```
/* SELF ASSESSMENT
   1.  clearBoard:
Did I use the correct method definition?
Mark out of 5:        5
Comment:        Yes, I used the correct method definition and made the function work as intended.
Did I use loops to set each position to the BLANK character?
Mark out of 5:        5
Comment:        I used the loop to set all characters in the array to ' '.
   2.  printBoard
Did I use the correct method definition?
Mark out of 5:        5
Comment:        Yes, I used the correct method definition and made the function work as intended.
Did I loop through the array and prints out the board in a way that it looked like a board?
Mark out of 5:        5
Comment:        I made sure to print out something that resembled a board.
   3.  canMakeMove
Did I have the correct function definition and returned the correct item?
Mark out of 5:        5
Comment:        Yes, I used the correct method definition and returned the correct item while making sure the function work as intended.
Did I check if a specified location was BLANK?
Mark out of 5:        5
Comment:        I made sure the position was not taken by a character already.
   4.  makeMove
Did I have the correct function definition?
Mark out of 5:        5
Comment:        Yes, I used the correct method definition and made the function work as intended.
Did I set the  currentPlayerPiece in the specified location?
Mark out of 5:        5
Comment:        I placed the player piece to the position that the player requested.
   5.  isBoardFull
Did I have the correct function definition and returned the correct item?
Mark out of 5:        5
Comment:           Yes, I used the correct method definition and returned the correct item.
Did I loop through the board to check if there are any BLANK characters?
Mark out of 5:        5
Comment:        I made sure that there were no blank characters in the program.
   6.  winner
Did I have the correct function definition and returned the winning character
Mark out of 5:        5
Comment:        Yes, I used the correct method definition and returned the winning character.
Did I identify all possible horizontal, vertical and diagonal winners
Mark out of 15:        15
Comment: Through a series of if statements I was able to identify the winner if there was one.
   7.main
Did I create a board of size 3 by 3 and use the clearBoard method to set all the positions to the BLANK character ('  ')?
Mark out of 3:        3
Comments:        I sucsesfully did so.
Did I loop asking the user for a location until wither the board was full or there was a winner?
Mark out of 5:        5
Comments:        I did so and it worked fine.
Did I call all of the methods above?
Mark out of 5:        5
Comments:        I called all of the above methods.
Did I handle incorrect locations provided by the user (either occupied or invalid locations)?
Mark out of 3:        3
Comments:        Yes. Incorrect locations are provided with a error screen.
Did I switch the current player piece from cross to nought and vice versa after every valid move?
Mark out of 3:        3
Comments:        I did so successfully.
Did I display the winning player piece or a draw at the end of the game?
Mark out of 3:        3
Comments:        I displayed the winning piece after the game if there was a winner.

   8.  Overall
Is my code indented correctly?
Mark out of 3:        3
Comments:        All of my code is indented correctly.
Do my variable names and Constants (at least four of them) make sense?
Mark out of 3:        3
Comments:        I had 4 constants and all variable names made sense.
Do my variable names, method names and class name follow the Java coding standard
Mark out of 2:        2
Comments:        All of he above followed the java coding standard.
      Total Mark out of 100 (Add all the previous marks):  100
*/

import java.util.Scanner;

public class NoughtsAndCrosses {

        public static final int NUMBER_OF_ROWS=3;
        public static final int NUMBER_OF_COLOUMS=3;
        public static final char PLAYER_1='X';
        public static final char PLAYER_2='O';

        public static void clearBoard(char[][] board)
```

```
{
        int counter=0;
        for (int count = 0; count < NUMBER_OF_ROWS; count++)
        {
                while ( counter< NUMBER_OF_COLOUMS)
                {
                        board[count][counter] = 32;
                        counter++;
                }
                counter=0;
        }
}

public static void printBoard(char[][] board)
{
        int counter=0;
        System.out.println("  1   2   3");
        for (int count = 0; count < NUMBER_OF_ROWS; count++)
        {
                System.out.print((count+1)+" ");
                while (counter< NUMBER_OF_COLOUMS)
                {
                        System.out.print(" "+board[count][counter]+" ");
                        counter++;
                }
                System.out.println("\n");
                counter=0;
        }
}

public static boolean canMakeMove (int row, int coloum, char board[][])
{
        if (board[coloum][row] == 32)
        {
                return true;
        }
        else
        {
                return false;
        }
}

public static void makeMove(char[][] board, char currentPlayerPiece, int row, int coloum)
{
        board[coloum][row]=currentPlayerPiece;
}

public static boolean isBoardFull ( char[][] board)
{
        int counter=0;
        for (int count = 0; count < NUMBER_OF_ROWS; count++)
        {
                while ( counter< NUMBER_OF_COLOUMS)
                {
                        if (board[counter][count] == 32)
                        {
                                return false;
                        }
                        counter++;
                }
                counter=0;
        }
        return true;
}

public static char winner (char[][] board)
{
        if ((board[0][0]==board[0][1]) && (board[0][0] == board[0][2])        && board[0][0] != 32)
        {
                return board[0][0];
        }
        else if ((board[1][0]==board[1][1]) && (board[1][0] == board[1][2]) && board[1][0] != 32)
        {
                return board[1][0];
        }
        else if ((board[2][0]==board[2][1]) && (board[2][0] == board[2][2])        && board[2][0] != 32)
        {
                return board[2][0];
        }

        else if ((board[0][0]==board[1][0]) && (board[0][0] == board[2][0])        && board[0][0] != 32)
        {
                return board[0][0];
        }
        else if ((board[0][1]==board[1][1]) && (board[0][1] == board[2][1])        && board[0][1] != 32)
        {
                return board[0][1];
```

```
		}
		else if ((board[0][2]==board[1][2]) && (board[0][2] == board[2][2])		&& board[0][2] != 32)
		{
			return board[0][2];
		}

		else if ((board[0][0]==board[1][1]) && (board[0][0] == board[2][2])		&& board[0][0] != 32)
		{
			return board[0][0];
		}
		else if ((board[0][2]==board[1][1]) && (board[0][2] == board[2][0])		&& board[0][2] != 32)
		{
			return board[0][2];
		}
		else
		{
			return 32;
		}
	}

	public static void main(String[] args)
	{
		int row=0;
		int coloum=0;
		char currentPlayerPiece = PLAYER_1;
		boolean gameFinished=false;

		Scanner input = new Scanner (System.in);
		char[][] board = new char[3][3];
		char[][] lastInvalidMove = new char[3][3];
		clearBoard( board);

		while(!gameFinished)
		{
			System.out.print("What row do you wish to go in? Choose from 1 to 3.");
			row=-1 + input.nextInt();
			System.out.print("What coloum do you wish to go in? Choose from 1 to 3.");
			coloum=-1 + input.nextInt();
			if ( row>2 || row<0 || coloum>2 || coloum<0)
			{
				System.out.println("One of the numbers you put in is invalid. Please try again.");
				gameFinished=true;
			}
			boolean canMakeMove=canMakeMove(row, coloum, board);
			if (!canMakeMove)
			{
				lastInvalidMove[coloum][row]=32;
				System.out.println("The space you wished to input was invalid as it was already filled. Please try a different
spot.");
			}
			else
			{
				makeMove(board, currentPlayerPiece, row, coloum);
				char whoWon=winner(board);
				if ( whoWon != 32)
				{
					System.out.println(whoWon+" won the game.");
					gameFinished=true;
				}
				boolean isBoardFull = isBoardFull(board);
				if (isBoardFull)
				{
					System.out.println("There was no winner.");
					gameFinished=true;
				}
				if (currentPlayerPiece == PLAYER_1)
				{
					currentPlayerPiece = PLAYER_2;
				}
				else
				{
					currentPlayerPiece = PLAYER_1;
				}
				printBoard(board);
			}
		}
		input.close();
	}


}
```