Player Class

```
package othello;

public class Player {

private String name;
private char symbol;

// Count of win games.
private int winGame;

// Default Constructor
public Player() {
	name = null;
	symbol = '\0';
	winGame = 0;
}

// Parameterised Constructor for Creating Player
public Player(String name, char symbol) {
	setName(name);
	setSymbol(symbol);
}

// Function to set name.
public void setName(String name) {
	if (!name.isEmpty()) {
		this.name = name;
	}
}

// Function to set symbol.
public void setSymbol(char symbol) {
	if (symbol != '\0') {
		this.symbol = symbol;
	}
}

// Function to get name.
public String getName() {
	return this.name;
}
// Function to get symbol.
public char getSymbol() {
        return this.symbol;
}

// Function to set count of win games.
public void setWinGames() {
        this.winGame++;
}

// Function to get count of win games.
public int getWinGames() {
        return this.winGame;
}

}
```

Board Class

```
package othello;
import java.util.ArrayList;
import javafx.util.Pair;

public class Board {
        private char board[][];
        private final int BOARD_SIZE = 8;
        private char p1Symbol, p2Symbol;

        // Count for Total Moves
        private int totalMoves;

        // Check for the Board is Full?????
        public boolean completeGame() {
                if (totalMoves == BOARD_SIZE * BOARD_SIZE) {
                        return true;
                }
                return false;
        }

        // Calculating number of Valid Moves.....
        public int noOfValidMoves(char symbol) {
                ArrayList<Pair<Integer, Integer>> noOfMoves = validMoves(symbol);
                return noOfMoves.size();
        }
```

```
// Constructor for creating Othello Board.....
public Board(char p1Symbol, char p2Symbol) {
        this.p1Symbol = p1Symbol;
        this.p2Symbol = p2Symbol;
        this.board = new char[BOARD_SIZE][BOARD_SIZE];
        for (int i = 0; i < BOARD_SIZE; i++) {
                for (int j = 0; j < BOARD_SIZE; j++) {
                        board[i][j] = ' ';
                }
        }
        board[3][3] = p1Symbol;
        board[3][4] = p2Symbol;
        board[4][3] = p2Symbol;
        board[4][4] = p1Symbol;
        totalMoves += 4;
}

// Displaying Current Board.....
public void printBoard() {
        String hline = "  +---+---+---+---+---+---+---+---+";
        System.out.println("X|Y 0   1   2   3   4   5   6   7");
        System.out.println(hline);
        for (int i = 0; i < BOARD_SIZE; i++) {
                System.out.print("" + i + " ");
                for (int j = 0; j < BOARD_SIZE; j++) {
                        System.out.print("| " + board[i][j] + " ");
                }
                System.out.println('|');
                System.out.println(hline);
        }
}

// Constructing list of Valid Moves.....
public ArrayList<Pair<Integer, Integer>> validMoves(char symbol) {
        ArrayList<Pair<Integer, Integer>> list = new ArrayList<>();
        for (int i = 0; i < BOARD_SIZE; i++) {
                for (int j = 0; j < BOARD_SIZE; j++) {
                        if (checkMove(symbol, i, j)) {
                                Pair<Integer, Integer> pair = new Pair<>(i, j);
                                list.add(pair);
                        }
                }
        }
return list;
```

```
}

// Checking and performing move......
public boolean move(char symbol, int x, int y) {
if (x < 0 || x >= BOARD_SIZE || y < 0 || y >= BOARD_SIZE || board[x][y] != ' ') {
        return false;
}
boolean ans = false;

// Array for movement for X
int[] xDir = { -1, -1, 0, 1, 1, 1, 0, -1 };

// Array for movement for Y
int[] yDir = { 0, 1, 1, 1, 0, -1, -1, -1 };

for (int i = 0; i < xDir.length; i++) {
        int xstep = xDir[i];
        int ystep = yDir[i];
        int xnew = x + xstep;
        int ynew = y + ystep;
        int count = 0;
        while (xnew >= 0 && xnew < 8 && ynew >= 0 && ynew < 8) {
                // empty cell
                if (board[xnew][ynew] == ' ') {
                        break;
                }
                else if (board[xnew][ynew] != symbol) {
                        xnew += xstep;
                        ynew += ystep;
                        count++;
                }

                // conversion is possible
                else {
                        if (count > 0) {
                        ans = true;
                        int convertX = xnew - xstep;
                        int convertY = ynew - ystep;
                        while (convertX != x || convertY != y) {
                                board[convertX][convertY] = symbol;
                                convertX -= xstep;
                                convertY -= ystep;
                        }
                }
                break;
        }
    }
}
if (ans) {
        board[x][y] = symbol;
        totalMoves++;
}

        return ans;
}

// helper function to generate list of Valid Moves.
public boolean checkMove(char symbol, int x, int y) {
        if (x < 0 || x >= BOARD_SIZE || y < 0 || y >= BOARD_SIZE || board[x][y] != ' ') {
                return false;
        }
        boolean ans = false;
        int[] xDir = { -1, -1, 0, 1, 1, 1, 0, -1 };
        int[] yDir = { 0, 1, 1, 1, 0, -1, -1, -1 };
                for (int i = 0; i < xDir.length; i++)  {
                int xstep = xDir[i];
                int ystep = yDir[i];
                int xnew = x + xstep;
                int ynew = y + ystep;
                int count = 0;
                while (xnew >= 0 && xnew < 8 && ynew >= 0 && ynew < 8) {
                        // empty cell
                        if (board[xnew][ynew] == ' ') {
                            break;
                        }
                        else if (board[xnew][ynew] != symbol) {
                            xnew += xstep;
                            ynew += ystep;
                            count++;
                        }
                        else {
                            // Move is valid
                            if (count > 0)
                                ans = true;
                                break;
                            }
                        }
                }
                return ans;
}

// Calculating total no of given symbol in board.....
public int countsymbol(char symbol) {
        int ans = 0;
        for (int i = 0; i < BOARD_SIZE; i++) {
                for (int j = 0; j < BOARD_SIZE; j++) {
                        if (board[i][j] == symbol) {
                            ans++;
                        }
                }
        }
        return ans;
  }
}
```

OTHALLO

```
package othello;

import java.util.ArrayList;
import java.util.Scanner;
import javafx.util.Pair;

public class Othello {
        private Board board;
        private Player player1, player2;
        private static boolean anotherGame = true;

        // Count for Total Number of Games
        public static int No_of_Games = 0;

        // Count for Number of Draw Games
        public static int Draw =  0;

        static Scanner sc = new Scanner(System.in);

        public static void main(String[] args) {
                No_of_Games++;
                System.out.println("\t\t\t OTHELLO GAME : ");
                Othello o = new Othello();
                o.startGame();
                while (anotherGame) {
                    o.create_Board();
                No_of_Games++;
            }
            System.out.println("************THANKS FOR PLAYING ************");
        }


        public void startGame() {
            // It comprises of 3 steps:
            // 1. Taking Player Information. (Player take_player_info(PLAYER_NUMBER))
            // 2. Creating Board with the Given Symbols &Conduct Game. (create_Board())
            // 3. Display Score Board. (printScoreBoard())

            player1 = take_player_info(1);
            player2 = take_player_info(2);
            while (player1.getName().compareTo(player2.getName()) == 0) {
            System.out.println("Name already taken !! Choose another name for Player 2 !!");
            System.out.println("Enter Player 2's name :");
                player2.setName(sc.nextLine());
            }
            while (player1.getSymbol() == player2.getSymbol()) {
            System.out.println("Symbol already taken !! Choose another Symbol for Player 2 !!");
            System.out.println("Enter Player 2's Symbol :");
            player2.setSymbol(sc.nextLine().charAt(0));
            }
        }

        // Taking Private information for the Players
        private Player take_player_info(int n) {
            System.out.println("Enter Player " + n + "'s name :");
            String name = sc.nextLine();
            System.out.println("Enter Player " + n + "'s Symbol :");
            char symbol = sc.nextLine().charAt(0);
            Player p = new Player(name, symbol);
            return p;
        }

        public void create_Board() {
        // Create board
            board = new Board(player1.getSymbol(), player2.getSymbol());
            System.out.println();
            System.out.println("OTHELLO Board : 8 X 8 Board");
            board.printBoard();
```

```
// Conducting Game
// Check for Turn
boolean p1Turn = true;

// Check for valid Move
        boolean validMove;

// List containing Valid moves for Player 1
ArrayList<Pair<Integer, Integer>> validMovesP1 = board.validMoves(player1.getSymbol());

// List containing Valid moves for Player 1
ArrayList<Pair<Integer, Integer>> validMovesP2 = board.validMoves(player1.getSymbol());

int no_of_valid_Moves_P1 = validMovesP1.size();
int no_of_valid_Moves_P2 = validMovesP2.size();
while(!board.completeGame()&&(no_of_valid_Moves_P1>0||no_of_valid_Moves_P2>0)) {
        validMovesP1 = board.validMoves(player1.getSymbol());
        validMovesP2 = board.validMoves(player2.getSymbol());
        no_of_valid_Moves_P1 = validMovesP1.size();
        no_of_valid_Moves_P2 = validMovesP2.size();
        if (p1Turn) {
                // PLAYER 1

                // VALID MOVES EXIST.....
                if (no_of_valid_Moves_P1 > 0) {
                System.out.println("Player 1 : " + player1.getName() + "'s Turn :");
                System.out.println("Enter row & column (X,Y):");
                int x = sc.nextInt();
                int y = sc.nextInt();

                // Checking and Making move if valid.
                validMove = board.move(player1.getSymbol(), x, y);

                // VALID MOVES.....
                if (validMove == true) {
                board.printBoard();
                p1Turn = false;
                } else {

                // INVALID MOVES.....
                System.out.println("!!!!!! INVALID MOVE !!!!!");
                System.out.println("So,Do u need Hints ??? (Enter Y/N)");

                // NEED HINTS......
```

```
                    sc.nextLine();
                    char choice = sc.nextLine().charAt(0);
                    if (choice == 'Y' || choice == 'y') {
                              printHint(validMovesP1);
                    }
          }
}else {
          p1Turn = false;
          System.out.println("OOPS!!!!!No Valid Moves for Player 1 : "+player1.getName());
          }
    }
 else {
                    // PLAYER 2

                    // VALID MOVES EXIST.....
                    if (no_of_valid_Moves_P2 > 0) {
                    System.out.println("Player 2 : " + player2.getName() + "'s Turn :");
                    System.out.println("Enter row & column (X,Y):");
                    int x = sc.nextInt();
                    int y = sc.nextInt();

                    // Checking and Making move if valid.
                    validMove = board.move(player2.getSymbol(), x, y);

                    // VALID MOVES.....
                    if (validMove == true) {
                              board.printBoard();
                              p1Turn = true;
                    } else {

                    // INVALID MOVES.....
                    System.out.println("!!!!!! INVALID MOVE !!!!!");

                    // NEED HINTS......
                    sc.nextLine();
                    System.out.println("So,Do u need Hints ??? (Enter Y/N)");
                    char choice = sc.nextLine().charAt(0);
                    if (choice == 'Y' || choice == 'y') {
                              printHint(validMovesP2);
                    }
          }
} else {
          // VALID MOVES DOES NOT EXIST.....
          p1Turn = true;
```

```
            System.out.println("OOPS!!!!!No Valid Moves for Player 1 : "+player1.getName());
                        }
                  }
            }

      // GAME OVER
      System.out.println("!!!!! GAME OVER !!!!!");

      // Calculating Result........
      int p1 = board.countsymbol(player1.getSymbol());
      int p2 = board.countsymbol(player2.getSymbol());
      if (p1 > p2) {
            player1.setWinGames();
            System.out.println("PLAYER 1 " + player1.getName() + " WINS !!!!!!");
      } else if (p1 < p2) {
            player2.setWinGames();
            System.out.println("PLAYER 2 " + player2.getName() + " WINS !!!!!!");
      }else {
            Draw++;
            System.out.println("MATCH DRAW !!!!!");
      }
      printScoreBoard();
      System.out.println();
      sc.nextLine();

      // Another Game ????
      System.out.println("Want to play a new Game ? (Enter Y/N)");
      char choice = sc.nextLine().charAt(0);
      if (choice == 'N' || choice == 'n') {
            anotherGame = false;
         }
      }

      // Display Score Board
      Private void printScoreBoard() {
            System.out.println("\tSCORE BOARD");
            System.out.println("Total number of games = " + No_of_Games);
            System.out.println(player1.getName()+" won "+player1.getWinGames()+" times");
            System.out.println(player2.getName()+" won "+player2.getWinGames()+" times");
            System.out.println("Number of tied games = " + Draw);
      }

      // Printing the Calculated Valid Moves for the required Player
      private void printHint(ArrayList<Pair<Integer, Integer>> validMoves) {
            int i = 0;
            System.out.print("Moves : { ");
            while (i < validMoves.size()) {
            System.out.print("("+validMoves.get(i).getKey()+","+validMoves.get(i).getValue()+"
            )");
            i++;
            }
            System.out.println(" }");
        }
    }
```