SE-1108 Self Study-4

Your task requires you to create a **ChessBoard** class that contains a list of **ChessPiece** objects. Each **ChessPiece** has a **BoardPosition** that represents its position by storing the row and column on the board, a **PieceColor** property that can be either White or Black, and finally **value** property changing according to the type of the piece as follows:

Pawn:1, Bishop:3 Knight 3 Rook:5 Queen:9 King:100

ChessPiece class contains abstract **canMove(BoardPosition target)** method returning if the piece can move to the target in one move. This method is supposed to be implemented in each subclass (i.e **Pawn**, **Rook**, **Queen**, **King**, **Knight**, and **Bishop**) differently according to the rules of chess as follows:

Pawn: They can move one square forward at a time if the target is empty. If the target is occupied by opponent piece, they can move one square forward-diagonal. They can not move to any square occupied by the same-colored piece.

Bishop: They can move diagonally in any direction to any distance if the target is empty or occupied by an opponent piece. They can not jump over any pieces, so the squares in-between must be empty They can not move to any square occupied by the same-colored piece.

Rook: They can move horizontally or vertically to any distance if the target is empty or occupied by an opponent piece. They can not jump over any pieces, so the squares in-between must be empty They can not move to any square occupied by the same-colored piece.

Queen: They can move horizontally or vertically or diagonally to any distance if the target is empty or occupied by an opponent piece. They can not jump over any pieces, so the squares in-between must be empty They can not move to any square occupied by the same-colored piece.

Knight: They can make L-move (check internet if it is not clear) in any direction if the target is empty or occupied by an opponent piece. They can jump over any pieces on the way. They can not move to any square occupied by the same-colored piece.

King: They can move single-square horizontally or vertically or diagonally if the target is empty or occupied by an opponent piece. They can not move to any square occupied by the same-colored piece. Ignore the check-conditions to keep the implementation simple.

You can also ignore rules like castling, the first-move of pawns, and en passant and promotion rules to keep it simple. **You can also assume that a ChessBoard object will always store a valid chess board state. Therefore in your implementation you can ignore invalid chess board states.**

Both ChessBoard and all ChessPiece subclasses have a proper toString method and a clone method implemented.

The ChessBoard class must implement several additional methods, including at least the following:

- **materialAdvantage**(PieceColor color): returns the difference between the total value of the pieces of the given color and that of the opponent color.

- **attackedBy(BoardPosition p, PieceColor c)** : returns the number of pieces of the given color that can move to the given position in one move
- **safety**(ChessPiece piece): returns the difference between the total value of the attacking pieces of opponent color that can move to the piece's position in one move and the total value of the defending pieces of the same color that can move to the piece's position in one move.
- **centralControl(PieceColor color):** returns the number of pieces of the given color that can move one of the central squares on the board (i.e. (D4,D5,E4,E5) in one move.

Finally, you should create a Main class with a main function, where you will create instances of the above classes and test their functions by calling them.