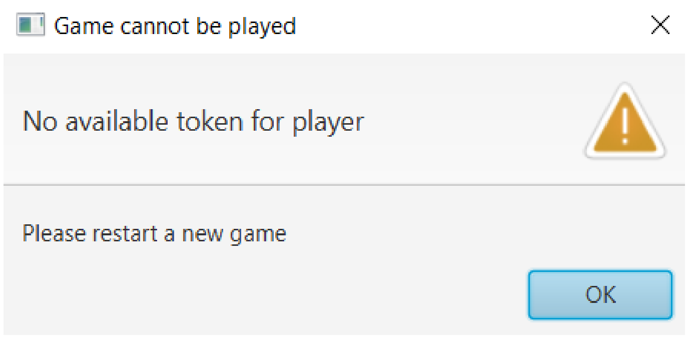
AI playing mode has been terminated.

please start a new game

Figure 17: Terminate AI Running

1.4.4 No available hand token

error message	cause	corrective action
Full of action token and the chessboard has no token	If the player has all action tokens on hand, meanwhile the board is empty without any symbole token to allow action token to implement its function	This game needs to be restart by clicking New Game button
Full of Replace Token and No symbol Token	If the player has clicked their token hand, which contains three replace tokens and no symbol tokens, the game cannot be continuously played.	The game should be restarted by clicking the New Game button on the top left.


Game cannot be played

No available token for player

Please restart a new game

Figure 18: No available token hand

Chapter 2

Developer manual

2.1 Development configuration

This project is mainly based on the JAVA programming language and the IDEs of this project are using IntelliJ IDEA. To develop a desktop application, the java library JavaFX is utilized for building a GUI platform. Moreover, to port FXML mark-up to an IDE, Scene Builder is chosen for this project to facilitate. All the specified compiled versions of the software components are shown in Table 2 below.

Components	Version
Java Development Kit (JDK)	JDK 17
Scene builder	18.0.0 (the latest version)

Table 2: A software version table.

2.2 Problem analysis and realization

This section presents the problem which the designer faced when he or she developed and designed the process of the game CrossWise. The problems are related to the GUI interface, the game logic design, or the necessary elements which are considered to implement. In addition, this section not only contains the analysis of the problem but also gives some specific realizations of the problems.

2.2.1 GUI representation of the game windows

2.2.1.1 Problem analysis

Considering the fluent game-playing process, the setting of the player name or whether some AI players are participating must be decided at the beginning. Thus, how to manage those setting is an issue. There are two ways to implement it, either in one- window or in a separate window.

2.2.1.1.1 All in One window

An ideal all-in-one window is that selecting the player amount and input player name can be determined in the main interface. All the possible situations for the human players and computer players are created in the menu bar by using the menu item. The advantage of this option is that all the game settings can be determined in one window. However, after during the game for a while, once users want to change their name and chooser another mode of the game, the previous setting will disappear.

2.2.1.1.2 Tow separated windows

An ideal of this situation is using two windows to represent the whole game running. One is for selecting the player's amount, determining the human player or the computer player, and also available to input the player's name. Another window is used to display the whole game field which is the main interface of the whole game. The first window can be invoked from the menu bar of the second game interface which named as "New game".

2.2.1.2 Realization analysis

To solve the difficulty of switching the players' related settings, it is achievable and possible to manage all initial settings in the two windows. Here include several workable realizations below:

1. **UserInterfacePlayer.fxml**
   In order to initiate the player setting, an FXML interface needs to be created which means an additional scene builder has to be used. Many elements of the items will be managed properly on this scene builder, for example, GridPane, Label and CheckBox, and so on.

2. **Stage**
   A Stage class is used to invoke an owner window. In this problem consideration, we will use the Stage class to invoke the second window for initializing the player settings. After the player setting is done, a Stage object also will be used to switch back to the primary interface.

3. **CheckBox**
   According to different modes of the player setting, we need to decide which player is played by a Bot or a Human. Thus, a checkBox item can be used to determine this option. Each player has a checkBox item to let the user decide except for the first player. The first player must always be played by humans.

2.2.1.3 Realization description

The primary window has one branch from the menu bar. The one is originally responsible for the fundamental configuration (e.g., New game, save game, and close game.). The second window is invoked from the new game options. when the new game item is pressed, the second window will be shown up and replace every content of the first window. In the second interface of the players-related setting, there are four labels for four players, for example, player1, player2, player, and player4. Each label has a text field after itself, and the player's name can be input into it. The last column of this player setting has three checkboxes to decide human player or a Bot Player. During the game playing, the user can reselect the desired case from the second window. This two-separated-window setting does not increase the difficulty for the designer and even simplifies the data passing process of the players' names.

2.2.2 Passing data between two windows

2.2.2.1 Problem analysis

The following analysis and description refer to how could the players' data be transferred from a Scene B controller to a Scene A controller which is the primary interface. To do so, two ways could be implemented. None of them is the perfect solution, but each has benefits and downsides to being considered in a different situation.

2.2.2.1.1 A class Controller

Logically, when an FXML file is created in Scene B, the IDE automatically creates a controller with it and uses the root tag to link it together. Also, this controller can create a new instance in the B class, however, if we want to pass information to another controller. An instance must be created manually in class A which will accept the new information and use this new instance to obtain the accessibility of all public methods which is necessary to be used.

2.2.2.1.2 The singleton pattern

An idea of the singleton pattern is that two classes use the same instance of the singleton class. This class contains all the necessary information that needs to be passed. In other words, this singleton class has a getter and setter function that is able to get the data and also set the data for different situations. Broadly speaking, it can be treated as a database.

2.2.2.2 Realization analysis

This project is passing information related to players. All players' data that need to be transferred is analyzed in several attributes. Those attributes can be treated as objects. A getter and setter function is important to exist and is utilized in the progress of passing data. The explanation of these two tools is described below.

1. **Controller Instance**
   An instance of a scene controller can be displayed as a class object. In addition,

the direction of passing information can decide which controller class should obtain the instance. Normally, the class which accepts data has the demands to require this instance.

2. **Getter and Setter**
   Getter and setter functions are used frequently when the data is transferred. Every passing attribute needs its own getter in order to get the data. In the meantime, the setter can be utilized purposely in some situations where the attribute needs to set some value.

2.2.2.3 Realization description

Using the controller instance is a prompt instrument for transferring all needed data. Regarding all the attributes that need to be passed and updated in this project, there are only four attributes that need to consider. For instance, the players' names, the active state, and the state represent whether the player is a human or a bot and the amount number of players in a round. Broadly speaking, there is not so much information that needs to be passed. Therefore, a controller calling is thoroughly enough for these demands.

2.2.3 Data structure of players' hands

2.2.3.1 Problem analysis

The playing hand consists of four random tokens, which can be used to proceed with a game. In each round of the game, a token could be removed from the hand and another token could be added. Until the game approaches the end, each hand's total length may not remain at four. Moreover, the hand element always needs to be updated in each round, and also each token is able to be adjustably utilized. Thus, a data structure for adopting to build a hand should consider the following conditions.

1. Although the length of the hand is fixed in the beginning, the data structure should be able to adjust the size when the game approaches the end.
2. A data structure should involve some characteristic that can easily access the element of the hand, for example., the ability to remove, insert, and get the element from the index.

2.2.3.1.1 Array

An array is created to hold a fixed number of values of a single type. The length of an array is determined when the array is created. Therefore, in this case, a hand can give a fixed length to the array and put token elements into the array. However, in this game, hand length cannot always be the same. It can be less than the original size of the array. As this situation happens, the way to solve this is that keeps the array has still existed and the empty position of the array can be assigned as null.

In addition, A good point by using an array to construct a hand is that it is much easier to get the token element by calling the index. However, there is an issue need to be noticed. While one token element is removed from the array, the index of the removed element should be recorded in order to store put another token into it. Alternatively, there can be another way to do it every time a token is being removed and the array must be sorted. This is helpful to insert a new token at the end of the array instead of recoding the index of the removed token.

2.2.3.1.2 LinkedList

A Linked list is a linear data structure in that elements are stored by using pointers. In other words, a linked list consists of several nodes, and a node contains a data field and a reference to the next node. With the help of the linked list structure, a hand can be designed as a linked list and each token is the node of the list. Each node is linked to each by using the link and reference.

The advantage of the linked list structure is that there is no need to establish a fixed size for the linked list. The length of the linked list is relatively flexible. Nevertheless, a downside of the linked list is to access elements. If a token need to be taken and placed on the board game, it must be gotten by using a pointer from the first node of the linked list and moving down until reaching the target that is required.

2.2.3.1.3 ArrayList

An ArrayList is using a dynamic array for storing the elements. It can be treated as a traditional array except it has no size limit. Also, the ArrayList allows random access by calling the array of the index and it also implements the List interface so the user can use the method of the List interface.

Based on the condition of the hand structure here, those characteristics of ArrayList are fulfilled with the condition of this hand structure.

2.2.3.2 Realization analysis

An ArrayList structure is suitable for the players' hand structure. It contains many different methods to be utilized. Here have some methods that can be used frequently in this project. Those methods are shown and introduced below:

1. **add(int index, E element)**
   An add() method can insert a specified element at the specified position in the ArrayList. This method can be used to add any new Token to the hand.

2. **remove(int index)**
   A remove() method can remove an element at the specified position in the ArrayList. Once a token is chosen in a hand and it needs to be removed from the hand, this method can be called by giving a specific index.

3. **toArray()**
   A toArray() method contains all of the elements in the ArrayList in proper sequence. This method is useful to check each element in an array and also can be used to display in the toString() method in order to show elements of each hand.

2.2.3.3 Realization description

A packed class is being created as a superclass of a hand class. In the pack class, the ArrayList structure is being used and called in the pack's constructor. A hand class can straightly inherit all the methods including the Arraylist of the pack class. The length of the ArrayList can be passed as a parameter in a pack construe. Therefore, Every time user creates a hand object with a specific size of the ArrayList. Once the game happens near the end, a length size can be decreased from the parameter of the constructor to create a new array list.

2.2.4 A moving animation for board

2.2.4.1 Problem analysis

Considering that the game can be played with all AI players, the running time of the whole game is considerably fast and uninterrupted. Hence, in order to slow down the whole running process for the whole game, completing a token action and utilizing an animation is considered.

Indeed, there are some possible animation types that can be utilized, The following description explains two different kinds of animation that are being considered.

2.2.4.1.1 Translate Transition

The TranslateTransition is represented by the class javafx.animation.TranslateTransition. It can translate the node from one position to another position depending on the specified duration. Those positions are defined by providing the translateX and translateY properties of the node. Also, the speed of the transition can be specified by the user.

2.2.4.1.2 Fade Transition

In JavaFx, the class javafx.animation.FadeTransition represents FadeTransition. It animates the opacity of the node so that the image color of the node becomes dull. Also, the specified duration also can be determined by the user.

2.2.4.2 Realization analysis

Due to the logical approach of this designed game, the fade transition could be the most efficient and achievable method to reach our target. Here have some specific features to describe why the fade transition could be the better approach in this project.

1. **Locating the cell coordinate of the board**
   After doing some attempts of the Translate Transition, a difficulty is shown up that giving the translateX and translateY properties could actuate the game developing progress become much harder than Fade Transition. The target of the moving node cannot be specified clearly. Relatively, the fade transition does not have this issue at all. It only needs to change the node opacity to let the user observe a difference after a token is placed.

2. **Synchronized changing behaviour**
   Given there are some action tokens that also need to do animation, some of them require switching to two tokens. Hence Fade Transition enables two nodes to do the animation simultaneously. However, even the Translate transition could also reach the same result, but it is more complex than the Fade transition.

2.2.4.3 Realization description

From the above explanation of two types of animation, Fade Transition eventually be implemented to reduce the speed of running the whole code. We create a few methods of different Fade Transitions in JavFxGui in order to be invoked when the logic of placing a token is completed. And by passing the coordinate of the node, the image of the given node can be adjusted to the opacity from the Fade transition. Even in the case of using action tokens, the coordinate of two nodes can be passed as parameters to the animation method. and call twice Fade transition to complete the fading behavior.

2.2.5 The conversion between GSON and java object

2.2.5.1 Problem analysis

In this project, creating a file to save and load the state of the whole game, and this file is recorded using GSON library.

Due to the expectation of the display format and the logical design of this game, a sequence of class objects is being used and some of them are documented in an array or a list.

However, during the implementation, an error is encountered, which is called java.lang.reflect.InaccessibleObjectException. The details of this error are that the java class object could not be binding or transferred. And moreover, this error also passed a piece of information that there is a java.util.Random object is tried to be serialized.

The following description analyze the solution to this issue:

2.2.5.1.1 Save

In the file of saving action, there is a class Player being used to record the player's name, activated state, and the state of representing whether is manipulated by AI and the token hand. Because of the logic designing, each player will get a series of random tokens for his hand when the game is started. This hand becomes an unpredictable variable. It will be changed at every turn.

2.2.5.1.2 Load

Likewise, when a file is transferred and bound to the current java objects in the program. But the Player class contains several attributes and this player class is also being recorded as an array type. Thus, those attributed to the player cannot be straightly assigned to the file's object.

2.2.5.2 Realization analysis

After considering a Player class contains several attributes, those attributes must have corresponding binding objects to allow the program to record their current value of them. Thus, the following solution "inner class " is precisely the lack of the clue of this converting method.

1. **Inner class**
   Creating an Inner class in the class GraphDataJson that includes four essential elements, name of the player, activate state, the state of representing whether is manipulated by AI and the token hand.

2.2.5.3 Realization description

From the above possible solution, an inner class is created and one more vital procedure needs to be implemented. In the method of transferring objects to the GSON format, those attributes need to be bound with the corresponding attributes in the main logic class instead of only transferring the class object Player.

2.2.6 Reducing the velocity of executing a game

2.2.6.1 Problem analysis

Because this game design enables some participants to be manipulated by the bot, once all participants are a bot and the whole program can be executed extremely fast and no interruption in it.

And considering the visibility of when a token is placed on the board, it enables the player to know which token is being placed and be able to observe how is the game played by the bot.

Thus, we use animation to complete this issue. However, there is a problem that how can allow the whole program to wait for the animation to be done and keep doing the following code. In other words, the animation should be implemented properly and in the meantime reduce the speed of the whole program running. The following analysis describes the possible solutions:

2.2.6.1.1 GUI animating

Here is the first consideration. Once the animation is completed successfully, a way to inform the logic part that the animation works are done and also needs to be set up smoothly.

2.2.6.1.2 Logic waiting

Second, the logic part will not be interrupted or paused due to any conditions set. The noticeable issue is that once the GUI is doing animation, the logic part will execute continuous code and not wait for GUI animation.

2.2.6.2 Realization analysis

From the problem analysis above, it can easily be observed that an element to connect the logic part and GUI part is indispensable. Especially the characteristic of connecting two packages. In addition, there have some valid solutions described below:

1. **Callback**
   A callback function can be called after the animation is done and coherent with the whole process of the code running. When the animation method is being called and executed, the GUI starts doing animation. But the logic part keeps running further down. And this callback function can lead the logic to wait for the animation to be done and follow up on the continuous work.

2. **Method reference**
   The callback function is located at the logic part, and in order to let the GUI know which method should be called once the animation is completed. Hence, a method reference approach can be passed this callback function to the GUI part as a parameter.

3. **SetOnFinished**
   The OnFinished function of the animation can sequence the whole process in a logical and reasonable direction.

2.2.6.3 Realization description

According to the problem analysis above, the structure of implementing the callback function plays an important role. From Figure 19 below, it can easily be comprehended that using animation to slow down the velocity of executing the program process is only demanded when the bot player is playing. However, no matter which turn is the bot or human player, the player hand token needs to increase and switch the turn player to the next. Therefore, those actions should be implemented in the callback function even if the combination of participants has human and bot, and the checking condition of whether the next player is a bot is also required to carry out here.

Additionally, this call-back function can be passed as a parameter to an animation method by using method reference. And on the GUI, the run method of the Runnable Interface should be called to match with this callback function and invoked it.

At last, the matched run method can be called in the onFinished method which is provided by the animation. This method will execute the callback method while the animation is entirely completed.

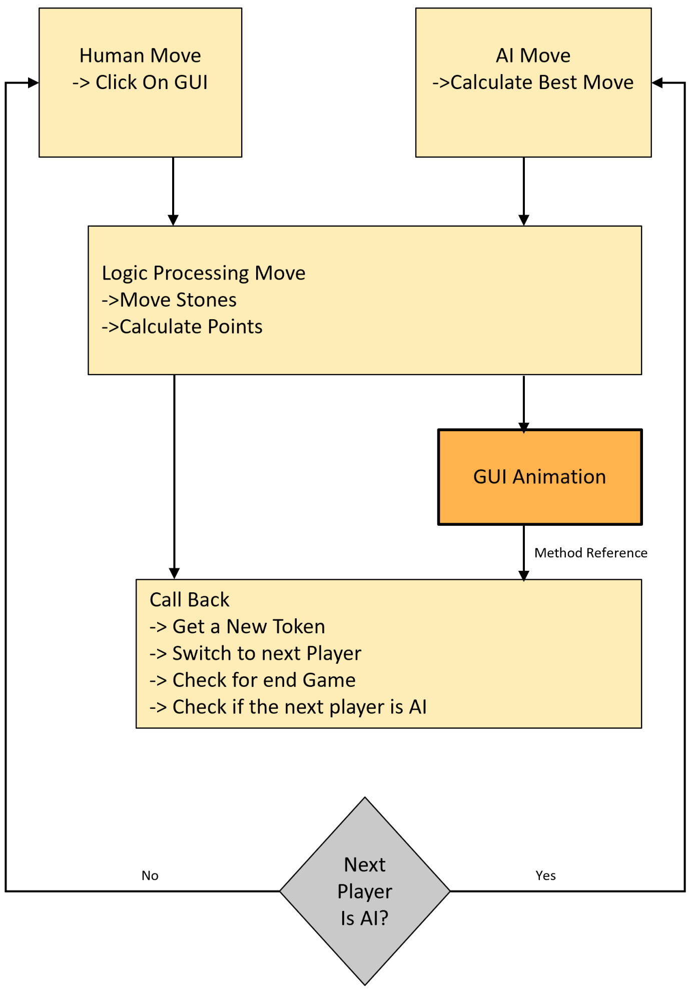


Figure 19: The Structure of callback function.

2.3 Program organization plan

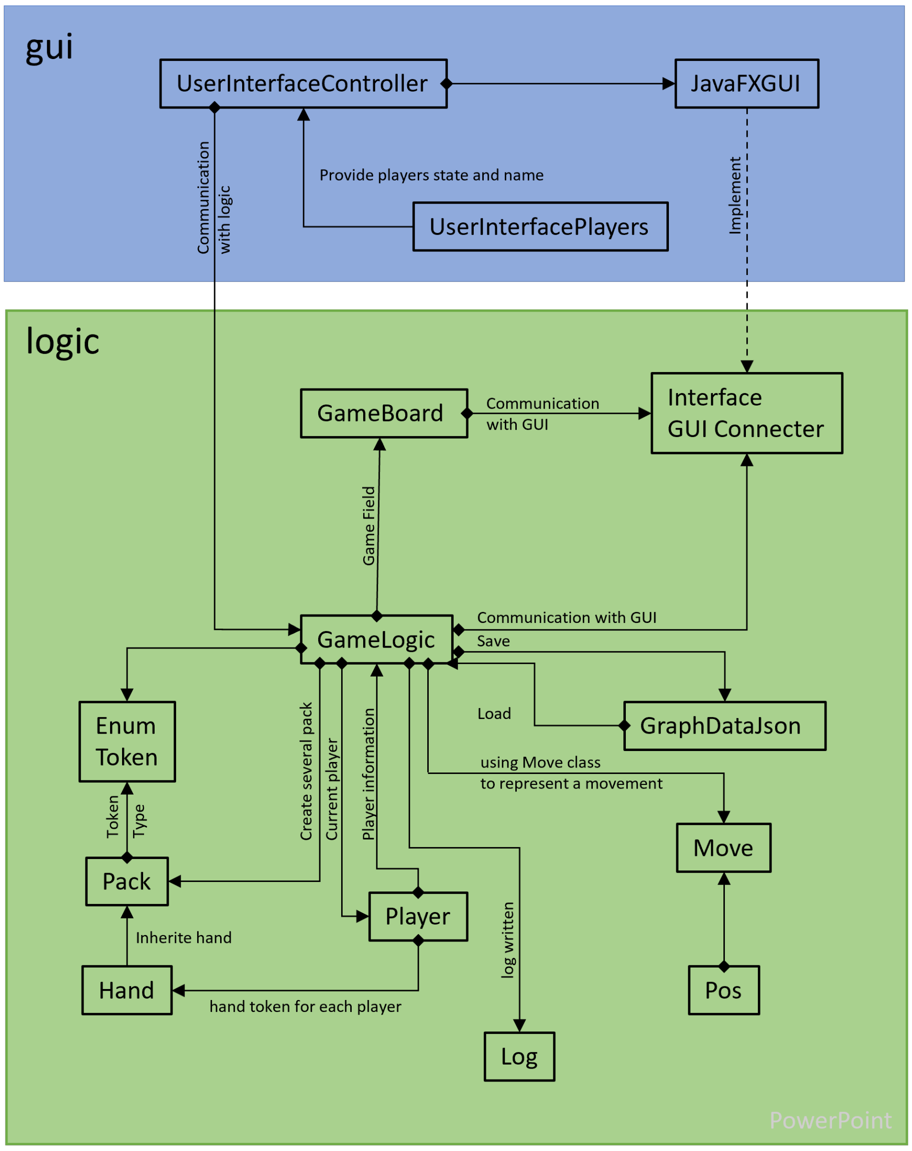


Figure 20: The classes diagram

2.3.1 The package of GUI

2.3.1.1 UserInterface Controller and UserInterface Player

The first FXML interface controller is responsible for the main scene and communicates with the logic package. Any performance of the users is exhibited on this primary User Interface, for example, players can operate the game, select the provided options on the menu bar, and observe the current game situation.

For the second scene UserInterfacePlayer, it manages the initialization of the state of participants. Players can manually insert their names and decide which players will be played by the bot.

With regards to the communication of these two windows as the Figure 21 below, the first interface establishes a reference of the second window in order to getter method to pass players' names and states to the main window. It can clearly conclude that it is a one-direction for passing information from the first scene to the second scene.

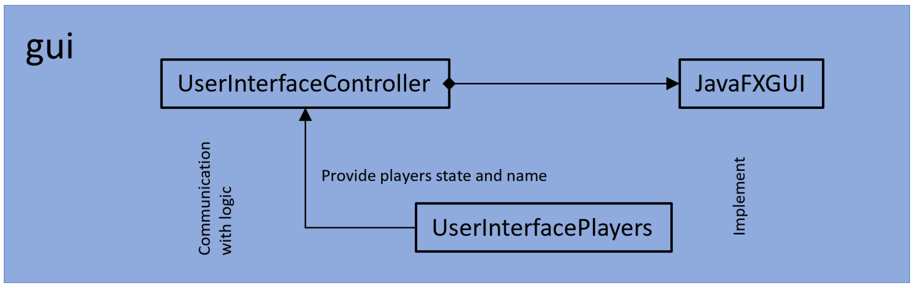


Figure 21: The GUI package

2.3.1.2 JavaFxGUI

The aims of all the methods in JavaFxGUI allow the logic class to change the output GUI. In fact, the JavaFxGUI can be implemented by the class GUIConnector, which is being used in the logic package. For example, while the logic class needs to modify the chessboard by placing a symbol token, it is undoubtedly that the display of the interface also needs to update the game field simultaneously. Thus, the function of the class JavaFxGUI can be utilized for this purpose.

2.3.2 The package of Logic

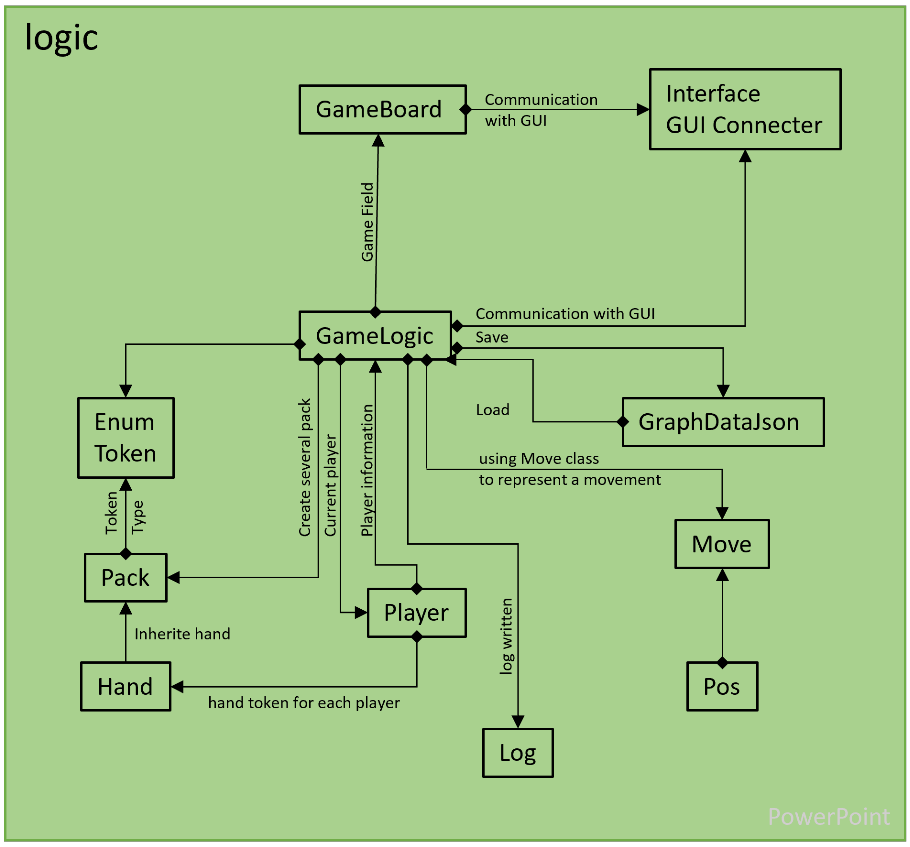


Figure 22: The Logic package

2.3.2.1 Game Logic

The class Logic is mainly controlled the whole program of the game, the most important core class among it. All other classes will be implemented here and be integrated into the logic of the whole game such as the logic algorithm of the AI, the logic program to determine the winner, and the initialization of the whole game.

From Figure 23 below, it is apparent to recognize that the class Logic also needs to communicate bidirectionally with other classes to update some features of the game.

In detail, the class logic frequently interacts with the class player to track the feature of the current player. And lead this feature to the GUI package to display on the main interface in order to become a condition, which is for the participant to realize who needs to play now.

Additionally, the class Logic also produces an interaction with the class Pack to have a global constant of a full pack, which functions to count the number amount of tokens. Not only does a full pack of all tokens exist but also a pack keeping every action token is required in the program. While a token is being used, the token pack needs to be diminished as a used token each time.

Furthermore, as mentioned in the top description, the class Move and the class Position are also being used in some methods of analyzing AI movement. As a tool, they are generated as an object corresponding to on their class type.

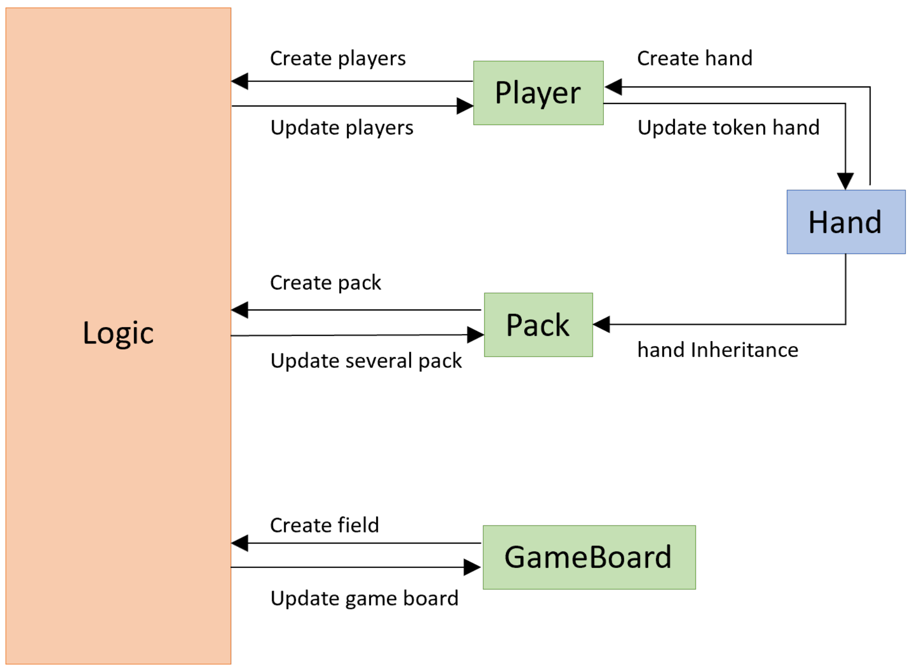


Figure 23: The relationship between GameLogic class and other classes.

Here has two essential and noticeable developing logics that needs to be mentioned below:

2.3.2.1.1 Human

Regards to the logic of the human player playing the game, a few coordinates of the game board and the player's hands are considered the fundamental gist. While a human player uses click or drag action to play the game, the coordinate of those actions needs to be recorded and passed to other functions to accomplish other aims.

2.3.2.1.2 AI logic

In the AI logic, it is important to claim that there are some rules that existed in the game. As the turn is in AI, the idea for AI to pick up the best token to get the best score or even attempt to win the game has its own logical design. In the developmental of our logical design, the two main types of token: Symbol token and Action token are introduced and follow the rules below:

1. **Winning a game by laying a line of six**
   The player should be able to use the token to complete six tokens in a line and win the game. This condition is the priority of winning the whole game. However, one thing that needs to realize is that the action token must always be utilized first and then the symbol token is the second option.

2. **Prevent another team to complete six**
   Since another team has accumulated five tokens in a line on the chess board, the player should be aware of and use tokens to interrupt the possibility of winning the game for another team. Likewise, the action token must be used in advance.

3. **Achieve the best score**
   However, achieving the best score has a different arrangement that the symbol token must be considered first when the player has a symbol token on hand. The action token will only be considered when there is no line of six to be completed and there are no symbols token on the player's hand,

2.3.2.2 Game board

A class Game board is represented as a field generated by every token which has been arranged on the chess board. All the methods existing in this class are related to the changes in behaviour or some identified use. It is apparent to observe that the game board class is directly output to the logic class as the player drag and drops a token on the board or modifies some changes.

2.3.2.3 Player

The class Player aims to create an object player in a game. The object of the player involves several attributes, for instance, the name of the participant, the active state, the AI state, and the hand token for the participant. In the constructor of the class logic, the player class can be implemented with the purpose of creating several players for game playing. And also the logic class can keep tracking the current player with the player object.

2.3.2.4 Pack and Hand

According to the understanding of the game playing, a pack of storing tokens and a hand token of each player is an indispensable element in a game. Hence, with the help of many methods, the class logic can utilize the Class pack to insert, remove or increase tokens in a pack or player's hands. Moreover, because of the inheritance relationship of the class pack and class hand as shown in Figure 24 below, those methods in the pack class are also applicable to be utilized for hand class.

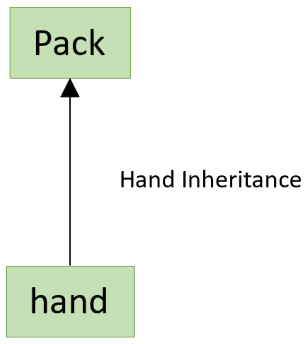


Figure 24: Pack and Hand

2.3.2.5 Enum Token

Regarding the Enum class Token, every token is numbered in an enum category. An advantage of sorting those tokens in the sequence is that other classes are able to invoke the specified token by an ordinal. The sequence of tokens is shown below Figure 25.

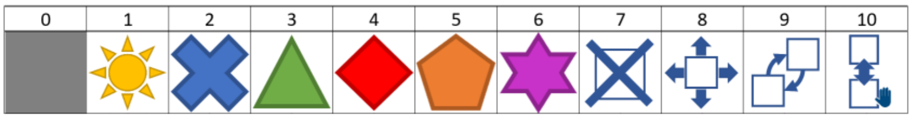


Figure 25: The sequence of Enum tokens

2.3.2.6 Move and Position

The function of the class Move and the class Position is principally the same. Mostly they have a constructor which includes an object to record the necessary attribute and this object can be passed to any class which is required. Indeed, the class Move even contains a position object to perform the movement position as shown in Figure 26 below. Among the position object, it has two attributes x and y, which are the coordinate of the chess board.

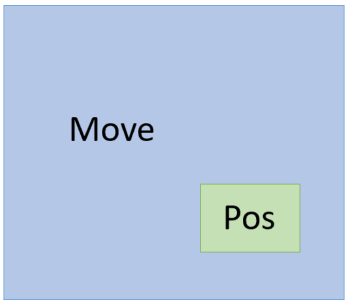


Figure 26: The Move and Position class.

2.3.2.7 GraphDataJson

In this project, the conception of Save and Load is based on the JSON file. To parse a JSON file, GSON is chosen for implementation. And this class contains all information, which is necessary for generating an unfinished game that would be established as a variable in class logic. No matter which operation of Save and Load to use, both of them must through this variable to transform data. The process of transforming data is as below in Figure 27.

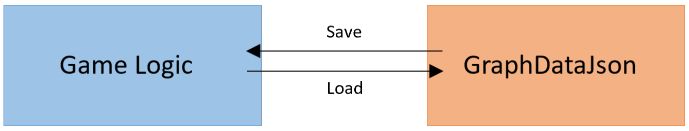
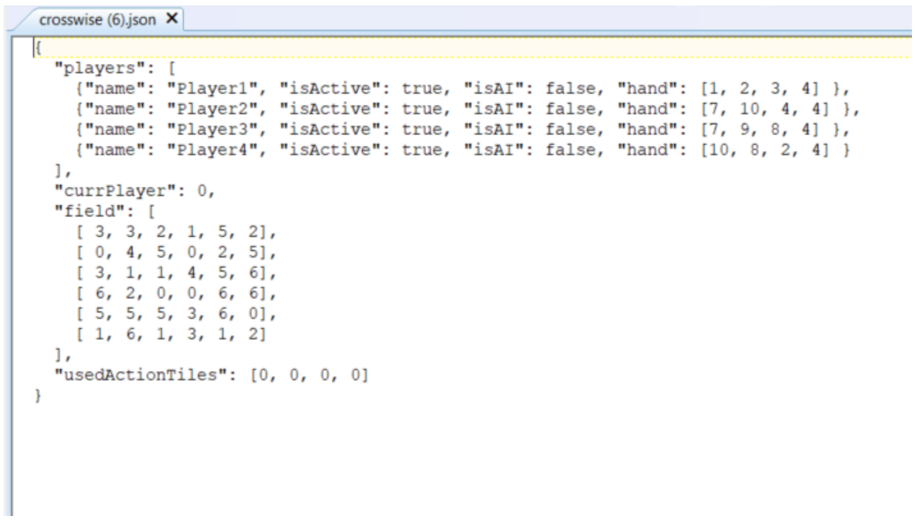


Figure 27: The GraphDataJson class and GameLogic class.

2.4 Files

2.4.1 Save and Load File

A file existed for a game to record the game situation, and continuous playing of an unfinished game or a customized game can be executed. The file is written as a text by using GSON format to represent the whole game situation, just as the Figure 28 below.



crosswise (6).json

```
{
  "players": [
    {"name": "Player1", "isActive": true, "isAI": false, "hand": [1, 2, 3, 4] },
    {"name": "Player2", "isActive": true, "isAI": false, "hand": [7, 10, 4, 4] },
    {"name": "Player3", "isActive": true, "isAI": false, "hand": [7, 9, 8, 4] },
    {"name": "Player4", "isActive": true, "isAI": false, "hand": [10, 8, 2, 4] }
  ],
  "currPlayer": 0,
  "field": [
    [ 3, 3, 2, 1, 5, 2],
    [ 0, 4, 5, 0, 2, 5],
    [ 3, 1, 1, 4, 5, 6],
    [ 6, 2, 0, 0, 6, 6],
    [ 5, 5, 5, 3, 6, 0],
    [ 1, 6, 1, 3, 1, 2]
  ],
  "usedActionTiles": [0, 0, 0, 0]
}
```

Figure 28: The representation of GSON

2.4.1.1 Save

In the program, the player class could provide the name of the player, active state, AI situation, and the hand token of this player and transfer them into text format.

Additionally, the current player and the field of the whole game board could be transformed into a number representation. Although the board tokens are presented as an enum element, in the board class, we create a method that can be transforming that enum element into an int representation.

At last, this archive also must involve the list of the used action token, which can remind players of the amount remained number of the action token. The list of action tokens has already used the int array to track in the logic class.

Those transferring behaviors would need to use the getter methods of the class GraphDataJson to complete.

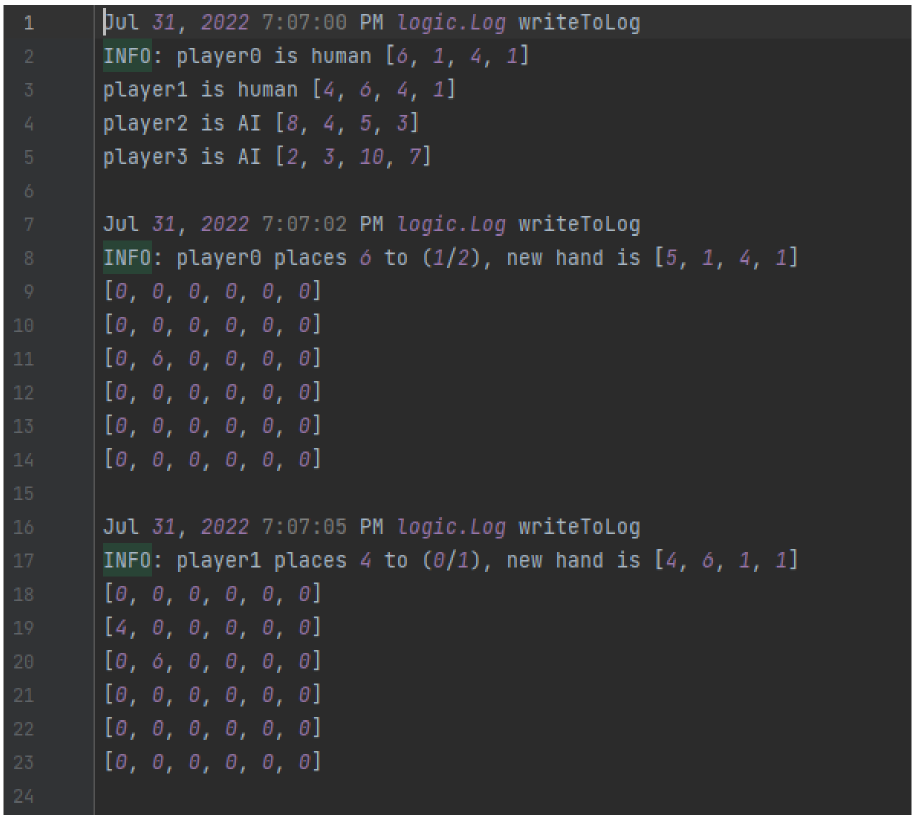
The ultimate text representation is shown as Figure 28.

2.4.1.2 Load

Likewise, loading a game with an archive, in which the file format is as the Figure 28. And the recorded attributes must be bound with the attribute in the program. With the purpose of assigning the value to the program attribute, the setter method of the class GraphDataJson could be used to set up.

2.4.2 Log File

A log is written to a file so that the programmer can retrace the steps in the event of an unexpected program response. This file is overwritten or recreated each time the game is started. In the program, a class Log is created to use a logger object to log messages for a specific system or application component. Furthermore, a simple method "WriteToLog " is created in order to let Class logic implement and conveniently document when an event happens, for example, a player places a token on the board. Furthermore, there is also some situation when the game encounter a problem such as the no Token in the deck and the player could not have a new Token. The representation of the Log File is shown as Figure 29 below.



```
Jul 31, 2022 7:07:00 PM logic.Log writeToLog
INFO: player0 is human [6, 1, 4, 1]
player1 is human [4, 6, 4, 1]
player2 is AI [8, 4, 5, 3]
player3 is AI [2, 3, 10, 7]

Jul 31, 2022 7:07:02 PM logic.Log writeToLog
INFO: player0 places 6 to (1/2), new hand is [5, 1, 4, 1]
[0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0]
[0, 6, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0]

Jul 31, 2022 7:07:05 PM logic.Log writeToLog
INFO: player1 places 4 to (0/1), new hand is [4, 6, 1, 1]
[0, 0, 0, 0, 0, 0]
[4, 0, 0, 0, 0, 0]
[0, 6, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0]

```

Figure 29: The representation of Log file

2.5 Program testing

2.5.1 AI algorithm

In this section of testing removing the action token for AI, there are three main essential to evaluate: Has six same tokens for the current team, prevent another team from reaching six tokens, and achieve the best score. Additionally, this testing includes external files. Follow the steps below:

- Check the folder ”AI algorithm”, and the subfolder ”remove token”.
- Select which teams: ”Orange” and ”Green”.
- Each had three test cases.

2.5.1.1 Symbol Token

Test case	Result
While there are no action tokens on hand and the current team has a specific token to accomplish the six in a line, this specific token need to be chosen and placed where it should be. In fact, the coordinate of lacking the specific token must be empty as a condition.	The expectation is that the player could place the right token in the right place. And the result shows as expected.
While there are no action tokens on hand and the opposite team has a specific token to cease the six tokens accomplished by the opposite team this specific token need to be chosen and placed where it should be. In fact, the coordinate of lacking the specific token must be empty as a condition.	The expectation is that the player could place the right token in the right place to interrupt the opposite team win a game. And the result shows as expected.
Generally speaking, when the player has a series token on hand and achieving the best score is the priority of the game. And the symbol token needs to accomplish this goal. The test case is that the current player will use one of the symbols tokens to achieve the best score for his/her team.	The expectation is that the difference between the two team scores should be as large as possible. And the result is shown as expected even though there is no chance to reduce the opposite team.

2.5.1.2 Remove Token

Test case	Result
The current team has a chance to complete six same tokens. Use remove token to remove the unique Token on a line that has accumulated five same tokens.	The expectation is that the unique token has been removed and returned back to the players' hands. And the result shows as expected.
The opposite team has a chance to complete six same tokens. Use remove token to remove one of the same Tokens on a line that has accumulated five same tokens.	The expectation is that the coordinate of the removed token is smaller the further it is up and the further it is to the left. And the result shows as expected.
The current has a full action token on hand including one remove token, which has the smallest index on the hand. The test case is that the current player will use this removed token to achieve the best score for his/her team.	The expectation is that the difference between the two team scores should be as large as possible. And the result is shown as expected even though there is no chance to reduce the opposite team.

2.5.1.3 Shifter Token

Test case	Result
The current team has a chance to complete six same tokens. Use shift token to shift the unique Token on a line that has accumulated five same tokens to the empty cell on the line.	The expectation is that the unique token has been founded on the board and shifted to the empty cell of the line in order to complete six tokens. And the result shows as expected.
The opposite team has a chance to complete six same tokens. Use a shifter token to shift one of the same Tokens on a line that has accumulated five same tokens to the empty cell which has the smallest row and column.	The expectation is that the coordinate of the shifter token start from the first same token of the line and the target empty coordinate is smaller the further it is up and the further it is to the left. And the result shows as expected.
The current has a full action token on hand including one shifter token, which appears on the smallest hand index. The test case is that the current player will use this shifter token to achieve the best score for his/her team.	The expectation is that the difference between the two team scores should be as large as possible. And the program will find the token that can cause the smallest effect on the current team and the most serious losing point to the opposite team. And the result is shown as expected.

2.5.1.4 Exchange Token

Test case	Result
The current team has a chance to complete six same tokens. Use Exchange token to Exchange the unique Token on a line that has accumulated five same tokens to the position which acquired this unique token.	The expectation is that the unique token has been founded and switched to the position in which the line has had five same tokens.If there are two needed tokens appear on board, then the program chooses the smallest row and column index. And the result shows as expected.
The opposite team has a chance to complete six same tokens. Use Exchange token to exchange one of the same Tokens on a line that has accumulated five same tokens to the token which is located as close as the left top corner.	The expectation is that one of the exchanging tokens is decided as the lowest index on the line no matter what row or column. And another exchanging token is the token that appears closest to the left top of the chess board. And the result shows as expected.
The current has a full action token on hand including one Exchange token, which has the smallest index on the hand. The test case is that the current player will use this Exchange token to achieve the best score for his/her team.	The expectation is that the difference between the two team scores should be as large as possible. And the program will find the token that can cause the smallest effect on the current team and the most serious losing point to the opposite team. And the result is shown as expected even though there is no chance to reduce the opposite team.

2.5.1.5 Replace Token

In the test case of the Replace token, only two considerations are being tested. The "Achieve the best score" test is not unable to implement in the case of the Replace tokens. Because of the characteristic of the Replace token, it needs to contain at least one symbol token on the token hand. However, according to the designing logic, achieving the best score always start from the symbol token.

Test case	Result
The current team has a chance to complete six same tokens. Use Replace token to switch the needed Token on a line that has accumulated five same tokens to the cell which needed this special token.	The expectation is that the unique token has been switched with the token, which appears on the gap of the line and it returns back to the player's hand. And the result shows as expected.
The opposite team has a chance to complete six same tokens. Use Replace token to switch one of the same Tokens on a line that has accumulated five same tokens with one token on hand, which is not related to the line.	The expectation is that the index of the replaced token from hand is as low as possible. And this token must be different from that special token on the line. And another switching token also has the lowest index of column and row. And the result shows as expected.

2.5.2 Test Load

Test case	Result
Test if the current player is AI, and check if the program is able to execute perfectly. A created testing file in the folder "Test Load". After creating a new game, select the Load menu button and choose "Current Player is AI".	The expectation is that the program will still work perfectly and the AI player will start to place tokens on the board. And the result shows as we expected.
Test if the current player is human, and check if the program is able to execute perfectly. A created testing file in the folder "Test Load". After creating a new game, select the Load menu button and choose "Current Player is Human".	The expectation is that the program will still work perfectly and the human player can start to play the game. And the result shows as we expected.
Test if all participants are AI, and check if the program is able to execute perfectly. A created testing file in the folder "Test Load". After creating a new game, select the Load menu button and choose "All AI player".	The expectation is that the program will still work perfectly. Each AI participant places their desired token on the chess board correctly. And the result shows as we expected.

2.5.3 Test Save

The purpose of this test case is that a game has been started to play and store as a JSON file by saving. And those saved files need to be invoked and loaded to ensure the game situation is displayed as the same. Thus, there are few external files that have been created in advance.

Following the step below to load the specific test case :

- Check the folder "Test Save".
- Select the specific case.

Test case	Result
From JSON format, a hand token contains null.	The expectation is that the GUI will display the null expression on this hand. For example. there is no image on the required position. However, the result did not show as expected. The position of the null still has an image on it. This image is the same as the image of the one smaller index position. But one noticeable point of the null token is that even if the null position has an image, the human player cannot do any action in this area. Hence, the null position will not cause a NullPointer exception at last.
Test a saved file in two players. When a game with only two players is stored in JSON format, the third player and fourth player have not been activated. Then their hand also has no Token. Nevertheless, the syntax of the token hand will be stored as empty. This test case needs to ensure those empty hands could be transferred and bounded.	The expectation is that due to the third player and fourth player do not attend the game, the game must not show the GUI display. And also the turn of the player will only switch between first and second. The result is entirely shown as expected.
Test case for customized used action token. The display of the remaining action token must not as the same as the content of the used toke array.	The expectation of this test case is that there is one action token is being used for Shifter, Exchange, and Replacer. But the removed token stays the same. Therefore, the display number of the action token must be 3, 2, 2, 2. In comparison, the customized used action token array is 0, 1, 1, 1. Fortunately, the results show as expected.

2.5.4 The restriction of different tokens and the duration of highlighting the test

2.5.4.1 Remove Token

Test case	Result
While the chess board does not contain any token, and the human player tries to use the Remove token function.	The expectation of this test case is that the human player could not select Remove token on hand. And the result of this testing shows precisely what we have expected. The human player cannot click the Remove token when there is nothing to remove from the board.
A Remove token is being triggered and observe the changes on the player's hand.	The expectation is that the player's hand will not get a new token and simply just get the token, which is removed from the board. And the final result shows the expectation perfectly.
Test duration of remove token. The removed token should be marked and the marking sign should only exist for a short time. After beginning a game, click on the options menu to decide the duration time of highlighting.	The expectation is only the highlighted region of the removed token on board should fade out. And the marked region of the removed sign should not change and stay as normal. And the result shows as expected.

2.5.4.2 Shifter Token

Test case	Result
While the chess board does not contain any token, and the human player tries to use the shifter token function.	The expectation of this test case is that the human player could not select the Shifter token on hand. And the result of this testing shows precisely what we have expected. The human player cannot click the shifter token when there is nothing to select and shift.
Test duration of shifter token. The shifter token should be marked and the marking sign should only exist for a short time. After beginning a game, click on the options menu to decide the duration time of highlighting.	The expectation is both coordinates of start and target on board should fade out. And the marked region of the shifter sign should not change and stay as normal. And the result shows as expected.

2.5.4.3 Exchange Token

Test case	Result
While the chess board does not contain any token, and the human player tries to use the Exchange token function.	The expectation of this test case is that the human player could not select Exchange token on hand. And the result of this testing shows precisely what we have expected. The human player cannot click the Exchange token when there is nothing to switch.
While the chessboard only contains a symbol token, and the human player tries to use the Exchange token.	The expectation is that the Exchange token cannot be triggered on hand. And there is a warning alert shown up to remind players cannot use the Exchange tokens.
Test duration of Exchange token. The exchange token should be marked and the marking sign should only exist for a short time. After beginning a game, click on the options menu to decide the duration time of highlighting.	The expectation is both coordinates of first changing and second changing on the chess board should fade out. And the marked region of the exchange sign should not change and stay as normal. And the result shows as expected.

2.5.4.4 Replace Token

Test case	Result
While the chess board does not contain any token, and human players try to use Replace token function.	The expectation of this test case is that the human player could not select replace token on hand. And the result of this testing shows precisely what we have expected. The human player cannot click the Replace token when the board is empty.
Test duration of Replace token. The replacement token should be marked and the marking sign should only exist for a short time. After beginning a game, click on the options menu to decide the duration time of highlighting.	The expectation is both coordinates of the first changing on hand and the second changing on the chess board should fade out. And the marked region of the Replacer sign should not change and stay as normal. And the result shows as expected.

2.5.5 During the game

Test case	Result
While no token on the board, the player holds four action tokens. There is a created testing file in the folder "During the game". After creating a new game, select the Load menu button and choose "No available token".	The expectation is that the current player can still click or drag the token on hand in attempting. However, a warning alert pops out that No available token for the player, and please start a new game. The result is shown as expected because the current player cannot do anything and the game should be restarted.
Players' hand has three replace token and no symbol token. A warning window appeared to aware the player cannot play further. For this test case, there is an external file created as the name "All replace token".	The expectation is that the player's hand will show three replaced tokens and one empty blank. However, the final result show four replace tokens. But the warning window has shown successfully. The purpose of this test is still worked as expected.

2.5.6 End the game

Test case	Result
While no token on the board, the player holds four action tokens. There is a created testing file in the folder "During the game". After creating a new game, select the Load menu button and choose "No available token".	The expectation is that the current player can still click or drag the token on hand in attempting. However, a warning alert pops out that No available token for the player, and please start a new game. The result is shown as expected because the current player cannot do anything and the game should be restarted.