Programação Avançada

2020/21

Trabalho Prático

Enunciado

Pretende-se desenvolver uma versão do conhecido jogo quatro-em-linha, que é um jogo de tabuleiro para dois jogadores (ver figura). Em cada novo jogo deve ser possível optar por uma partida entre dois jogadores humanos, ou um jogador humano e um virtual, ou dois jogadores virtuais. Um jogador virtual deve jogar corretamente (respeitando as regras do jogo), mas sem a necessidade de recorrer a estratégias avançadas como técnicas específicas de Inteligência Artificial.

O tabuleiro está na posição vertical possui uma largura de 7 células e uma altura de 6 células. Cada jogador, alternadamente, deixa cair uma peça numa coluna à sua escolha, fazendo com que as peças se acumulem até um máximo de 6 peças por coluna (altura do tabuleiro). Ganha o jogador que conseguir colocar 4 peças seguidas em linha, na vertical, horizontal ou diagonal.

O jogo inicia-se com a introdução do nome do jogador, sendo os nomes obrigatoriamente diferentes. De seguida é sorteado qual o jogador (seja ele virtual ou humano) a efetuar a primeira jogada.

Depois de iniciado, o jogo desenrola-se da forma tradicional (do jogo 4 em linha tradicional), mas com a seguinte diferença: após cada quatro jogadas de cada jogador (ou seja, após 8 jogadas alternadas), aos jogadores humanos (nota: apenas para os humanos) é dada a hipótese (que pode não aproveitar) de realizar uma atividade suplementar (mini-jogo independente) para ganhar uma peça especial. A atividade a propor ao jogador será alternadamente uma das seguintes:

- Serão propostos cálculos matemáticos simples usando os operadores básicos (+,-,/,x) e dois inteiros positivos (de 1 ou 2 dígitos cada um). Os números e os operadores a usar são sorteados. Caso o jogador acerte em 5 cálculos em menos de 30 segundos ganha a peça especial;
- São apresentadas 5 palavras escolhidas aleatoriamente de um ficheiro de texto com 100 ou mais palavras, cada uma de 5 ou mais letras. O jogador deve escrever essas palavras e será avaliada a sua rapidez. É contabilizado o tempo que passa entre a apresentação da sequência dessas palavras e a indicação de finalização da escrita por parte do jogador. O jogador ganha a peça especial se digitar corretamente as palavras num período inferior ou igual ao número de segundos que corresponde a metade do número de caracteres apresentados, incluindo os espaços em branco.

Quando um jogador humano ganha um mini-jogo, obtém uma peça especial que poderá jogar imediatamente ou guardar para uma jogada posterior. A peça especial que o jogador ganha tem o efeito de fazer desaparecer todas as peças da coluna onde é jogada.

Quando o jogador ganha o mini-jogo, ou seja, ganha a peça especial, mantém a sua vez de jogar. Significa que a participação no mini-jogo não substitui a sua vez de jogar no jogo principal (4 em linha). No entanto, caso o jogador perca a atividade proposta, então perde a sua vez de jogar no 4 em linha, passando a vez ao outro jogador (virtual ou humano).

Um jogador virtual não tem acesso aos mini-jogos e, consequentemente, não poderá jogar peças especiais.

A evolução do contexto (informação) do jogo quatro-em-linha, resultante das jogadas, deve ser armazenada para ser possível "voltar atrás" (reverter o jogo a uma situação anterior). Esta hipótese deverá ser possível enquanto o jogo não termina (assim que um jogador, humano ou virtual, ganhar, deixa de ser possível "voltar atrás"). Cada jogada que é desfeita consome um crédito, sendo dados inicialmente 5 créditos a cada jogador humano. A jogada desfeita é sempre a anterior, independentemente de ter sido feita pelo próprio ou pelo outro jogador. O jogador pode optar por usar um crédito de cada vez ou vários em conjunto. Quando volta atrás, o jogador não recupera as peças especiais ganhas e a contagem de grupos de 4 jogadas para acesso aos mini-jogos é colocada a zero. Esta funcionalidade não abrange os mini-jogos, ou seja, a ocorrência de um mini-jogo é ignorada na identificação de qual é "jogada anterior" (passa para a jogada antes do mini-jogo).

Quando um jogo termina, deverá ser dada a oportunidade de iniciar um novo jogo.

Deverá também ser mantido um histórico dos últimos 5 jogos completos para possibilitar o "replay" completo desses jogos. Este histórico, não inclui o desenrolar dos mini-jogos, mas apenas os resultados destes. Note-se que esta funcionalidade não é a mesma do "voltar atrás" referido acima.

Regras gerais

O trabalho deve ser realizado de forma **individual**.

A elaboração do trabalho está dividida em duas fases separadas.

As datas de entrega do trabalho nas duas fases são as seguintes:

1. Primeira fase, que inclui o modelo do jogo utilizando os padrões apresentados nas aulas e interface do utilizador em modo texto: **23 de maio**;

2. Segunda fase, correspondente à funcionalidade completa do jogo, incluindo uma interface do utilizador em modo gráfico baseada em JavaFX: **13 de junho**.

Mais abaixo são dados mais pormenores acerca dos requisitos a cumprir em cada meta.

As entregas correspondentes às duas fases do trabalho devem ser feitas através do *moodle* num ficheiro compactado em formato ZIP. O nome deste ficheiro deve incluir o primeiro nome, o último nome e o número (novo) de estudante do aluno, bem como a indicação da turma prática a que pertence (ex: P3-201906104-Antonio-Ferreira.zip; utilize a designação PL para a turma do pós-laboral).

O ficheiro ZIP deve conter, pelo menos:

- O projeto com todo o código fonte produzido;
- Eventuais ficheiros de dados e recursos auxiliares necessários à execução do programa;
- O relatório em formato pdf.

O relatório deve incluir em ambas as fases:

1. Uma descrição sintética acerca das opções e decisões tomadas na implementação (máximo uma página);

2. O diagrama da máquina de estados que controla o jogo, devidamente explicado; neste diagrama, o nome atribuído às transições de estado deve corresponder ao nome dado às funções da hierarquia de estados a que correspondem e o nome dos estados deve também corresponder ao nome das classes que os representam;

3. Diagramas de outros padrões de programação que tenham eventualmente sido aplicados no trabalho;

4. A descrição das classes utilizadas no programa (o que representam e os objetivos);

5. A descrição do relacionamento entre as classes (podem ser usados diagramas);

6. Em ambas as fases, para cada funcionalidade ou regra do jogo, a indicação de cumprido/implementado totalmente ou parcialmente (especificar o que foi efetivamente

cumprido neste caso) ou não cumprido/implementado (especificar a razão). O uso de uma tabela pode simplificar a elaboração desta parte.

Ambas as fases estão sujeitas a defesas que incluem a apresentação, explicação e discussão do trabalho apresentado. Estas podem incluir a realização de alterações ao que foi entregue.

À primeira e segunda fases do trabalho correspondem, respetivamente, as cotações de **8** e **6** valores**.**

Objetivos e requisitos

Os objetivos das duas fases do trabalho prático são os seguintes:

1. **Primeira Fase**:
    a. Implementação do jogo em modo texto;
    b. Suporte para dois jogadores humanos, humano vs. computador ou computador vs. computador;
    c. Gravação/carregamento do jogo (um jogo carregado de ficheiro deve permitir a sua continuação).
2. **Segunda Fase** (adicionalmente ao implementado na primeira fase):
    a. Implementação do jogo em modo gráfico
    b. A interface do utilizador em modo gráfico deve estar de acordo com o padrão de notificações assíncronas estudado nas aulas (deve ser usado JavaFX e não são aceites soluções baseadas em Swing).

A implementação do trabalho deve **obrigatoriamente** obedecer aos requisitos seguintes:

1. Devem ser seguidos os padrões apresentados nas aulas;
2. Deve ser aplicado, de forma adequada, o padrão máquina de estados para concretizar a lógica do jogo;
3. Ao terminar um jogo, a máquina de estados deve permitir jogar de novo ou terminar a aplicação;
4. Os momentos do jogo em que o jogador tem que tomar decisões devem corresponder a estados. Podem existir estados cujo objetivo é principalmente informar o jogador acerca do ponto da situação e aguardar que indique para prosseguir. A existência de estados nestas situações depende apenas da dinâmica que se pretende dar à interação com o utilizador;
5. A aplicação deve apresentar toda a informação necessária ao acompanhamento e verificação do bom funcionamento do jogo;

6. O modelo do jogo (máquina de estados) deve, através de um *log* (“registo”) interno, gerar e disponibilizar informação detalhada sobre o estado interno e o resultado das várias ações. Ter em atenção que “disponibilizar” não significa “mostrar”, mas sim “tornar acessível” ou “retornar”;

7. Em ambas as fases do trabalho, o código deve ser estruturado de maneira que a lógica não dependa da forma de interação com o utilizador (na lógica não pode ser pedida ou mostrada informação ao utilizador). Nas classes relacionadas com a interação com o utilizador não pode haver processamento direto de regras nem dados internos da lógica: apenas se podem invocar funções cuja execução a lógica gere internamente.

Estrutura de projecto

A aplicação deve estar organizada em *packages*, incluindo os seguintes:

- ***jogo*** – package que abrange toda a aplicação
- ***jogo.logica*** –tem a classe que constitui a fachada de acesso à lógica, que internamente distribui
- as responsabilidades que lhe são pedidas, gerindo a dinâmica de processamento através de uma máquina de estados;
- ***jogo.logica.estados*** – contém a hierarquia dos estados;
- ***jogo.logica.dados*** – contém as classes que representam as estruturas de dados e que disponibilizam toda a funcionalidade;
- ***jogo.iu.texto*** – classe(s) que implementa(m) o interface em modo texto;
- ***jogo.iu.gui*** – classes que implementam a interface em modo gráfico em JavaFx (apenas na segunda fase).