Description of the algorithms

MIN-MAX ALGORITHM

- Code:

```
public static int minimax(Board board, int depth, Player.Side side, boolean maximizing) {

    if (depth == 0) return board.board_evaluation(side, maximizing);

    ArrayList<Move> possibleMoves = board.getPossibleMoves(maximizing, side);

    Board tempBoard = board.copy();

    if (maximizing) {

        int maxEva = Integer.MIN_VALUE;

        for (Move m: possibleMoves) {

            tempBoard = board.makeMove(m, maximizing);

            int eva;

            if (side == Player.Side.BLACK) eva = minimax(tempBoard, depth-1, 
Player.Side.WHITE,false);

            else eva = minimax(tempBoard, depth-1, Player.Side.BLACK,false);

            maxEva = Math.max(maxEva, eva);

        }

        return maxEva;

    }

    else {

        int minEva = Integer.MAX_VALUE;


        for (Move m: possibleMoves) {

            tempBoard = board.makeMove(m, maximizing);

            int eva;

            if (side == Player.Side.BLACK) eva = minimax(tempBoard, depth-1, 
Player.Side.WHITE,true);

            else eva = minimax(tempBoard, depth-1, Player.Side.BLACK,true);

            minEva = Math.min(minEva, eva);

        }

        return minEva;

    }

}
```

- Explanation:

The minimax function is a recursive function that returns an integer that is the value of the evaluation that has been given to the board in case it is maximizing or minimizing. As parameters we have the board we are studying, the depth where we are currently, the color of the player's tiles, and whether it is maximizing or minimizing.

If the depth reaches 0 we return the value of the board evaluation. On the other hand, if the value is greater than 0, we create a copy of the board and make alternating calls of maximization and minimization until reaching depth 0 that then will be returning the calls and looking for maxima and minima depending on the level in which be.

We will always use this call when the AI player touches it to find the best possible play in a particular state of the board.

ALPHA BETA PRUINING ALGORITHM

```
public static int alpha_beta(Board board, int depth, Player.Side side, boolean
maximizing, double alpha, double beta)

{

    if(depth == 0) {

        return board.board_evaluation(side, maximizing);

    }

    ArrayList<Move> possibleMoves = board.getPossibleMoves(maximizing, side);


    int initial = 0;

    Board tempBoard = null;

    if(maximizing)

    {

        initial = Integer.MIN_VALUE;

        for (Move possibleMove : possibleMoves) {

            tempBoard = board.copy();

            tempBoard.makeMove(possibleMove, maximizing);


            int result;

            if (side == Player.Side.BLACK)

                result = alpha_beta(tempBoard, depth - 1, Player.Side.WHITE, !maximizing,
alpha, beta);
```

```
            else result = alpha_beta(tempBoard, depth - 1, Player.Side.BLACK, !
maximizing, alpha, beta);

            initial = Math.max(result, initial);
            alpha = Math.max(alpha, initial);

            if (alpha >= beta)
                break;
        }
    }
    //minimizing
    else
    {
        initial = Integer.MAX_VALUE;
        for (Move possibleMove : possibleMoves) {
            tempBoard = board.copy();
            tempBoard.makeMove(possibleMove, maximizing);

            int result;
            if (side == Player.Side.BLACK)
                result = alpha_beta(tempBoard, depth - 1, Player.Side.WHITE, !maximizing,
alpha, beta);
            else result = alpha_beta(tempBoard, depth - 1, Player.Side.BLACK, !
maximizing, alpha, beta);

            initial = Math.min(result, initial);
            alpha = Math.min(alpha, initial);

            if (alpha >= beta)
                break;
        }
    }

    return initial;
}
```

- Explanation:

The alpha-beta function is a recursive function that is based on the minimax function but that adds the alpha and beta parameters to make a cut in the search in case of finding a better solution than the one of a higher level in case of maximization. or to find a worse solution than a higher level in case of minimization.