Relatório do trabalho prático “Smoothy” relativo à disciplína de Programação 2

Trabalho realizado por:

Miguel Maia nº33456
Luís Rosado nº34249

Ano lectivo 2015/2016
2º Semestre

Introdução

Este trabalho é constituído por 4 classes, uma que gere o jogo, Classe Smoothy; uma que gere o tabuleiro, Classe Tabuleiro; uma que gere as jogadas, Classe Jogadas e uma que gere os inputs, Classe Inputs. A classe Smoothy gere o jogo e chama todos os métodos usados no jogo.

Classe “Smoothy”

Como não poderia deixar de ser, a primeira coisa que tivemos de fazer nesta classe foi fazer o import das funcionalidades em geral do Java. De seguida declaramos as variáveis do enunciado do trabalho, nomeadamente, o tamanho do tabuleiro, o número de cores no tabuleiro e uma seed para, no random, termos uma espécie de número de jogo. Também invocamos o scanner.

De seguida fizemos o método start para começar o programa. Criámos um booleano para nos ajudar a fazer os ciclos while. Com o booleano como parâmetro, fizemos o ciclo while, para que pudéssemos utilizar o try e o catch até que fosse introduzido um valor dentro dos parâmetros. Se o utilizador introduzir um valor fora do intervalo entre 4 e 26, inclusive, este pede novamente para por um valor dentro do intervalo. Caso o utilizador escreve algo que não seja um número, ou que seja um número muito grande, o catch é activado, o programa dá erro e volta novamente a pedir para o utilizador escrever um número dentro do intervalo permitido. O pensamento é igual para o número de cores no tabuleiro. Para a seed, como pode receber qualquer número, apenas nos preocupamos com o facto de o utilizador não colocar um número. Portanto, fizemos um catch semelhante ao em cima descrito. O valor do booleano é alterado de forma a que se o utilizador coloque algo errado, este fique falso e que só permita que o programa continue caso seja verdadeiro.

Seguidamente, fizemos o método main. Esta chama todas as outras classes e as funções necessárias para começar o jogo. Ainda nesta classe, fizemos um ciclo while, para que o programa peça o input da jogada ao utilizador até deixar de haver jogadas possíveis. Quando terminar, chamamos o método pontuação.

Classe “Input”

A primeira coisa que fizemos nesta classe, foi declarar as variáveis linha e coluna. Criamos o método pedir e as variáveis onde íamos guardar o que o utilizador colocasse. É pedida uma letra ao utilizador, podendo esta ser maiúscula ou minúscula. Com a ajuda do código ascii e do código (char)(x) onde x significa o número da letra no código ascii, introduzida pelo utilizador, podemos passar da letra que o utilizador pediu para um número, este que será a linha e a coluna, que começam em zero. Por exemplo, se o utilizador pedir o “a”, então o código converte-o no numero “0”, o que significa que o utilizador escolheu a primeira linha. Caso a letra não esteja no intervalo do tamanho escolhido para o tabuleiro, ou seja outra coisa que não seja uma letra, o programa pede novamente, até que seja introduzido uma letra que esteja no intervalo do tamanho pretendido para o tabuleiro.

Depois das verificações, guardamos o valor correspondente à letra escolhida, nas variáveis linha. Fizemos exactamente o mesmo para a coluna.

Classe “Tabuleiro”

Nesta classe, bastou-nos fazer o import do random, pois era a única funcionalidade deste género que íamos utilizar. Criamos um array bidimensional para fazermos o tabuleiro.

Criamos o método gerarTabuleiro, para criar o tabuleiro, com o tamanho dado pelo input do utilizador. Declaramos um random, com o parâmetro seed, que gera uma sequência fixa, sempre que os parâmetros sejam iguais. Com dois ciclos for, um para as linhas [i] e as colunas [j], preenchemos todas as posições com números aleatórios, gerados pelo random. Cada número é uma cor, o espaço vazio é 0, por isso soma-se 1 a todos os números gerados.

De seguida, fizemos o método mostrarTabuleiro. Primeiro faz-se print das letras, de acordo com o tamanho do tabuleiro, com a ajuda do código ascii e do código (char) (x), novamente. Depois imprimimos uma letra e todos os números dentro do array, por linha, ou seja, por cada linha aparece uma letra e os números de uma linha, ficando no final com todos os integrantes do array e por cada linha uma letra.

O método atualizaTabuleiro percorre o tabuleiro, coluna a coluna, começando pelo fim. Cada vez que encontra uma posição com 0, copia o primeiro numero não 0 dessa coluna para essa posição. Faz essa verificação para todas as colunas do tabuleiro. No final o tabuleiro fica com os 0, na parte de cima.

Depois do tabuleiro ter as peças jogáveis em baixo é verificada se alguma coluna esta toda a 0. Se estiver é copiada a primeira coluna com algum número não 0 para essa posição.

O último método desta classe é a pontuação. Declaramos um contador para somar 1, sempre que alguma posição do tabuleiro não tivesse a zero. Com dois ciclos for, corremos o tabuleiro todo e no contador fica o número de posições do tabuleiro que não estavam a zero. Declaramos também um inteiro para fazer a fórmula da pontuação. Com if’s verificamos se o jogador tinha ganho o jogo, ou se ficavam peças no tabuleiro, sem que se possa fazer mais nenhuma jogada. Por fim, o método imprime a pontuação obtida.

Classe “Jogadas”

A primeira coisa que fizemos nesta classe foi fazer o import de ArrayLists e de Lists, pois guardámos as peças que iam ser eliminadas numa ArrayList. Antes de podermos utilizar a ArrayList, tivemos que a criar. Também declaramos o inteiro cor.

Criamos o método verificarJogada. Como parâmetros tem a linha e a coluna desejadas. Declarámos um booleano para saber se a posição escolhida pode ser alterada. A cor passa a ser o número que estava nessa posição no tabuleiro. Através de vários if’s e for fizemos as condições para verificar tanto na horizontal, como na vertical, se os números adjacentes são iguais à cor escolhida. Para a linha seguinte, verifica-se se é igual á cor, e se for passa essa peça a “0”. Também guarda a coordenada no ArrayList para depois verificar se essa coordenada tem adjacentes iguais à cor. O pensamento é semelhante para verificar as linhas a cima, tal como as colunas á esquerda e á direita. A única coisa que muda, é se verificamos a linha ou a coluna, e se incrementamos, ou decrementamos. Dentro de cada ciclo, utilizamos o booleano, para sabermos se foi alterado alguma coisa, pois se escolhermos uma cor (número) que não tenha nenhum número igual adjacente, então não vai haver alteração nenhuma, e o booleano fica falso.

No método verificarMaisJogadas criamos uma ArrayList copia, onde são copiados os elementos da ArrayList eliminar, e posteriormente é eliminada a ArrayList eliminar. Enquanto a ArrayList eliminar tiver elementos, estes são copiados para uma nova ArrayList copia, e eliminados da ArrayList eliminar. Para o conjunto de coordenadas do ArrayList copia é chamado o metodo verificarJogada.

Por fim, criamos o método continuarJogo, que verifica se alguma peça diferente de “0” tem ao seu lado o mesmo número. Caso tenha a função retorna True.

Conclusão

Inicialmente pensamos apenas em fazer duas classes para a realização do trabalho, uma para o tabuleiro e outra para tudo o resto, mas concluímos que para trabalharmos melhor e para o trabalho ficar mais organizado, quatro classes era o ideal.