```
class Solution
{
static class Move
{
  int row, col;
};

static char player = 'x', opponent = 'o';

static Boolean isMovesLeft(char board[][])
{
  for (int i = 0; i < 3; i++)
    for (int j = 0; j < 3; j++)
      if (board[i][j] == '_')
        return true;
  return false;
}

static int evaluate(char b[][])
{
  for (int row = 0; row < 3; row++)
  {
    if (b[row][0] == b[row][1] &&
      b[row][1] == b[row][2])
    {
      if (b[row][0] == player)
        return +10;
      else if (b[row][0] == opponent)
        return -10;
    }
  }

  for (int col = 0; col < 3; col++)
  {
    if (b[0][col] == b[1][col] &&
      b[1][col] == b[2][col])
    {
      if (b[0][col] == player)
        return +10;

      else if (b[0][col] == opponent)
        return -10;
    }
  }

  if (b[0][0] == b[1][1] && b[1][1] == b[2][2])
  {
    if (b[0][0] == player)
      return +10;
    else if (b[0][0] == opponent)
      return -10;
  }

  if (b[0][2] == b[1][1] && b[1][1] == b[2][0])
  {
    if (b[0][2] == player)
      return +10;
    else if (b[0][2] == opponent)
      return -10;
  }

  return 0;
}

static int minimax(char board[][],
,         int depth, Boolean isMax)
{
  int score = evaluate(board);

  if (score == 10)
    return score;

  if (score == -10)
    return score;

  if (isMovesLeft(board) == false)
    return 0;

  if (isMax)
  {
    int best = -1000;

    for (int i = 0; i < 3; i++)
    {
      for (int j = 0; j < 3; j++)
      {
        if (board[i][j]=='_')
        {
          board[i][j] = player;

          best = Math.max(best, minimax(board,
                  depth + 1, !isMax));

          board[i][j] = '_';
        }
      }
    }
    return best;
  }

  else
  {
    int best = 1000;

    for (int i = 0; i < 3; i++)
    {
      for (int j = 0; j < 3; j++)
      {
        if (board[i][j] == '_')
        {
          board[i][j] = opponent;

          best = Math.min(best, minimax(board,
                  depth + 1, !isMax));

```

```java
          board[i][j] = '_';
        }
      }
    }
    return best;
  }
}

static Move findBestMove(char board[][])
{
  int bestVal = -1000;
  Move bestMove = new Move();
  bestMove.row = -1;
  bestMove.col = -1;

  for (int i = 0; i < 3; i++)
  {
    for (int j = 0; j < 3; j++)
    {
      if (board[i][j] == '_')
      {
        board[i][j] = player;

        int moveVal = minimax(board, 0, false);

        board[i][j] = '_';

        if (moveVal > bestVal)
        {
          bestMove.row = i;
          bestMove.col = j;
          bestVal = moveVal;
        }
      }
    }
  }

  System.out.printf("The value of the best Move " +
              "is : %d\n\n", bestVal);

  return bestMove;
}

public static void main(String[] args)
{
  char board[][] = {{ 'x', 'o', 'x' },
          { 'o', 'o', 'x' },
          { '_', '_', '_' }};

  Move bestMove = findBestMove(board);

  System.out.printf("The Optimal Move is :\n");
  System.out.printf("ROW: %d COL: %d\n\n",
      bestMove.row, bestMove.col );
}
}
```