UNIVERSIDADE
DE ÉVORA

FUSION

```
------- Menu ------
1- Play The Game
2- Exit
Choose your option: 1

What's the size of the grid N x N: 9
How many colors do you wanna play with 2-9: 9

     0 1 2 3 4 5 6 7 8
     ▼ ▼ ▼ ▼ ▼ ▼ ▼ ▼ ▼

0 ▶  1 3 9 3 5 2 5 2 3
1 ▶  1 2 5 1 4 7 3 2 5
2 ▶  4 3 7 3 7 8 4 3 2
3 ▶  7 1 6 1 7 1 7 3 6
4 ▶  1 2 2 5 4 3 3 8 8
5 ▶  1 6 8 4 8 5 2 8 9
6 ▶  8 3 5 2 5 5 3 8 2
7 ▶  4 1 7 6 4 9 4 9 9
8 ▶  2 4 6 3 7 4 3 2 9

Row: 5
Col: 7

Score: 16
```

Introdução

Este trabalho consiste na criação de um código cujo objetivo é desenvolver um jogo apelativo, de fácil entendimento.

Foi implementado na linguagem java, através dos conhecimentos adquiridos em aula.

Este relatório explica o modo de funcionamento do jogo e explica detalhadamente o código que o originou.

Esta também associada uma parte gráfica.

Breve Apresentação

O jogo começa com um "Menu" que permite ao utilizador selecionar uma das duas opções possíveis: "Play the Game" ou "Exit".

Caso a opção selecionada seja a primeira (através do input do número "1"), o jogo cria uma tabela com vários números.

Após decidir começar a jogar, é-lhe possível escolher o tamanho da grelha de jogo e o número de cores com que pretende jogar.

Se a opção selecionada for "2" o jogo termina.

É possível obter o score após cada jogada. Esse score corresponde a um valor de pontuação calculado pelo quadrado da soma das casas eliminadas.

Sempre que o jogador visualizar três ou mais números seguidos, deve escrever, após ser dada a indicação, o número da linha e coluna associadas.

Neste caso, pode escolher qualquer um dos números que se encontram seguidos, isto é, o primeiro, último ou o/os do meio que o programa é capaz de reconhecer os que estão à sua volta, apaga-los e substituir a linha ou a coluna por novos números.

Código (sem a componente gráfica)

- **Main**
  A classe é responsável por iniciar o jogo automaticamente.

- **Fusion**
  Dentro da classe Fusion encontram-se variáveis globais e métodos essenciais ao programa.

  O programa é composto por variáveis privadas (n e c), um array bidimensional "grid", dois arrayList (uma de inteiros e outra de strings), um booleano e vários inteiros públicos (score, choice, row, col) .

  input_num – Este método contém um scanner para ter acesso ao tamanho do tabuleiro que se pretende criar. No caso da deteção de qualquer input que não seja válido, o programa reconhece-o como exceção, através de um try/catch , e devolve a mensagem correspondente ao erro.

  input_color – É dada a oportunidade ao utilizador para escolher com quantas cores pretende jogar (entre 2 a 9). Contém um scanner para o input do utilizador e, qualquer input inválido, passa por uma exceção.

  set_grid – Para se obter a grelha de n x n (com n definido pelo utilizador no inicio do jogo), passa-se para um ciclo "for" no qual se cria o tabuleiro.

  Cada coluna e cada linha estão devidamente identificadas nas laterais, desde 0 (zero) até n-1 (tamanho de grelha escolhido menos um).

  get_grid – Mostra ao utilizador o tabuleiro criado, previamente gerado pelo método set_grid.

  gameOver – Verifica se existe a possibilidade de jogar, isto é, se existem pelo menos 3 casas com o mesmo valor (cor) seguidas. Caso contrário retorna state igual a false.

  checkingHorizontal – Verifica as posições na horizontal iguais à posição selecionada.

  checkingVertical- Verifica as posições na vertical e substitui na grid, desce os números que forem necessários descer e, eventualmente, podem ser criados um ou mais valores na grid aleatoriamente entre 1 e c, este método retorna também o valor counter, que é o número de casas removidas.

  pop – Estão envolvidos um random e dois arraylist (removable e passed). Chama o checkingHorizontal dentro de uma variável checkerH. Se o valor de checkerH for maior que 3 vai eliminar o número de casas que estiverem na arraylist "removable". Adicionando um valor random entre 1 e c às casas que estiverem na row = 0. O pop chama também o método checkingVertical, com este completa o valor de score que é o quadrado da soma das casas removidas.

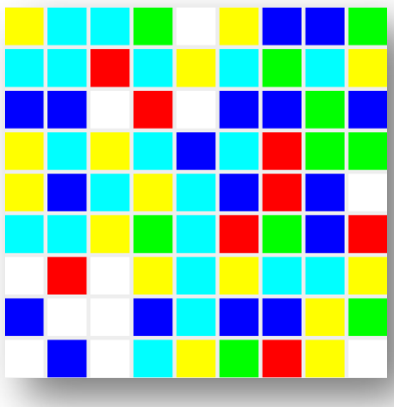
menu – Dá origem a um menu em que as opções disponíveis são jogar ou sair, através do input "1" ou "2", respetivamente. Caso os valores inseridos pelo utilizador sejam diferentes de estes dois, o input é reconhecido como um erro, por um Try/Catch e o utilizador recebe uma mensagem de erro. Neste caso, volta a poder introduzir uma das opções do menu. Este processo repete-se sempre, até que sejam obtidos inputs válidos.

selection – Após ser visualizada a grelha com todos os números gerados, o jogador deve escolher a linha (row) e a coluna (col) de uma das diferentes casas seguidas identificadas pelo mesmo. Caso a coluna ou linha escolhidas não estejam compreendidas entre 0 e n-1, segue-se para uma exceção e é dada a oportunidade de o utilizador fazer um novo input, após a visualização da respetiva mensagem de erro.

runner – Se a opção, no método "menu" for a de jogar ("1"), são chamados os métodos responsáveis pela escolha da grelha e do número de cores (números) com que o jogador pretende jogar ("input_num" e "input_color") e, através de um ciclo, corre sempre que o "state" for true, ou seja, sempre que for possível gerar um tabuleiro de jogo, pelo método "set_grid". Chama ainda os métodos "pop", "get_grid", "selection", e "game over" e permite também ter acesso ao "score". É o método que permite a interação entre todos os outros, tornando o jogo eficiente e funcional.

Sempre que não existirem mais possibilidades de jogo, o programa termina e mostra a pontuação obtida durante o jogo.

Código (sem a componente gráfica)



- **Main1Graphics**
  É a classe responsável por iniciar o jogo automaticamente.

- **Fusion1Graphics**
  Dentro da classe Fusion encontram-se variáveis globais e métodos essenciais ao programa.

O programa é composto por variáveis privadas (n e c),
um array bidimensional "grid", dois arrays List (uma de inteiros e outra de strings), um booleano definido como falso, vários inteiros públicos (score, rowIndex, columnIndex), um Jbutton e um Jframe.

input_num – É o método que pede ao utilizador para definir o tamanho do jogo. Sempre que o input não seja um número inteiro ou que não esteja dentro do valor pretendido 3 – 50, o programa volta a pedir um input válido.

input_color – Nesta fase, pergunta-se o número de cores com que o jogador prefere jogar, definindo, assim, um nível de dificuldade. Com o input diferente de um número compreendido entre 2-9 cores, será pedido um input que seja válido.

set_grid – É o método capaz de criar o jogo, isto é, através do n introduzido o programa cria uma tabela e uma Jframe, sendo que o tamanho da tabela está compreendido entre 0 a n-1.

buttons – É o método responsável pela criação dos botões, por lhes dar a cor e a capacidade de terem ação após o mouseclick.

get_grid – É apresentado o jogo, criado pelo método set_grid. Este método inclui a possibilidade de visualizar uma janela com vários botões de diferentes cores, chamando o método buttons.

graphics – Associa a cada número uma cor para ser depois distribuída aleatoriamente a cada botão.

gameOver – Verifica se existe a possibilidade de jogar, isto é, se existem pelo menos 3 casas com o mesmo valor (cor) seguidas. Caso contrário retorna state igual a false.

checkingHorizontal – Verifica as posições na horizontal iguais à posição selecionada.

checkingVertical- Verifica as posições na vertical e substitui na grid, desce os números que forem necessários descer e, eventualmente, podem ser criados um ou mais valores na grid aleatoriamente entre 1 e c, este método retorna também o valor counter, que é o número de casas removidas.

pop - Estão envolvidos um random e dois arraylist (removable e passed). Chama o checkHorizontal dentro de uma variável checkerH. Se o valor de checkerH for maior que 3 vai eliminar o número de casas que estiverem na arraylist “removable”. Adicionando um valor random entre 1 e c às casas que estiverem na row = 0. O pop chama também o método checkingVertical, com este completa o valor de score que é o quadrado da soma das casas removidas. Remove, também, todo o conteúdo do Jframe, sendo este adicionado com as novas alterações pelo método get_grid, é verificado também se existem possíveis jogadas, caso não existam, o programa termina devolvendo o Score ao utilizador.

Conclusão

No âmbito da cadeira Programação II, foi desenvolvido, com relativa facilidade, um jogo na linguagem Java. Foram encontradas dificuldades na realização da parte gráfica, no que diz respeito à movimentação de peças, todas ultrapassadas à medida em que se foi ganhando mais à-vontade com a linguagem.

Este trabalho ajudou a descobrir novas formas mais eficientes de realizar as mesmas operações.

Pensamos ter ido ao encontro de tudo o que era pedido e ter chegado ao objetivo do trabalho.