Engenharia Informática – Programação II
Semestre Par - 2018

UNIVERSIDADE DE ÉVORA

Fusion

Trabalho realizado por:

Maria Correia 39836

Mafalda Rosa 40021

Introdução

Este trabalho foi-nos proposto como projeto final da disciplina de Programação II e tem como nome “Fusion”. É uma versão simplificada do jogo “Candy Crush” mas tem apenas um tabuleiro quadrado, com o mesmo número de linhas e colunas e com peças numeradas e aleatórias.

Em cada jogada, o jogador insere coordenadas, escolhendo a peça a ser retirada. Esta deve estar adjacente a pelo menos duas peças com o mesmo número para que sejam eliminadas, tanto a selecionada como as peças adjacentes com a mesma numeração.

As peças só são consideradas adjacentes quando estão na mesma linha e em colunas seguidas ou na mesma coluna e em linhas seguidas, assims duas peças na diagonal não são adjacentes.

O objetivo será eliminar peças e fazer com que as que se encontram por cima caiam verticalmente.

Caso uma das colunas fique sem peças, então as colunas que se encontrarem à esquerda dessa mesma coluna, deslocar-se-ão para a direita para preencher a coluna vazia.

Quando todas as peças forem eliminadas ou não houver mais jogadas possíveis, o jogo irá terminar. Será dado ao utilizador uma pontuação total que será a soma acumulada do quadrado do número de células eliminadas.

O jogo tem como parâmetros:

- **Dimensões** - o tamanho de um lado do tabuleiro;

- **Cores** - o número de cores distintas, neste caso associámos a cada cor um número, visto que a nossa interface é textual;

1 - vermelho

2 -verde

3 - amarelo

4 – azul

5 - rosa

6 – laranja

7 – magenta

8 – cinzento

9 – preto

- **Gerador** - a raiz do gerador de números aleatórios usado, que define um tabuleiro.

Classe Menu

Esta classe contém a main e é maioritariamente o "cérebro" do projeto. Nela escolhem-se as opções que queremos relativamente ao tabuleiro e inserimos as coordenadas para jogar.

Classe Tabuleiro

É nesta classe que ocorrem as alterações ao tabuleiro de jogo, sendo que esta contém um método recursivo para verificar as posições adjacentes no tabuleiro que é a base deste projeto.

Main

Na nossa main decidimos simplesmente iniciar o menu. O resto das funções controlam o jogo, pois acreditamos que tornará o nosso trabalho mais simples.

Classe Menu

options

É neste método que se dá a inicialização das variáveis principais. Começamos por fazer um print das opções do menu e de seguida entramos num ciclo while que faz uso do try/catch para evitar exceções. Consoante a escolha do utilizador este método irá chamar outros métodos, tanto para inserir os valores de inicialização do tabuleiro como os créditos, ou voltar para ele mesmo.

creditos

Este método dá uso a três prints para mostrar os autores do trabalho.

tabInputs

Este método vai fazer uso a um ciclo infinito (while(true)) para obrigar o utilizador a fornecer parâmetros válidos para a inicialização do tabuleiro.
Como todos os outros métodos que lidam com inputs dá uso a try e catch(InputMismatchException) para que quando o utilizador inserir outros caracteres que não números, lidar com a exceção.

coordenadas

Neste método é pedido ao utilizador que insira as coordenadas da posição onde quer jogar. Mais uma vez dá uso a um ciclo while com try/catch e procede à verificação das coordenadas inseridas pelo utilizador: se se encontram dentro do tabuleiro ou não. Após o utilizador inserir coordenadas dentro do tabuleiro, a função retorna esse par de coordenadas

jogo

Começamos por chamar a função que imprime o tabuleiro e após esta ser executada, o jogo começa. Aqui encontramos um while com uma condição: se já não se puderem efetuar mais jogadas sobre o tabuleiro, este quebra o ciclo. Dentro deste while iniciamos um array preenchido por 33 em cada posição. Este 33 serve, simplesmente, para assinalar que o espaço se encontra vazio, para se poderem inserir coordenadas neste array, visto que num array de inteiros não se podem meter elementos a null.
Nesta função procedemos à verificação da vizinhança das coordenadas. Caso o par de coordenadas não tenha posições adjacentes com o mesmo valor, dá um print a afirmar que a posição é inválida, voltando a realizar o ciclo e a pedir as coordenadas de novo. Caso contrário, são chamadas as funções jogadas e printTabuleiro do tabuleiro.

Classe Tabuleiro

tabuleiro

Aqui se encontra o construtor da classe. Neste método iniciamos todas as variáveis da classe e chamamos a função que preenche os valores do tabuleiro com valores random, bem como as função do botão.

graphics

Neste método definimos apenas as cores associadas a cada número, ou seja os botões do jogo.

criarTabuleiro

Neste método usamos dois ciclos for para percorrer o array tab de duas dimensões de modo a ser preenchido com valores do random(gerador) diferentes de 0.

printTabuleiro

Método que percorre o tabuleiro, adicionando todos os valores do mesmo a uma string s, para no fim imprimir o seu valor na consola.

verifica_vizinhos

Método recursivo que percorre todas as posições adjacentes com valores iguais à posição inicial e adiciona-as para um array bidimensional para mais tarde serem colocadas a 0. Este começa por utilizar try/catch de modo a que as exceções lançadas pelas verificações fora do tabuleiro não originem erros.
Caso as posições adjacentes tenham o mesmo valor, não exista no array coords e seja diferente de 0, chama-se a função insereCoord para inserir as coordenadas atuais no array coords e chama-se recursivamente para a posição seguinte. Quando não existirem mais posições para onde se possa deslocar a função, guarda a última posição no array e começa a retornar o array coords atualizado.

verifica

Este método usa um ciclo iterativo, de modo a percorrer o array coords para verificar se umas determinadas coordenadas x,y se encontram no mesmo. Retorna verdade quando as encontrarmos. Caso contrário, retorna falso.

insereCoord

Percorrendo novamente o array coords usando recursividade, esta função procura um espaço livre no array coords para inserir novas coordenadas lá.

verifica_pos

Aqui verificamos se a posição atual é válida, simplesmente averiguando se o array coords contém mais do que uma coordenada. Se estas condições se verificarem, retorna true. Caso contrário, retorna false.

verificaColunas

Neste método usamos três ciclos for de modo a que o array tab seja percorrido na vertical, e caso a coluna esteja toda a 0. este vai percorrer o array colunas e verificar se o x atual existe no array colunas. Caso exista coloca a variável booleana var1 a falso e verifica as restantes colunas, caso não aconteça retorna a var1.

arrumaTabuleiro

Este método utiliza um inteiro change para alternar entre puxar os elementos do tabuleiro para baixo ou para a direita. Ambas as duas “subfunções” são iguais, diferindo simplesmente em que uma usa os valores x do tabuleiro e outra utiliza os valores y.
No caso da função que puxa os elementos para a direita, utilizamos três ciclos for para percorrer o array tab por linhas e comparar se a posição à esquerda é diferente de 0 e se a atual é igual a 0. Caso seja, troca os elementos. Nesta função, o terceiro for serve para que este processo se repita várias vezes para a mesma linha

de modo a garantir que todos os elementos ficam o mais à direita possível.

jogadas

Esta função simplesmente elimina células o caso de posições adjacentes.

fim_de_jogo

Método que verifica todas as colunas e linhas do tabuleiro e a existência (ou não) de vizinhos. Caso estes não existam em nenhuma das coordenadas, retorna true. Se continuarem a existir, retorna false.

pontos

Esta função dá uma pontuação parcial a cada eliminação, igual ao quadrado do número de células eliminadas. A pontuação total, será a soma de todas as pontuações parciais.

Conclusão

Primeiramente há que referir que é de extrema importância para nós, iniciantes em Java, conseguir alcançar as metas propostas nestes projetos. Adquirimos, nas aulas teóricas e práticas, bastante conhecimento acerca desta disciplina e de como ela é tão importante: não só na criação de jogos, como também em tantas tarefas do quotidiano.

A programação é, portanto, uma ferramenta bastante útil, podendo ser utilizada em tantas e diversas situações. Não basta saber a linguagem, há que adquirir uma prática e um pensamento diferentes do habitual. Porém, foi um desafio bastante agradável, elevando todos os nossos conhecimentos nesta área.