

# Warring States Game

## Group members

Chunxiang Song u6302158

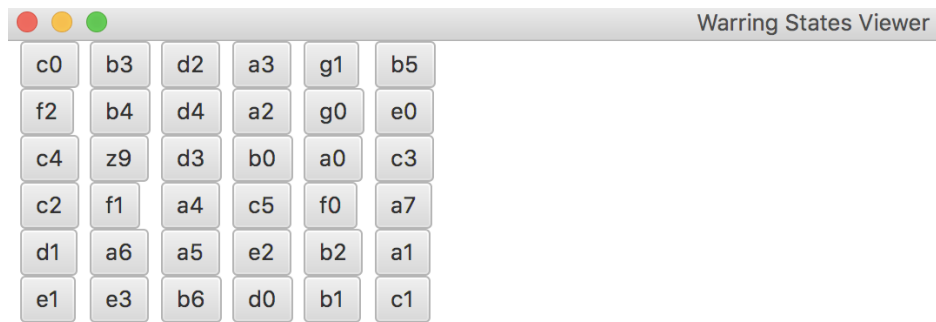
Chucheng Qian u6451847

Jiayang Li u6360655

COMP6710  
2018



---



Warring States Viewer					
c0	b3	d2	a3	g1	b5
f2	b4	d4	a2	g0	e0
c4	z9	d3	b0	a0	c3
c2	f1	a4	c5	f0	a7
d1	a6	a5	e2	b2	a1
e1	e3	b6	d0	b1	c1

In Task 5, its function is to determine whether a given move is legal. So I implemented this function by judging whether there is a card in the target position and the position of the card relative to Zhang Yi. In Task 10, The purpose of Task 10 is to generate a legal move based on the supplied string. So, I first get all the positions in the same row or the same column as Zhang Yi, and then determine if the randomly generated position code is legal.

## Jiayang Li part

In this project, I independently complete task 2, task3, and task 7. Besides, in the task 2, Chunxiang help me revise my code to make it concisely, and in the task7, I create another simplified function to support the job in task 8 as well. The best code for me can be the part in task 7, just because it is the basic function in the game, which may be used in task 8, 9, 11 to get supporters, and implement the game.

In task2, I just divide the String into 3 different char types and judge each of them based on the rules of the game. In task3, I use the function in task2 to test if each three of the char in the placement is correct firstly. Then, I use two double for loops for testing the duplicated characters and locations separately. In task7, using complementation and quotient to judge the direction from Zhang Yi can be the core. I set up four string to store all supporters at a time based on the move sequence, and the number of players will decide if the four string will be used. Finally, the output can be the sorted supporters of the input player ID. Besides, to make the code slightly concise, the sorting function will be separated.

---

## Chucheng Qian part

In this project, I have done task6, task8, task9 and task11.

For Task6, some structures about finding Zhangyi and searching the 4 directions of Zhangyi are from Task5 written by Chunxiang Song. I create two new methods "CountMoveSequenceValid" which will count the number of all valid moves from the original move sequence depends on the current board and "GetNewSetup" which will turn the board setup from its "ArrayList<String>" form to the "String". I update the board with a single move each time as an iteration is used in "CountMoveSequenceValid". Then I compare the number of all the valid moves with the number of all moves and return the result in the "isMoveSequenceValid".

For Task8, I did it with the new method "getEachPace" written by Jiangyang Li. In the Task8 method "getFlags", I check the flag of each Kingdom and then store the owner of each flag followed by the number of supporters of this owner for this Kingdom in the record char list step by step. Each step I use one move, then update the board and continue.

For Task9:

Some supplementary method/class:

1. An inner class "Card" which extends "Button" to set the event handler to the object 'card'.
2. A Class "CardImage": It contains two methods which set images to cards and supporters respectively.

It also contains many Image and ImageView objects used.

3. 'generateRandomSetup()': Take 'generateRandomSetup()' from TestUtility.java with little bit of changes to generate a random setup for the game.
4. "updateBoard(ArrayList<String> cardLocalList1, int target)" to update the current board according to the target location.
5. "WhetherFinished(ArrayList<String> presentBoard)": Search the upper, lower, right and left side of Zhangyi to make sure the game is finished.
6. "Winners": Find the winners when the game is finished

---

7. "Notice": Show some notice to the user, depending on the input String.

8. "showSupporters": Show supporters of one player on the view.

"GetFlagofSepecificPlayer" and "showFlags(ArrayList<Character> list,double x,double y)" :Get all the flags owned by one sepecific player and show them.

Main Structure:

1. "putPlacement": It puts the object card into the grid and used the "makePlacement" in Task4.

2. "NumberofPlayerEqualtoX": Methods named like this show flags and supporters and allow the next player to put his move.

3. "StartingView","WelcomeView" and "SetupGrid": Show the start view, welcome view and grid.

4. "NumberOfPlayerWhenWithHuman": Allow the user to type in the number of players from 2 to 4 and start the game.

5. "start(Stage primaryStage)": add things to root and call the "WelcomeView()" to start the game.

For Task10:

1." NumberOfPlayerWhenWithComputer": Allow the user to type in the number of players from 2 to 4 and start the game. As there is only one human player this time, therefore I set the id to be 50 to make this situation can be separated when updating the board and setting flags and supporters.

1. Methods named like"NumberofPlayerEqualtoXWhenComputer": It updates the board with the move put by the user and then checks whether there is no valid move at first. Then it continues to generate a new move, updates the board and checks whether there is no valid move. The times of this process depend on the number of players involved.

## Screen shots of the game

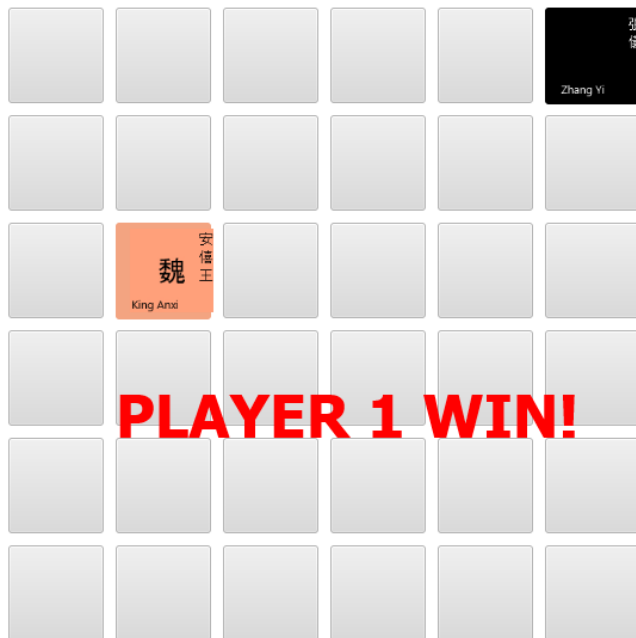


# Warring States Game



Invalid move, try again!

# Warring States Game



PLAYER 1 WIN!



Player0  
You



Player1