package vier\_gewinnt;  
  
import static gdi.MakeItSimple.readInt;  
  
public class programm {  
  
 private static char[][] board;  
 private final static int WIDTH = 7;  
 private final static int HEIGHT = 6;  
 private final static char EMPTY = ' ';  
 private final static char PLAYER\_1 = 'X';  
 private final static char PLAYER\_2 = 'O';  
 private final static int TOKENS\_TO\_WIN = 4;  
   
 public static void main(String[] args) {  
 System.out.println("1. Spiel starten\n2. Programm beenden");  
 int userInput = -1;  
 boolean exit = true;  
   
 do {  
 userInput = readInt();  
   
 switch(userInput) {  
 case 1:  
 startGame();  
 break;  
 case 2:  
 exit = true;  
 break;  
 }  
 } while(!exit);  
 }  
   
 public static void startGame() {  
 init();  
 print();  
   
 char winner = EMPTY;  
 char currentPlayer = PLAYER\_1;  
 boolean drawn = false;  
 do {  
 makeTurn(currentPlayer);  
 currentPlayer = nextPlayer(currentPlayer);  
 winner = checkVictory();  
 drawn = checkDrawn();  
 } while(winner == EMPTY && !drawn);  
   
 if(winner != EMPTY) {  
 System.out.println(getPlayerName(winner) + " hat gewonnen.");  
 } else {  
 System.out.println("Unentschieden.");  
 }  
 }  
   
 public static char nextPlayer(char currentPlayer) {  
 if(currentPlayer == PLAYER\_1) {  
 currentPlayer = PLAYER\_2;  
 } else {  
 currentPlayer = PLAYER\_1;  
 }  
 return currentPlayer;  
 }  
   
 public static void makeTurn(char currentPlayer) {  
 int userColumn = -1;  
 boolean didTurn = false;  
   
 do {  
 String currentPlayerName = getPlayerName(currentPlayer);  
 System.out.println(currentPlayerName + " ist am zug.");  
 userColumn = readInt();  
 didTurn = putToken(userColumn, currentPlayer);  
  
 if (didTurn)  
 print();  
 else  
 System.out.println("Ungültige Spalte.");  
 } while(!didTurn);  
 }  
   
 public static String getPlayerName(char currentPlayer) {  
 String playerName = "";  
 if(currentPlayer == PLAYER\_1) {  
 playerName = "Spieler 1";  
 } else {  
 playerName = "Spieler 2";  
 }  
 return playerName;  
 }  
   
 public static void init() {  
 board = new char[WIDTH][HEIGHT];  
   
 for(int width = 0; width < WIDTH; width++) {  
 for(int height = 0; height < HEIGHT; height++) {  
 board[width][height] = EMPTY;  
 }  
 }  
 }  
   
 public static boolean putToken(int column, char token) {  
 boolean successful = false;  
 int topPosition = getTopPosition(column);  
   
 if(topPosition != -1) {  
 board[column-1][topPosition] = token;  
 successful = true;  
 } else {  
 successful = false;  
 }  
 return successful;  
 }  
   
 private static int getTopPosition(int column) {  
 int topPosition = -1;  
 boolean foundTopPosition = false;  
 boolean validColumn = column-1 >= 0 && column-1 < WIDTH;  
   
 if(validColumn) {  
 for(int height = HEIGHT - 1; !foundTopPosition && height >= 0; height--) {  
 if(board[column-1][height] == EMPTY) {  
 topPosition = height;  
 foundTopPosition = true;  
 }  
 }  
 }  
   
 return topPosition;  
 }  
   
 public static char checkVictory() {  
 for(int width = 0; width < WIDTH; width++) {  
 for(int height = 0; height < HEIGHT; height++) {  
 char currentToken = board[width][height];  
   
 char wonBottomLeft = checkBottomLeft(currentToken, width, height);  
 char wonBottom = checkBottom(currentToken, width, height);  
 char wonBottomRight = checkBottomRight(currentToken, width, height);  
 char wonRight = checkRight(currentToken, width, height);  
   
 if(wonBottomLeft != EMPTY || wonBottom != EMPTY || wonBottomRight != EMPTY || wonRight != EMPTY)  
 return currentToken;  
 }  
 }  
 return EMPTY;  
 }  
   
 private static char checkBottom(char token, int columnIndex, int heightIndex) {  
 int currentTokensInARow = 0;  
   
 do {  
 if(token == board[columnIndex][heightIndex]) {  
 heightIndex++;  
 currentTokensInARow++;  
 if(currentTokensInARow == TOKENS\_TO\_WIN)  
 return token;  
 } else {  
 return EMPTY;  
 }  
 } while(columnIndex < WIDTH && heightIndex < HEIGHT);  
   
 return EMPTY;  
 }  
   
 private static char checkRight(char token, int columnIndex, int heightIndex) {  
 int currentTokensInARow = 0;  
   
 do {  
 if(token == board[columnIndex][heightIndex]) {  
 columnIndex++;  
 currentTokensInARow++;  
 if(currentTokensInARow == TOKENS\_TO\_WIN)  
 return token;  
 } else {  
 return EMPTY;  
 }  
 } while(columnIndex < WIDTH && heightIndex < HEIGHT);  
   
 return EMPTY;  
 }  
   
 private static char checkBottomRight(char token, int columnIndex, int heightIndex) {  
 int currentTokensInARow = 0;  
   
 do {  
 if(token == board[columnIndex][heightIndex]) {  
 columnIndex++;  
 heightIndex++;  
 currentTokensInARow++;  
 if(currentTokensInARow == TOKENS\_TO\_WIN)  
 return token;  
 } else {  
 return EMPTY;  
 }  
 } while(columnIndex < WIDTH && heightIndex < HEIGHT);  
   
 return EMPTY;  
 }  
   
 private static char checkBottomLeft(char token, int columnIndex, int heightIndex) {  
 int currentTokensInARow = 0;  
   
 do {  
 if(token == board[columnIndex][heightIndex]) {  
 columnIndex--;  
 heightIndex++;  
 currentTokensInARow++;  
 if(currentTokensInARow == TOKENS\_TO\_WIN)  
 return token;  
 } else {  
 return EMPTY;  
 }  
 } while(columnIndex >= 0 && heightIndex < HEIGHT);  
   
 return EMPTY;  
 }  
   
 public static boolean checkDrawn() {  
 for(int width = 0; width < WIDTH; width++) {  
 if(board[width][0] == EMPTY)  
 return false;  
 }  
 return true;  
 }  
   
 public static void print() {  
 System.out.println("Board:");  
   
 for(int height = 0; height < HEIGHT; height++) {  
 System.out.print("|");  
 for(int width = 0; width < WIDTH; width++) {  
 System.out.print(board[width][height] + "|");  
 }  
 System.out.println();  
 }  
 }  
  
}