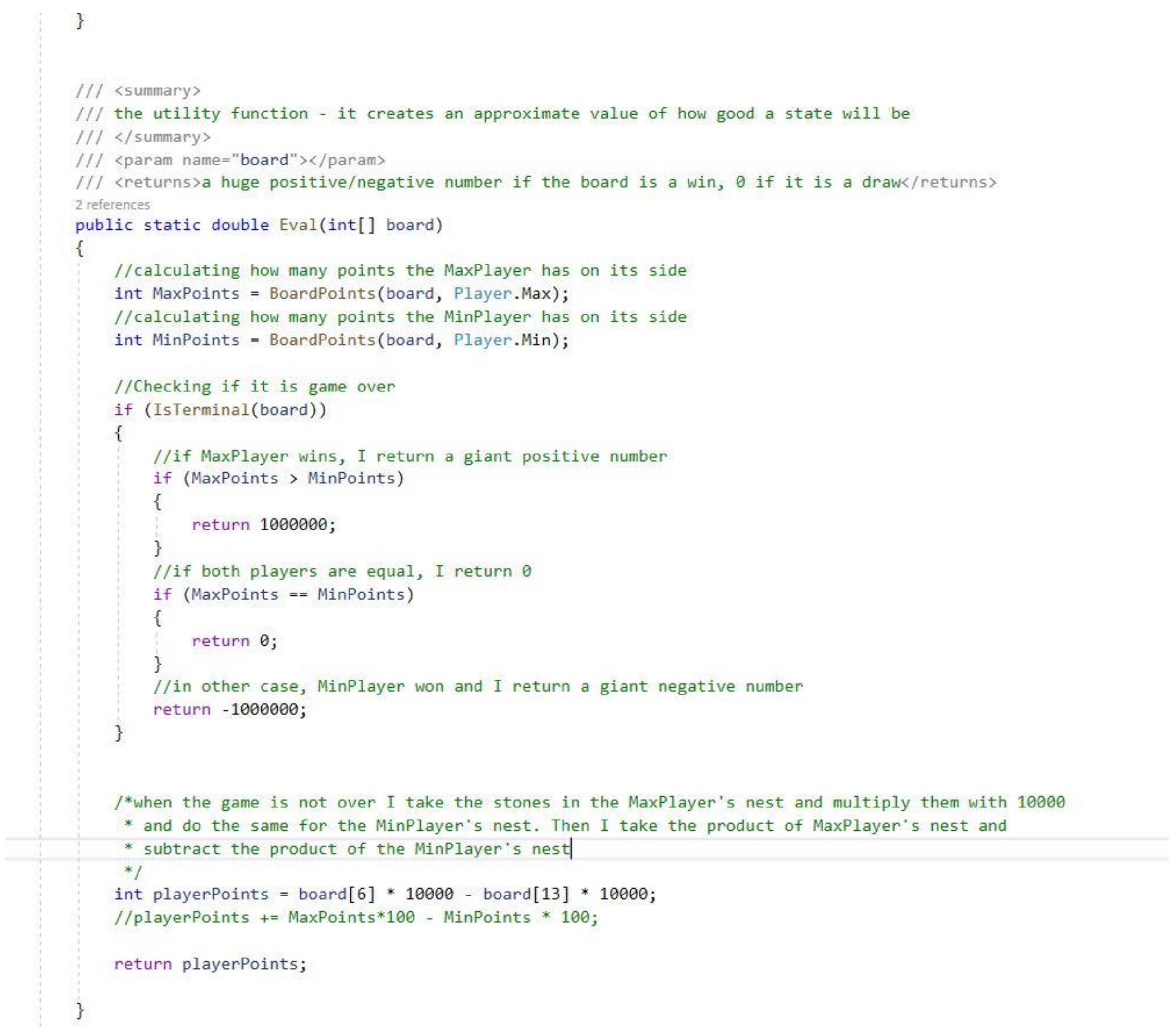
Minimax

The utility function, or evaluation as I call it in my solution, is a function that gives an estimate how good or bad a certain move would be for the players.

In the function, I begin with calculating how many points each player got on its side of the board. Later on, I explore if the game has terminated or not. The game terminates when one player does not have any more stones left in any of the six wholes on its side of the board.

If the game is terminated, I explore who the winner is. If it is a draw between the players, the function will return 0. If Max wins, the function will maximize the value and return a big positive number. If Min wins, the function minimizes the value and return a very low negative number.

If the game is not terminated, the function will provide a value that gives an estimate on how it is going for the players. Therefore, I take all the stones in Max's nest multiplies with 10000 and do the same for Min's nest. The product of Min's nest is then subtracted from the product of Max's nest and the function returns the value.



```
}


/// <summary>
/// the utility function - it creates an approximate value of how good a state will be
/// </summary>
/// <param name="board"></param>
/// <returns>a huge positive/negative number if the board is a win, 0 if it is a draw</returns>
2 references
public static double Eval(int[] board)
{
    //calculating how many points the MaxPlayer has on its side
    int MaxPoints = BoardPoints(board, Player.Max);
    //calculating how many points the MinPlayer has on its side
    int MinPoints = BoardPoints(board, Player.Min);

    //Checking if it is game over
    if (IsTerminal(board))
    {
        //if MaxPlayer wins, I return a giant positive number
        if (MaxPoints > MinPoints)
        {
            return 1000000;
        }
        //if both players are equal, I return 0
        if (MaxPoints == MinPoints)
        {
            return 0;
        }
        //in other case, MinPlayer won and I return a giant negative number
        return -1000000;
    }


    /*when the game is not over I take the stones in the MaxPlayer's nest and multiply them with 10000
     * and do the same for the MinPlayer's nest. Then I take the product of MaxPlayer's nest and
     * subtract the product of the MinPlayer's nest
     */
    int playerPoints = board[6] * 10000 - board[13] * 10000;
    //playerPoints += MaxPoints*100 - MinPoints * 100;

    return playerPoints;

}
```