Programação

ISEL INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Área Departamental de Engenharia de Electrónica e Telecomunicações e de Computadores

Rui Paças	44815	3º Trabalho Prático
João Sousa	44839	
Leonardo Silva Carreira	44849	<13.12.2017>

Neste trabalho foi nos proposto a realização do jogo 2048 em Java pelo professor Pedro Pereira. Para isso é dada a class **console** já implementada e pronta para utilizar no nosso programa.

O professor deu-nos no começo os ficheiros fonte das classes Game2048, Panel, TopScores e Scores. A classe Panel é usada para a apresentação e já está implementada. As classes TopScores e Scores estão ainda por implementar sendo que são responsáveis pela apresentação dos resultados obtidos no jogo. A classe Game2048 é a classe principal responsável pela lógica do jogo e teve de ser modificada.

1º passo (método dos movimentos)

No começo, focámo-nos no método responsável pelos movimentos, sendo que no jogo é possivel mexer as peças para as 4 direções, para cima, baixo, esquerda e direita.

Neste método usamos a intrução for para percorrer toda o quadro do jogo, e consoante a direção que o utilizador manda mexer as peças, o programa verifica se há uma peça numa posição, e em caso afirmativo, o programa entra num ciclo e verifica se a posição seguinte está vazia e se estiver ele mexe a peça anterior para essa posição. Usamos uma instrução if para evitar que o programa realize vários movimentos seguidos sem parar, como por exemplo uma linha com 4 peças seguidas iguais, sem esta instrução ele juntava as 4 e é suposto fazer duas junções de cada vez.

Caso o utilizador faça um movimento numa direção em que a peça seguinte é de igual valor, ele mexe a peça na direção pedida e cria uma nova peça com o valor a dobrar.

2º passo (modificar o método processKey)

Este método foi criado pelo professor, mas achámos necessidade de o modificar. Simplificámos e criámos um caso para que se utilizador prima a tecla N o programa chama o método newGame para começar um jogo novo.

O método newGame começa por perguntar ao utilizador se queremos fazer um novo jogo. Em caso afirmativo o programa grava o score feito e limpa o quadro de jogo com o método resetTiles. É também iniciado o método init responsável pela apresentação inicial. Esta apresentação é composta pela explicação das teclas de jogo e pela tabela dos top 5 scores feitos, inicia o score e o número de movimentos a 0.

3º passo (método resetTiles)

Para fazer reset percorremos o array Tiles com a instrução for e colocamos cada indice do array vazio a 0.

4º passo (criar o método randomTile)

Este método cria uma peça numa posição aleatória que esteja vazia.

Para isso criámos um ciclo que usa duas variáveis para os indices do array Tiles da grelha em que são afetadas por valores aleatorios com a condição de essa posição gerada aleatóriamente não estar a afetar nenhuma posição já preenchida. Esta peça criada tem como valores, 2 e 4, com o 4 a ter 10% de probabilidade em relação ao 2.

5º passo (método que avalia a condição de vitória)

Aqui criámos um método que percorre todos os índices do array Tiles para verificar a ocorrencia de uma peça com o valor 2048, o objetivo deste jogo é chegar a este valor.

Caso o score feito seja válido para entrar no top 5 dos scores ele grava no array dos Scores o valor e elimina o score mais baixo da tabela.

6º passo (criar o método gridFull)

Neste método percorremos todos os indices do array Tiles e verifica se cada indice está ocupado. A cada verificação, caso esteja a posição ocupada, incrementa uma variável **a** criada. No final, se a variável **a** tem o valor do numero total de quadrados da grelha significa que a grelha está totalmente ocupada e o método retorna true.

7º passo (método noMoves)

Este método serve para diferenciar os casos em que a grelha está totalmente preenchida, mas com possibilidade de movimentos do caso em que não dá para mexer mais.

Para isso chama o método gridFull como condição na instrução if, com um for percorre os indices do array e a cada indice compara a peça em baixo e à direita. Tivemos de criar duas condições para os casos em que o for chega à ultima linha ou à ultima coluna, nesses casos, caso chegue à ultima linha compara para a direita apenas, caso chegue à ultima coluna compara para baixo.

Se chegar à peça que está na ultima linha e na ultima coluna dá break.

8º passo (método da derrota)

O método que verifica a condição de derrota chama os métodos noMoves e o gridFull como condições. Caso a grelha esteja cheia e não haja possibilidades de movimento o programa mostra uma mensagem a dizer que perdemos e adiciona o score à tabela dos topScores caso seja elegível para tal e o jogo é terminado após isso.

9º passo (Classe TopScores)

Tivemos necessidade de alterar a classe TopScores fornecidade pelo professor.
Alterámos os 3 métodos pedidos pelo professor, o canAdd e o addRow e o getNumOfRows.

O getNumOfRows serve apenas para retornar o valor de linhas ocupadas na tabela dos scores.

O canAdd verifica se o score com que acabámos o jogo é superior ao ultimo score obtido, ou se o número de linhas ocupadas na tabela é menor ao numero máximo de linhas a ocupar, para saber se é elegível para entrar no topscores. Caso seja, retorna true.

O método addRow é o método responsável pela criação da coluna com o score, primeiramente analisámos a condição em que ainda não existe nenhuma linha ocupada, e depois outra para os restantes casos, sendo que se o número de linhas ocupadas fosse menor que o numero de linhas a ocupar, é criada uma nova linha no fim. Existem 2 ciclos repetitivos neste método, o 1º serve para percorrer o score de todas as linhas ocupadas sendo que ele para na linha em que o novo score vai ter de ficar, o outro ciclo serve para percorrer todos as linhas desde a ultima até à que o novo score vai ficar e vai colocar todos os scores que são menores que o novo na linha abaixo, sendo que o ultimo score vai desaparecer.