TCP2201
Project Trimester 1, 2019/2020
by Perilanda Chess Offline

Team Leader:

Low Chi Han, 016-2836838, 1161104278@student.mmu.edu.my

Team Members:

Justin Tey, 012-9831832, 1161104265@student.mmu.edu.my
Chan Hao Kang, 012-2089296, 1161104114@student.mmu.edu.my

Compile Instruction **2**

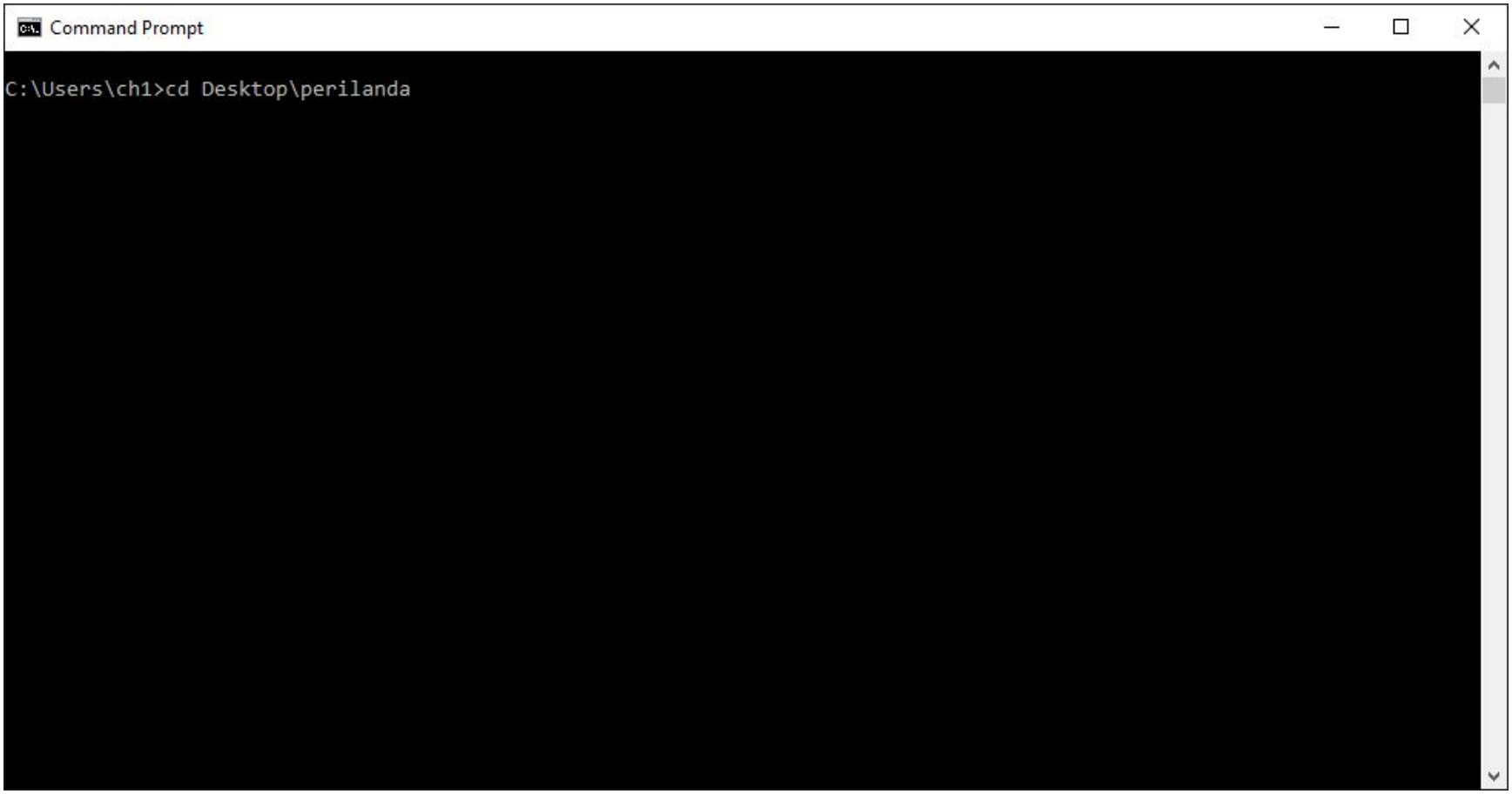
UML Class Diagram **6**

Use Case Diagram **9**

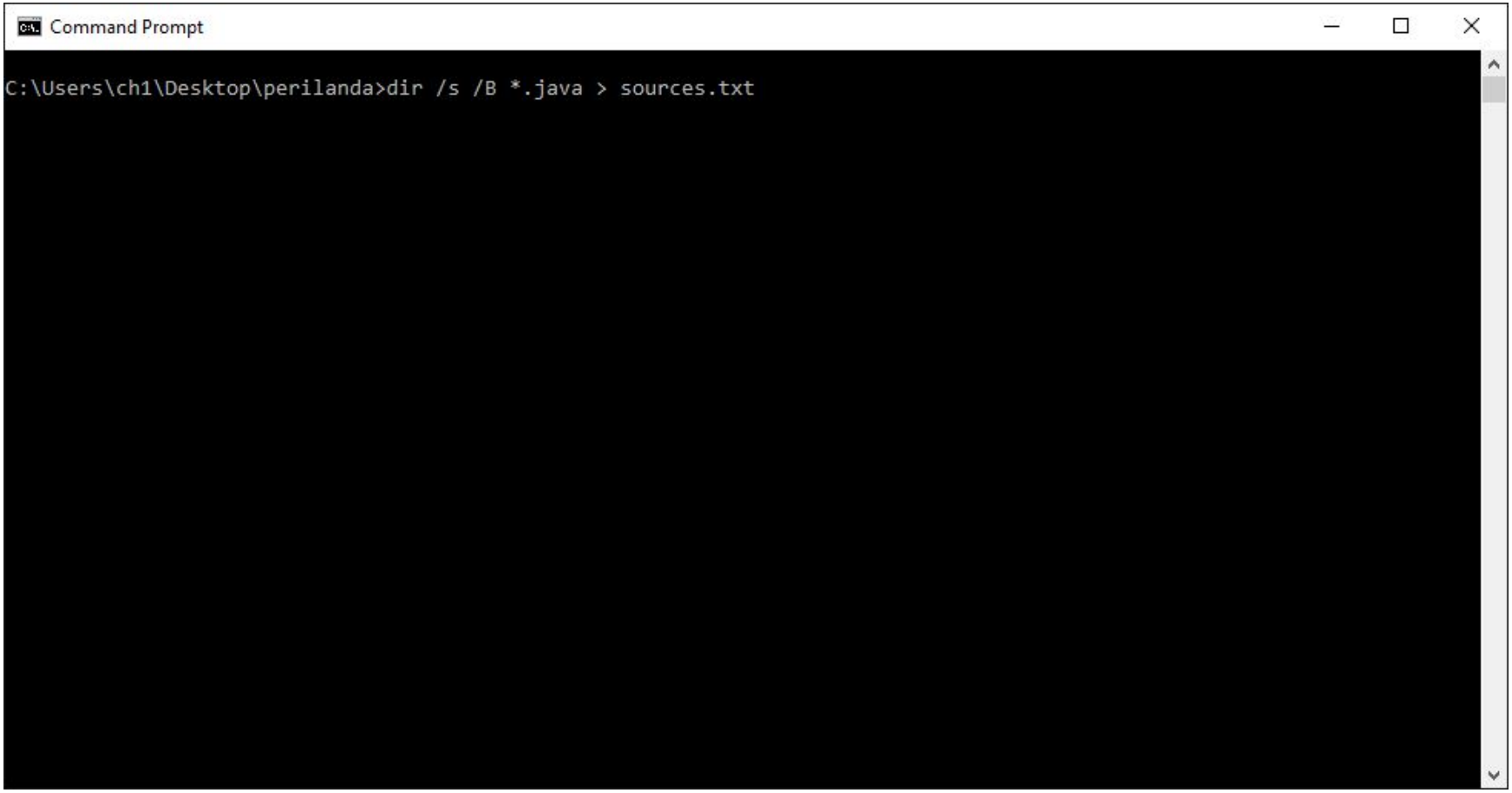
Sequence Diagram **10**

User Guidelines **15**

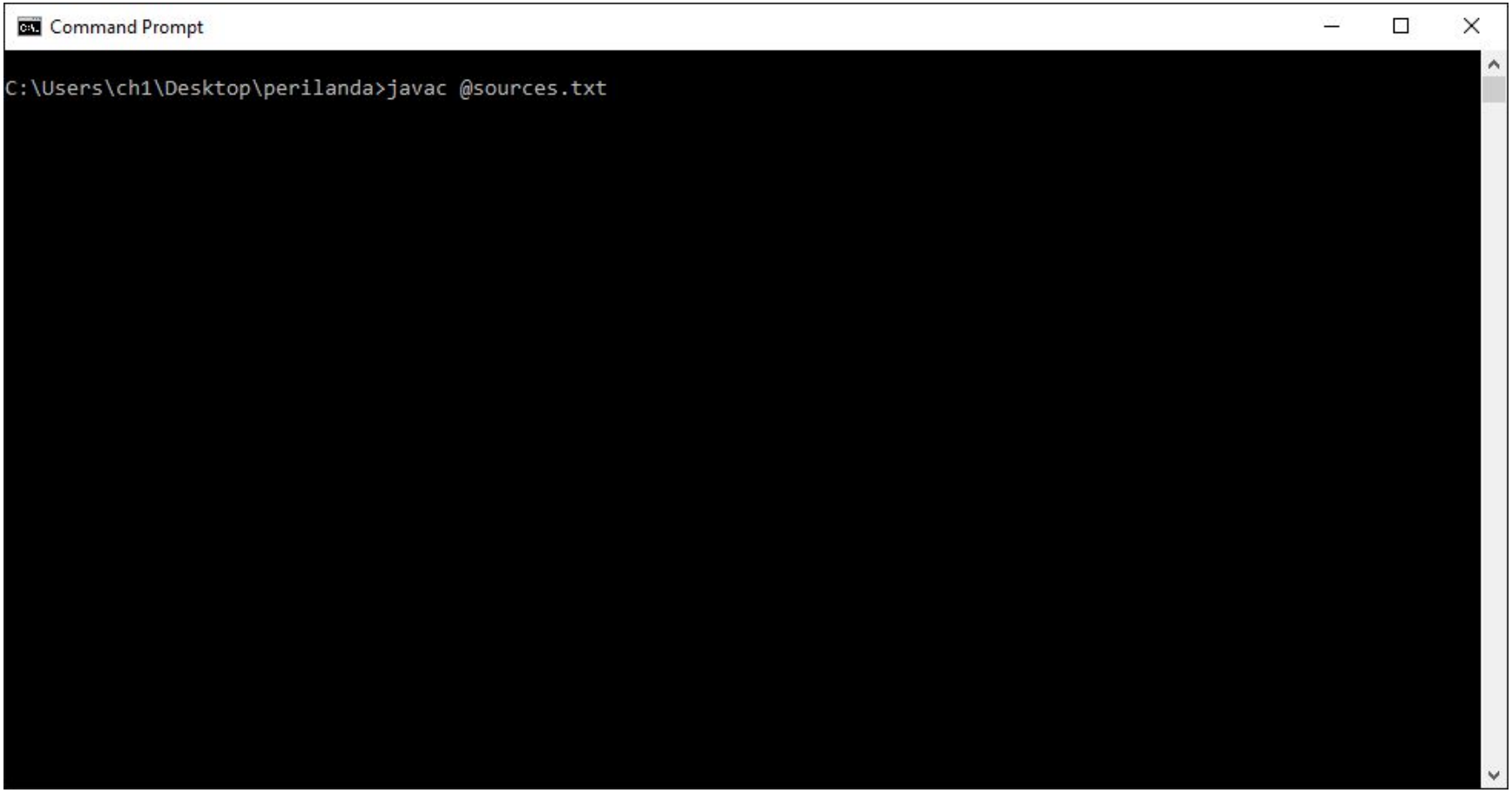
Compile Instruction



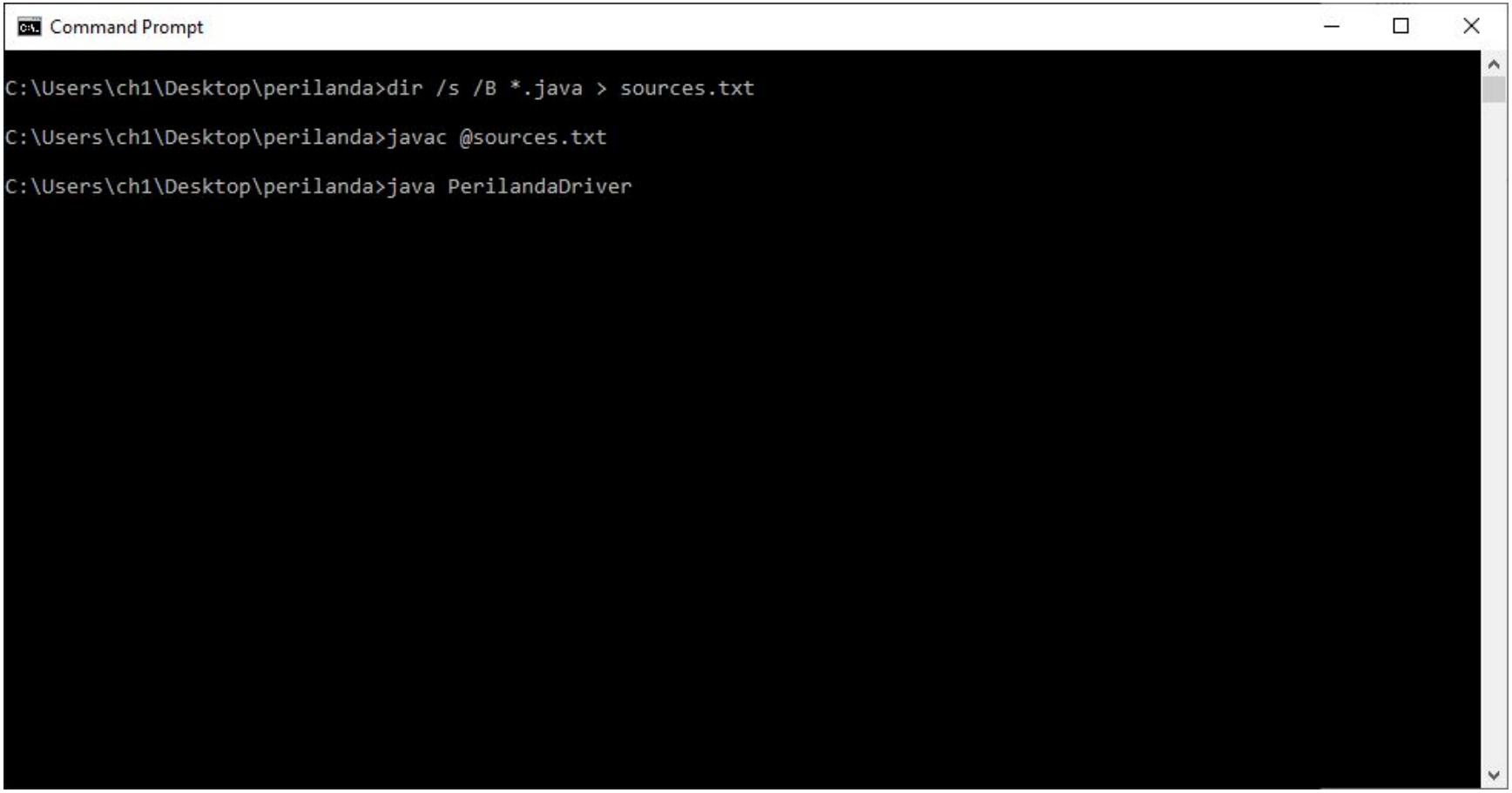
To compile all the java source code of the program, you first need to navigate to the folder containing the java source code from command line. In this case, the java source code are located in the perilanda folder in Desktop.



After you get into the directory containing the java source code, enter the command “dir /s /B *.java > sources.txt”. This command will store the directories of all the java source code in the current directory inside sources.txt.

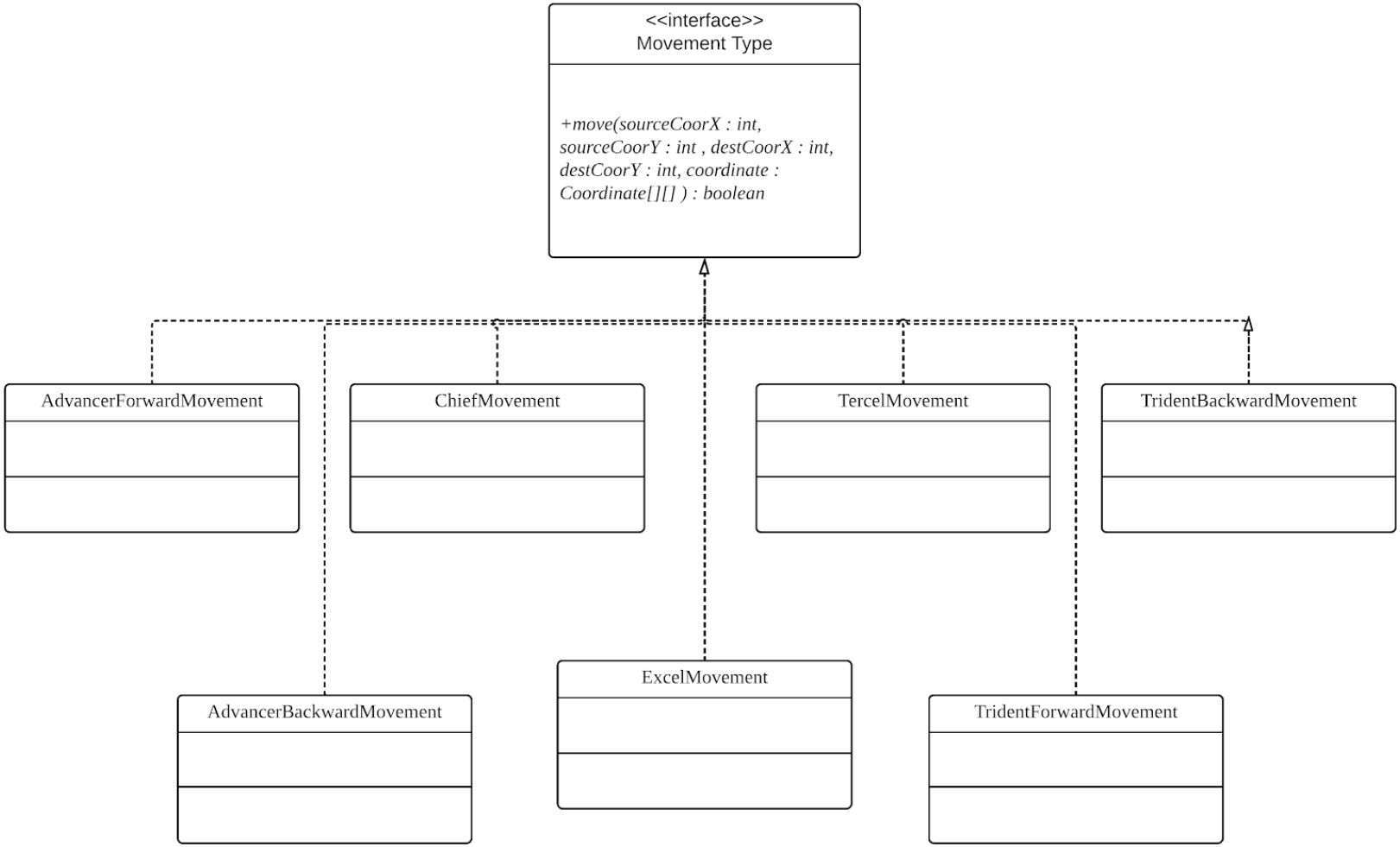


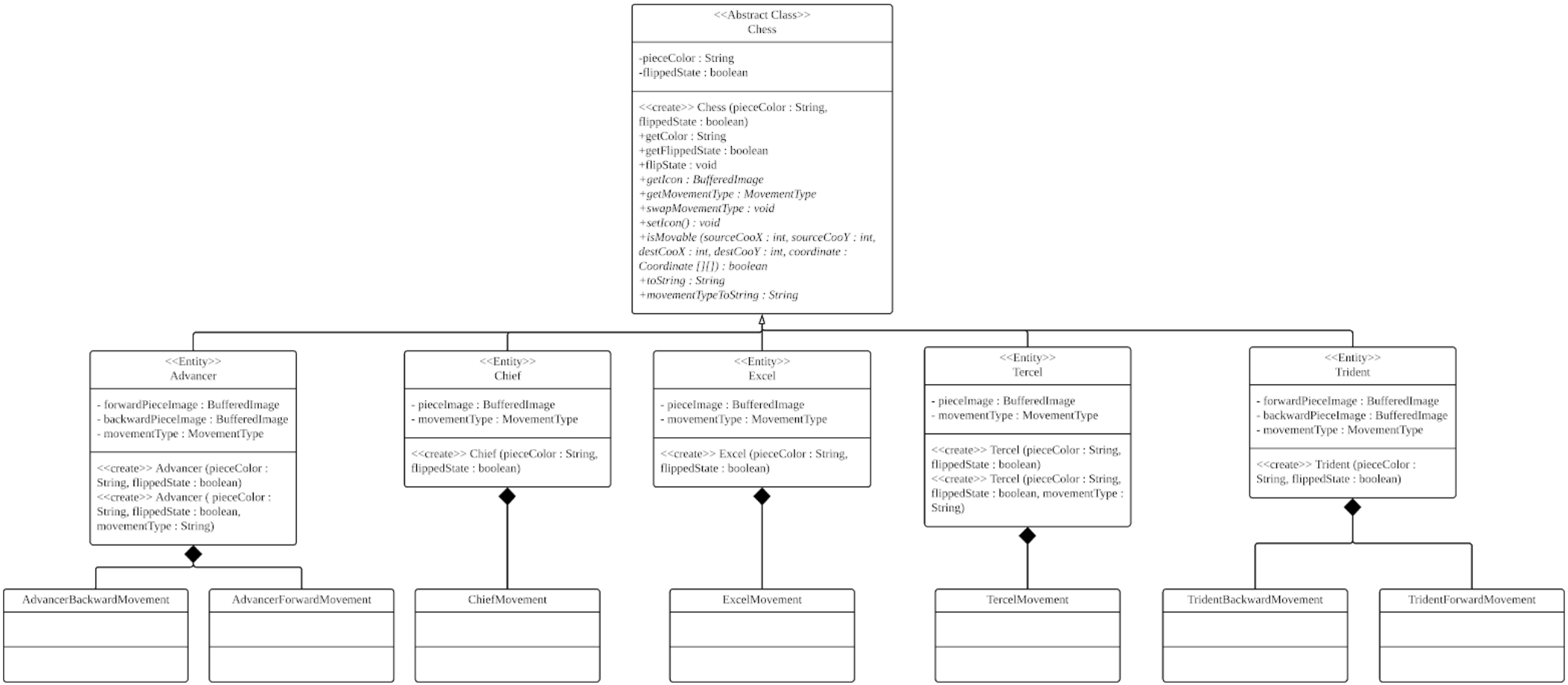
To compile all the java source code at once, enter the command “javac @sources.txt “.

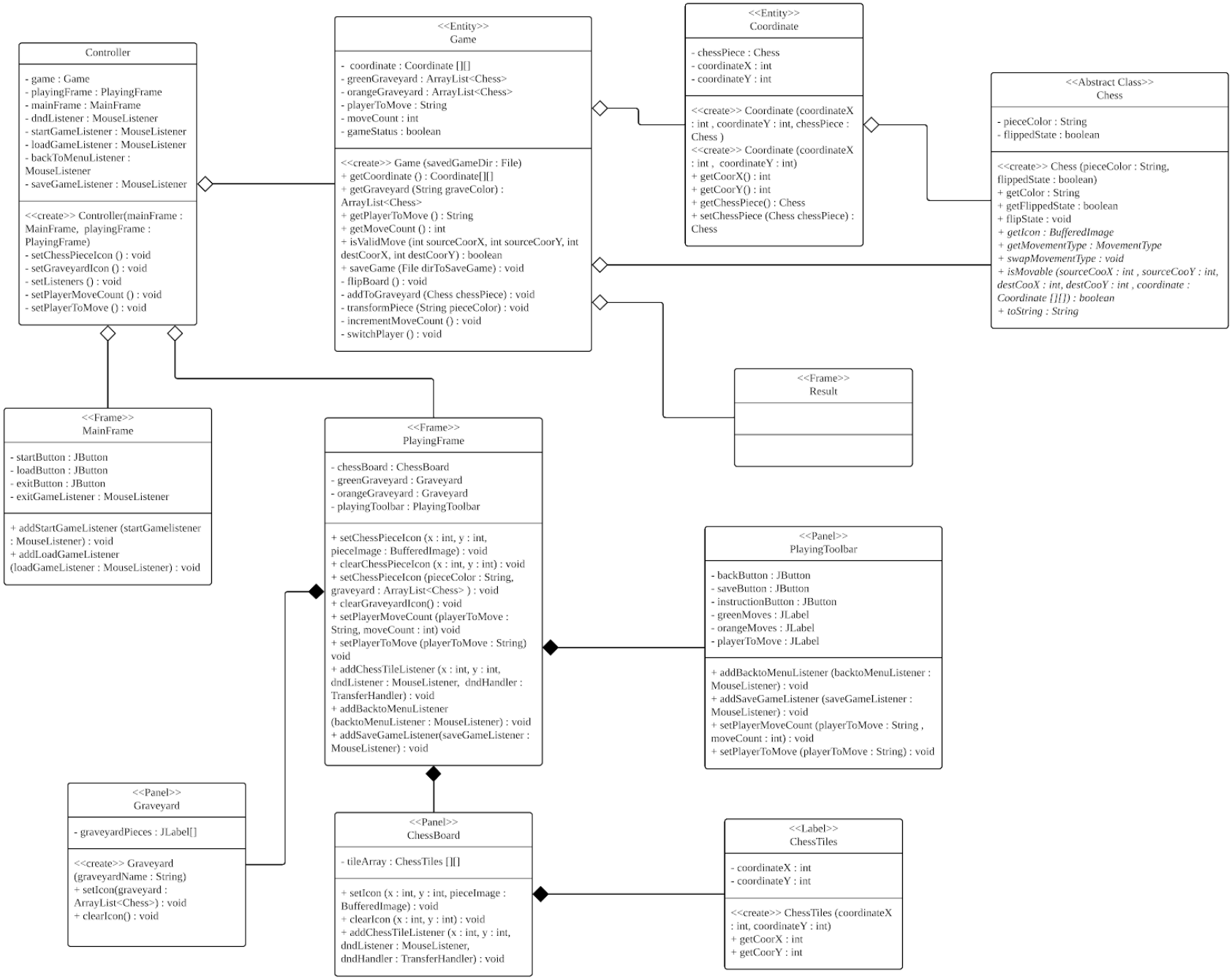


To run the program, enter the command “java PerilandaDriver”. PerilandaDriver.java serves as the “Driver class” of the program.

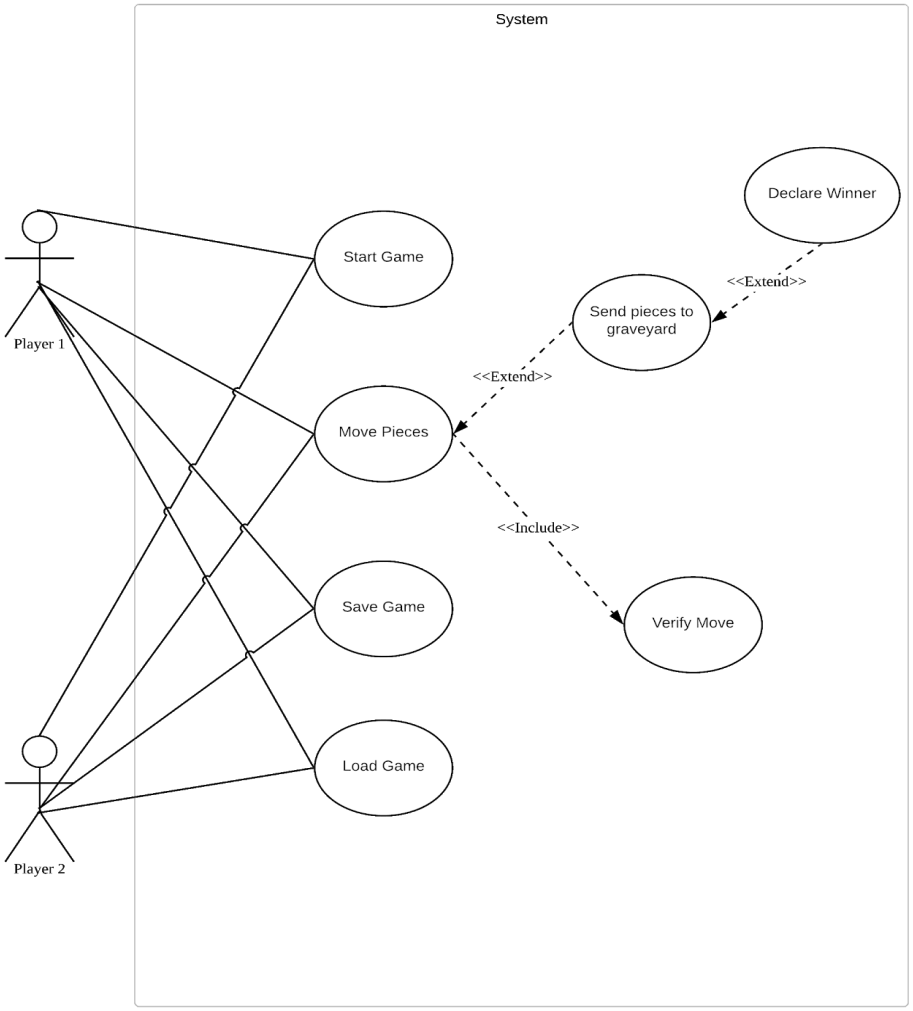
UML Class Diagram



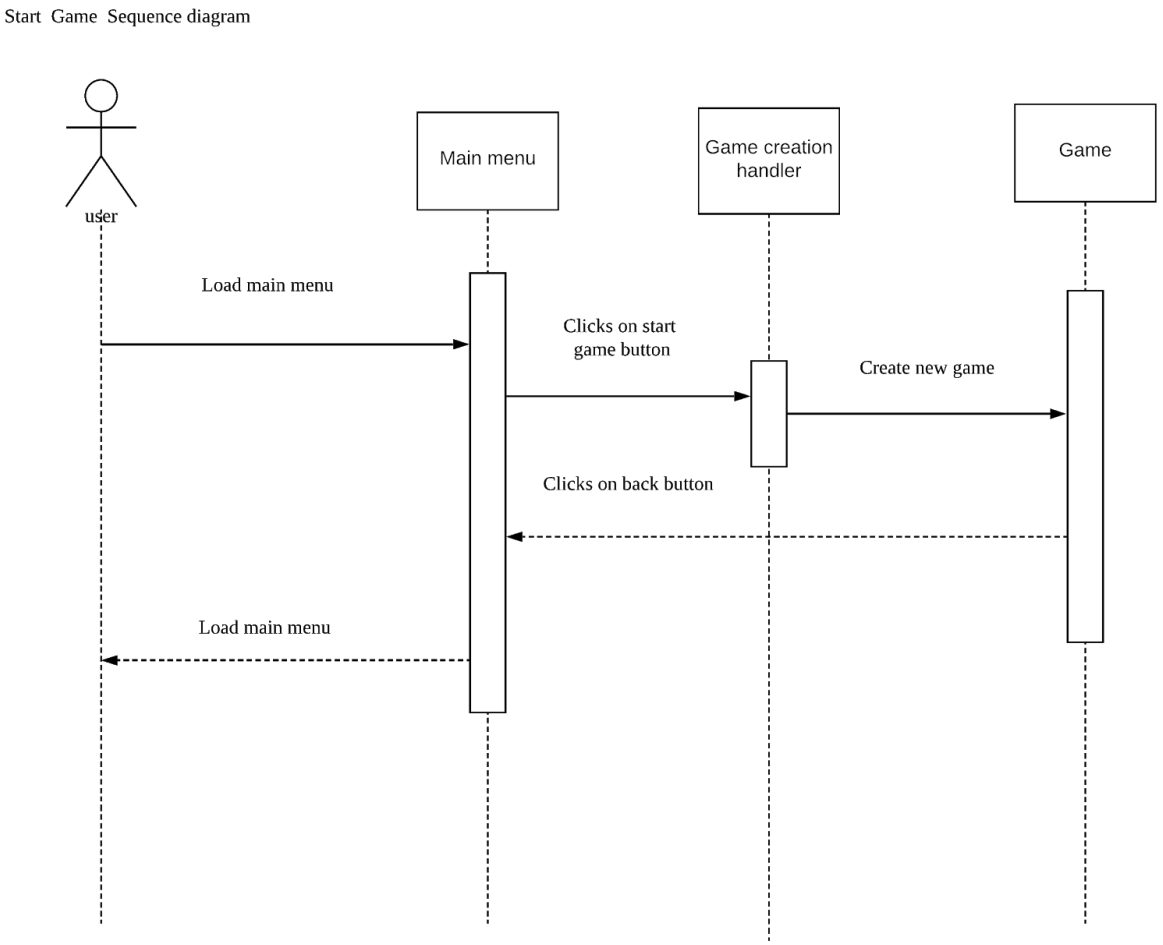

<<Abstract Class>>
Chess
-pieceColor : String
-flippedState : boolean
<<create>> Chess (pieceColor : String, flippedState : boolean)
+getColor : String
+getFlippedState : boolean
+flipState : void
+getIcon : BufferedImage
+getMovementType : MovementType
+swapMovementType : void
+setIcon() : void
+isMovable (sourceCooX : int, sourceCooY : int, destCooX : int, destCooY : int, coordinate : Coordinate [][]) : boolean
+toString : String
+movementTypeToString : String
<<Entity>>
Advancer
- forwardPieceImage : BufferedImage
- backwardPieceImage : BufferedImage
- movementType : MovementType
<<create>> Advancer (pieceColor : String, flippedState : boolean)
<<create>> Advancer (pieceColor : String, flippedState : boolean, movementType : String)
<<Entity>>
Chief
- pieceImage : BufferedImage
- movementType : MovementType
<<create>> Chief (pieceColor : String, flippedState : boolean)
<<Entity>>
Excel
- pieceImage : BufferedImage
- movementType : MovementType
<<create>> Excel (pieceColor : String, flippedState : boolean)
<<Entity>>
Tercel
- pieceImage : BufferedImage
- movementType : MovementType
<<create>> Tercel (pieceColor : String, flippedState : boolean)
<<create>> Tercel (pieceColor : String, flippedState : boolean, movementType : String)
<<Entity>>
Trident
- forwardPieceImage : BufferedImage
- backwardPieceImage : BufferedImage
- movementType : MovementType
<<create>> Trident (pieceColor : String, flippedState : boolean)
AdvancerBackwardMovement
AdvancerForwardMovement
ChiefMovement
ExcelMovement
TercelMovement
TridentBackwardMovement
TridentForwardMovement


Controller
- game : Game
- playingFrame : PlayingFrame
- mainFrame : MainFrame
- dndListener : MouseListener
- startGameListener : MouseListener
- loadGameListener : MouseListener
- backToMenuListener : MouseListener
- saveGameListener : MouseListener
<<create>> Controller(mainFrame : MainFrame, playingFrame : PlayingFrame)
- setChessPieceIcon () : void
- setGraveyardIcon () : void
- setListeners () : void
- setPlayerMoveCount () : void
- setPlayerToMove () : void
<<Entity>>
Game
- coordinate : Coordinate [][]
- greenGraveyard : ArrayList<Chess>
- orangeGraveyard : ArrayList<Chess>
- playerToMove : String
- moveCount : int
- gameStatus : boolean
<<create>> Game (savedGameDir : File)
+ getCoordinate () : Coordinate[][]
+ getGraveyard (String graveColor) : ArrayList<Chess>
+ getPlayerToMove () : String
+ getMoveCount () : int
+ isValidMove (int sourceCoorX, int sourceCoorY, int destCoorX, int destCoorY) : boolean
+ saveGame (File dirToSaveGame) : void
- flipBoard () : void
- addToGraveyard (Chess chessPiece) : void
- transformPiece (String pieceColor) : void
- incrementMoveCount () : void
- switchPlayer () : void
<<Entity>>
Coordinate
- chessPiece : Chess
- coordinateX : int
- coordinateY : int
<<create>> Coordinate (coordinateX : int , coordinateY : int, chessPiece : Chess)
<<create>> Coordinate (coordinateX : int , coordinateY : int)
+ getCoorX() : int
+ getCoorY() : int
+ getChessPiece() : Chess
+ setChessPiece (Chess chessPiece) : Chess
<<Abstract Class>>
Chess
- pieceColor : String
- flippedState : boolean
<<create>> Chess (pieceColor : String, flippedState : boolean)
+ getColor : String
+ getFlippedState : boolean
+ flipState : void
+ getIcon : BufferedImage
+ getMovementType : MovementType
+ swapMovementType : void
+ isMovable (sourceCooX : int , sourceCooY : int, destCooX : int, destCooY : int , coordinate : Coordinate [][]) : boolean
+ toString : String
<<Frame>>
Result
<<Frame>>
MainFrame
- startButton : JButton
- loadButton : JButton
- exitButton : JButton
- exitGameListener : MouseListener
+ addStartGameListener (startGamelistener : MouseListener) : void
+ addLoadGameListener (loadGameListener : MouseListener) : void
<<Frame>>
PlayingFrame
- chessBoard : ChessBoard
- greenGraveyard : Graveyard
- orangeGraveyard : Graveyard
- playingToolbar : PlayingToolbar
+ setChessPieceIcon (x : int, y : int, pieceImage : BufferedImage) : void
+ clearChessPieceIcon (x : int, y : int) : void
+ setChessPieceIcon (pieceColor : String, graveyard : ArrayList<Chess>) : void
+ clearGraveyardIcon() : void
+ setPlayerMoveCount (playerToMove : String, moveCount : int) void
+ setPlayerToMove (playerToMove : String) void
+ addChessTileListener (x : int, y : int, dndListener : MouseListener, dndHandler : TransferHandler) : void
+ addBacktoMenuListener (backtoMenuListener : MouseListener) : void
+ addSaveGameListener(saveGameListener : MouseListener) : void
<<Panel>>
PlayingToolbar
- backButton : JButton
- saveButton : JButton
- instructionButton : JButton
- greenMoves : JLabel
- orangeMoves : JLabel
- playerToMove : JLabel
+ addBacktoMenuListener (backtoMenuListener : MouseListener) : void
+ addSaveGameListener (saveGameListener : MouseListener) : void
+ setPlayerMoveCount (playerToMove : String , moveCount : int) : void
+ setPlayerToMove (playerToMove : String) : void
<<Panel>>
Graveyard
- graveyardPieces : JLabel[]
<<create>> Graveyard (graveyardName : String)
+ setIcon(graveyard : ArrayList<Chess>) : void
+ clearIcon() : void
<<Panel>>
ChessBoard
- tileArray : ChessTiles [][]
+ setIcon (x : int, y : int, pieceImage : BufferedImage) : void
+ clearIcon (x : int, y : int) : void
+ addChessTileListener (x : int, y : int, dndListener : MouseListener, dndHandler : TransferHandler) : void
<<Label>>
ChessTiles
- coordinateX : int
- coordinateY : int
<<create>> ChessTiles (coordinateX : int, coordinateY : int)
+ getCoorX : int
+ getCoorY : int

Use Case Diagram

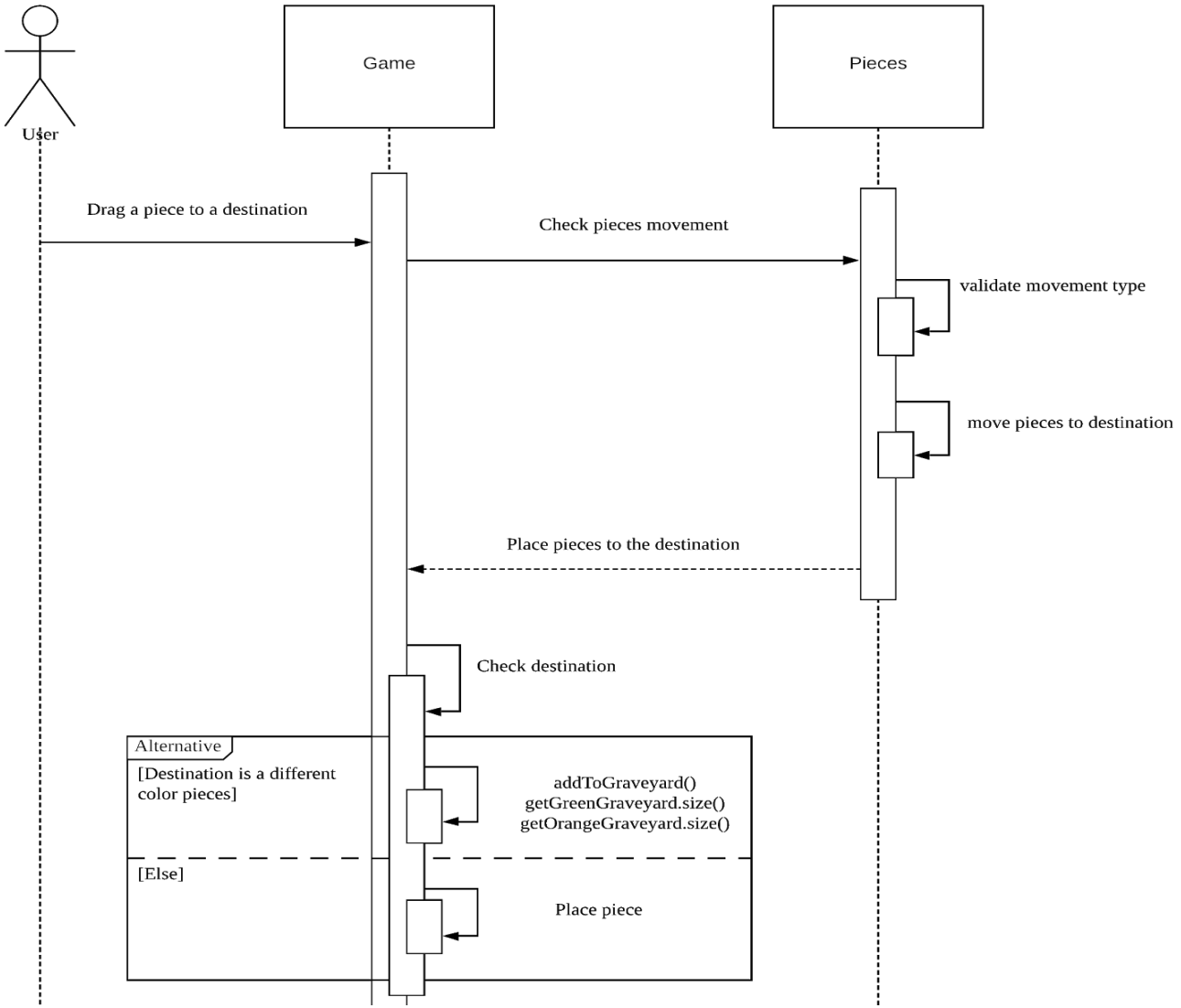


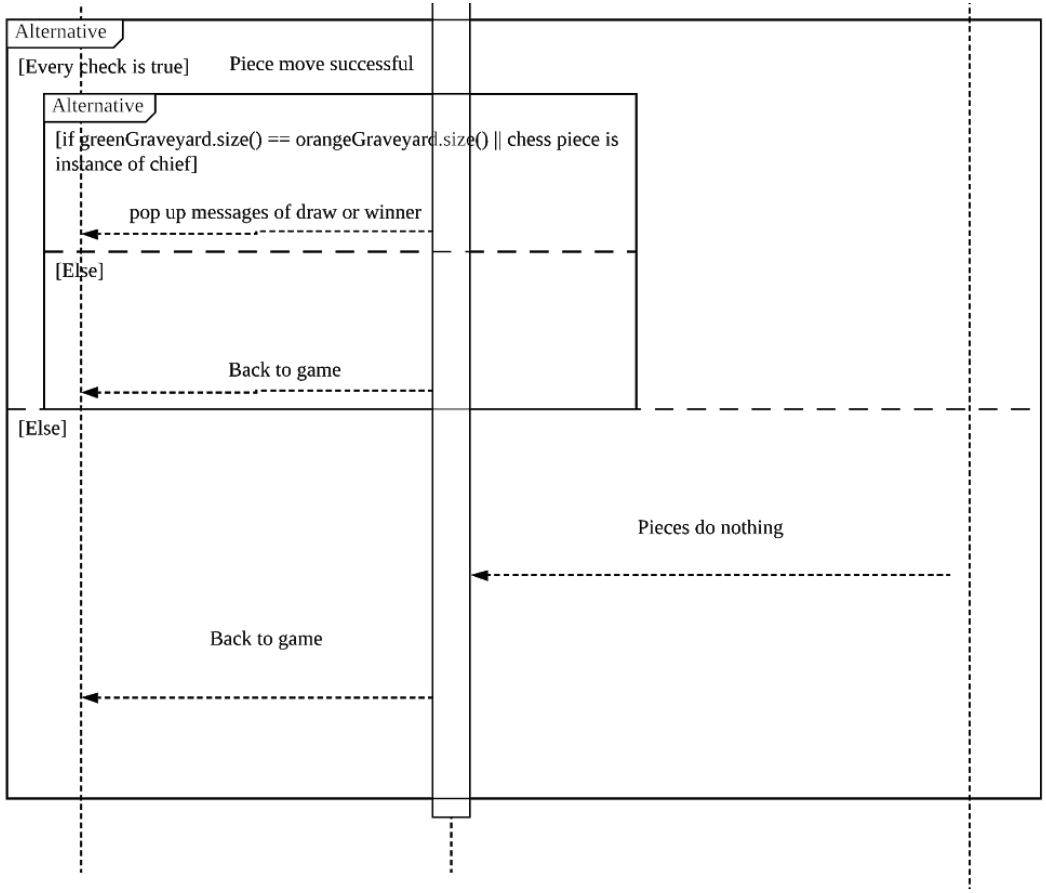
Sequence Diagram



Start Game Sequence Diagram

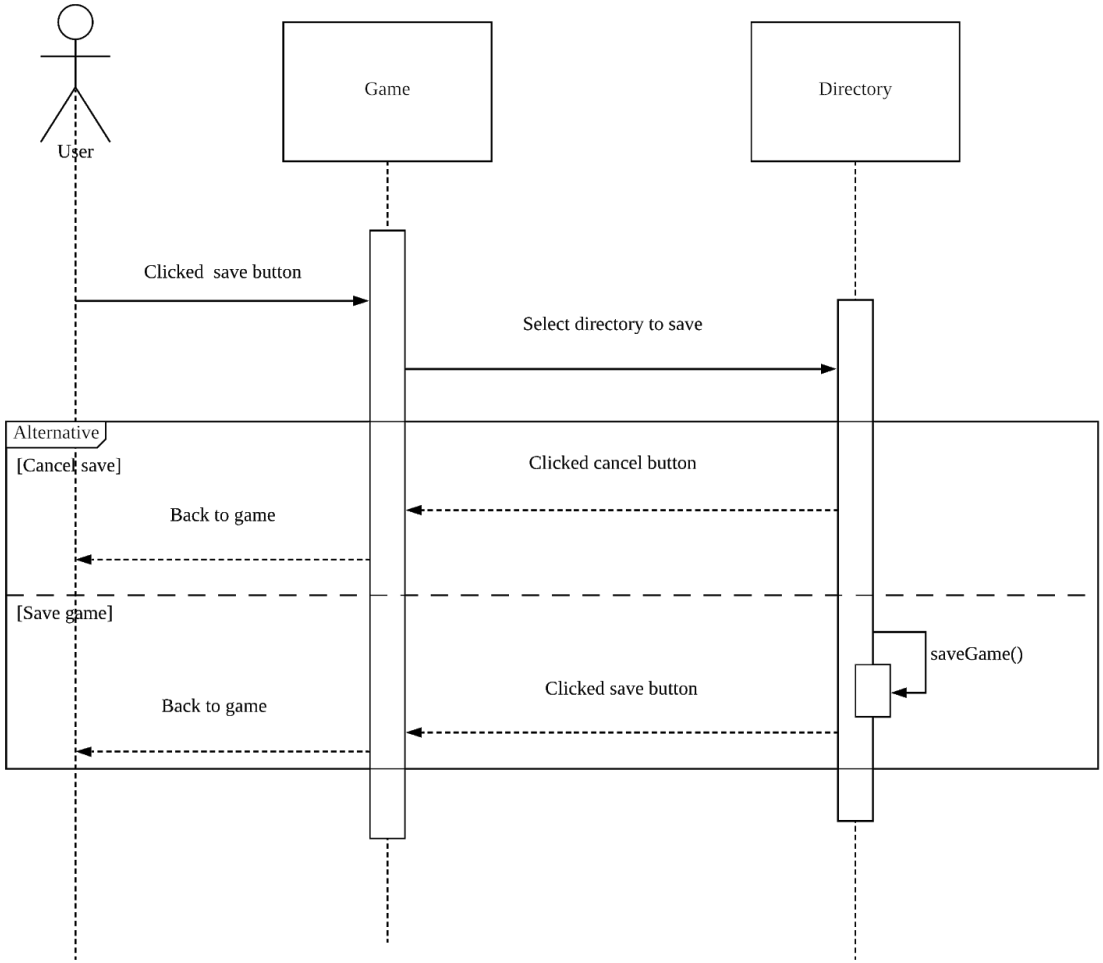
Move pieces sequence diagram



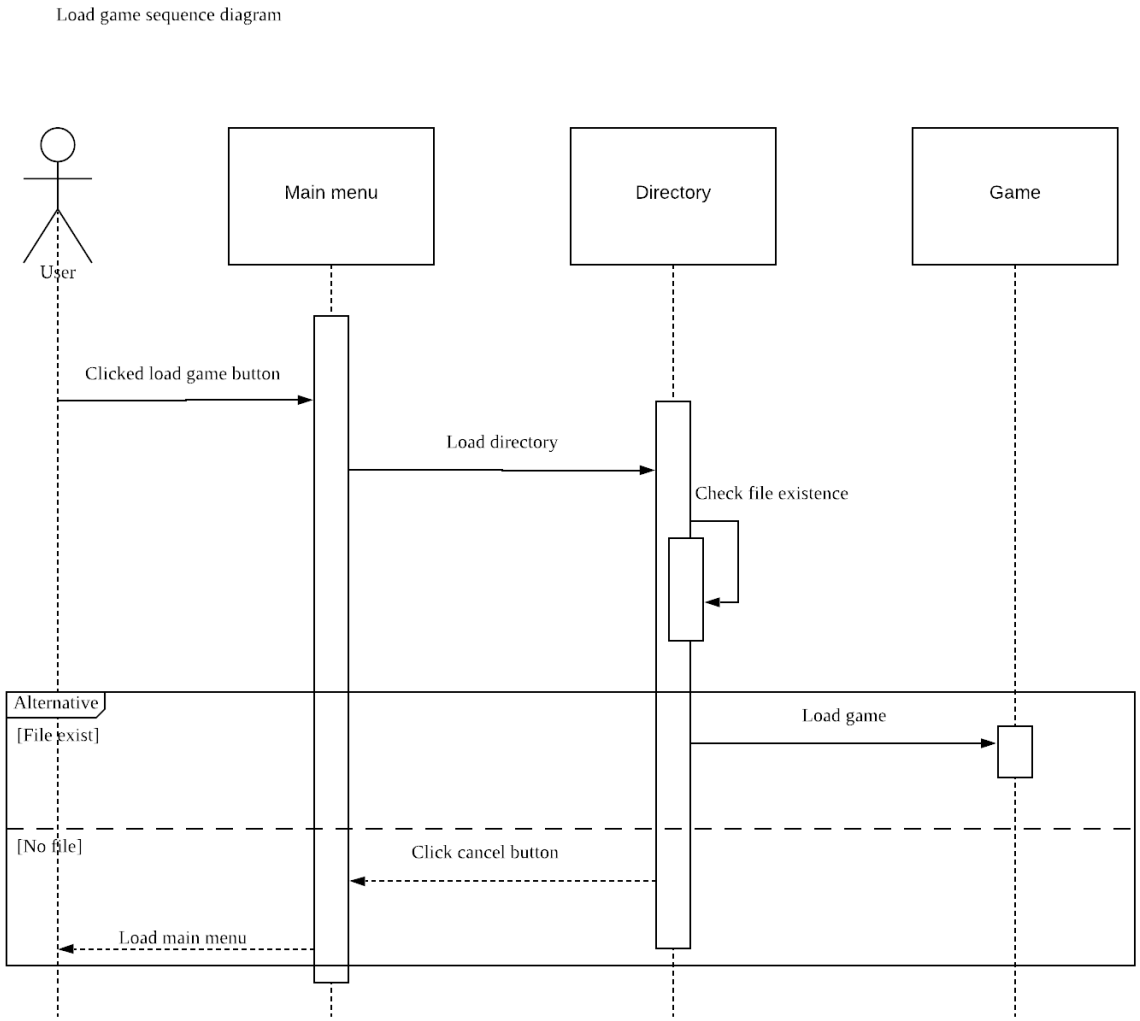


Move Pieces Sequence Diagram

Save game sequence diagram



Save Game Sequence Diagram



Load Game Sequence Diagram

User Guidelines

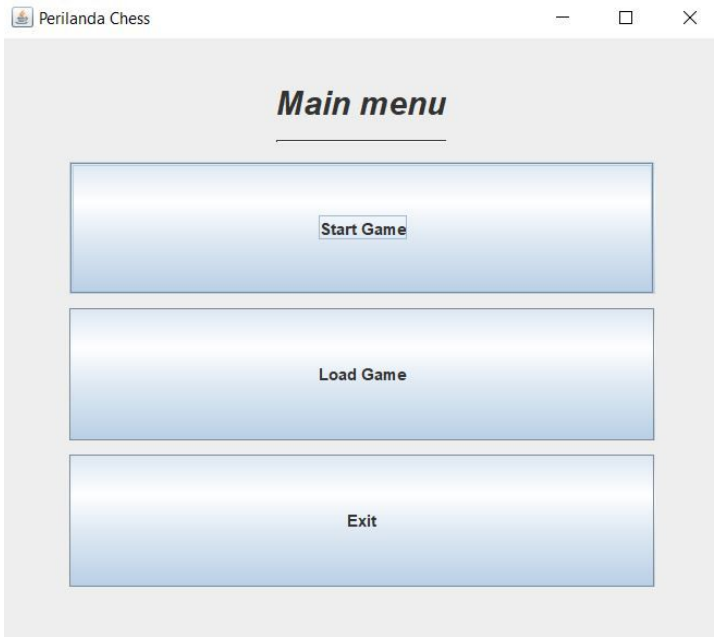


Figure 1: Main menu

When the user executes the program, the program will direct them to the main menu. The main menu will contain three buttons namely, “Start Game”, “Load Game”, and “Exit”. If the user wishes to exit, the user can either select the “Exit” button or the ‘x’ button on the top right corner of the window.

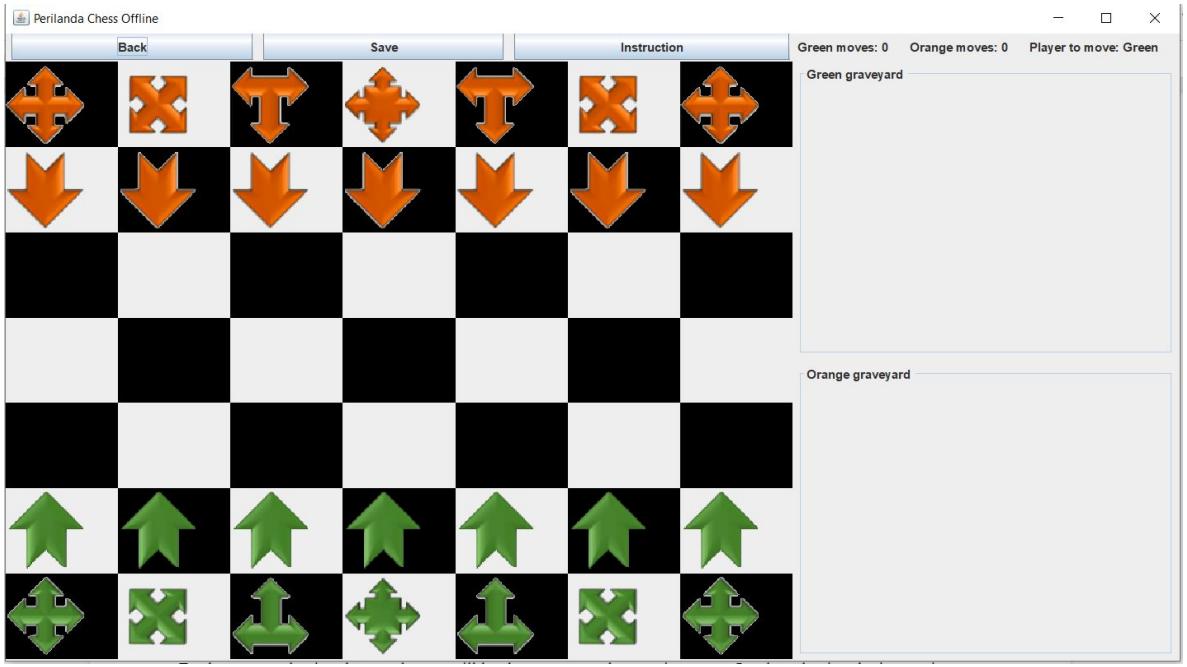


Figure 2: Game page

After selecting the start button, it will lead to the gameplay window. Inside this window, there are three buttons, namely the “Back” button, “Save” button and “Instruction” button. Selecting the “Back” button will lead the user back to the main menu page. Furthermore, selecting the “Save” button will let the user specify where to save the current game state. Lastly, selecting the “Instruction” button will bring up a pop-up showing the users the rules of the game such as the movement of each piece, the polymorphing nature of certain pieces and so on. Besides that, information such as how many times Green or Orange has moved, the player to move, and dead pieces are shown.

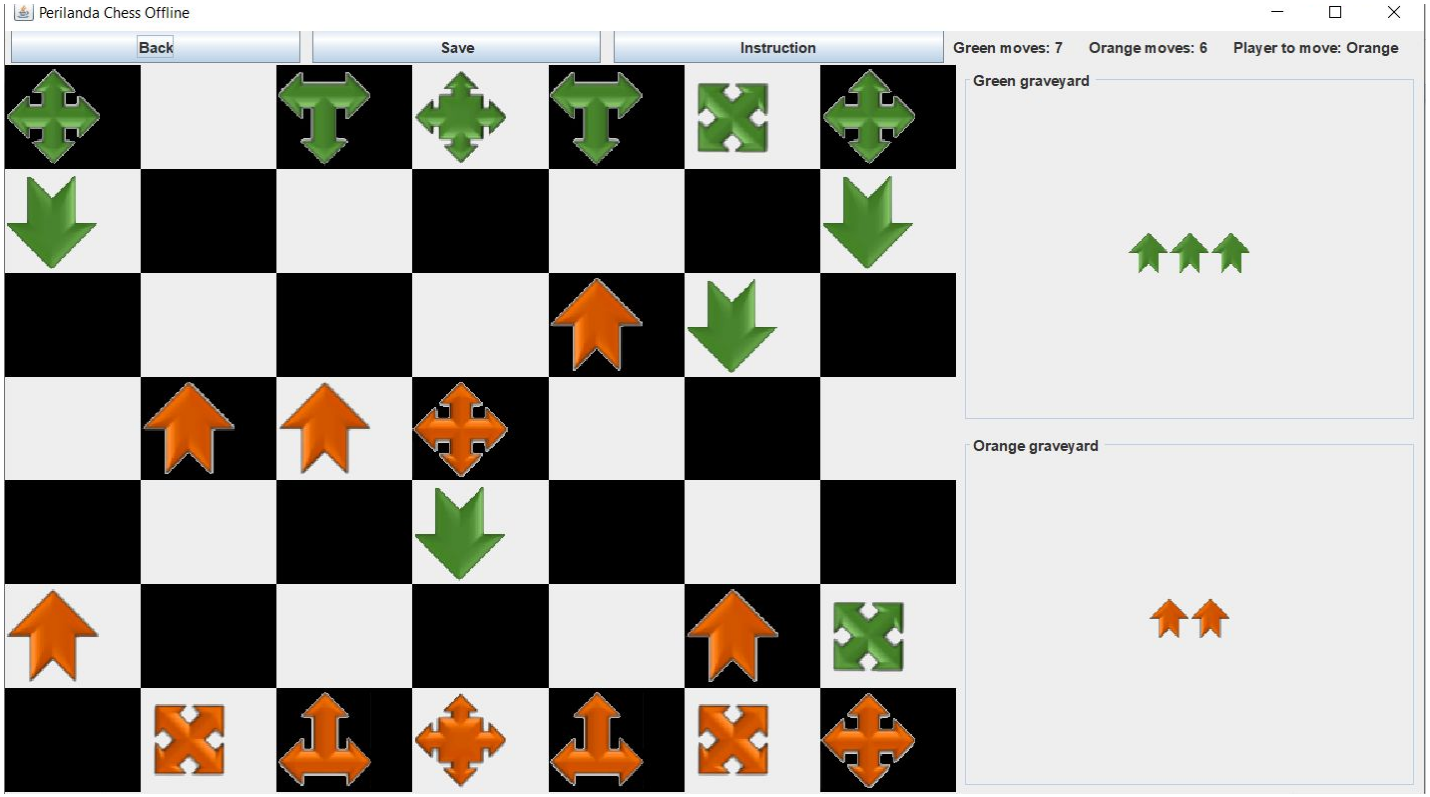


Figure 3: Shows ongoing game

In the figure above, it can be seen that the graveyards each show the dead pieces of the game.

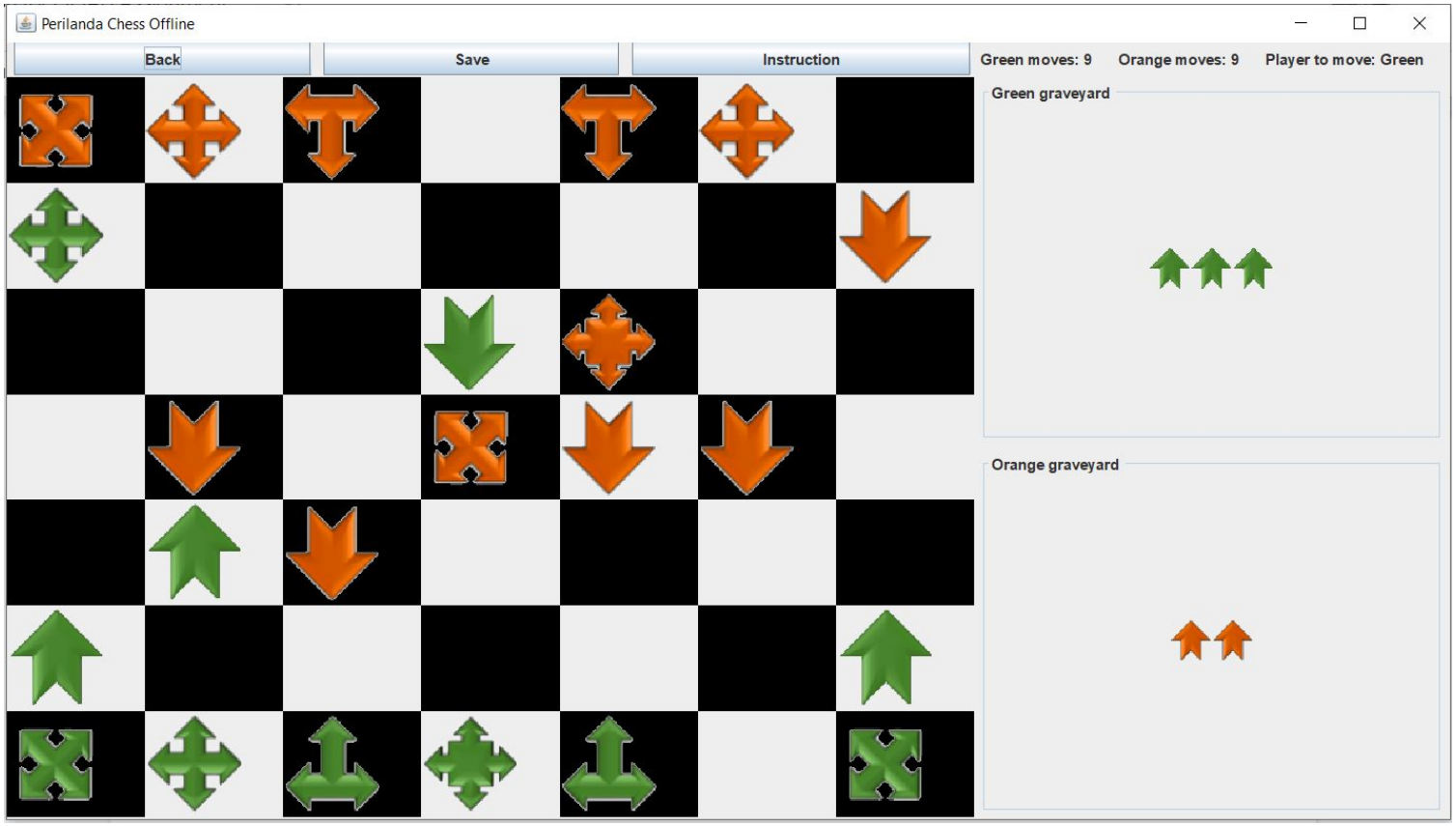


Figure 4: Shows ongoing game

Advancer and Trident pieces will start moving backwards when they reach the opposite side. Looking from the top, the Advancer piece in the third row, fourth column, is an example of a piece moving backwards.

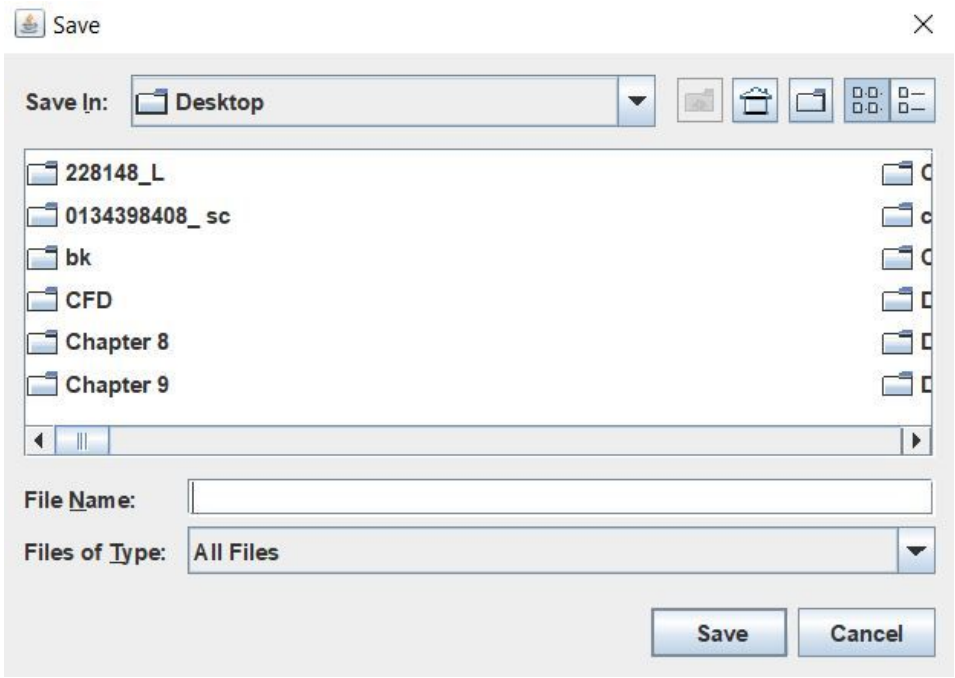


Figure 5: Save page

During the game, the players can save the game by pressing the "Save" button. Then, the user may choose where to store the save files.

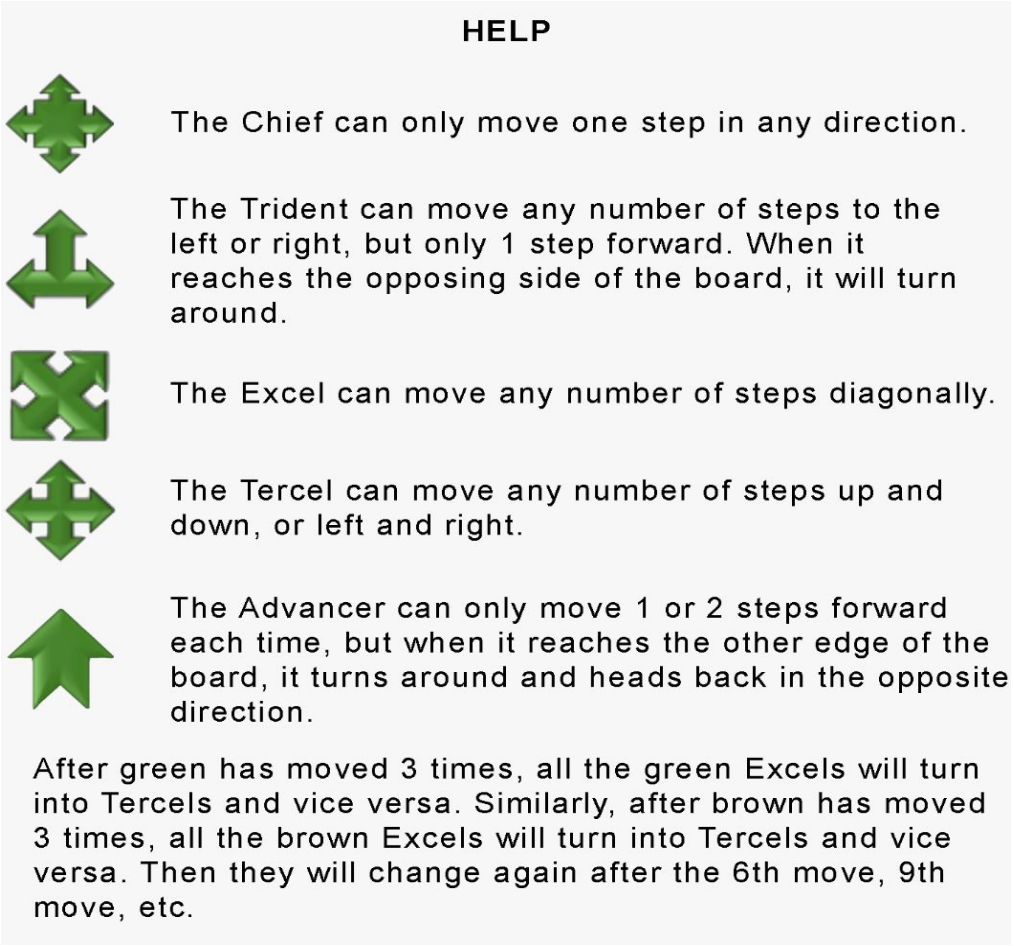


Figure 6: Instruction page

After selecting the "Instruction" button, a pop-up of the game instruction that describes the movement of each chess piece and the rules of the game is shown.

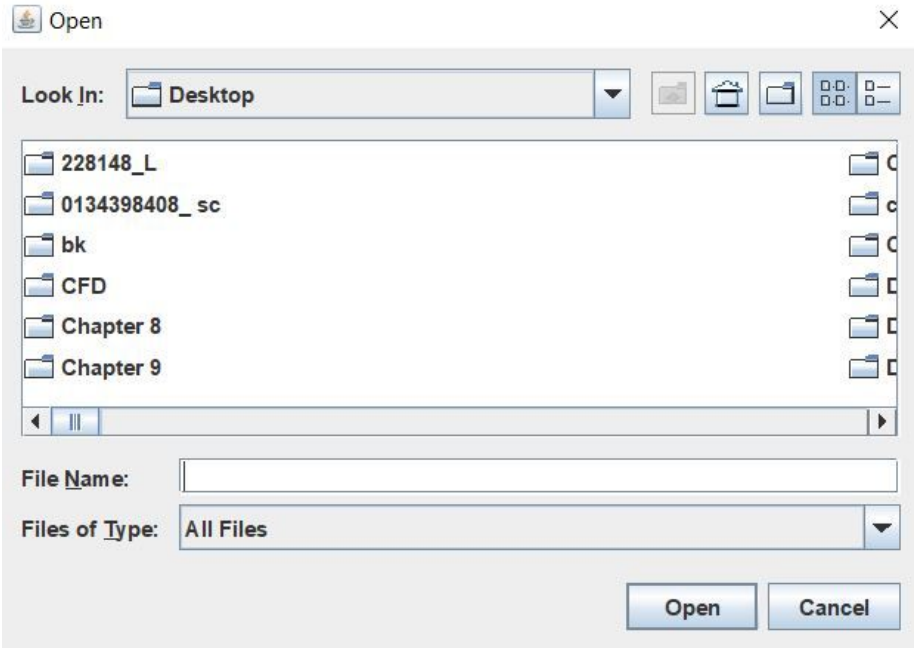


Figure 7: Load game page

After selecting the "Load Game" button from the main menu, the user may choose the save file to load the board state from.

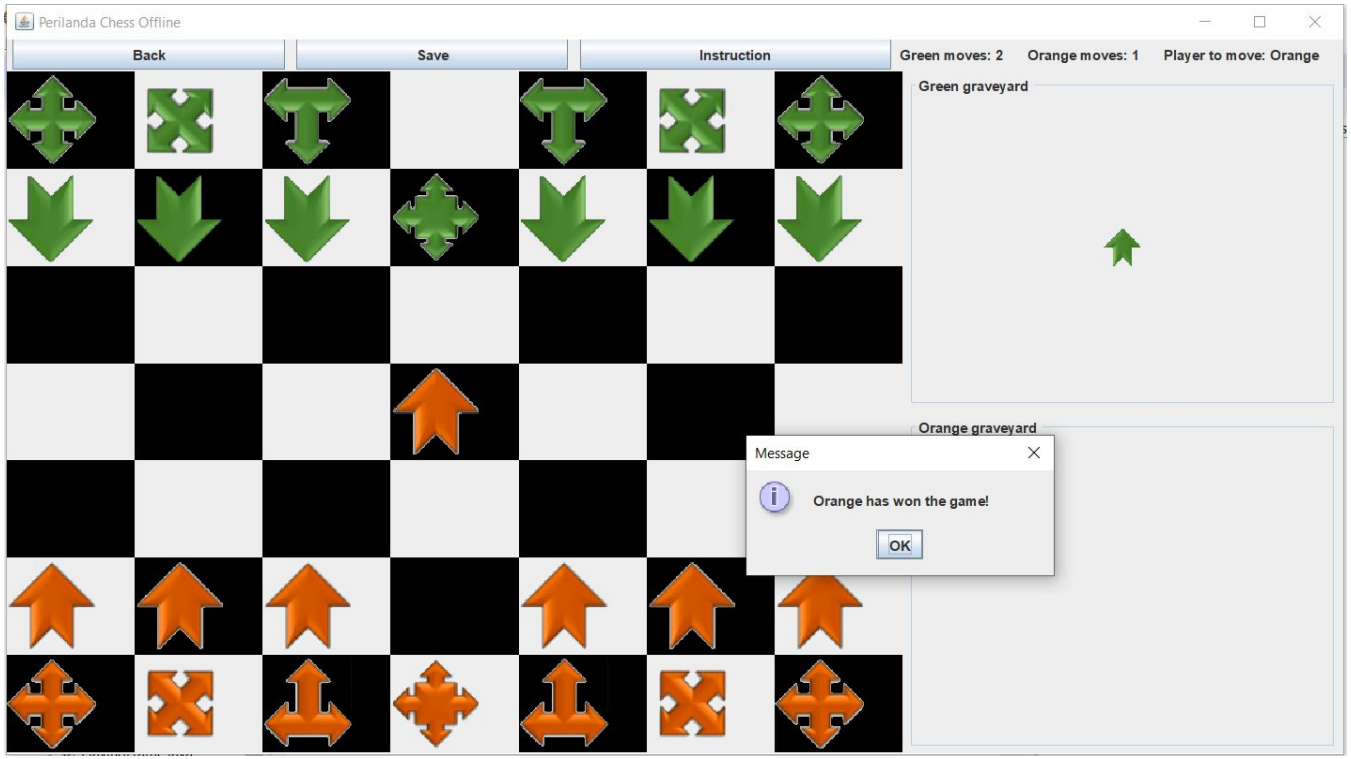


Figure 8: Pop-up message to announce the winner

When a valid move involves the capturing of a chief, a pop-up message will appear declaring the winner. After pressing the “OK” button, the board state will be fixed such as no move is considered valid. At this point, the player will be able to analyze the board and save it if he chooses so.

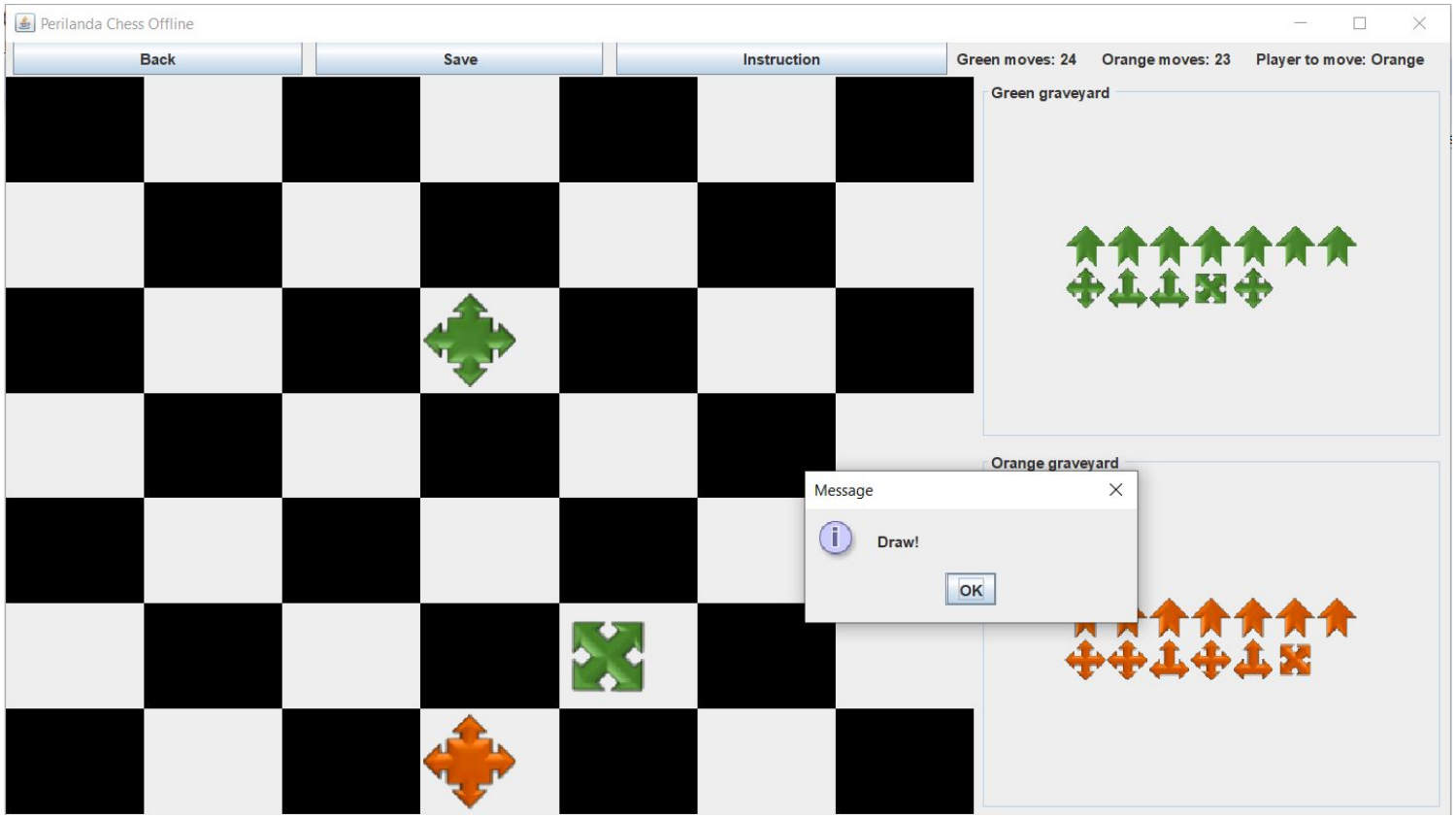


Figure 9: Draw condition

The status of the game is declared a draw in the case that, after a valid move is executed only the chiefs are left.