

```

1  package Pieces;
2
3  import Game.Board;
4  import Game.Colour;
5  import Game.Player;
6
7  /**
8   * This class represents the Knight piece
9   * @author Param
10  */
11  public class Knight extends Piece {
12
13      public boolean canMove;
14
15      public Knight(Colour colour) {
16          super(PieceType.Knight, colour, 3);
17      }
18
19      @Override
20      public int threats(Board board, int row, int column) {
21          Piece[][] currentBoard = board.getBoard();
22          Piece toExamine;
23          int threatened = 0;
24          // top-left
25          if (row >= 2 && column >= 1) {
26              toExamine = currentBoard[row - 2][column - 1];
27              if (toExamine != null) {
28                  if (this.isOppositeColour(toExamine)) {
29                      threatened += toExamine.weight;
30                  }
31              }
32          }
33          if (row >= 1 && column >= 2) {
34              toExamine = currentBoard[row - 1][column - 2];
35              if (toExamine != null) {
36                  if (this.isOppositeColour(toExamine)) {
37                      threatened += toExamine.weight;
38                  }
39              }
40          }
41          // top-right
42          if (row >= 2 && column <= 6) {
43              toExamine = currentBoard[row - 2][column + 1];
44              if (toExamine != null) {
45                  if (this.isOppositeColour(toExamine)) {
46                      threatened += toExamine.weight;
47                  }
48              }
49          }
50          if (row >= 1 && column <= 5) {
51              toExamine = currentBoard[row - 1][column + 2];
52              if (toExamine != null) {
53                  if (this.isOppositeColour(toExamine)) {
54                      threatened += toExamine.weight;
55                  }
56              }
57          }
58          // bottom-left
59          if (row <= 5 && column >= 1) {
60              toExamine = currentBoard[row + 2][column - 1];
61              if (toExamine != null) {
62                  if (this.isOppositeColour(toExamine)) {
63                      threatened += toExamine.weight;
64                  }
65              }
66          }
67          if (row <= 6 && column >= 2) {
68              toExamine = currentBoard[row + 1][column - 2];
69              if (toExamine != null) {

```

```

70         if (this.isOppositeColour(toExamine)) {
71             threatened += toExamine.weight;
72         }
73     }
74 }
75 // bottom-right
76 if (row <= 5 && column <= 6) {
77     toExamine = currentBoard[row + 2][column + 1];
78     if (toExamine != null) {
79         if (this.isOppositeColour(toExamine)) {
80             threatened += toExamine.weight;
81         }
82     }
83 }
84 if (row <= 6 && column <= 5) {
85     toExamine = currentBoard[row + 1][column + 2];
86     if (toExamine != null) {
87         if (this.isOppositeColour(toExamine)) {
88             threatened += toExamine.weight;
89         }
90     }
91 }
92 return threatened;
93 }
94
95 @Override
96 public int[][] attacks(Board board, int row, int column) {
97     int[][] attacked = new int[8][8];
98     // top-left
99     if (row >= 2 && column >= 1) {
100         attacked[row - 2][column - 1]++;
101     }
102     if (row >= 1 && column >= 2) {
103         attacked[row - 1][column - 2]++;
104     }
105     // top-right
106     if (row >= 2 && column <= 6) {
107         attacked[row - 2][column + 1]++;
108     }
109     if (row >= 1 && column <= 5) {
110         attacked[row - 1][column + 2]++;
111     }
112     // bottom-left
113     if (row <= 5 && column >= 1) {
114         attacked[row + 2][column - 1]++;
115     }
116     if (row <= 6 && column >= 2) {
117         attacked[row + 1][column - 2]++;
118     }
119     // bottom-right
120     if (row <= 5 && column <= 6) {
121         attacked[row + 2][column + 1]++;
122     }
123     if (row <= 6 && column <= 5) {
124         attacked[row + 1][column + 2]++;
125     }
126     return attacked;
127 }
128
129 @Override
130 public boolean[][] validMoves(Player opponent, Board board, int row, int column) {
131     Piece[][] currentBoard = board.getBoard();
132     Piece toExamine;
133     // reset to false and check
134     canMove = false;
135     boolean[][] validPositions = new boolean[8][8];
136     // top-left
137     if (row >= 2 && column >= 1) {
138         toExamine = currentBoard[row - 2][column - 1];

```

```

139         if (toExamine == null | toExamine != null &&
140             this.isOppositeColour(toExamine)) {
141             validPositions[row - 2][column - 1] = true;
142             canMove = true;
143         }
144     }
145     if (row >= 1 && column >= 2) {
146         toExamine = currentBoard[row - 1][column - 2];
147         if (toExamine == null | toExamine != null &&
148             this.isOppositeColour(toExamine)) {
149             validPositions[row - 1][column - 2] = true;
150             canMove = true;
151         }
152     }
153     // top-right
154     if (row >= 2 && column <= 6) {
155         toExamine = currentBoard[row - 2][column + 1];
156         if (toExamine == null | toExamine != null &&
157             this.isOppositeColour(toExamine)) {
158             validPositions[row - 2][column + 1] = true;
159             canMove = true;
160         }
161     }
162     if (row >= 1 && column <= 5) {
163         toExamine = currentBoard[row - 1][column + 2];
164         if (toExamine == null | toExamine != null &&
165             this.isOppositeColour(toExamine)) {
166             validPositions[row - 1][column + 2] = true;
167             canMove = true;
168         }
169     }
170     // bottom-left
171     if (row <= 5 && column >= 1) {
172         toExamine = currentBoard[row + 2][column - 1];
173         if (toExamine == null | toExamine != null &&
174             this.isOppositeColour(toExamine)) {
175             validPositions[row + 2][column - 1] = true;
176             canMove = true;
177         }
178     }
179     if (row <= 6 && column >= 2) {
180         toExamine = currentBoard[row + 1][column - 2];
181         if (toExamine == null | toExamine != null &&
182             this.isOppositeColour(toExamine)) {
183             validPositions[row + 1][column - 2] = true;
184             canMove = true;
185         }
186     }
187     // bottom-right
188     if (row <= 5 && column <= 6) {
189         toExamine = currentBoard[row + 2][column + 1];
190         if (toExamine == null | toExamine != null &&
191             this.isOppositeColour(toExamine)) {
192             validPositions[row + 2][column + 1] = true;
193             canMove = true;
194         }
195     }
196     if (row <= 6 && column <= 5) {
197         toExamine = currentBoard[row + 1][column + 2];
198         if (toExamine == null | toExamine != null &&
199             this.isOppositeColour(toExamine)) {
200             validPositions[row + 1][column + 2] = true;
201             canMove = true;
202         }
203     }
204     return validPositions;
205 }

```

@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
```