```java
package Pieces;

import Game.Board;
import Game.Colour;
import Game.Player;

/**
 * This class represents the Knight piece
 * @author Param
 */
public class Knight extends Piece {

    public boolean canMove;

    public Knight(Colour colour) {
        super(PieceType.Knight, colour, 3);
    }

    @Override
    public int threats(Board board, int row, int column) {
        Piece[][] currentBoard = board.getBoard();
        Piece toExamine;
        int threatened = 0;
        // top-left
        if (row >= 2 && column >= 1) {
            toExamine = currentBoard[row - 2][column - 1];
            if (toExamine != null) {
                if (this.isOppositeColour(toExamine)) {
                    threatened += toExamine.weight;
                }
            }
        }
        if (row >= 1 && column >= 2) {
            toExamine = currentBoard[row - 1][column - 2];
            if (toExamine != null) {
                if (this.isOppositeColour(toExamine)) {
                    threatened += toExamine.weight;
                }
            }
        }
        // top-right
        if (row >= 2 && column <= 6) {
            toExamine = currentBoard[row - 2][column + 1];
            if (toExamine != null) {
                if (this.isOppositeColour(toExamine)) {
                    threatened += toExamine.weight;
                }
            }
        }
        if (row >= 1 && column <= 5) {
            toExamine = currentBoard[row - 1][column + 2];
            if (toExamine != null) {
                if (this.isOppositeColour(toExamine)) {
                    threatened += toExamine.weight;
                }
            }
        }
        // bottom-left
        if (row <= 5 && column >= 1) {
            toExamine = currentBoard[row + 2][column - 1];
            if (toExamine != null) {
                if (this.isOppositeColour(toExamine)) {
                    threatened += toExamine.weight;
                }
            }
        }
        if (row <= 6 && column >= 2) {
            toExamine = currentBoard[row + 1][column - 2];
            if (toExamine != null) {
```

```
                if (this.isOppositeColour(toExamine)) {
                    threatened += toExamine.weight;
                }
            }
        }
        // bottom-right
        if (row <= 5 && column <= 6) {
            toExamine = currentBoard[row + 2][column + 1];
            if (toExamine != null) {
                if (this.isOppositeColour(toExamine)) {
                    threatened += toExamine.weight;
                }
            }
        }
        if (row <= 6 && column <= 5) {
            toExamine = currentBoard[row + 1][column + 2];
            if (toExamine != null) {
                if (this.isOppositeColour(toExamine)) {
                    threatened += toExamine.weight;
                }
            }
        }
        return threatened;
    }

    @Override
    public int[][] attacks(Board board, int row, int column) {
        int[][] attacked = new int[8][8];
        // top-left
        if (row >= 2 && column >= 1) {
            attacked[row - 2][column - 1]++;
        }
        if (row >= 1 && column >= 2) {
            attacked[row - 1][column - 2]++;
        }
        // top-right
        if (row >= 2 && column <= 6) {
            attacked[row - 2][column + 1]++;
        }
        if (row >= 1 && column <= 5) {
            attacked[row - 1][column + 2]++;
        }
        // bottom-left
        if (row <= 5 && column >= 1) {
            attacked[row + 2][column - 1]++;
        }
        if (row <= 6 && column >= 2) {
            attacked[row + 1][column - 2]++;
        }
        // bottom-right
        if (row <= 5 && column <= 6) {
            attacked[row + 2][column + 1]++;
        }
        if (row <= 6 && column <= 5) {
            attacked[row + 1][column + 2]++;
        }
        return attacked;
    }

    @Override
    public boolean[][] validMoves(Player opponent, Board board, int row, int column) {
        Piece[][] currentBoard = board.getBoard();
        Piece toExamine;
        // reset to false and check
        canMove = false;
        boolean[][] validPositions = new boolean[8][8];
        // top-left
        if (row >= 2 && column >= 1) {
            toExamine = currentBoard[row - 2][column - 1];
```

```
            if (toExamine == null | toExamine != null &&
            this.isOppositeColour(toExamine)) {
                validPositions[row - 2][column - 1] = true;
                canMove = true;
            }
        }
        if (row >= 1 && column >= 2) {
            toExamine = currentBoard[row - 1][column - 2];
            if (toExamine == null | toExamine != null &&
            this.isOppositeColour(toExamine)) {
                validPositions[row - 1][column - 2] = true;
                canMove = true;
            }
        }
        // top-right
        if (row >= 2 && column <= 6) {
            toExamine = currentBoard[row - 2][column + 1];
            if (toExamine == null | toExamine != null &&
            this.isOppositeColour(toExamine)) {
                validPositions[row - 2][column + 1] = true;
                canMove = true;
            }
        }
        if (row >= 1 && column <= 5) {
            toExamine = currentBoard[row - 1][column + 2];
            if (toExamine == null | toExamine != null &&
            this.isOppositeColour(toExamine)) {
                validPositions[row - 1][column + 2] = true;
                canMove = true;
            }
        }
        // bottom-left
        if (row <= 5 && column >= 1) {
            toExamine = currentBoard[row + 2][column - 1];
            if (toExamine == null | toExamine != null &&
            this.isOppositeColour(toExamine)) {
                validPositions[row + 2][column - 1] = true;
                canMove = true;
            }
        }
        if (row <= 6 && column >= 2) {
            toExamine = currentBoard[row + 1][column - 2];
            if (toExamine == null | toExamine != null &&
            this.isOppositeColour(toExamine)) {
                validPositions[row + 1][column - 2] = true;
                canMove = true;
            }
        }
        // bottom-right
        if (row <= 5 && column <= 6) {
            toExamine = currentBoard[row + 2][column + 1];
            if (toExamine == null | toExamine != null &&
            this.isOppositeColour(toExamine)) {
                validPositions[row + 2][column + 1] = true;
                canMove = true;
            }
        }
        if (row <= 6 && column <= 5) {
            toExamine = currentBoard[row + 1][column + 2];
            if (toExamine == null | toExamine != null &&
            this.isOppositeColour(toExamine)) {
                validPositions[row + 1][column + 2] = true;
                canMove = true;
            }
        }
        return validPositions;
    }

    @Override
```

```
200         public boolean validSpecial() {
201             return false;
202         }
203 
204         @Override
205         public void modifySpecial() {
206             // nothing
207         }
208 
209         @Override
210         public String printToBoard() {
211             return this.colour == Colour.White ? "\u2658" : "\u265E";
212         }
213 
214         @Override
215         public String printToLog() {
216             return "N";
217         }
218 
219         @Override
220         public boolean getCanMove() {
221             return canMove;
222         }
223 
224     }
225 
```