Relatório Trabalho Prático de Programação II

Introdução:

A partir da elaboração do trabalho prático "Smoothy" foi-nos possível adquirir e consolidar conhecimentos na área da POO, leccionados na cadeira de Programação II.

Com este relatório pretendemos facilitar o entendimento do código por nós escrito através da explicação por palavras, daquelas que para nós, são as classes/métodos essenciais.

Classes/métodos:

Decidimos Utilizar duas Classes java e vários métodos para a elaboração do trabalho sendo eles:

public class Jogo;

- Classe principal do jogo pois contém o Main.

Métodos da classe Jogo:

public static void main;

- Contém os inputs através de Scanners e também a verificação dos mesmos em relação a possíveis erros introduzidos por parte do utilizador.

- Contém o Loop do jogo, que não é mais que um ciclo While que apenas termina quando o jogo chega ao fim.

public class Tabuleiro;

-Classe que contém todos os métodos relacionados com o tabuleiro de jogo.

Métodos referentes ao tabuleiro:

public void criaTabuleiro(**int** raiz, **int** cor){

-Método que cria um tabuleiro preenchido com números gerados aleatóriamente diferentes de 0, pois 0 no tabuleiro significa vazio.

}

public void mostraTabuleiro(){

-Método que formata o tabuleiro de forma a parecer "quadrado".

Métodos referentes à jogada:

public boolean horizontal(**int** x,**int** y){

-Função booleana que verifica a partir da posição inserida pelo utilizador a existência de peças semelhantes tanto à esquerda como à direita.
-Caso existam peças para qualquer um dos lados, o método horizontal altera o tabuleiro de forma a eliminar as peças semelhantes, "desce" as respectivas colunas e coloca zeros no topo do tabuleiro, de forma a simular a gravidade quando as peças são eliminadas.

}

public boolean vertical(**int** x,**int** y){

-Função booleana que verifica a partir da posição inserida pelo utilizador a existência de peças semelhantes na vertical.
-Começa por incrementar um contador para baixo de forma a encontrar peças semelhantes.
-O valor do contador é depois adicionado à coordenada "x" da posição escolhida pelo utilizador, criando assim uma novaPosição.
-A partir da novaPosição é incrementado um contador para cima, que através de um ciclo For altera a coluna de forma a eliminar as peças semelhantes, "desce" as respectivas colunas e coloca zeros no topo do tabuleiro, de forma a simular a gravidade quando as peças são eliminadas.

}

public void groundZero(**int** x,**int** y){

-Método que altera a coluna da coordenada Y inserida pelo utilizador quando
horizontal == true & vertical ==false.
-A criação deste método deve-se ao facto de não querermos alterar a coluna onde se encontra a posição escolhida pelo utilizador antes de serem feitas as verificações em todas as direções, pois esta alteração iria modificar o tabuleiro, levando a uma verificação errada em métodos que fossem utlizados posteriormente.

}

public void empurraColuna(){

-Método que em todas as jogadas, através de Ciclos For incrementa um contador sempre que encontra um zero no tabuleiro.
-Quando o contador iguala o tamanho do tabuleiro, então significa que encontrou uma coluna cheia de zeros (vazia).
Posteriormente "empurra essa coluna para a esquerda e todas as outras para a direita, para simular o seu desaparecimento.

}

public void jogada(**int** x,**int** y){

-Método que “chama” todos os métodos anteriores referentes à jogada, com as devidas condições e restrições de forma a não comprometer nenhuma verificação no tabuleiro.

Nota: Decidimos utilizar o operador “&” em vez do tradicional AND “&&” porque apesar da condição para ground zero ser, horizontal == true && vertical == false, precisámos que ambas as condições fossem verificadas em qualquer caso, e o operador “&” resolve esse problema, ao contrário do AND que abandona a verificação imediatamente caso a primeira função devolva o valor booleano false.

}

Métodos referentes à Verificação da jogada:

public boolean verificaHorizontal(**int** x, **int** y){

-Método que verifica as posições imediatamente a seguir (horizontalmente) à posição inserida pelo utilizador, de forma a verificar se há jogadas possiveis, incluindo casos particulares.

}

public boolean verificaVertical(**int** x, **int** y){

-Método semelhante ao anterior com a diferença de funcionar no eixo vertical do tabuleiro.
}

public boolean verificaFimJogo(){

-Método que “chama” a funções anteriores e as introduz num ciclo de forma a verificar todo o tabuleiro. Desta maneira é possível analizar todas as posições de forma a encontrar alguma jogada possivel.
-Caso não encontre nenhuma jogada possível, devolve o valor false, e consequentemente o loop do jogo termina no main.

}

public void pontuacao(){

-Após terminado o jogo este método verifica o tabuleiro e incrementa um contador quando encontra valores diferentes de zero.
-De seguida aplica a formula “pontuacao = R*R-N”, e imprime o resultado da mesma ao utilizador.

}

Conclusão:

Através deste trabalho foi possível avaliar as nossas falhas enquanto programadores e pela prática que nos proporcionou melhorar em vários aspetos que serão concerteza valiosos para nós futuramente enquanto Engenheiro Informáticos.

Ficámos bastante satisfeitos com o resultado final e com a aprendizagem que nos proporcionou.

Trabalho realizado por:

José Serra 33289
João Pica 35921