Projeto de Programação Multiparadigma [v1.0]
2023/2024

Enunciado

Um projeto de programação multiparadigma tira partido das vantagens de diferentes paradigmas de programação de forma a produzir uma solução mais adequada (p.e. desempenho, leitura/interpretação, manutenção). Uma divisão clássica consiste na utilização do paradigma de programação orientado aos eventos para o desenvolvimento da **camada de apresentação** (*User Interface*) e outro(s) paradigma(s) (funcional, orientado aos objetos, imperativo, lógico) para o desenvolvimento da **camada de negócio**. A **camada de dados** completa uma típica arquitetura de 3 camadas.

Aviso: O trabalho entregue deve corresponder ao esforço individual de cada grupo, e em nenhum caso deve ser copiado código que será entregue. A deteção de código copiado será realizada por software especializado bastante sofisticado. Casos de plágio óbvio serão penalizados com a anulação do projeto, o que implica a reprovação à Unidade Curricular (UC). Adicionalmente, a situação será reportada à Comissão Pedagógica da ISTA/Conselho Pedagógico do ISCTE-IUL. Serão penalizados da mesma forma tanto os alunos que fornecem código (se for o caso) como os que copiam código.

Grupos de Trabalho

O trabalho deverá ser realizado por grupos de **3 elementos** (exceções necessitam de aceitação por parte do coordenador da UC).

Datas

- Entrega nº1: até 8 de abril (via Moodle) contendo a 1ª parte (no mínimo com as tarefas T1, T2, T3, T4, T5 e T8)
- Discussão nº1: 15 a 19 de abril (no horário habitual das aulas)
- Entrega nº2: até 12 de maio (via Moodle) contendo a 1ª parte (melhorada) e a 2ª parte
- Discussão nº2: durante uma das aulas (podem ser necessárias *slots* adicionais) da última semana de aulas – 13 a 17 de maio

Introdução

O objetivo deste projeto é o desenvolvimento de competências de programação que permitam desenvolver, em colaboração, aplicações multiparadigma interativas de média escala. O projeto divide-se em duas partes. A primeira parte corresponde ao desenvolvimento da camada de negócio utilizando, obrigatoriamente, a linguagem de programação **Scala** seguindo os princípios de programação funcional pura (exceções serão explicitamente indicadas). A segunda parte corresponde ao desenvolvimento da camada de apresentação (interativa) que deverá ser desenvolvida utilizando **JavaFX**.

Parte 1: Camada de Negócio

Objetivos: Desenvolver um conjunto de métodos, utilizando o paradigma de programação funcional, capazes de garantir os objetivos funcionais da solução proposta.

Parte 2: Camada de Apresentação Interativa

Objetivos: Desenvolver uma *text-based User Interface* (TUI) e uma *Graphical User Interface* (GUI) que permitam uma interação fácil e intuitiva com a aplicação.

Requisitos

O projeto deverá, adicionalmente, satisfazer os seguintes requisitos:

Funcionais

1. Permitir executar com sucesso as várias tarefas/métodos (ver secção Tarefas);
2. Manter estado entre execuções, i.e., guardar o estado de jogos não terminados para execuções futuras.

Não funcionais

1. Apresentar ambas interfaces: gráfica (GUI) e textual (TUI);
2. Os utilizadores deverão conseguir interagir corretamente (com o menor número de erros cometidos) e com facilidade (equilíbrio adequado entre quantidade de informação apresentada e número de ações necessárias para realizar uma tarefa) sem necessidade de treino.

Tema – Jogo de tabuleiro *Sopa de Letras "ZigZag"*

Neste trabalho prático ocupar-nos-emos com a implementação do jogo de tabuleiro Sopa de Letras[1] alterado. Ou seja, nesta versão alterada, as letras (somente A-Z) de cada palavra não têm de estar na mesma direção (i.e., horizontal, vertical ou diagonal) podendo estar em qualquer direção (p.e., em zigzag). No entanto, cada letra só pode ser utilizada uma única vez (ver Figura 1). O objetivo do jogo mantem-se inalterado, i.e., é necessário encontrar as palavras escondidas tão rapidamente quanto possível. A lista de palavras escondidas é fornecida.

Tarefas

- **T1**: implementar o método `randomChar(rand:MyRandom):(Char, MyRandom)` responsável por gerar um caracter aleatório válido (i.e., A-Z). A assinatura do método recebe e devolve `MyRandom` para poder ser puro.
  O tabuleiro deve ser preenchido de forma aleatória diferente em cada execução do programa;
- **T2**: implementar o método `fillOneCell(board:Board, letter: Char, coord:Coord2D):Board` responsável por atribuir um caracter dado válido a uma coordenada do tabuleiro dada;
  Nota: pode utilizar o método `fill` (da classe `List`) para inicializar o `board`.
- **T3**: implementar o método `setBoardWithWords(board:Board, words:List[String], positions:List[List[Coord2D]]: Board` responsável por adicionar ao tabuleiro, uma lista de palavras nas coordenadas dadas (ambas palavras e coordenadas podem ser obtidas a partir da leitura de ficheiro de texto com formatação à sua escolha);
- **T4**: implementar o método de ordem superior `completeBoardRandomly(board:Board, r:MyRandom, f: MyRandom => (Char, MyRandom)):(Board, MyRandom)` responsável por completar de forma aleatória as coordenadas "livres" (i.e., sem letras correspondentes a uma palavra a procurar) do tabuleiro. Este método irá operar sobre o resultado do método anterior;
- **T5**: implementar o método `play` responsável por verificar se a palavra dada existe no tabuleiro a partir da posição (i.e., `Coord2D`) e direção inicial (e.g., `NorthWest` do enumerado `Direction` - ver secção seguinte e Fig. 1) indicadas;
- **T6**: implementar o método `checkBoard` responsável por validar o tabuleiro. Ou seja, verificar que o tabuleiro contém todas as palavras (pode definir um número máximo) que devem ser procuradas, sejam válidas e, nenhuma esteja duplicada.
- **T7**: adicionar um temporizador (à sua escolha), pontos (regras de cálculo à sua escolha), e um método para determinar se o jogo terminou.
  Pode utilizar o método `System.currentTimeMillis()` para obter o tempo atual;
- **T8**: desenvolver uma TUI disponibilizando as várias opções de interação (p.e., iniciar tabuleiro, selecionar palavra, reiniciar, alterar cor do texto) do jogo (tabuleiro de dimensão configurável);

[1] https://pt.wikipedia.org/wiki/Caça-palavras

- **T9**: desenvolver uma GUI (tabuleiro de dimensão fixa: 5x5) com as mesmas opções que a TUI e, adicionalmente, uma interação com usabilidade mais adequada (p.e., ao identificar uma palavra no tabuleiro).

Fatores de valorização:

- Otimizar a eficiência do preenchimento do tabuleiro substituindo os métodos das tarefas T3 e T4 por um método que percorre o tabuleiro uma só vez;
  Para efeitos de avaliação mantenha os métodos das tarefas T3 e T4 no projeto.
- Otimizar a eficiência da tarefa T6;
- Utilização de conceitos de programação funcional "avançada" (p.e., *fold*, *tail recursion*);

Tipo de Dados a Utilizar Obrigatoriamente

```
type Board = List[List[Char]]

type Coord2D = (Int, Int) //(row, column)

object Direction extends Enumeration {
     type Direction = Value
     val North, South, East, West,
     NorthEast, NorthWest, SouthEast, SouthWest = Value

     //(…)

}
```

Exemplos de Execução do Jogo *Sopa de Letras "ZigZag"*

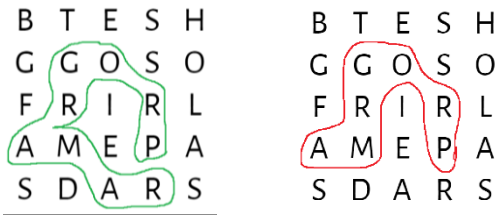

Figura 1 – Exemplos de tabuleiros de sopa de letras "Zigzag". À esquerda: seleção válida da palavra *programar* (a começar na coordenada (3,3) e direção inicial *North*). À direita: seleção inválida da palavra *programar*

Avaliação

A realização do projeto é obrigatória para obter aprovação à UC. Não existe qualquer possibilidade de obter aprovação à UC sem realizar o projeto. O projeto é composto por duas partes e realizado em grupo, no entanto, as discussões e aferições são individuais. As classificações possíveis na aferição individual são A, B ou C, definindo a nota final individual do projeto (*nf*):

- A: *nf* igual à classificação do projeto;
- B: *nf* igual a 80% da classificação do projeto;
- C: reprovação à UC.

A **primeira parte** do projeto (com peso de **80%**) será inicialmente avaliado em termos funcionais (i.e., se os métodos produzem os resultados esperados) e se cumpre os requisitos funcionais (para ser usado numa primeira fase via TUI). A solução desenvolvida deverá, obrigatoriamente, aplicar as seguintes matérias de programação funcional introduzidos na UC:

- Recursividade (no lugar de iteratividade) e *Pattern matching*;
- Imutabilidade (exceto na camada interativa);
- *Pure functions* (exceto na camada interativa);
- Função de ordem superior;
- *Functional Object-Oriented programming*.

Ainda que a funcionalidade exigida tenha sido concretizada, a não aplicação ou aplicação inadequada das matérias mencionadas poderão diminuir a classificação. A avaliação também poderá ser penalizada em função da reduzida qualidade do código (p.e., difícil manutenção, baixa eficiência, dificuldade de interpretação, duplicação, complexidade desnecessária).

A **segunda parte** do projeto (com peso de **20%**) será avaliada em termos da qualidade e usabilidade das interfaces (TUI e GUI) desenvolvidas, do cumprimento dos requisitos e da correta aplicação da matéria sobre engenharia de sistemas interativos (*Event-driven Programming*).

Apesar de não ter um peso na avaliação a utilização de tecnologias para suporte à programação em equipa (e.g., Git, GitHub) é aconselhado.

Os alunos poderão obter feedback juntos dos professores das respetivas turmas sobre o progresso do projeto, e deverão seguir as recomendações dadas.

Os projetos são classificados de acordo com os seguintes pesos:

1ª parte (80%):

- Requisitos funcionais – 40%
- Fatores de valorização – 15%
- Correta e completa aplicação da matéria de Programação Funcional – 20%
- Qualidade do código – 5%

2ª parte (20%):

- Qualidade das interfaces desenvolvidas – 15%
- Correta aplicação da matéria de sistemas interativos – 5%

A versão do projeto da entrega nº1 terá um peso de 10% na nota final e será classificado da seguinte forma:

- 2 valores – no mínimo, 5 das tarefas solicitadas para a entrega nº1 implementadas de forma adequada;
- 1 valor – no mínimo, 3 das tarefas solicitadas para a entrega nº1 implementadas de forma adequada;
- 0.5 valores – no mínimo, 2 das tarefas solicitadas para a entrega nº1 implementadas de forma adequada;
- 0 valores – caso contrário.

As tarefas da entrega nº1 podem ser corrigidas/melhoradas na entrega nº2 para melhorar a avaliação final do projeto, no entanto, a nota da entrega nº1 ficará inalterada.

Entrega

Submissões: ambas as entregas deverão ser, obrigatoriamente, realizadas por cada grupo através do Moodle.

Dados e formatação:

- Projeto com todo o código fonte desenvolvido em formato *Zip* ("File/Export/Project to Zip File…") **contendo todos os dados necessários para poder testá-lo**. O nome do projeto deve incluir o **número do grupo e nome dos vários membros** (*NúmeroGrupo_Nome1_Nome2_Nome3*).
  Sugestão: verifiquem que a importação do vosso projeto numa máquina diferente ocorre com sucesso antes de efetuarem a submissão;
- Possíveis comentários/considerações (ficheiro: Readme.txt).

O ficheiro Readme.txt deve ser adicionado à raiz do projeto.

Cada membro do grupo deve submeter, individualmente, na entrega final a autoavaliação do trabalho em grupo (ficheiro: grupo_*NúmeroGrupo*.pdf) – ver tabela no anexo I. Este ficheiro deverá ser submetido através do Moodle num outro link de submissão que será criado para o efeito.

Discussão: a discussão é individual e todos os alunos terão de demonstrar ser capazes de fazer um trabalho com o nível de qualidade igual ao que entregaram. Ao entregar o trabalho, os alunos implicitamente afirmam que são os únicos responsáveis pelo código entregue e que todos os membros do grupo participaram de forma equilibrada na sua execução, tendo todos adquirido os conhecimentos necessários para produzir, individualmente, um trabalho equivalente.

Anexo I - Avaliação do trabalho em grupo

Classifique os membros do seu grupo (incluindo-o a si próprio) acerca dos itens seguintes numa escala até 20. Coloque o nome de cada membro no cabeçalho da respetiva coluna.

Responsabilidades	***Minha (preencher)***	***MembroA (preencher)***	***MembroB (preencher)***
Contribuiu para a sua parte do trabalho			
Assistiu a todas as reuniões calendarizadas			
Foi cooperativo			
Teve uma atitude positiva			
Foi tolerante a outros pontos de vista			
Teve espírito de grupo			
Completou todo o trabalho a si atribuído e no tempo estipulado			
Esforçou-se para produzir resultados de qualidade			
Incluiu os outros membros nas decisões tomadas e discussões			
Ajudou os outros membros do grupo			