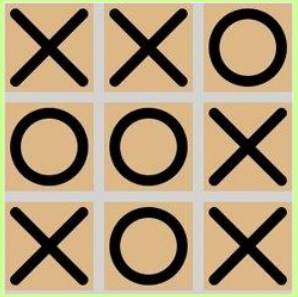
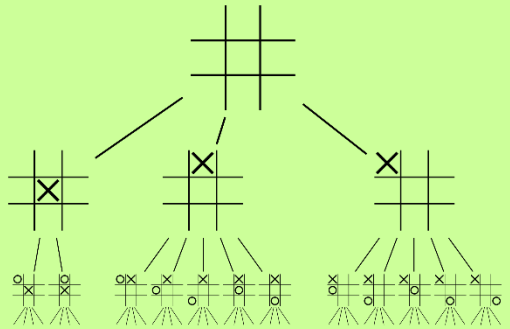
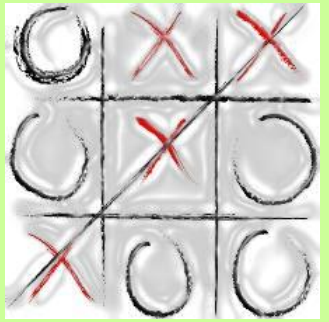
Tic Tac Toe

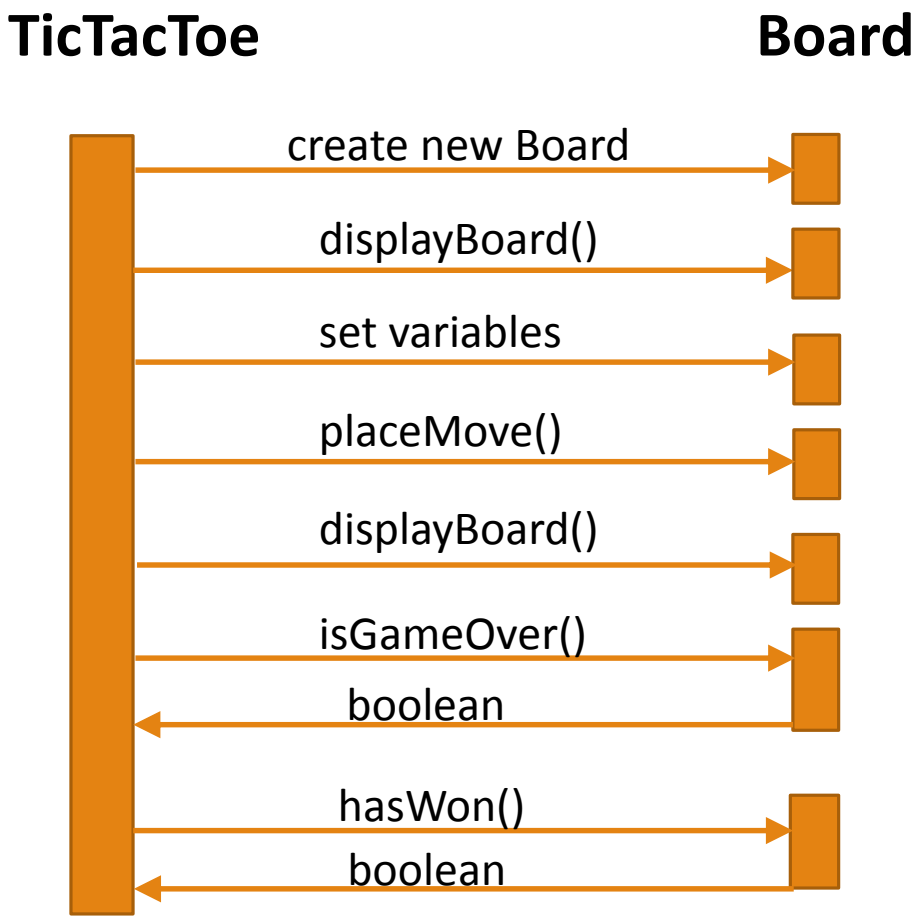
AI MOVES USING MINIMAX ALGORITHM



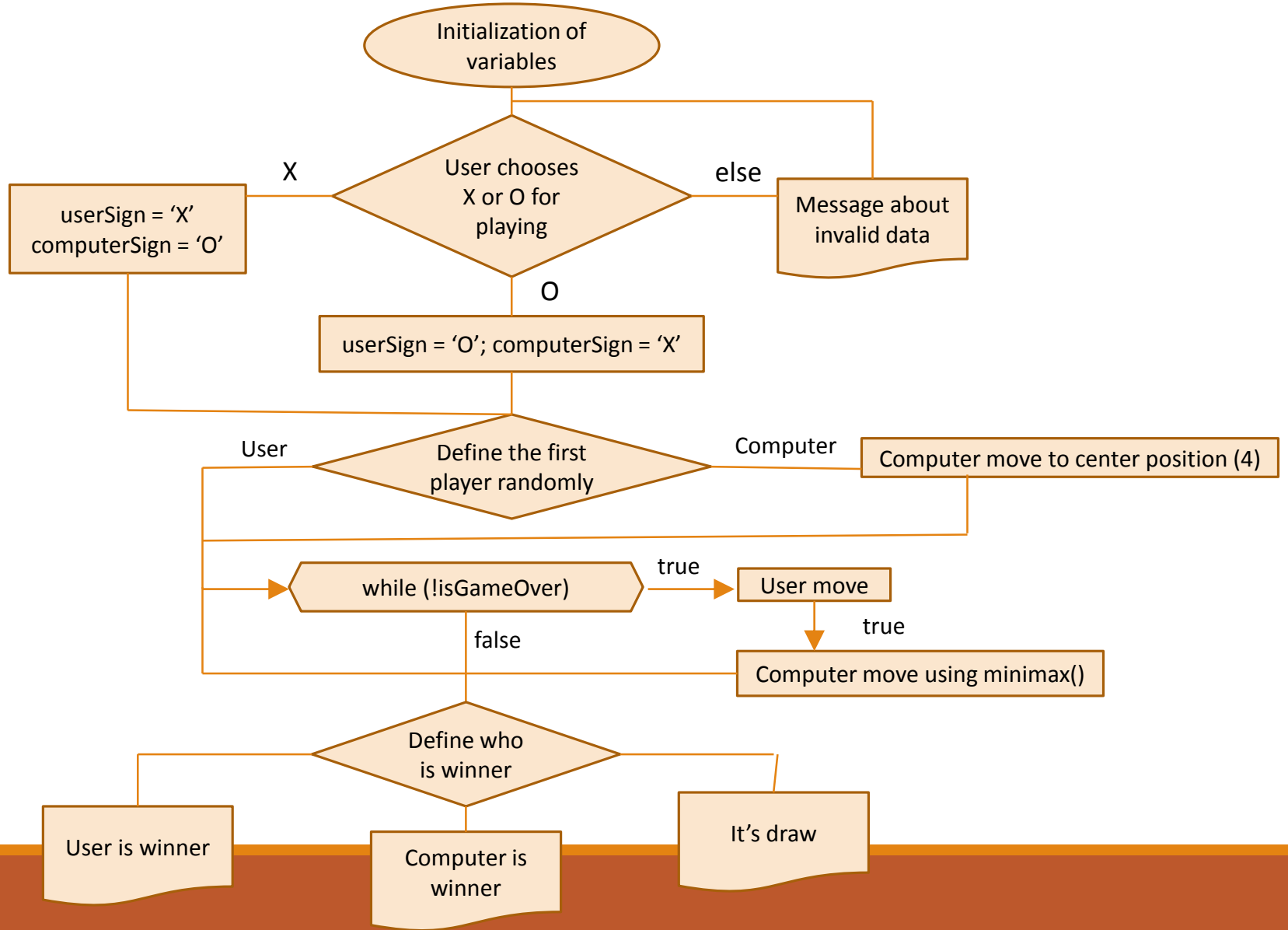




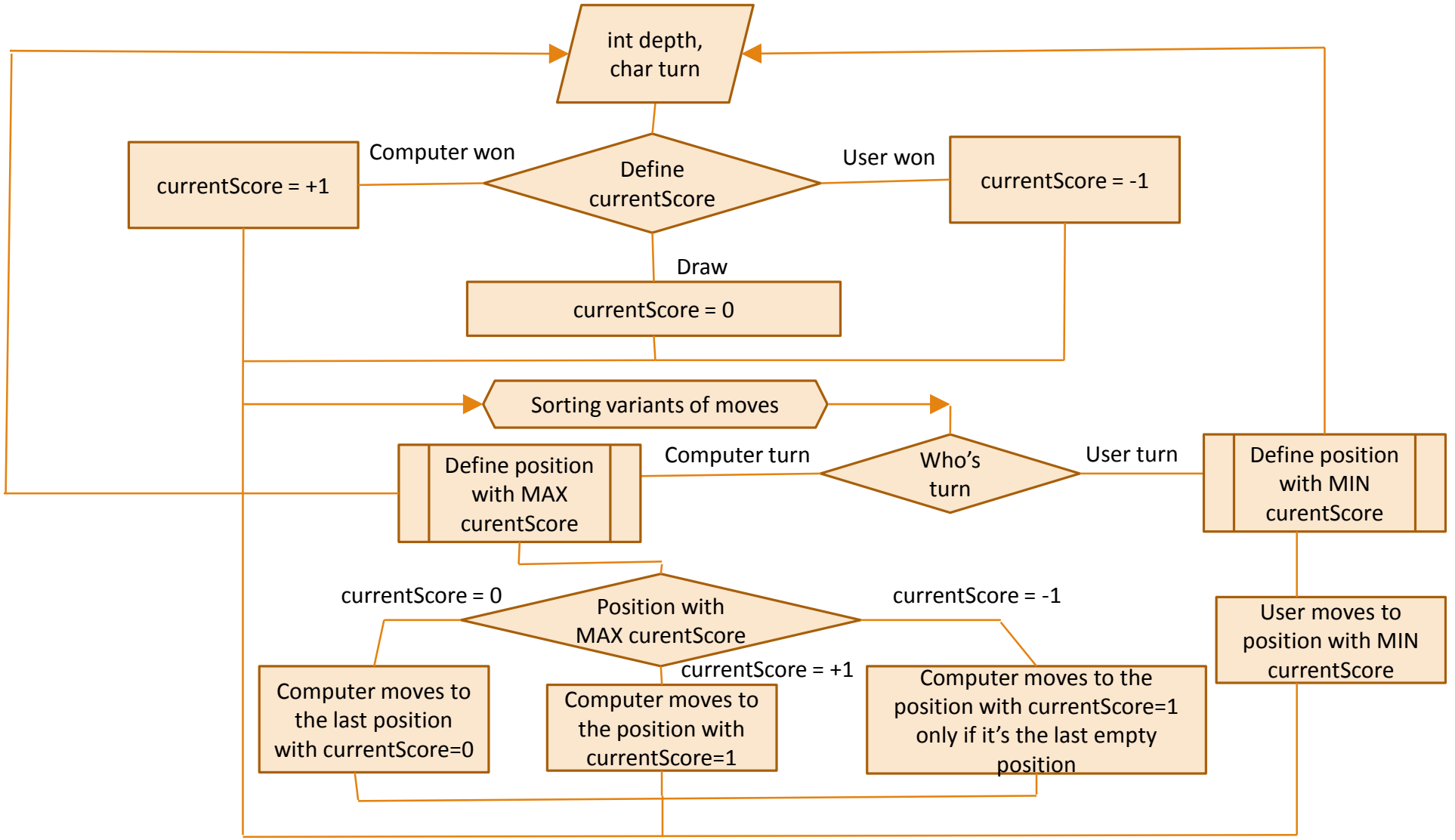
Sequence Class Diagram



Algorithm TicTacToe



Algorithm minimax()



Code: Class TicTacToe (main)

```
import java.util.Random;
import java.util.Scanner;


public class TicTacToe {
    public static void main(String[] args) {
        final char SYMBOL_FOR_X = 'X';
        final char SYMBOL_FOR_0 = 'O';
        Board b = new Board();

        System.out.println("Hello, player! Do you want to play with cross (X) or zero (O)?");
        b.displayBoard();

        char computerSign = SYMBOL_FOR_0;
        boolean enterValidSymbol = false;
        char userSign;
        Scanner scan;

        do {
            scan = new Scanner(System.in);
            userSign = scan.next().charAt(0);


            if (userSign == 'X' || userSign == 'x') {
                System.out.println("You'll play with X!");
                userSign = SYMBOL_FOR_X;
                computerSign = SYMBOL_FOR_0;
                enterValidSymbol = true;
            }
            else if (userSign == 'O' || userSign == 'o' || userSign == '0') {
                System.out.println("You'll play with 0!");
                userSign = SYMBOL_FOR_0;
                computerSign = SYMBOL_FOR_X;
                enterValidSymbol = true;
            } else {
                System.out.println("This move at (" + userSign
                        + ") is not valid. Try again...");

            }
        } while (!enterValidSymbol);
```

Code: Class TicTacToe (main)

```
Board.computerSign = computerSign;
Board.userSign = userSign;
String[] playersName = {"YOU", "COMPUTER"};
Random rand = new Random();
int firstPlayer = rand.nextInt(2);
System.out.println("The first move will be done by " + playersName[firstPlayer] + ".");
if (firstPlayer == 1){
    b.placeAMove(4, computerSign);
    b.displayBoard();
}
while (!b.isGameOver()) {
            System.out.println("Your move: ");
            int userMove;
            try {
                userMove = scan.nextInt();

                if ((userMove < 0) || (userMove > 8)){
                    throw new InputDataCheck(userMove);
                }
                if (b.board[userMove] == 0) {

                    b.placeAMove(userMove, userSign);
                    b.displayBoard();
                    if (b.isGameOver()) {break;}
                    b.minimax(0, computerSign);
                    b.placeAMove(Board.computerMove, computerSign);
                    b.displayBoard();
                } else {
                    System.out.println("This position isn't empty! Choose another one, please.");
                }
            }
            catch (InputDataCheck ex){
                System.out.println(ex.getMessage());
                scan.nextLine();
            }
            catch (Exception ex){
                System.out.println("Exception was handled. Enter number of position from 0 to 8, please.");
                scan.nextLine();
            }
```

Code: Class TicTacToe (main)

```
if (((b.hasXWon()) && (computerSign == SYMBOL_FOR_X)) || ((b.hasOWon()) && (computerSign == SYMBOL_FOR_0)))
    System.out.println("Unfortunately, you lost!");
else if (((b.hasXWon()) && (computerSign == SYMBOL_FOR_0)) || ((b.hasOWon()) && (computerSign == SYMBOL_FOR_X)))
    System.out.println("You win!");
else
    System.out.println("It's a draw!");
```

Code: Class Board

```
public class Board {
    final char SYMBOL_FOR_X = 'X';
    final char SYMBOL_FOR_O = 'O';
    char[] board = new char[9];
    public Board() {
    }
    public boolean isGameOver() {
        if (hasXWon() || hasOWon() || getAvailableStates().isEmpty()) {
            return true;
        }
        return false;
    }
    public boolean hasXWon() {
        if (((board[0] == board[4]) && (board[0] == board[8]) && (board[0] == SYMBOL_FOR_X))
                || ((board[2] == board[4]) && (board[2] == board[6]) && (board[2] == SYMBOL_FOR_X))) {
            return true;
        }
        for (int i = 0; i < 3; ++i) {
            if ((board[i] == board[i+3] && board[i] == board[i+6] && board[i] == SYMBOL_FOR_X)
                    || (board[3*i] == board[3*i+1] && board[3*i] == board[3*i+2] && board[3*i] == SYMBOL_FOR_X))
                    {
                return true;
            }
        }
        return false;
    }
    public boolean hasOWon() {
        if ((board[0] == board[4] && board[0] == board[8] && board[0] == SYMBOL_FOR_O)
                || (board[2] == board[4] && board[2] == board[6] && board[2] == SYMBOL_FOR_O)) {
            return true;
        }
        for (int i = 0; i < 3; ++i) {
            if ((board[i] == board[i+3] && board[i] == board[i+6] && board[i] == SYMBOL_FOR_O)
                    || (board[3*i] == board[3*i+1] && board[3*i] == board[3*i+2] && board[3*i] == SYMBOL_FOR_O))
            {
                return true;
            }
        }
        return false;
    }
```

Code: Class Board

```
public List<Integer> getAvailableStates() {
    List<Integer> availablePoints = new ArrayList<>();
    for (int i = 0; i < 9; ++i) {
            if (board[i] == 0) {
                availablePoints.add(i);
            }
        }
    return availablePoints;
}
public void placeAMove(int i, char player) { board[i] = player; }
public void displayBoard() {
    System.out.println();
    processBoard(board);
    System.out.println("|-----------|");
    System.out.println(String.format("| %s | %s | %s |", board[0], board[1], board[2]));
    System.out.println("|-----------|");
    System.out.println(String.format("| %s | %s | %s |", board[3], board[4], board[5]));
    System.out.println("|-----------|");
    System.out.println(String.format("| %s | %s | %s |", board[6], board[7], board[8]));
    System.out.println("|-----------|");
}
public static void processBoard(char[] board){
    final char SYMBOL_FOR_X = 'X';
    final char SYMBOL_FOR_0 = 'O';

    for(int i = 0; i < board.length; i++){
        if(board[i] == 'X' || board[i] == 'x'){
            board[i] = SYMBOL_FOR_X;
        }
        else if(board[i] == 'o' || board[i] == '0' || board[i] == 'O'){
            board[i] = SYMBOL_FOR_0;
        }
    }
}

private void print(String str) { System.out.println(str); }
```

Code: Class Board

```
public int minimax (int depth, char turn) {
    if (computerSign == SYMBOL_FOR_X) {
        if (hasXWon()) return +1;
        if (hasOWon()) return -1;
    }
    if (computerSign == SYMBOL_FOR_O) {
        if (hasOWon()) return +1;
        if (hasXWon()) return -1;
    }
    int next_depth = depth-1;
    List<Integer> pointsAvailable = getAvailableStates();
    if (pointsAvailable.isEmpty()) return 0;
    int min = Integer.MAX_VALUE, max = Integer.MIN_VALUE;

    for (int i = 0; i < pointsAvailable.size(); ++i) {
        if (turn == computerSign) {
            placeAMove(pointsAvailable.get(i), computerSign);
            int currentScore = minimax(next_depth, userSign);
            max = Math.max(currentScore, max);
            if(currentScore >= 0) {
                if(depth == 0) {
                    print("Score for position "+pointsAvailable.get(i)+ " = "+currentScore);
                    computerMove = pointsAvailable.get(i);
                }
            }
            if(currentScore == 1) {
                board[pointsAvailable.get(i)] = 0; break;
            }
            if(i == pointsAvailable.size()-1 && max < 0) {
                if(depth == 0) {
                    print("Score for position "+pointsAvailable.get(i)+ " = "+currentScore);
                    computerMove = pointsAvailable.get(i);
                }
            }
        } else {
                placeAMove(pointsAvailable.get(i), userSign);
                int currentScore = minimax(next_depth, computerSign);
                min = Math.min(currentScore, min);
        }
        board[pointsAvailable.get(i)]= 0;
    }
    return turn == computerSign ?max:min; //если ход компьютера, то возвращает максимум, если поль
```

Tests

```
public class BoardTest {

    @Test
    public void testHasXWon() {
        char[] board = { 'X', 'O', 0, 'X', 0, 'O', 'X', 'O', 0 };

        Board testBoard = new Board(board);

        boolean result = testBoard.hasXWon();

        Assert.assertEquals(result, true);

    }

    @Test
    public void testHasNotXWon() {
        char[] board = { 'X', 'O', 0, 'X', 0, 'O', 'O', 'X', 0 };

        Board testBoard = new Board(board);

        boolean result = testBoard.hasXWon();

        Assert.assertEquals(result, false);

    }

    @Test
    public void testIsGameOver() {
        char[] board = { 'X', 'O', 'O', 'O', 'X', 'X', 'X', 'O', 'O' };

        Board testBoard = new Board(board);

        boolean result = testBoard.isGameOver();

        Assert.assertEquals(result, true);

    }
```

List of information sources

1. http://www3.ntu.edu.sg/home/ehchua/programming/java/javagame_tictactoe_ai.html#zz-1.5
2. http://www.codebytes.in/2014/08/minimax-algorithm-tic-tac-toe-ai-in.html
3. http://neverstopbuilding.com/minimax
4. http://shkolo.ru/blok-shema-algoritma/
5. http://stackoverflow.com/questions/17311797/how-to-generate-a-sequence-diagram-from-java-source-code