FINALS KING JAVA

Language used: Java 8

```
class Result {

 static char[][] board;

 static class pos {
 int i, j;

 pos(int i, int j){
 this.i =i;
 this.j=j;
 }

 boolean inBounds() {
 return i >=0 && j >= 0 && i < 8 && j < 8;
 }

char piece() {
 if (inBounds()) {
 return board[i][j];
 }
 else {
 return 0;
 }
 }

 boolean emptyOrEnemy() {
```

```
return piece() == 0 || piece() == 'K';

}


pos add(int i, int j) {

return new pos(this.i + i, this.j + j);

}

}


static ArrayList<pos> getMoves(pos from) {

ArrayList<pos> result = new ArrayList<>();

char piece = from.piece();

if (piece == 'P') {

pos front = from.add(1, 0);

if (front.piece() == 0) {

result.add(front);

}

pos frontl = from.add(1, 1);

if (frontl.piece() == 'K') {

result.add(frontl);

}

pos frontr = from.add(1, -1);

if (frontr.piece() == 'K') {

result.add(frontr);

}

}

else if (piece == 'N') {

int[] dx = { 2, 1, -2, -1, 2, 1, -2, -1 };

int[] dy = { 1, 2, 1, 2, -1, -2, -1, -2 };

for (int i = 0; i < 8 ; ++i ){
```

```
pos potential = from.add(dx[i], dy[i]);
if (potential.emptyOrEnemy()) {
result.add(potential);
}
}
}
else if (piece == 'R' || piece == 'Q' || piece == 'B') {
int[] dxs, dys;
if (piece == 'R') {
dxs = new int[] { 0, 0, 1, -1 };
dys = new int[] { 1, -1, 0, 0 };
}
else if (piece == 'Q') {
dxs = new int[] { 1, -1, 1, -1, 0, 0, 1, -1 };
dys = new int[] { 1, -1, -1, 1, 1, -1, 0, 0 };
}
else /*if (piece == 'B')*/ {
dxs = new int[] { 1, -1, 1, -1 };
dys = new int[] { 1, -1, -1, 1 };
}

for (int i = 0; i < dxs.length; ++i) {
int dx = dxs[i];
int dy = dys[i];
y y [ ];
pos at = from;
while (at.inBounds()) {
at = at.add(dx, dy);
if (at.piece() == 0) {
```

```
result.add(at);

}

else if (at.piece() == 'K') {

result.add(at);

break;

}

else {

break;

}

}

}

}

for (int i = result.size() - 1; i >= 0 ;--i ){

if (!result.get(i).inBounds()) {

result.remove(i);

}

}

return result;

}


static void cloneBoard() {

char[][] result = new char[8][8];

for (int i =0 ; i < 8 ;++i) {

for (int j = 0; j< 8 ; ++j) {

result[i][j] = board[i][j];

}

}

board = result;

}
```

```
static pos getKingPos() {
pos king = null;
out: for (int i = 0; i<8;++i){
for(int j=0;j<8;++j){
if (board[i][j] == 'K') {
king = new pos(i, j);
break out;
}
}
}
return king;
}

static boolean isInCheck() {
pos king = getKingPos();
for (int i = 0; i < 8; ++i) {
for (int j = 0; j < 8; ++j) {
if (board[i][j] != 0 && board[i][j] != 'K') {
ArrayList<pos> moves = getMoves(new pos(i, j));
for (pos p : moves) {
if (p.i == king.i && p.j == king.j) {
return true;
}
}
}
}
}
return false;
```

```
}

/*
 * Complete the 'find_king_status' function below.
 *
 * The function is expected to return a STRING.
 * The function accepts STRING pieces as parameter.
 */
public static String find_king_status(String pieces) {
board = new char[8][8];
for (char[] arr : board) {
Arrays.fill(arr, (char) 0);
}
for (String piece : pieces.split(" ")) {
int col = piece.charAt(1) - 'a';
int row = piece.charAt(2) - '1';
char t = piece.charAt(0);
board[row][col] = t;
}
pos king = getKingPos();
char[][] template = board;
int[] dxs = { -1, 0, 1, -1, /*0,*/ 1, -1, 0, 1 };
int[] dys = { -1, -1, -1, 0, /*0,*/ 0, 1, 1, 1 };
int stuck = 0;
int stuckInBounds = 0;
for (int i = 0; i< 8 ; ++i) {
int dx = dxs[i];
int dy = dys[i];
pos newPos = king.add(dx, dy);
if (newPos.inBounds()) {
board = template;
cloneBoard();
board[king.i][king.j] = 0;
board[newPos.i][newPos.j] = 'K';
if (isInCheck()) {
stuck += 1;
}
stuckInBounds += 1;
}
}
board = template;
cloneBoard();
if (isInCheck()) {
if (stuck == stuckInBounds) {
return "CHECKMATE";
}
else {
return "CHECK";
}
}
else {
if (stuck == stuckInBounds) {
return "STALEMATE";
}
else {
return "SAFE";
}
}
```

```
 }
}
```