CENG 1004 Spring 2020 Midterm Homework Documentation

A) Printing Board

Describe how you manage the print the board representation? How did you construct loops?

I manage the print the board representation with ChessBoard's toString method. toString method have empty String brdStr. ChessBoard have String array harf. Harf contains chess board's letters. Harf does not change, constant. Harf is specific to the ChessBoard class. Harf does not create when an object from each ChessBoard class is produced. Harf is created only when it is first called. toString method first add to brdStr letters at the top. Chess board have 8 rows so toString method have 8 times for loop for rows. Each loop add to brdStr line and row. Second for loop runs 8 times because of harf. Each loop take one letter from harf. This letter and row creates coordinate. After that second for loop take piece which have this coordinate p. If p is undefined for this coordinate, brdStr is added " | ". Otherwise p have piece, if p's rank is ROOK, if p's color is 0 that means p is white so brdStr is added R otherwise r. If p's rank is KNIGHT, if p's color is 0, brdStr is added N otherwise n. If p's rank is BISHOP, if p's color is 0, brdStr is added B otherwise b. If p's rank is QUEEN, if p's color is 0, brdStr is added Q otherwise q. If p's rank is KING, if p's color is 0, brdStr is added K otherwise k. If p's rank is PAWN, if p's color is 0, brdStr is added P otherwise p. After second for loop brdStr is added " | " and row. After first loop brdStr is added line. toString method last add to brdStr letters at the bottom. After that toString method returns brdStr.

What are the challenges? How did you manage to solve them?

The challenge is which piece is which one. I solved this problem with if method. If method takes one piece and control this piece is rook or knight or bishop or queen or king or pawn.

B) Defining Board and Square Classes

A board can have 64 squares. How did you define the relation between Board and Square objects?

ChessBoard class have 2 dimension 8x8 Square array squares. ChessBoard's constructor assign new Square objects to squares.

C) Implementing methods of Board and Square Classes

-ChessBoard class

public boolean isWhitePlaying()

- This method checks White playing or not.
- This method has not any parameter.
- ChessBoard has boolean flag and initial value is true. This method returns flag.

public boolean isGameEnded()

- This method counts how many white and black pieces.
- This method has not any parameter.
- This method returns if whites or blacks is 0 true, otherwise false.
- This method has 2 int variables whites and blacks. This method has nested for loops. Each for loop runs 8 times because of chess board's square. Nested for loops takes one Square from squares and controls this Square has Piece or not. If this square has piece, takes Piece's color. If color is 0, increments whites, otherwise increments blacks.

public void nextPlayer()

- This method changes flag value.
- This method has not any parameter.
- This method is not return any value. This method is void.
- This method updates flag value. If flag is true, update to false, otherwise update to true.

public Piece getPieceAt(String from)

- This method takes Piece which's location is String from.
- String from is piece's location.
- This method returns Piece if squares has Piece which's location is from, otherwise null.
- This method converts String from to upper case String fromUpper. fromUpper is added one character space. fromUpper is splited index 0 to index 1 and assign to String col. fromUpper is splitted index 1 to index 2. This value is converted to integer and assign to int row. This method has nested for loops. Each for loop runs 8 times because of chess board's square. Nested for loops takes one Square from squares and controls if this Square's column equals to col and this Square's row equals to row. If that is returns this Square's Piece, otherwise null.

public Square getSquareAt(String to)

- This method takes Square which's location is String to.
- String to is piece's location.
- This method returns Square if squares has Square which's location is to, otherwise null.
- This method converts String to to upper case String toUpper. toUpper is added one character space. toUpper is splited index 0 to index 1 and assign to String col. toUpper is splitted index 1 to index 2. This value is converted to integer and assign to int row. This method has nested for loops. Each for loop runs 8 times because of chess board's square. Nested for loops takes one Square from squares and controls if this Square's column equals to col and this Square's row equals to row. If that is returns this Square, otherwise null.

public Square[ ] getSquaresBetween(Square location, Square targetLocation, String rank, int color)

- This method takes Squares between Square location and Square targetLocation.
- location: the beginning square ; targetLocation: final square; rank: which piece is; color: Piece's color.
- This method returns Square array which has Squares between Square location and Square targetLocation.
- This method controls if rank is "P" or "R" or "B". If rank is "P", this method creates Square array squares and takes Square in front of location and add this Square to squares. If rank is "R", this method creates Square array squares and checks location's column same as targetLocation's column, if that is takes Squares in front of and back location and add this Squares to squares. If location's row same as targetLocation's row, takes Squares right and left of location and add this Squares to squares. If rank is "B", this method creates Square array squares and checks location and targetLocation are

same diagonal, if that is takes Squares diagonal location to targetLocation and add this Squares to squares. This method returns squares. If rank is not “P” or “R” or “B”, this method returns null.

public String stringColumn(int col)

- This method converts int col to String.
- int col is Square’s column.
- This method returns String column.
- This method controls if col is 1, returns “A”, if col is 2, returns “B”, if col is 3, returns “C”, if col is 4, returns “D”, if col is 5, returns “E”, if col is 6, returns “F”, if col is 7, returns “G”, if col is 8, returns “H”, otherwise returns null.

public void removeList(Square target)

- This method creates new Square with target’ column, target’s row and Piece which is null, from Square target.
- Square target has column and row which are new Square’s location.
- This method is not return any value. This method is void.
- This method takes target’s column to String col and target’s row to String row. This method has nested for loops. Each for loop runs 8 times because of chess board’s square. Nested for loops takes one Square from squares and controls if this Square equals to target. If that is, puts new Square with col, row and null to squares.

public void putNewQueen(int color, Square target)

- This method creates new Square with target’ column, target’s row and Piece which has int color and String “QUEEN”, from Square target.
- Square target has column and row which are new Square’s location. İnt color is Piece’s color.
- This method is not return any value. This method is void.
- This method takes target’s column to String col and target’s row to String row. This method has nested for loops. Each for loop runs 8 times because of chess board’s square. Nested for loops takes one Square from squares and controls if this Square equals to target. If that is, puts new Square with col, row and new Queen with color and String col and row, to squares.

public void setPiece(Piece piec, Square target)

- This method creates new Square with target’ column, target’s row and Piece which is Piece pie.
- Square target has column and row which are new Square’s location. Piece piec is new Square’s Piece
- This method is not return any value. This method is void.
- This method takes target’s column to String col and target’s row to String row. It takes Piece which has location is piec’s location to Piece pie. It updates pie’s location with col and row. This method has nested for loops. Each for loop runs 8 times because of chess board’s square. Nested for loops takes one Square from squares and controls if this Square’s column equals to col and Square’s row equals to row, if pie instance of Pawn, puts new Square with col, row and new Pawn with pie’s color and pie’s location. If pie instance of Bishop, puts new Square with col, row and new Bishop with pie’s color and pie’s location. If pie instance of King, puts new Square with col, row and new King with pie’s color and pie’s location. If pie instance of Knight, puts new Square with col, row and new Knight with pie’s color and pie’s location. If pie instance of Queen, puts new Square with col, row and new Queen with pie’s color and pie’s location. If pie instance of Rook, puts new Square with col, row and new Rook with pie’s color and pie’s location.

-Square class

public boolean isEmpty()

- This method checks if Square's Piece is null or not.
- This method has not any parameter.
- This method returns true or false.
- This method returns if Square's Piece is null, true, otherwise false.

public boolean isAtSameColumn(Square s)

- This method checks Square and Square s same column or not.
- Square s is the Square to compare.
- This method returns true or false.
- This method returns if Square's column equals to s' column, true, otherwise false.

public boolean isAtSameRow(Square targetLocation)

- This method checks Square and Square targetLocation same row or not.
- Square targetLocation is the Square to compare.
- This method returns true or false.
- This method returns if Square's row equals to targetLocation' row, true, otherwise false.

public int getRowDistance(Square location)

- This method takes differences between Square's row and Square location's row.
- Square location is the square which has row, to get the difference.
- This method returns row distance between Square and location.
- This method returns differences between Square's row and Square location's row.

public boolean isNeighborColumn(Square targetLocation)

- This method checks Square and Square targetLocation are neighbor column or not.
- Square targetLocation is the Square to compare.
- This method returns true or false.
- This method creates new ArrayList with String neighborColumn. This method checks if Square's column equals to "A", add "B" to neighborColumn or if Square's column equals to "B", add "A" and "C" to neighborColumn or if Square's column equals to "C", add "B" and "D" to neighborColumn or if Square's column equals to "D", add "C" and "E" to neighborColumn or if Square's column equals to "E", add "D" and "F" to neighborColumn or if Square's column equals to "F", add "E" and "G" to neighborColumn or if Square's column equals to "G", add "F" and "H" to neighborColumn otherwise add "G" to neighborColumn. This method returns if neighborColumn contains targetLocation's column, true, otherwise false.

public boolean isAtLastRow(int color)

- ❖ This method checks Square's row is last row or not.
- ❖ int color controls which's color last row white or black.
- ❖ This method returns true or false.
- ❖ This method checks if color is White and Square's row is 8, returns true, or color is black and Square's row is 1, returns true, otherwise returns false.

public int intColumn()

- ❖ This method converts Square's column to integer.
- ❖ This method has not any parameter.
- ❖ This method returns integer.
- ❖ This method controls if Square's column is "A", returns "1", if Square's column is "B", returns 2, if Square's column is "C", returns 3, if Square's column is "D", returns 4, if Square's column is "E", returns 5, if Square's column is "F", returns 6, if Square's column is "G", returns 7, if Square's column is "H", returns 8, otherwise returns 0.

D) Defining Piece Hierarchy

Explain how Main class benefits from polymorphism:

Main class has Piece. Piece is superclass of King, Queen, Rook, Bishop, Knight and Pawn. Main class knows Piece is which one so we do not have to write specific Piece's canMove and move metods. This is Main class benefits from polymorphism.

Explain, which methods and classes can be defined abstract in Piece hierarchy. Is there a code reuse in your implementation?

move and canMove metods defined abstract in Piece hierarchy. No, there is not any code reuse in my implementation.

E) Implementing methods in Piece Hierarchy

-**Pawn class**

public boolean canMove(String to, ChessBoard board)

- ❖ This method controls Pawn moves right or not.
- ❖ String to is Pawn will move this location. ChessBoard board holds ChessBoard's data and metods for canMove method.
- ❖ This method returns true or false.
- ❖ This method creates boolean validMove and puts false to validMove. It creates Square location from Pawn's location and Square targetLocation from to. This method calculates row distance between targetLocation and location and puts int rowDistance. It checks if location and targetLocation are at same column and Pawn's color is white and rowDistance greater than 0 and rowDistance less or equals to 2 and rowDistance equals to 2 and initialLocation is true, it takes Squares between location and targetLocation and puts to Square array between. If targetLocation is empty and the square in front of the location is empty, puts true to validMove. This method updates initialLocation to false. If rowDistance not equal to 2, if targetLocation is empty, puts true to validMove. This method updates initialLocation to false. It returns validMove. If Pawn's color is black and rowDistance less than 0 and rowDistance greater or equals to -2 and rowDistance equals to -2 and initialLocation is true, this method does same things for black Pawn. If location is neighbor column with targetLocation that

means Pawn is attacking. It puts color from Piece which from to, to int color. This method checks if Pawn's color is white and rowDistance equals to 1, if targetLocation is not empty and color is black, it puts true to attacking and updates validMove with attacking. If Pawn's color is black and rowDistance equals to -1, this method does same things for black Pawn. If moves is false, this method returns validMove which is false.

public void move(String to, ChessBoard board)

- Pawn is move to String to by this method.
- String to is Pawn will move this location. ChessBoard board holds ChessBoard's data and metods for move method.
- This method is not return any value. This method is void.
- It creates Square location from Pawn's location and Square targetLocation from to. If targetLocation is at last row for Pawn's color this means Pawn promotes to Queen. This method puts new Queen with Pawn's color to targetLocation. If Pawn is attacking, it removes targetLocation's piece and puts Pawn to targetLocation. If Pawn is not promote to Queen and is not attacking that means Pawn moves normally. This method puts Pawn to targetLocation. It removes location's Piece and updates location with targetLocation. This method turns to next player.

-Rook class

public boolean canMove(String destination, ChessBoard board)

- This method controls Rook moves right or not.
- String destination is Rook will move this location. ChessBoard board holds ChessBoard's data and metods for canMove method.
- This method returns true or false.
- This method creates boolean validMove and puts false to validMove. It creates Square location from Rook's location and Square targetLocation from destination. If location and targetLocation are at same column or same row, it takes Squares between location and targetLocation and puts to Square array between. This method checks all Squares in between are empty or not. If all Squares in between are empty, it puts true to validMove. If targetLocation is not empty, this means Rook is attacking. It puts color from Piece which from destination, to int color. This method checks if Rook's color is not equal to color, updates attacking with reverse validMove. Because if Rook is attacking, validMove must be false and attacking must be true. If Rook's color equals to color, it returns validMove. This method returns if attacking true, attacking, otherwise validMove.

public void move(String destination, ChessBoard board)

- Rook is move to String destination by this method.
- String destination is Rook will move this location. ChessBoard board holds ChessBoard's data and metods for move method.
- This method is not return any value. This method is void.
- It creates Square location from Rook's location and Square targetLocation from destination. If Rook is attacking, it removes targetLocation's piece and puts Rook to targetLocation. If Rook is not attacking that means Rook moves normally. This method puts Rook to targetLocation. It removes location's Piece and updates location with targetLocation. This method turns to next player.

-Knight class

public boolean canMove(String destination, ChessBoard board)

- This method controls Knight moves right or not.
- String destination is Knight will move this location. ChessBoard board holds ChessBoard's data and metods for canMove method.
- This method returns true or false.
- This method creates boolean validMove and puts false to validMove. It creates Square location from Knight's location and Square targetLocation from destination. This method takes location's integer column to int intLcol, location's row to int lRow, targetLocation's integer column to int intTcol and targetLocation's row to int tRow. If Knight moves correctly and if targetLocation is not empty, this means Knight is attacking. It puts color from Piece which from destination, to int color. This method checks if Knight's color is not equal to color, updates attacking with true. If Knight's color equals to color, it returns false. If targetLocation is not empty, This method returns true. If moves is false, this method returns false.

public void move(String destination, ChessBoard board)

- Knight is move to String destination by this method.
- String destination is Knight will move this location. ChessBoard board holds ChessBoard's data and metods for move method.
- This method is not return any value. This method is void.
- It creates Square location from Knight's location and Square targetLocation from destination. If Knight is attacking, it removes targetLocation's piece and puts Knight to targetLocation. If Rook is not attacking that means Knight moves normally. This method puts Rook to targetLocation. It removes location's Piece and updates location with targetLocation. This method turns to next player.

-Bishop class

public boolean canMove(String destination, ChessBoard board)

- This method controls Bishop moves right or not.
- String destination is Bishop will move this location. ChessBoard board holds ChessBoard's data and metods for canMove method.
- This method returns true or false.
- This method creates boolean validMove and puts false to validMove. It creates Square location from Bishop's location and Square targetLocation from destination. This method takes location's integer column to int intLcol, location's row to int lRow, targetLocation's integer column to int intTcol and targetLocation's row to int tRow. It calculates absolute diffrences between intLcol and intTcol and puts to int colDif. It calculates absolute diffrences between lRow and tRow and puts to int rowDif. If colDif equals to rowDif, it takes Squares between location and targetLocation and puts to Square array between. This method checks all Squares in between are empty or not. If all Squares in between are empty, it puts true to validMove. If targetLocation is not empty, this means Bishop is attacking. It puts color from Piece which from destination, to int color. This method checks if Bishop's color is not equal to color, updates attacking with reverse validMove. Because if Bishop is attacking, validMove must be false and attacking must be true. If Bishop's color equals to color, it returns validMove. This method returns if attacking true, attacking, otherwise validMove.

public void move(String destination, ChessBoard board)

- Bishop is move to String destination by this method.
- String destination is Bishop will move this location. ChessBoard board holds ChessBoard's data and metods for move method.
- This method is not return any value. This method is void.
- It creates Square location from Bishop's location and Square targetLocation from destination. If Bishop is attacking, it removes targetLocation's piece and puts Bishop to targetLocation. If Bishop is not attacking that means Bishop moves normally. This method puts Bishop to targetLocation. It removes location's Piece and updates location with targetLocation. This method turns to next player.

-Queen class

public boolean canMove(String destination, ChessBoard board)

- This method controls Queen moves right or not.
- String destination is Queen will move this location. ChessBoard board holds ChessBoard's data and metods for canMove method.
- This method returns true or false.
- This method creates boolean validMove and puts false to validMove. It creates Square location from Queen's location and Square targetLocation from destination. This method takes location's integer column to int intLcol, location's row to int lRow, targetLocation's integer column to int intTcol and targetLocation's row to int tRow. It calculates absolute diffrences between intLcol and intTcol and puts to int colDif. It calculates absolute diffrences between lRow and tRow and puts to int rowDif. If location and targetLocation are at same column or same row, it takes Squares between location and targetLocation and puts to Square array between. This method checks all Squares in between are empty or not. If all Squares in between are empty, it puts true to validMove. If targetLocation is not empty, this means Queen is attacking. It puts color from Piece which from destination, to int color. This method checks if Queen's color is not equal to color, updates attacking with reverse validMove. Because if Queen is attacking, validMove must be false and attacking must be true. If Queen's color equals to color, it returns validMove. This method returns if attacking true, attacking, otherwise validMove. If colDif equals to rowDif, it takes Squares between location and targetLocation and puts to Square array between. This method checks all Squares in between are empty or not. If all Squares in between are empty, it puts true to validMove. If targetLocation is not empty, this means Queen is attacking. It puts color from Piece which from destination, to int color. This method checks if Queen's color is not equal to color, updates attacking with reverse validMove. Because if Queen is attacking, validMove must be false and attacking must be true. If Queen's color equals to color, it returns validMove. This method returns if attacking true, attacking, otherwise validMove. If moves is false, this method returns validMove which is false.

public void move(String destination, ChessBoard board)

- Queen is move to String destination by this method.
- String destination is Queen will move this location. ChessBoard board holds ChessBoard's data and metods for move method.
- This method is not return any value. This method is void.
- It creates Square location from Queen's location and Square targetLocation from destination. If Queen is attacking, it removes targetLocation's piece and puts Queen to targetLocation. If Queen is not attacking that means Queen moves normally. This method puts Queen to targetLocation. It removes location's Piece and updates location with targetLocation. This method turns to next player.

-King class

public boolean canMove(String destination, ChessBoard board)

- This method controls King moves right or not.
- String destination is King will move this location. ChessBoard board holds ChessBoard's data and metods for canMove method.
- This method returns true or false.
- This method creates boolean validMove and puts false to validMove. It creates Square location from King's location and Square targetLocation from destination. This method takes location's integer column to int intLcol, location's row to int lRow, targetLocation's integer column to int intTcol and targetLocation's row to int tRow. It calculates absolute diffrences between intLcol and intTcol and puts to int colDif. It calculates absolute diffrences between lRow and tRow and puts to int rowDif. If location and targetLocation are at same column or same row and King moves 1 square, if targetLocation is empty, this method puts true to validMove, otherwise false to validMove. If targetLocation is not empty, this means King is attacking. It puts color from Piece which from destination, to int color. This method checks if King's color is not equal to color, updates attacking with reverse validMove. Because if King is attacking, validMove must be false and attacking must be true. If King's color equals to color, it returns validMove. This method returns if attacking true, attacking, otherwise validMove. If colDif equals to rowDif and King moves 1 square, it does same things. If moves is false, this method returns validMove which is false.

public void move(String destination, ChessBoard board)

- King is move to String destination by this method.
- String destination is King will move this location. ChessBoard board holds ChessBoard's data and metods for move method.
- This method is not return any value. This method is void.
- It creates Square location from King's location and Square targetLocation from destination. If King is attacking, it removes targetLocation's piece and puts King to targetLocation. If King is not attacking that means King moves normally. This method puts King to targetLocation. It removes location's Piece and updates location with targetLocation. This method turns to next player.