



Engenharia Informática

Programação I

2009/2010

Trabalho Prático



Trabalho Realizado Por:

Marlene Oliveira nº 25999

Pedro Mateus nº 26048

Índice

1. O Problema – Construir o Jogo Reversi em Java
 - a. Etapas de Resolução do Problema
 0. Classe Reversi e Método Estático “Main”
 - i. Menu da Entrada
 - ii. Tabuleiro Inicial
 - iii. Jogo Entre Humanos
 - iv. Tabuleiro Final
 2. Conclusão.
- Bibliografia

1. O Problema - Construir o Jogo Reversi em Java

No âmbito da disciplina de Programação I, foi-nos pedida a elaboração de um programa, usando classes do Java, que permita jogar uma versão em texto do jogo Reversi (ou Othello, como é muitas vezes designado). Para o efeito, criámos a classe Reversi.java que permitirá alcançar esse objectivo.

Segundo o enunciado, o programa deverá satisfazer os seguintes requisitos:

- i. Menu da Entrada: Para que se possa iniciar o jogo será necessário invocar o comando java Reversi e o programa deverá apresentar um menu em que seja possível escolher entre as opções "o – Sair" e "1 – Jogar". Caso seja escolhida a opção "o – Sair", o programa deve sair.
- ii. Tabuleiro inicial: Caso seja escolhida a opção "1 – Jogar", referida no ponto anterior, o programa deve perguntar qual é o tamanho do tabuleiro pretendido e, depois, deve inicializar um tabuleiro com as dimensões definidas anteriormente pelo utilizador.
- iii. Jogo entre humanos: O jogador que utiliza as peças X será o primeiro a começar (satisfazem-se assim as regras do jogo), o programa deverá disponibilizar as seguintes opções ao jogador: "o – Passar" e "1 – Jogar". Caso seja escolhida a opção que possibilita ao jogador passar a vez, o programa deverá alternar para o jogador que utiliza as peças o. Caso o jogador pretenda jogar, o programa deverá solicitar a escolha de um inteiro correspondente à linha e a escolha do carácter correspondente à coluna, o que, em conjunto, corresponderá ao movimento pretendido pelo jogador. Pede-se ainda que o programa impeça a ocorrência de fraude, ou seja, os jogadores não podem passar caso exista jogada possível.
- iv. Tabuleiro Final: Caso ambos os jogadores passem sucessivamente, o programa deverá ostentar o tabuleiro final com o número de peças de cada um dos jogadores, bem como o nome do vencedor.

Para que fosse mais fácil a leitura do código, decidimos distribuí-lo por várias classes. Assim, para além da classe Reversi.java, existem também as classes Contador.java; JogadasReversi.java; JogadoresReversi.java; MenuGameOver.java; TabuleiroReversi.java; ValidarO.java; ValidarX.java; VirarPeçasO.java e, finalmente, VirarPeçasX.java. Estas classes possuem o código necessário para o decorrer normal do jogo Reversi UE.

a. Etapas da Resolução do Problema

O. Classe Reversi e Método Estático "Main":

A classe Reversi.java contém o método Main, cuja descrição consta na tabela abaixo.

<pre>public static void main(String args[]){ Menu(); }</pre>	No método estático "main" é inicializado o método estático "Menu", que nos permitirá ter acesso ao menu da entrada.
--	---

i. Menu da Entrada:

O menu da entrada do jogo está contido dentro da classe Reversi.java. A sua descrição detalhada encontra-se na tabela que se segue.

<pre>public static void Menu(){ Scanner s = new Scanner(System.in); System.out.println("Reversi UE."); int escolha,N; do{ System.out.println("0-Sair 1-Jogar"); escolha=s.nextInt(); } while(escolha!=1&&escolha!=0); if(escolha==0) return; do{ System.out.println("Insira um número par que corresponda ao tamanho do tabuleiro:"); N=s.nextInt(); } while(N<4 N>26 N%2!=0); int [][]a = new int [N+2][N+2]; TabuleiroReversi.Tabuleiro(a); }</pre>	<p>No método “Menu” são dadas duas opções aos jogadores humanos. Os jogadores podem seleccionar uma destas opções usando um inteiro dado na linha de comandos (0 ou 1), o que corresponderá ao inteiro “escolha”.</p> <p>Caso o jogador escolha a opção “0 – Passar”, ou seja, se o valor do inteiro “escolha” for zero, o programa não retorna nada e sai.</p> <p>Caso o jogador escolha a opção “1 – Jogar”, ou seja, se o valor do inteiro “escolha” for um, o utilizador terá ainda de inserir o tamanho do tabuleiro, que corresponderá ao inteiro N e, de seguida o programa invoca o método “Tabuleiro”.</p> <p>NOTA: Existem duas restrições neste método (ambas implementadas com recurso à instrução <i>do-while</i>): na primeira restrição, o programa não permite que sejam inseridos na linha de comandos inteiros diferentes de zero e de um; na segunda restrição, o programa não permite que sejam inseridos números inteiros menores que quatro ou maiores que vinte e seis ou que não sejam pares (que não tenham o resto da divisão inteira por dois igual a zero).</p> <p>Neste método é também invocado o Tabuleiro de jogo, presente na classe TabuleiroReversi.</p>
--	--

ii. Tabuleiro Inicial:

A classe *TabuleiroReversi.java* contém os métodos que são necessários para escrever o tabuleiro do jogo. A descrição dos métodos encontra-se na tabela seguinte.

<pre>public static void Tabuleiro(int p[][]){ Tabuleiroinicial(p); Tabuleirofisico(p); JogadoresReversi.menujogadorX(p); }</pre>	<p>O método “Tabuleiro” inicializa os métodos “Tabuleiroinicial”, “Tabuleirofisico” e o método “menujogadorX” da classe <i>JogadoresReversi.java</i>. Na prática, este método serve apenas para inicializar o Tabuleiro como um todo.</p>
<pre>public static void Tabuleirofisico(int b [][]){ String [][]tabuleiro= new String [b.length][b.length]; JogadasReversi.Pecas(b,tabuleiro); Letras(b); Traçosvert(b); for(int linha =1; linha<b.length-1;linha++){ for(int coluna =1; coluna<b.length-1;coluna++){ if(coluna==1){ System.out.print(linha+ " "); } System.out.print(tabuleiro[linha] [coluna]+ " "); if(coluna==b.length-2) System.out.print(linha); } System.out.println(); Traçosvert(b); } Letras(b); System.out.println();</pre>	<p>Neste método são inicializados os métodos “Peças” (que consta na classe <i>JogadasReversi.java</i>), “Letras” e “TraçosVert”. De seguida o programa percorre o array bidimensional (através do ciclo for) e escreve a String “ ”, que será a linha que divide as colunas do tabuleiro. Depois disto, o método “TraçosVert” é inicializado e escreve os limites das linhas, ou seja, escreve a String “ -” que irá compor uma linha que limitará as diversas linhas do tabuleiro de jogo. O método “Letras” também é inicializado, o que permite que sejam escritos o topo e no final das colunas os caracteres que assinalam as coordenadas das mesmas. No final deste método, o programa escreve o número de peças de cada jogador (recorre aos métodos “ContarPeçasX” e “ContarPeças0”, que estão incluídos na classe <i>Contador.java</i>).</p>

<pre> System.out.println("Nº de peças X - "+Contador.ContadorPeçasX(b)+" Nº de peças O - "+Contador.ContadorPeçasO(b)); System.out.println(); } </pre>	
<pre> public static void Letras(int d[][]){ char []letras= {'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z'}; System.out.print(" "); for(int linha=0;linha<d.length;linha++) System.out.print(letras[linha]+ " "); System.out.println(); } </pre>	<p>Neste método é criado um array de caracteres onde constam todas as letras que compõem o alfabeto e que posteriormente é percorrido pelo ciclo <i>for</i>.</p> <p>No final do ciclo, o programa escreve os caracteres necessários para escrever as coordenadas das colunas do tabuleiro de jogo.</p>
<pre> public static void Traçosvert(int e[][]){ System.out.print(" "); for(int linha=0;linha<e.length;linha++) System.out.print("- "); System.out.println(); } </pre>	<p>O método “Traçosvert” percorre o array através do ciclo <i>for</i> e escreve os limites das linhas, ou seja, escreve a String “ -” que irá compor uma linha que limitará as diversas linhas do tabuleiro de jogo.</p>
<pre> public static void Tabuleiroinicial(int c[][]){ c[c.length/2][(c.length/2)-1]=1; c[(c.length/2)-1][c.length/2]=1; c[c.length/2][c.length/2]=2; c[(c.length/2)-1][(c.length/2)-1]=2; } </pre>	<p>Este método coloca os valores das peças iniciais, escritas através do método “Peças” (incluído na classe <i>JogadasReversi.java</i>) no centro do tabuleiro de jogo.</p> <p>(NOTA: Neste caso não é necessário percorrer o array, basta assinalar o local específico onde se irão localizar as peças iniciais.)</p>

iii. Jogo Entre Humanos

A classe *Contador.java* contém os métodos estáticos que contam as peças "X" e as peças "O", bem como o número de jogadas de cada um dos jogadores. O funcionamento de cada um destes métodos encontra-se descrito na tabela subsequente.

<pre>public static int ContadorPeçasX(int tabuleiro[][]){ int peçasX=0; for(int coluna=1;coluna<tabuleiro.length-1;coluna++) for(int linha=1;linha<tabuleiro.length-1;linha++){ if(tabuleiro[coluna][linha]==1) peçasX+=1; } return peçasX; }</pre>	Neste método estático o programa percorre o array de inteiros que constitui a base do tabuleiro de jogo e conta o número de peças "X", que são reconhecidas pelo programa como o algarismo 1. No final, o método retorna o número total de peças "X" que se encontram no tabuleiro.
<pre>public static int ContadorPeçasO(int tabuleiro[][]){ int peçasO=0; for(int coluna=1;coluna<tabuleiro.length-1;coluna++) for(int linha=1;linha<tabuleiro.length-1;linha++){ if(tabuleiro[coluna][linha]==2) peçasO+=1; } return peçasO; }</pre>	Neste método estático o programa percorre o array de inteiros que constitui a base do tabuleiro de jogo e conta o número de peças "O", que são reconhecidas pelo programa como o algarismo 2. No final, o método retorna o número total de peças "O" que se encontram no tabuleiro.
<pre>public static int ContadorPeçasTotal(int tabuleiro[][]){ int peçasTotal=0; for(int coluna=1;coluna<tabuleiro.length-1;coluna++) for(int linha=1;linha<tabuleiro.length-1;linha++){ if(tabuleiro[coluna][linha]!=0) peçasTotal+=1; } return peçasTotal; }</pre>	Neste método estático o programa percorre o array de inteiros que constitui a base do tabuleiro de jogo e conta o número de peças de cada um dos jogadores, que são reconhecidas pelo programa como os algarismos 1 e 2. No final, o método retorna o número total de peças que se encontram no tabuleiro.

<pre> public static int ContarJogadasX(int tabuleiro[][]){ int contador=0; for(int a=1;a<tabuleiro.length-1;a++){ for(int b=1;b<tabuleiro.length-1;b++){ if(JogadasReversi.JogadaX(tabuleiro, a, b)==true) contador+=1; } } return contador; } </pre>	<p>O programa percorre todo o array de inteiros que constitui a base do tabuleiro de jogo e conta as jogadas validadas pelo método JogadaX que se encontra na classe JogadasReversi.java. Finalmente, o método retorna o número de jogadas possíveis do jogador X.</p>
<pre> public static int ContarJogadasO(int tabuleiro[][]){ int contador=0; for(int a=1;a<tabuleiro.length-1;a++){ for(int b=1;b<tabuleiro.length-1;b++){ if(JogadasReversi.JogadaO(tabuleiro, a, b)==true) contador+=1; } } return contador; } </pre>	<p>O programa percorre todo o array de inteiros que constitui a base do tabuleiro de jogo e conta as jogadas validadas pelo método JogadaO que se encontra na classe JogadasReversi.java. Finalmente, o método retorna o número de jogadas possíveis do jogador O.</p>

Os métodos que contêm os menus de cada um dos jogadores encontram-se na classe JogadoresReversi.java. O seu funcionamento é explicado na tabela seguinte.

<pre> public static void menujogadorX(int x []){ Scanner b= new Scanner(System.in); int escolhaX; do{ System.out.println("Jogador X pretende: 0: Passar; 1: Jogar;"); escolhaX = b.nextInt(); } while(escolhaX!=1 && escolhaX!=0); if (escolhaX==0){ menujogador0(x); } } </pre>	<p>Tal como o nome indica, este método representa o menu do primeiro jogador que, segundo as regras do Reversi, é o primeiro a jogar. Neste caso, designamos este jogador por “jogadorX”. Quando é inicializado, este método apresenta um pequeno menu, que permite ao jogadorX decidir se quer jogar ou se prefere passar a jogada. Caso o jogador decida passar a jogada, o programa invoca o menu do jogador contrário (neste caso será o jogador 0). Se o jogador escolher a opção “1: Jogar”, o programa pede ao jogador para seleccionar</p>
---	--

<pre> } else{ int linha1,coluna1; do{ System.out.println("Insira a linha entre 1 e "+(x.length-2)+ " em que pretende jogar:"); linha1 = b.nextInt(); System.out.println("Insira uma coluna entre a e "+ JogadasReversi.Abc(x.length-2)+ " em que pretende jogar:"); char coluna1Temp = (b.next()).charAt(0); coluna1=JogadasReversi.AbcInverso(coluna1Temp); }while(JogadasReversi.JogadaX(x, linha1, coluna1)==false); TabuleiroReversi.Tabuleirofisico(x); if(Contador.ContadorPeçasTotal(x)==(x.length-2)*(x.length- 2) Contador.ContadorPeçasO(x)==0) MenuGameOver.Vencedor(x); else menujogador0(x); } </pre>	<p>uma linha, introduzindo um inteiro, e uma coluna, introduzindo um carácter (que, posteriormente será convertido em inteiro). A linha e a coluna corresponderão à coordenada da jogada pretendida. De seguida, o programa valida a jogada. Caso esta seja válida, o programa escreve o tabuleiro com a peça no local pretendido e vira peças do adversário (caso seja possível). Se a jogada não for válida, o programa continua a pedir a linha e a coluna onde o jogador pretende jogar. Depois de escrever o tabuleiro com a jogada e de virar as peças (caso isso seja permitido) o programa conta as peças de cada jogador e escreve o resultado final do jogo e o nome do vencedor, isto, se o número de peças total for igual ao número total de espaços do tabuleiro. Caso ainda existam espaços livres, o programa conta as peças e invoca o método do outro jogador.</p> <p>NOTA: Existem duas restrições neste método (ambas implementadas com recurso à instrução <i>do-while</i>): na primeira restrição, o programa não permite que sejam inseridos na linha de comandos inteiros diferentes de zero e de um; na segunda restrição, o programa não permite que sejam inseridas jogadas que não sejam válidas (para verificar se a jogada é válida, o programa invoca o método “JogadaX” da classe JogadasReversi.java).</p>
<pre> public static void menujogador0(int z[][]){ Scanner t= new Scanner(System.in); </pre>	<p>Este método funciona de uma forma idêntica à do método que contém menu do jogador X.</p>

<pre> int escolha0; do{ System.out.println("Jogador 0 pretende: 0: Passar; 1: Jogar;"); escolha0 = t.nextInt(); } while(escolha0!=1&&escolha0!=0); if ((escolha0)==0){ menujogadorX(z); } else{ int linha2,coluna2; do{ System.out.println("Insira a linha entre 1 e "+(z.length-2)+" em que pretende jogar:"); linha2 = t.nextInt(); System.out.println("Insira uma coluna entre a e "+ JogadasReversi.Abc(z.length-2)+" em que pretende jogar:"); char coluna2Temp=(t.next()).charAt(0); coluna2=JogadasReversi.AbcInverso(coluna2Temp); }while(JogadasReversi.JogadaO(z, linha2, coluna2)==false); TabuleiroReversi.Tabuleirofisico(z); if(Contador.ContadorPeçasTotal(z)==(z.length-2)*(z.length- 2) Contador.ContadorPeçasX(z)==0) MenuGameOver.Vencedor(z); else menujogadorX(z); } } </pre>	<p>Tal como no método do menu do jogador X, este método permite ao jogador 0 passar a jogada ou jogar Reversi (caso opte pela opção “1: Jogar”) e conta as peças de cada jogador depois de escrever o tabuleiro de jogo com as peças e de validar a jogada.</p> <p>Tal como no método “menujogadorX”, o programa também termina o jogo se não existirem mais jogadas possíveis e escreve o nome do vencedor.</p> <p>Caso ainda existam jogadas possíveis, o programa invoca o método do jogador X.</p> <p>NOTA: Existem duas restrições neste método (ambas implementadas com recurso à instrução <i>do-while</i>): na primeira restrição, o programa não permite que sejam inseridos na linha de comandos inteiros diferentes de zero e de um; na segunda restrição, o programa não permite que sejam inseridas jogadas que não sejam válidas (para verificar se a jogada é válida, o programa invoca o método “Jogada0” da classe JogadasReversi.java).</p>
--	--

A classe *JogadasReversi.java* contém os métodos que são necessários para escrever as jogadas de cada um dos jogadores e para validar as mesmas. Para além disto, nesta classe estão também contidos os métodos que reconhecem o carácter dado pelo jogador na linha de comandos e que o convertem num inteiro. A descrição detalhada do funcionamento de cada um destes métodos está presente na tabela seguinte.

<pre>public static void Peças(int s[], String w[]){ for(int i=0;i<s.length;i++) for(int j=0;j<s.length;j++){ w[i][j]=" "; if(s[i][j]==1) w[i][j]="X"; if(s[i][j]==2) w[i][j]="O"; } } }</pre>	<p>Neste método, o programa percorre o array que constitui a base do tabuleiro e escreve as peças do jogo.</p> <p>(NOTA: Este método não coloca as quatro peças iniciais no centro do tabuleiro.)</p>
<pre>public static char Abc(int a){ char []letras= {'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z'}; char letra; letra=letras[a-1]; return letra; } }</pre>	<p>Este método tem como finalidade reconhecer o número do tamanho do tabuleiro e converte-lo em carácter, para que o utilizador possa saber qual o limite máximo, conforme a sequência do abecedário, da letra que poderá seleccionar.</p>
<pre>public static int AbcInverso(char a){ char []letras= {'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z'}; int numero=0; for(int coluna=0;coluna<letras.length;coluna++){ numero+=1; if(a==letras[coluna]) break; } }</pre>	<p>O método “AbcInverso” converte os caracteres dados pelos jogadores na linha de comandos em inteiros, que são usados pelo programa para assinalar o local onde deverá ser colocada a jogada. Isto é, o programa identifica no array “letras” o carácter dado pelo utilizador na linha de comandos e retorna o número correspondente à posição desse carácter. De seguida, o programa utiliza</p>

<pre> } return numero; } </pre>	<p>este inteiro que irá corresponder à linha da posição da peça no tabuleiro.</p>
<pre> public static boolean JogadaX(int tab[],int linha, int coluna){ boolean validoX=false; if(tab[linha][coluna]==0){ for(int a=linha-1;a<=linha+1;a++){ for(int b=coluna-1;b<=coluna+1;b++){ if(tab[a][b]==2){ if(a==linha-1&&b==coluna) if(ValidarX.VerticalCima(a, b, tab)==true){ tab[linha][coluna]=1; VirarPeçasX.Vertical(a, b, tab); validoX=true; } if(a==linha+1&&b==coluna) if((ValidarX.VerticalBaixo(a, b, tab))==true){ tab[linha][coluna]=1; VirarPeçasX.Vertical(a, b, tab); validoX=true; } if(a==linha&&b==coluna-1) if(ValidarX.HorizontalEsq(a, b, tab)==true){ tab[linha][coluna]=1; VirarPeçasX.Horizontal(a, b, tab); validoX=true; } if(a==linha&&b==coluna+1) if(ValidarX.HorizontalDir(a, b, tab)==true){ tab[linha][coluna]=1; </pre>	<p>Este método permite-nos validar a jogada X, colocá-la no local especificado pelo jogador X e virar peças “0”.</p> <p>Caso o espaço assinalado pelo jogador esteja vazio, o programa percorre o array que compõe o tabuleiro e valida a jogada. O programa começa por verificar em que direcção é que se encontra as peças adversárias (Vertical Cima, Vertical Baixo, Horizontal Esquerda, Horizontal Direita, Diagonal Esquerda Cima, Diagonal Esquerda Baixo, Diagonal Direita Cima e Diagonal Direita Baixo) depois irá validar essa jogada nas direcções onde se encontram as peças adversárias, se a jogada for válida pelo menos numa dessas direcções, o programa escreve a peça “X” no local assinalado e inicia o método de virar as peças na direcção validada.</p> <p>NOTA: A peça “X” é reconhecida pelo programa no array de inteiros sempre que existe um algarismo 1 num espaço do tabuleiro.</p>

```

        VirarPeçasX.Horizontal(a, b, tab);

        validoX=true;
    }

    if(a==linha-1&&b==coluna-1)

        if(ValidarX.DiagonalEsqCima(a, b, tab)==true){

            tab[linha][coluna]=1;

            VirarPeçasX.DiagonalCimaBaixo(a, b, tab);

            validoX=true;
        }

    if(a==linha+1&&b==coluna-1)

        if(ValidarX.DiagonalEsqBaixo(a, b, tab)==true){

            tab[linha][coluna]=1;

            VirarPeçasX.DiagonalBaixoCima(a, b, tab);

            validoX=true;
        }

    if(a==linha-1&&b==coluna+1)

        if(ValidarX.DiagonalDirCima(a, b, tab)==true){

            tab[linha][coluna]=1;

            VirarPeçasX.DiagonalBaixoCima(a, b, tab);

            validoX=true;
        }

    if(a==linha+1&&b==coluna+1)

        if(ValidarX.DiagonalDirBaixo(a, b, tab)==true){

            tab[linha][coluna]=1;

            VirarPeçasX.DiagonalCimaBaixo(a, b, tab);

            validoX=true;
        }
    }
}
}
return validoX;
}

```

<pre> public static boolean JogadaO(int tab[],int linha, int coluna){ boolean validoO=false; if(tab[linha][coluna]==0){ for(int a=linha-1;a<=linha+1;a++) for(int b=coluna-1;b<=coluna+1;b++){ if(tab[a][b]==1){ if(a==linha-1&&b==coluna) if(ValidarO.VerticalCima(a, b, tab)==true){ tab[linha][coluna]=1; VirarPeçasO.Vertical(a, b, tab); validoO=true; } if(a==linha+1&&b==coluna) if((ValidarO.VerticalBaixo(a, b, tab))==true){ tab[linha][coluna]=2; VirarPeçasO.Vertical(a, b, tab); validoO=true; } if(a==linha&&b==coluna-1) if(ValidarO.HorizontalEsq(a, b, tab)==true){ tab[linha][coluna]=2; VirarPeçasO.Horizontal(a, b, tab); validoO=true; } if(a==linha&&b==coluna+1) if(ValidarO.HorizontalDir(a, b, tab)==true){ tab[linha][coluna]=2; VirarPeçasO.Horizontal(a, b, tab); validoO=true; } } } } } </pre>	<p>Este método permite-nos validar a jogada O, colocá-la no local especificado pelo jogador O e virar peças “X”.</p> <p>Caso o espaço assinalado pelo jogador esteja vazio, o programa percorre o array que compõe o tabuleiro e valida a jogada. O programa começa por verificar em que direcção é que se encontra a peça adversária (Vertical Cima, Vertical Baixo, Horizontal Esquerda, Horizontal Direita, Diagonal Esquerda Cima, Diagonal Esquerda Baixo, Diagonal Direita Cima e Diagonal Direita Baixo) e depois irá validar a jogada, se a jogada for válida nessa direcção, o programa escreve a peça “O” no local assinalado e inicia o método de virar peças consoante a direcção detectada.</p> <p>NOTA: A peça “O” é reconhecida pelo programa no array de inteiros sempre que existe um algarismo 2 num espaço do tabuleiro.</p>
---	--

<pre> if(a==linha-1&&b==coluna-1) if(ValidarO.DiagonalEsqCima(a, b, tab)==true){ tab[linha][coluna]=2; VirarPeçasO.DiagonalCimaBaixo(a, b, tab); validoO=true; } if(a==linha+1&&b==coluna-1) if(ValidarO.DiagonalEsqBaixo(a, b, tab)==true){ tab[linha][coluna]=2; VirarPeçasO.DiagonalBaixoCima(a, b, tab); validoO=true; } if(a==linha-1&&b==coluna+1) if(ValidarO.DiagonalDirCima(a, b, tab)==true){ tab[linha][coluna]=2; VirarPeçasO.DiagonalBaixoCima(a, b, tab); validoO=true; } if(a==linha+1&&b==coluna+1) if(ValidarO.DiagonalDirBaixo(a, b, tab)==true){ tab[linha][coluna]=2; VirarPeçasO.DiagonalCimaBaixo(a, b, tab); validoO=true;} } } } return validoO; } </pre>	
---	--

A classe ValidarO.java contém os métodos: VerticalCima, VerticalBaixo, HorizontalEsq, HorizontalDir, DiagonalEsqCima, DiagonalEsqBaixo, DiagonalDirCima,

DiagonalDirBaixo, que verificam se a jogada pretendida é válida. A sua descrição pode ser consultada na tabela abaixo.

<pre> public static boolean VerticalCima(int linha, int coluna, int tab[][]){ boolean Valido=false; for(int a=linha;a>=0;a--){ for(int b=coluna;b==coluna;b++){ if(tab[a][b]==0) break; if(tab[a][b]==2) return(Valido=true); } return Valido; } </pre>	<p>Neste método o programa percorre o array que compõe o tabuleiro de jogo na direcção vertical e no sentido ascendente e, caso a posição seleccionada pelo jogador seja válida (ou seja, se detectar uma peça “O” antes que detecte um espaço branco), o programa retorna o booleano Valido com “True”, ou seja, permite que a jogada seja efectuada. Caso contrário irá interromper o ciclo e devolver o Valido como “False”.</p>
<pre> public static boolean VerticalBaixo(int linha, int coluna, int tab[][]){ boolean Valido=false; for(int a=linha;a<tab.length;a++){ for(int b=coluna;b<=coluna;b++){ if(tab[a][b]==0) break; if(tab[a][b]==2) return(Valido=true); } return Valido; } </pre>	<p>Neste método o programa percorre o array que compõe o tabuleiro de jogo na direcção vertical e no sentido descendente e, caso a posição seleccionada pelo jogador seja válida, o programa retorna o booleano Valido com “True”, ou seja, permite que a jogada seja efectuada. Caso contrário irá interromper o ciclo e devolver o Valido como “False”.</p>
<pre> public static boolean HorizontalDir(int linha, int coluna, int tab[][]){ boolean Valido=false; for(int a=linha;a<=linha;a++){ for(int b=coluna;b<tab.length;b++){ if(tab[a][b]==0) </pre>	<p>Neste método o programa percorre o array que compõe o tabuleiro de jogo na Horizontal e para a direita e, caso a posição seleccionada pelo jogador seja válida, o programa retorna o booleano Valido com “True”, ou seja,</p>

<pre> break; if(tab[a][b]==2) return(Valido=true); } return Valido; } </pre>	<p>permite que a jogada seja efectuada. Caso contrário irá interromper o ciclo e devolver o Valido como "False".</p>
<pre> public static boolean HorizontalEsq(int linha, int coluna, int tab[][]){ boolean Valido=false; for(int a=linha;a<=linha;a++) for(int b=coluna;b>=0;b--){ if(tab[a][b]==0) break; if(tab[a][b]==2) return(Valido=true); } return Valido; } </pre>	<p>Neste método o programa percorre o array que compõe o tabuleiro de jogo na Horizontal e para a esquerda e, caso a posição seleccionada pelo jogador seja válida, o programa retorna o booleano Valido com "True", ou seja, permite que a jogada seja efectuada. Caso contrário irá interromper o ciclo e devolver o Valido como "False".</p>
<pre> public static boolean DiagonalEsqCima(int linha, int coluna, int tab[][]){ boolean Valido=false; int diagonal=linha-coluna; for(int a=linha;a>=0;a--) for(int b=coluna;b>=0;b--){ if(a-b==diagonal){ if(tab[a][b]==0) break; if(tab[a][b]==2) return(Valido=true); } } return Valido; } </pre>	<p>Neste método o programa percorre a diagonal esquerda do array que compõe o tabuleiro no sentido ascendente e, caso a posição seleccionada pelo jogador seja válida, o programa retorna o booleano Valido com "True", ou seja, permite que a jogada seja efectuada. Caso contrário irá interromper o ciclo e devolver o Valido como "False".</p>
<pre> public static boolean DiagonalEsqBaixo(int linha, int coluna, int tab[][]){ boolean Valido=false; </pre>	<p>Neste método o programa percorre a diagonal esquerda do</p>

<pre> int diagonal=linha+coluna; for(int a=linha;a<tab.length;a++) for(int b=coluna;b>=0;b--){ if(a+b==diagonal){ if(tab[a][b]==0) break; if(tab[a][b]==2) return(Valido=true); } } return Valido; } </pre>	<p>array que compõe o tabuleiro no sentido descendente e, caso a posição seleccionada pelo jogador seja válida, o programa retorna o booleano Valido com "True", ou seja, permite que a jogada seja efectuada. Caso contrário irá interromper o ciclo e devolver o Valido como "False".</p>
<pre> public static boolean DiagonalDirCima(int linha, int coluna, int tab[][]){ boolean Valido=false; int diagonal=linha+coluna; for(int a=linha;a>=0;a--){ for(int b=coluna;b<tab.length;b++){ if(a+b==diagonal){ if(tab[a][b]==0) break; if(tab[a][b]==2) return(Valido=true); } } } return Valido; } </pre>	<p>Neste método o programa percorre a diagonal direita do array que compõe o tabuleiro no sentido ascendente e, caso a posição seleccionada pelo jogador seja válida, o programa retorna o booleano Valido com "True", ou seja, permite que a jogada seja efectuada. Caso contrário irá interromper o ciclo e devolver o Valido como "False".</p>
<pre> public static boolean DiagonalDirBaixo(int linha, int coluna, int tab[][]){ boolean Valido=false; int diagonal=linha-coluna; for(int a=linha;a<tab.length;a++){ for(int b=coluna;b<tab.length;b++){ </pre>	<p>Neste método o programa percorre a diagonal direita do array que compõe o tabuleiro no sentido descendente e, caso a posição seleccionada pelo jogador seja válida, o programa retorna o booleano Valido com "True", ou seja,</p>

<pre> if(a-b==diagonal){ if(tab[a][b]==0) break; if(tab[a][b]==2) return(Valido=true); } } return Valido; } </pre>	<p>permite que a jogada seja efectuada. Caso contrário irá interromper o ciclo e devolver o Valido como "False".</p>
--	--

A classe ValidarX.java contém os métodos que validam a jogada do jogador X e que funcionam de uma forma muito semelhante à dos métodos que compõem a classe ValidarO.java. A sua descrição pode ser consultada na tabela abaixo.

<pre> public static boolean VerticalCima(int linha, int coluna, int tab[][]){ boolean Valido=false; for(int a=linha;a>=0;a--){ for(int b=coluna;b==coluna;b++){ if(tab[a][b]==0) break; if(tab[a][b]==1) return(Valido=true); } } return Valido; } </pre>	<p>Neste método o programa percorre o array que compõe o tabuleiro de jogo na direcção vertical e no sentido ascendente e, caso a posição seleccionada pelo jogador seja válida (se detectar uma peça "X" antes de um espaço vazio), o programa retorna o booleano Valido com "True", ou seja, permite que a jogada seja efectuada. Caso contrário irá interromper o ciclo e devolver o Valido como "False".</p>
<pre> public static boolean VerticalBaixo(int linha, int coluna, int tab[][]){ boolean Valido=false; for(int a=linha;a<tab.length;a++){ for(int b=coluna;b<=coluna;b++){ if(tab[a][b]==0) break; if(tab[a][b]==1) return(Valido=true); } } } </pre>	<p>Neste método o programa percorre o array que compõe o tabuleiro de jogo na direcção vertical e no sentido descendente e, caso a posição seleccionada pelo jogador seja válida, o programa retorna o booleano Valido com "True", ou seja, permite que a jogada seja efectuada. Caso contrário irá interromper o ciclo e devolver o Valido como "False".</p>

<pre> } return Valido; } </pre>	<p>“False”.</p>
<pre> public static boolean HorizontalDir(int linha, int coluna, int tab[][]){ boolean Valido=false; for(int a=linha;a<=linha;a++) for(int b=coluna;b<tab.length;b++){ if(tab[a][b]==0) break; if(tab[a][b]==1) return(Valido=true); } return Valido; } </pre>	<p>Neste método o programa percorre o array que compõe o tabuleiro de jogo na Horizontal e para a direita e, caso a posição seleccionada pelo jogador seja válida, o programa retorna o booleano Valido com “True”, ou seja, permite que a jogada seja efectuada. Caso contrário irá interromper o ciclo e devolver o Valido como “False”.</p>
<pre> public static boolean HorizontalEsq(int linha, int coluna, int tab[][]){ boolean Valido=false; for(int a=linha;a<=linha;a++) for(int b=coluna;b>=0;b--){ if(tab[a][b]==0) break; if(tab[a][b]==1) return(Valido=true); } return Valido; } </pre>	<p>Neste método o programa percorre o array que compõe o tabuleiro de jogo na Horizontal e para a esquerda e, caso a posição seleccionada pelo jogador seja válida, o programa retorna o booleano Valido com “True”, ou seja, permite que a jogada seja efectuada. Caso contrário irá interromper o ciclo e devolver o Valido como “False”.</p>
<pre> public static boolean DiagonalEsqCima(int linha, int coluna, int tab[][]){ boolean Valido=false; int diagonal=linha-coluna; for(int a=linha;a>=0;a--) for(int b=coluna;b>=0;b--){ if(a-b==diagonal){ if(tab[a][b]==0) </pre>	<p>Neste método o programa percorre a diagonal esquerda do array que compõe o tabuleiro no sentido ascendente e, caso a posição seleccionada pelo jogador seja válida, o programa retorna o booleano Valido com “True”, ou seja, permite que a jogada seja efectuada. Caso contrário</p>

<pre> break; if(tab[a][b]==1) return(Valido=true); } } return Valido; } </pre>	<p>irá interromper o ciclo e devolver o Valido como "False".</p>
<pre> public static boolean DiagonalEsqBaixo(int linha, int coluna, int tab[][]){ boolean Valido=false; int diagonal=linha+coluna; for(int a=linha;a<tab.length;a++){ for(int b=coluna;b>=0;b--){ if(a+b==diagonal){ if(tab[a][b]==0) break; if(tab[a][b]==1) return(Valido=true); } } } return Valido; } </pre>	<p>Neste método o programa percorre a diagonal esquerda do array que compõe o tabuleiro no sentido descendente e, caso a posição seleccionada pelo jogador seja válida, o programa retorna o booleano Valido com "True", ou seja, permite que a jogada seja efectuada. Caso contrário irá interromper o ciclo e devolver o Valido como "False".</p>
<pre> public static boolean DiagonalDirCima(int linha, int coluna, int tab[][]){ boolean Valido=false; int diagonal=linha+coluna; for(int a=linha;a>=0;a--){ for(int b=coluna;b<tab.length;b++){ if(a+b==diagonal){ if(tab[a][b]==0) break; if(tab[a][b]==1) return(Valido=true); } } } } </pre>	<p>Neste método o programa percorre a diagonal direita do array que compõe o tabuleiro no sentido ascendente e, caso a posição seleccionada pelo jogador seja válida, o programa retorna o booleano Valido com "True", ou seja, permite que a jogada seja efectuada. Caso contrário irá interromper o ciclo e devolver o Valido como "False".</p>

<pre> } return Valido; } </pre>	
<pre> public static boolean DiagonalDirBaixo(int linha, int coluna, int tab[][]){ boolean Valido=false; int diagonal=linha-coluna; for(int a=linha;a<tab.length;a++){ for(int b=coluna;b<tab.length;b++){ if(a-b==diagonal){ if(tab[a][b]==0) break; if(tab[a][b]==1) return(Valido=true); } } } return Valido; } </pre>	<p>Neste método o programa percorre a diagonal direita do array que compõe o tabuleiro no sentido descendente e, caso a posição seleccionada pelo jogador seja válida, o programa retorna o booleano Valido com "True", ou seja, permite que a jogada seja efectuada. Caso contrário irá interromper o ciclo e devolver o Valido como "False".</p>

A classe VirarPeças.java contém os métodos Vertical, Horizontal, DiagonalCimaBaixo e DiagonalBaixoCima, que têm como principal função virar peças "X" e substituí-las por peças "O". A descrição da função destes métodos encontra-se na tabela seguinte.

<pre> public static void Vertical(int linha, int coluna, int tab[][]){ boolean virar=false; int repetirO=0; int repetirX=0; for(int a=0;a<tab.length;a++){ for(int b=coluna;b==coluna;b++){ if(tab[a][b]==2){ if(repetirO==0 (repetirO==1&&repetirX>0)) repetirO+=1; virar=true; } } } } </pre>	<p>Depois de colocar a peça "O" num espaço vazio e de validar a jogada (como vimos nas classes anteriores), o programa percorre a coluna do local da peça, e onde existirem peças "X" transforma-as em peças "O", ou seja, o programa substitui o valor das peças "X" e coloca o valor das peças "O". Primeiro irá detectar a primeira peça "O", somar 1 ao contador de repetições de peças "O", "repetirO", e depois mudar o valor boolean "virar" para "True". Isto irá permitir que se inicie o</p>
---	--

<pre> if(virar=true&&repetirO<2){ if(tab[a][b]==1){ tab[a][b]=2; repetirX+=1; } } if(repetirO==2) virar=false; } } } </pre>	<p>processo da troca das peças. Quando for detectada uma peça “X” altera-se para o valor de uma peça “O” e soma 1 ao contador de repetição de peças “X”. A partir daqui se for detectada outra peça “O” irá somar mais 1 ao contador das peças “O”, o que irá depois modificar o boolean “virar” para “false” mais uma vez. Terminando o processo de virar peças.</p>
<pre> public static void Horizontal(int linha, int coluna, int tab[][]){ boolean virar=false; int repetirO=0; int repetirX=0; for(int a=linha;a==linha;a++){ for(int b=0;b<tab.length;b++){ if(tab[a][b]==2){ if(repetirO==0 (repetirO==1&&repetirX>0)) repetirO+=1; virar=true; } if(virar=true&&repetirO<2){ if(tab[a][b]==1){ tab[a][b]=2; repetirX+=1; } } if(repetirO==2) virar=false; } } } </pre>	<p>Depois de colocar a peça “O” num espaço vazio e de validar a jogada (como vimos nas classes anteriores), o programa percorre a linha do local da peça, e onde existirem peças “X” transforma-as em peças “O”, ou seja, o programa substitui o valor das peças “X” e coloca o valor das peças “O”. Primeiro irá detectar a primeira peça “O”, somar 1 ao contador de repetições de peças “O” “repetirO”, e depois mudar o valor boolean “virar” para “True”. O que irá permitir que se inicie o processo da troca das peças. Quando for detectada uma peça “X” altera-se para o valor de uma peça “O” e soma 1 ao contador de repetição de peças “X”. A partir daqui se for detectada outra peça “O” irá somar mais 1 ao contador das peças “O”, o que irá depois modificar o boolean “virar” para “False” mais uma vez. Terminando o processo de virar peças.</p>

<pre> public static void DiagonalCimaBaixo(int linha, int coluna, int tab[][]){ boolean virar=false; int repetirO=0; int repetirX=0; int diagonal=linha-coluna; for(int a=0;a<tab.length;a++) for(int b=0;b<tab.length;b++){ if(a-b==diagonal){ if(tab[a][b]==2){ if(repetirO==0 (repetirO==1&&repetirX>0)) repetirO+=1; virar=true; } if(virar=true&&repetirO<2){ if(tab[a][b]==1){ tab[a][b]=2; repetirX+=1; } } if(repetirO==2) virar=false; } } } </pre>	<p>Depois de colocar a peça “O” num espaço vazio e de validar a jogada (como vimos nos métodos das classes anteriores), o programa percorre a diagonal, que corresponde à subtração das coordenadas da peça, (dirigindo-se do topo para a base do mesmo) do local da peça, e onde existirem peças “X” transforma-as em peças “O”, ou seja, o programa substitui o valor das peças “X” e coloca o valor das peças “O”. Primeiro irá detectar a primeira peça “O”, somar 1 ao contador de repetições de peças “O” “repetirO”, e depois mudar o valor boolean “virar” para “True”. O que irá permitir que se inicie o processo da troca das peças. Quando for detectada uma peça “X” altera-se para o valor de uma peça “O” e soma 1 ao contador de repetição de peças “X”. A partir daqui se for detectada outra peça “O” irá somar mais 1 ao contador das peças “O”, o que irá depois modificar o boolean “virar” para “False” mais uma vez. Terminando o processo de virar peças.</p>
<pre> public static void DiagonalBaixoCima(int linha, int coluna, int tab[][]){ boolean virar=false; int repetirO=0; int repetirX=0; int diagonal=linha+coluna; </pre>	<p>Depois de colocar a peça “O” num espaço vazio e de validar a jogada (como vimos nos métodos das classes anteriores), o programa percorre a diagonal, que corresponde à soma das coordenadas da peça, (dirigindo-se da base para o topo do mesmo) do</p>

<pre> for(int a=0;a<tab.length;a++) for(int b=0;b<tab.length;b++){ if(a+b==diagonal){ if(tab[a][b]==2){ if(repetirO==0 (repetirO==1&&repetirX>0)) repetirO+=1; virar=true; } if(virar=true&&repetirO<2){ if(tab[a][b]==1){ tab[a][b]=2; repetirX+=1; } } if(repetirO==2) virar=false; } } } </pre>	<p>local da peça, e onde existirem peças "X" transforma-as em peças "O", ou seja, o programa substitui o valor das peças "X" e coloca o valor das peças "O". Primeiro irá detectar a primeira peça "O", somar 1 ao contador de repetições de peças "O" "repetirO", e depois mudar o valor boolean "virar" para "True". O que irá permitir que se inicie o processo da troca das peças. Quando for detectada uma peça "X" altera-se para o valor de uma peça "O" e soma 1 ao contador de repetição de peças "X". A partir daqui se for detectada outra peça "O" irá somar mais 1 ao contador das peças "O", o que irá depois modificar o boolean "virar" para "False" mais uma vez. Terminando o processo de virar peças.</p>
---	--

A classe VirarPeçasX.java contém os métodos que têm como principal função virar peças "O" e substituí-las por peças "X". A descrição da função destes métodos encontra-se na tabela subsequente.

<pre> public static void Vertical(int linha, int coluna, int tab[][]){ boolean virar=false; int repetirX=0; int repetirO=0; for(int a=0;a<tab.length;a++){ for(int b=coluna;b==coluna;b++){ if(tab[a][b]==1){ if(repetirX==0 (repetirX==1&&repetirO>0)) </pre>	<p>Depois de colocar a peça "X" num espaço vazio e de validar a jogada (como vimos nas classes anteriores), o programa percorre a coluna do local da peça, e onde existirem peças "O" transforma-as em peças "X", ou seja, o programa substitui o valor das peças "O" e coloca o valor das peças "X". Primeiro irá detectar a primeira peça "X", somar 1 ao contador de</p>
--	---

<pre> repetirX+=1; virar=true; } if(virar=true&&repetirX<2){ if(tab[a][b]==2){ tab[a][b]=1; repetirO+=1; } } if(repetirX==2) virar=false; } } </pre>	<p>repetições de peças “X” “repetirX”, e depois mudar o valor boolean “virar” para “True”. O que irá permitir que se inicie o processo da troca das peças. Quando for detectada uma peça “O” altera-se para o valor de uma peça “X” e soma 1 ao contador de repetição de peças O. A partir daqui se for detectada outra peça “X” irá somar mais 1 ao contador das peças “X”, o que irá depois modificar o boolean “virar” para “False” mais uma vez. Terminando o processo de virar peças.</p>
<pre> public static void Horizontal(int linha, int coluna, int tab[][]){ boolean virar=false; int repetirX=0; int repetirO=0; for(int a=linha;a==linha;a++){ for(int b=0;b<tab.length;b++){ if(tab[a][b]==1){ if(repetirX==0 (repetirX==1&&repetirO>0)) repetirX+=1; virar=true; } if(virar=true&&repetirX<2){ if(tab[a][b]==2){ tab[a][b]=1; repetirO+=1; } } if(repetirX==2) virar=false; </pre>	<p>Depois de colocar a peça “X” num espaço vazio e de validar a jogada (como vimos nas classes anteriores), o programa percorre a linha do local da peça, e onde existirem peças “O” transforma-as em peças “X”, ou seja, o programa substitui o valor das peças “O” e coloca o valor das peças “X”. Primeiro irá detectar a primeira peça “X”, somar 1 ao contador de repetições de peças “X” “repetirX”, e depois mudar o valor boolean “virar” para “True”. O que irá permitir que se inicie o processo da troca das peças. Quando for detectada uma peça “O” altera-se para o valor de uma peça “X” e soma 1 ao contador de repetição de peças O. A partir daqui se for detectada outra peça “X” irá somar mais 1 ao contador das peças “X”, o que irá depois modificar o boolean “virar” para</p>

<pre> } } } </pre>	<p>“False” mais uma vez. Terminando o processo de virar peças.</p>
<pre> public static void DiagonalCimaBaixo(int linha, int coluna, int tab[][]){ boolean virar=false; int repetirX=0; int repetirO=0; int diagonal=linha-coluna; for(int a=0;a<tab.length;a++) for(int b=0;b<tab.length;b++){ if(a-b==diagonal){ if(tab[a][b]==1){ if(repetirX==0 (repetirX==1&&repetirO>0)) repetirX+=1; virar=true; } if(virar=true&&repetirX<2){ if(tab[a][b]==2){ tab[a][b]=1; repetirO+=1; } } if(repetirX==2) virar=false; } } } } </pre>	<p>Depois de colocar a peça “X” num espaço vazio e de validar a jogada (como vimos nas classes anteriores), o programa percorre a diagonal, que corresponde à subtração das coordenadas da peça, (dirigindo-se do topo para a base do mesmo) do local da peça, e onde existirem peças “O” transforma-as em peças “X”, ou seja, o programa substitui o valor das peças “O” e coloca o valor das peças “X”. Primeiro irá detectar a primeira peça “X”, somar 1 ao contador de repetições de peças “X” “repetirX”, e depois mudar o valor boolean “virar” para “True”. O que irá permitir que se inicie o processo da troca das peças. Quando for detectada uma peça “O” altera-se para o valor de uma peça “X” e soma 1 ao contador de repetição de peças O. A partir daqui se for detectada outra peça “X” irá somar mais 1 ao contador das peças “X”, o que irá depois modificar o boolean “virar” para “False” mais uma vez. Terminando o processo de virar peças.</p>
<pre> public static void DiagonalBaixoCima(int linha, int coluna, int tab[][]){ boolean virar=false; int repetirX=0; int repetirO=0; int diagonal=linha+coluna; </pre>	<p>Depois de colocar a peça “X” num espaço vazio e de validar a jogada (como vimos nas classes anteriores), o programa percorre a diagonal, que corresponde à soma das coordenadas da peça, (dirigindo-se da base para</p>

<pre> for(int a=0;a<tab.length;a++) for(int b=0;b<tab.length;b++){ if(a+b==diagonal){ if(tab[a][b]==1){ if(repetirX==0 (repetirX==1&&repetirO>0)) repetirX+=1; virar=true; } if(virar=true&&repetirX<2){ if(tab[a][b]==2){ tab[a][b]=1; repetirO+=1; } } if(repetirX==2) virar=false; } } } </pre>	<p>o topo do mesmo) do local da peça, e onde existirem peças "O" transforma-as em peças "X", ou seja, o programa substitui o valor das peças "O" e coloca o valor das peças "X". Primeiro irá detectar a primeira peça "X", somar 1 ao contador de repetições de peças "X" "repetirX", e depois mudar o valor boolean "virar" para "True". O que irá permitir que se inicie o processo da troca das peças. Quando for detectada uma peça "O" altera-se para o valor de uma peça "X" e soma 1 ao contador de repetição de peças O. A partir daqui se for detectada outra peça "X" irá somar mais 1 ao contador das peças "X", o que irá depois modificar o boolean "virar" para "False" mais uma vez. Terminando o processo de virar peças.</p>
---	--

iv. Tabuleiro Final

A classe *MenuGameOver.java* contém o método *Vencedor* que escreve no output o nome do vencedor no final do jogo. O seu funcionamento está descrito na tabela seguinte.


<pre> public static void Vencedor (int tab[][]){ if(Contador.ContadorPeçasX(tab)>Contador.ContadorPeçasO(tab)) System.out.println("O Jogador X é o vencedor!!"); if(Contador.ContadorPeçasX(tab)<Contador.ContadorPeçasO(tab)) System.out.println("O Jogador O é o vencedor!!"); if(Contador.ContadorPeçasX(tab)==Contador.ContadorPeçasO(tab)) </pre>	<p>Este método utiliza a contagem das peças "X" e das peças "O" para escrever no output, no final do jogo, o nome do vencedor. Caso o número de peças "X" seja maior do que o número de peças "O", o vencedor é o jogador X. Caso o número de peças "O" seja maior do que o número de peças "X", o</p>
---	--

<pre>System.out.println("É um empate!!!Ninguem ganhou..."); }</pre>	vencedor será o jogador O. Se o número de peças "X" for igual ao número de peças "O", o programa escreve no output a conclusão final: "É um empate!!! (...)".
---	---

2. Conclusão:

Tendo em conta o objectivo do trabalho, podemos concluir que o programa cumpre a maioria dos requisitos pretendidos e funciona de uma forma simples e, de certo modo, funcional. O programa foi desenvolvido de uma forma incremental. Este tipo de desenvolvimento permitiu-nos dividir o problema em fracções, o que facilitou a sua resolução.

3. Bibliografia

 *Slides da cadeira de Programação I por Vítor Beires Nogueira, Ano Lectivo 2009 – 2010.*