Project Report

Student 1 : Kim Chhorn Sambath

Student 2 : Het Patel

In this assignment 2 folder we have a sub folder SRC which contains 4 files:

1. checkergame.java
2. filehandler.java
3. playerCondition.java
4. playerRecord.csv

# Checkergame.java

this file is containing menu for the game. The menu is created using DoWhile loop which contains different options and ask user for an input from the above options. Then each option is checked through an if condition.

**Option 1**: Start game.

This option will first ask the user to enter the players name after which board method is called to create board. After than from playerCondition class playerXmove method is called which initializes player X move before which a display board method is called which will display the board. Later it checks the player x win condition with an if statement and if the conditions are true, it updates the record and then it breaks.

Moreover, later board is again displayed if win condition is not true and then playerOmove is called from playerCondition class then as same player win Condition is checked and if it's true it will update the record accordingly.

If user select **option 2** it displays the rules to play game. This is done using an elseif condition.

If user select **option 3** it calls readrecord method from filehandler class which prints the record previously updated. This will be repeated until user selects **option 4**.

Later, in this file it contains displayBoard method. This method first prints 0 – 7 which are indexes. Then a for loop is used which start from 0 until the boards row length and increased by 1 each time. In this loop it prints a line first and then it prints loop iteration number and then later it also has one nested for loop which starts from 0 to board current iteration length it prints on standing line and space which creates one cell like structure after which one more line is printed which is ending line.

Furthermore, it also contains one board method which contains a 2d array with correct placements of each element and this method return that board.

# playerCondition.java

in this file the first method is playerXmove.

## playerXmove()

in this method first it prints the respective player turn which is player x and then ask the user to move his element from destination after which it will check for user input that is that correct or not this is done by checkInput method. the user will enter two number when program ask from user “move x from” than the program will divide the user input from 10 and the remainder will be considered as column and rest will be considered as row so the user have to enter two numbers in row/column format for example (if user input 46 6 will be consider as column and 4 as row) . further it will ask for destination location and user have to enter in the same way as destination and then again checkInput( ) method will run to check if the user have entered valid destination location. After the inputs the first if checks if the “x” that user wants to move is a queen if it is then it will move with method “checkXorKmove()” else it will move with normal moves which is checkXorKmove without “k” parameter (so, we have used method overloading here to check normal move or queens move) rather than exceptional queen moves.

## playerOmove()

same as playerXmove method now it will ask player O to move his element by asking for destination and location oRow and oColumn are variables that represent current location of element (just like xRow and xColumn) while moveOrow and moveOcolumn represents destination location or location where it has to move. Rest in this method is same as “x” but instead we have a different method checkOorQmove which works same as in “x” to check queens' movement (just instead K it has Q, however method overloading is used in this one as well)

## checkInput()

in this method inside the do loop the first if condition check if the input length is 2 and if its length is 2 then it will check if the input is character or not (this is checked by using a for loop so we can check for the whole input the user gave.)

further the else if condition checks if the user input is more than two number and if it is it will assign true to wrong which will prevent the code to run further and ask the user to input again by showing an error.

So, the next if will be done when the user enters wrong input and that is why the above “If” and “else if” conditions meet it turns the wrong to true and the loop will not break until wrong is false.

After which we convert the user entered string into the integer so we can further use it as index for movements.

Overall, if the user enters a wrong input, it will turn the wrong to true which will display an error message and the user will be asked to input again.

## checkXorKmove()

this method has arguments that contains board, xRow , xColumn , moveXrow and moveXcolumn.

For Reference:

xRow : current elements row

xColumn : current elements column

moveXrow : destination row of element

moveXcoumn : destination column of element

the first if condition checks that the current location selected by user should have X, if the destination location has X and if the destination location has O.(here destination location refers to location where the user wants to move.) if this conditions will be true it will print wrong coordinate and again playerXmove will run with board value passed with it by recursion.

The else if condition checks if the sum of current location and the sum of destination location is same because it is diagonally right movement along with this one more condition must be true which is row of destination + 1 should be equal to row of current position and if this conditions are true than the current location of x will be converted to a blank space and destination location will be filled with “X” . the nested “if” checks if the movement leads the x to 0 and if it do so it will turn it into queen which is letter “K” in case of “x”(in case of “o” it is “Q”) and at the next step we have a return which will return to the first if or the last else condition of the same method when it has a recursion. And if the moveXrow is 0, it will turn array element to 'K' that is represent as a Queen of X.

The next else if checks the same movement but this time the jump is at the diagonally left side and that is why in the else if condition, we have added 2 to destination column.

The third else if condition below the comment “//jump over 0 or Q” is to check if we have 2 step jump and if we have a “O” in the middle of the current location and destination location (the middle location is: board[moveXrow+1] [moveXcolumn+1]). So this else if have 4 condition the first 2 checks diagonal movement while the other two checks if there is a O or Q(queen of O) in between , if it will be the case it will replace the current location of x with empty space along with the location at where the Q or O Is present And will display a X at destination location. Overall, this condition is for eliminating O or Q if present diagonally right.

The same way next else if condition is for elimination but this time the O or Q might be present at the diagonally left side.

If this all “if” and “else if” conditions don't meet, then it will run the else which will print a message “invalid move please enter again” and then will call playerXmove method to again enter the player x movement.

## checkXorKmove()

this next checkXorKmove method is checking movement for queen of player X the first if condition is checking if the element selected by the user is not queen of “x” or it is normal “x” or it is player “o” or it is queen of “O” that Is “Q” if this conditions are fulfilled it will print an error message and playerXmove() method is call again to enter the input again by recursion. The following four else if are same as above method that check upside right movement upside left movement, upside right jump 2 and upside left jump 2. the different is the fifth condition it checks for down movements because queen can even move downwards so just by checking conditions similarly up till now it will display a space at current location and at destination location it will display “K”. Furthermore, just like above methods it than checks for down right then down 2 steps left and down 2 steps right, at the end if none of the above conditions are meet so it will again print an error message and will call playerXmove() method to again enter input (this code is in “else” block.)

## checkOorQmove()

this method also works same as checkXorKmove() method but it is for player “o” so its condition will be different because “O’s” movement will be from upwards to downward. However, the ideology behind the code is same (means conditions for checking movement and checking for valid move is same as above with slight modifications).

## checkOorQmove()

the same goes with this method as well it will check movement for queen of “O” that is “Q” that is why it have “char Q” in its arguments. But the if conditions and rest is similar to checkXorKmove() method that checks for queen movement of “X” just the conditions are slightly modified because as mentioned “O” movement according to board will be from up to down.

## checkXwinner()

CheckXwinner method is to use for checking that whether there is any X block O or not and whether there is any O on the board or not. So, if X blocked O or there is no O on the board, so X is a winner.

this method checks the current conditions of the board to check the conditions for winner however this method checks the winning conditions for player “X” . the for loops start running from row 0 to board length and the inner for loops runs for columns same from 0 to board[row].length because it represents column length the first if condition checks if at the location “x” is not present and instead if “o” is present than x is not a winner. the next if condition checks if the row is less than 5 and the nested if in it checks if column is not 7 because if column is 7 checks diagonally it would be meaning less and will prompt an error of indexoutofbounds because of length of board. The next if condition is that if we are I column 6 we can't check the second row diagonally ahead (so to prevent indexoutofbound error). overall columns 1 and 6 are checked to prevent error for second row while columns 0 and 7 are checked to prevent first and second. So, it will only check row 1-5 because if we check more further it must check row 8 which will throw and error of index out of bound.

if there is queen on the board. the first if condition checks if the queen is located in a row position greater than or equal to 2 and smaller than or equal 5. Since a queen can move both backward and forward, this condition ensures that the code only checks for potential moves if the row position allows for at least two rows in front of the queen and further the code checks for empty cells around the opponent queen “Q” and If any of this cells are found it sets Xwinner to false because opponent still has moves left .

the next if condition check if the queen is in row 1 and its moves to right and left one row ahead of the queen is empty because it indicates the queen still has some moves left however if the queen is in between row 0 to 5 it checks for moves like two rows ahead with similar conditions.

## checkOwinner()

This method is just similar to checkXwinner() method, and it checks for the win condition of player O, it just with changes because player O is opposite side. Just like the above we have conditions for queen of player “O” which is “Q” but with opposite side.

# Filehandler.java

## writeRecord()

It is a method that write the string that display when either player X or player O win into the playerRecord.csv file, therefore the parameter of this method is string. The idea is that write that string into file every time when string is display when someone wins, so that string can use for displaying and use for a record of players.

## readRecord()

This method is used for only read the string in the playerRecord.csv and add a number before the string sentence so I use a loop every time when the file is read to show that number of pervious players.

# Instruction

1. The format of the coordinate for inputting to the program is RowColoum.for example, 12 is equivalent to row 1 and column 2. Note: Do not make a space between these two intergers. If the format of coordinate that the player input is wrong like more than 2 intergers or not interger, it will display error and input the coordinate again. Note. If the format of first input is wrong, it will display an error and ask to input again immediately after the first incorrect format input is entered.
2. first, Inputting the coordinate of location of what piece that player wants to move and then inputting the coordinate of destination that player wants their selected piece to move. If the move is invalid, the error message will display, so then player just inputs the coordinate again.
3. the symbol of queen of X is K and the symbol of queen of O is Q.
4. A player can only move diagonally upwards but just one step. This move is only applicable if the destination is empty.
5. A player can eliminate another player only if that player is exactly diagonally next to that another player and the next diagonal place to opponent which is that another player is vacant.

In short, jump over that piece and eliminate that piece.

1. A player X and O can only move one step diagonally.
2. A player X or O becomes queen if it reaches the other side.
3. queen pieces can move in both directions, forward and backward.
4. You win the game when the opponent has no more pieces or can't move (even if he/she still has pieces). If neither player can move, then it is a draw or a tie.