# Final Year Project Daybook

# Chess with Artificial Intelligence (in Java)

Monday 07/10/2019:

* Created a BoardTest JUnit test case class
* Within the BoardTest class, I wrote test cases for the tile initialisation method. I then wrote further test cases for the piece initialisation method.
* Created a Board Java class
* Within the Board class, I wrote a tile initialisation method which sets the properties of the chess board tiles to start a new game. The code sets the top and bottom 2 rows of tiles as occupied (as they will be used for pieces in their starting positions), then sets the rows in between as empty (vacant space which can be used to move a chess piece).
* I also wrote a chess piece initialisation method. This uses a HashMap which maps a tile co-ordinate to a piece name followed by the piece ID’s number if there is more than one of that type of piece. This tile co-ordinate will be the starting position of the piece mapped to it.

Tuesday 08/10/2019:

* Added JavaDoc comments to the Board class.
* Added JavaDoc comments to the BoardTest class.
* Started using Checkstyle and fixed Checkstyle issues in Board and BoardTest classes
* Changed HashMaps to TreeMaps, so that I have ordering on my keys, which was useful for my print status method mentioned later on.
* I wrote a getter for the tile occupation. This means I can pass a tile co-ordinate and the method will return if the tile is occupied or empty.
* I wrote a getter for obtaining what piece, if any, is in a given tile co-ordinate.
* I wrote a getter for obtaining the key (tile co-ordinate) with a given chess piece.
* I created a setter for setting a new tile co-ordinate of a chess piece. This can be used later on for moving a chess piece from one tile to another. The method also uses another existing method to obtain the previous tile the chess piece was in, in order to set the old tile’s occupation to empty and to set the current chess piece in the old tile to null.
* I created a print status method, which simply prints my Tile Occupation and Piece Position TreeMaps. This enables me to see where the pieces are and which tiles are occupied when testing the setter above in my JUnit tests, so that I can see if the setting of a new tile and unsetting of the old tile was successful. This will be useful until my GUI is created.

Wednesday 09/10/2019:

* Started committing work onto GitHub
* Explained what I have been doing in my Personal Diary so far this week
* Experimenting with potential game loop code in a test class
* Created a GameLoop test case class, which will eventually be used to test the GameLoop class.

Thursday 10/10/2019:

* Refactored code so that my TreeMaps are in a separate class named ‘TreeMaps’ and called from the Board class.
* Created a TestPlayer Junit test case class, which will contain my test cases for the Player class.
* Created a Player Java class. The Player class contains a constructor which takes a single string argument, which is “White” or “Black”. There is a getter for returning the current player, and there is also a boolean method for checking if it is the white player’s turn or not.
* Created an InvalidPlayer Exception. If a value other than “White” or “Black” is passed to the Player constructor, an InvalidPlayerException is thrown.

Friday 11/10/2019:

* Created an Invalid Piece Exception. If a given piece ID (comprising of the player colour, piece type and piece number) does not contain a piece type then the exception is thrown as the piece ID is not valid.
* Created a TestPawn Junit test case class. This will contain my test cases for the Pawn class.
* Created a Pawn class with a move pawn method. This method is responsible for validating a move from one tile to another, checking things such as the number of tiles the pawn is moving, if it’s the pawn’s first move, if the destination tile contains an opponent piece or is already occupied etc.

Tuesday 15/10/2019:

* Refactored BoardTest class to test the refactored code in the Board class.
* Refactored Board class to use the TreeMaps and methods in the TreeMap class. I also created a Board constructor which calls the initPieces and initTiles methods inside the class.

Monday 21/10/2019:

* Started research on the history of Chess, facts about its history, the rules of game, and different types of manoeuvres.

Monday 28/10/2019:

* Started my ‘History and Rules of Chess’ report, containing findings from my research which I started from the week previous.

Thursday 31/10/2019:

* Completed and uploaded my ‘History and Rules of Chess’ report to GitHub.

Monday 04/11/2019:

* Added a set position method to the Pawn class. This sets the new tile position of a selected pawn, as long as conditions in the move pawn class are met and the new tile does not contain another piece from the same player.
* Added a first move Boolean method to the Pawn class. This checks if the selected pawn has moved before or not. If it has, the pawn can only move one square forward or one square diagonally to capture an opponent piece. If it hasn’t, the pawn can move two squares forward. The piece ID is then added to an ArrayList named ‘movedPawns’ which is searched each time a pawn is selected to verify if it’s the pawns first move or not.

Wednesday 06/11/2019:

* Created a TestRook JUnit Test Case class. This will contain my test cases for the Rook Java class.
* Created a Rook Java class. This will contain validation and movement methods when the game is handling a Rook piece type.
* Started working on a moveRook method. This is the main method involved in validating if a move specified by the player with a Rook piece is valid or not. The only thing I need to work on next for this method is checking for any of the player’s own pieces, or the opponent’s pieces, in the Rook’s movement path.

Thursday 07/11/2019:

* Finished Javadoc commenting in Pawn and TestPawn classes so they appropriately explain/document the methods inside those classes.
* Attended meeting with supervisor to get feedback on my ‘History and Rules of Chess’ report. This feedback was very useful, and will take the feedback on-board and make relevant amendments to my report very soon.
* Added Javadoc commenting to TestRook and Rook classes to document the methods I have started writing so far.

Friday 15/11/2019:

* Started writing code responsible for checking if there is a piece between the origin tile and destination tile when moving a Rook in the Rook class. This is to ensure the Rook can’t ‘jump’ over pieces and to distinguish between a valid and invalid move.

Sunday 17/11/2019:

* Finished writing code responsible for checking if there is a piece in the Rook’s movement path in the Rook class.

Monday 18/11/2019:

* Started writing a setPos method, which will be responsible for setting the final position of the Rook after validation has taken place in the moveRook method. This new setPos method will also check for opponent pieces in the destination tile, so that the Rook can capture that piece after being moved.
* Refactored the setValue method in the TreeMaps class so that when setting a piece’s tile to null (thus removing the piece), it automatically sets the tileOccupation for that tile to “Empty”. Before this had to be done manually on another line of code. This was tested in a new Junit test case in the ‘TestTreeMaps’ class.
* Finished writing setPos method and committed this to GitHub.
* Started writing a Castling method in the Rook class to handle castling manoeuvres with the player’s King piece.

Tuesday 19/11/2019:

* Started revising my ‘History and Rules of Chess’ report, based on useful feedback given to me by my advisor in the previous meeting.
* Added an Abstract and Introduction to my report.

Wednesday 20/11/2019:

* Revised and lengthened the ‘History’ section in my report so that it focuses more on the actual history of the game, since this one of the main topics of the report.

Thursday 21/11/2019:

* Attended meeting with advisor to discuss work progress and receiving useful advice to help with my second report and interim report, both of which are due in the near future.
* Added citations to the ‘Facts About the Game and Its History’ and ‘Chess Manoeuvres’ sections of the report.

Friday 22/11/2019:

* Written new sections named “Check” and “Winning Conditions” in my report, as my advisor correctly pointed out that I have written rules for the game but did not discuss the winning conditions.
* Added short summaries to each of my new references/resources in my Bibliography section in my report.
* Finished all revisions to the History and Rules of Chess report. Committed this to GitHub.

Saturday 23/11/2019:

* Created TestKing Junit test class. Written test cases for the moveKing and setPos methods in the King class.
* Created King Java class.
* Added a moveKing method, which contains basic validation checks (just seeing if the method correctly validates the direction the King is moving in).
* Added a setPos method. This contains code to actually update/set the King’s position on the board after validation in the moveKing method.

Sunday 24/11/2019:

* Finished moveKing method in King class (Validation was all completed on 23/11/2019 but wasn’t fully completed until a call to the setPos method was made in each validation check).
* Finished setPos method in King class.
* Fixed Bug in Rook class and committed changes (Bug explained in commit comments)
* Created TestQueen Junit test class. Written test cases for all possible directions the Queen can move in and different scenarios of pieces being in her movement path and/or destination tile. This tests the moveQueen and setPos methods in the Queen class.
* Created Queen Java class.
* Added a moveQueen method, with code that checks if the Queen is moving in the Left, Right, Up, Down and North East direction with conditional statements

Monday 25/11/2019:

* Finished the moveQueen method by completing validation code for cases where the Queen is moving in the South East, South West, and North West directions.
* Added and finished the setPos method in the Queen class. This is important for validation checks in the Queen’s destination tile (different from checks for the movement path), before updating/setting the Queen’s position accordingly.

Thursday 28/11/2019:

* Created TestBishop Junit test class. This will contain all my tests for the Bishop Java class.
* Created Bishop Java class.
* Added a moveBishop method in the Bishop class. This method’s purpose is to validate a move with a Bishop piece in accordance to the game’s rules.

Saturday 30/11/2019:

* Started writing Coding Progression report.

Sunday 01/12/2019:

* Finished writing move bishop method in Bishop class. Started and finished writing the set position method in the Bishop class.

Monday 02/12/2019:

* Created TestKnight Junit test class and Knight Java class. Started and finished writing both move knight and set position methods in the Knight class.

Thursday 05/12/2019:

* Finished writing Coding Progression report.

Friday 06/12/2019:

* Finished writing Interim report.