Projeto

André Souto

Unidade Curricular de
Programação Centrada em Objectos

2024/2025

Objectivos

- Algoritmia
- Tipos e subtipos
- Genéricos
- Herança
- Polimorfismo
- Encapsulamento

Antes de Começar

De modo a poder realizar este projecto deverá estudar e recordar o que aprendeu sobre os assuntos mencionados anteriormente.

Enunciado

Descrição

O Mastermind é um jogo de tabuleiro inventado por um carteiro israelita perito em telecomunicações nos anos 70. É um puzzle pensado para dois jogadores, um denominado *CodeMaster* e o outro *CodeBreaker*. Inicialmente o *CodeMaster* cria um código secreto composto por uma sequência de quatro cores (de entre seis disponíveis) e o objetivo do jogo é o *CodeBreaker* decifrar (através de algumas pistas dadas pelo *CodeMaster*) o código escolhido. Para decifrar a sequência certa o *CodeBreaker* pode fazer tentativas, também elas compostas por sequências de 4 cores de entre as 6 disponíveis às quais o *CodeMaster* vai respondendo com dois tipos de respostas:

a. nº de cores certas no sítio certo (assinalados pelos pinos mais pequenos pretos).

b. nº de cores certas no sítio errado (e que não conta com as cores certas no sítio certo).

O objetivo do jogo é decifrar a sequência do *CodeMaster* no menor número de jogadas possível.

O jogo *Bulls and Cows*, é um precursor do Mastermind do século 19. O jogo é em tudo igual ao Mastermind mas em que os dois participantes só podem usar duas cores e portanto a informação de b, em vez de ser quantas cores estão certas no sítio errado é a soma do número máximo de cada cor em posições erradas. Tal como no Mastermind, o desafio para o *CodeBreaker* é resolver o enigma com o menor número de tentativas, utilizando a lógica para interpretar as pistas e deduzir a sequência correta.

Objectivo do trabalho a desenvolver

Pretende-se que seja criado um programa que simule o jogo do Mastermid e a sua variante Bulls and Cows com algumas modificações:

1. Os pinos coloridos são substituídos, no caso das sequências, por letras dadas pelos enumerados *BinaryColour* and *MultiColor* que implementam a interface *Colour*.
2. Relativamente aos pinos correspondentes às respostas, estes são representados por dois inteiros, sendo o primeiro correspondente às cores certas no sítio certo, e o segundo às cores certas mas no sítio errado conforme descrito acima.
3. Ao contrário do jogo tradicional de tabuleiro em que o número máximo de tentativas é 10, o número de tentativas que o CodeBreaker pode fazer é apenas limitado pelo número de todas as possíveis combinações de sequências. Contudo na sua representação apenas iremos representar as últimas 10 tentativas.
4. O *CodeMaster* será simulado pelo computador.

Objectivo do trabalho

O objetivo do trabalho é implementar uma solução que permita a um utilizador jogar um jogo do tipo descrito acima com a criação de classes descritas a seguir.

O que fazer

Os alunos devem implementar:

1. Classe Code que implementa o serviço de Cloneable e tem definido o seguinte construtor:

   public Code(List< ? **extends** Colour> code) que partindo de uma implementação de lista com valores que sejam subtipos de code, constroi um novo código como a lista dada.

   e tem os seguintes métodos

   List<Colour> getCode() que devolve uma cópia da lista com as cores do código.

   int getLength() que devolve o tamanho da sequência do código.

   int[] howManyCorrect (CodeI other) que devolve um vector com dois inteiros, um que corresponde ao **a** (número de pins com com e posição certa) e ao **b** (número de pinos com cor certa mas na posição errada - que estão na posição errada) definidos na introdução ao mastermind.

   String toString() que devolve uma representação textual deste Code na forma [c1,c2,c3,c4].

   Code clone() que re-implementa o clone da classe Object e devolve uma cópia deste Code.

   boolean equals(Object obj) que re-implementa o método equals da classe Object e verifica se um dado obj é igual a este código.

2. A classe BullsAndCowsCode que é um subtipo de Code e, tem definido o seguinte construtor:

   BullsAndCowsCode(List<BianryColour> code) que é responsável por inicializar os atributos da classe.

   e além de ter definidos os métodos da classe Code, reescreve o método

   int[] howManyCorrect (CodeI other) que é responsável por dizer quantos pins brancos e pretos de other estão em posição errada.

3. Uma interface MastermindGame que tem definida a constante:

   public static final int *MAX_TRIALS* = 25;

   e define os seguintes métodos:

   void play (Code x) que despoleta uma jogada, verificando se o jogo não está terminado, calcula os valores **a** e **b** do código **x** relativamente ao segredo guardado em cada ronda do jogo. Este método é responsável por guardar a tentativa (sem que haja repetições) com os respectivos resultados, atualizar o score caso a ronda fique terminada e o número de jogadas.

boolean isRoundFinished() que diz se a ronda está acabada, isto é, se o código secreto foi desvendado ou o número máximo de rondas permitido foi atingido.

void startNewRound() que providencia o serviço de gerar um novo segredo para o jogo e reiniciar o número de tentativas para esta ronda;

Colour hint() que permite ao jogador obter uma ajuda de uma cor que é usada na sequência secreta de forma aleatória dada pelo mesmo gerador aleatório que é usado para a geração do código secreto.

int getNumberOfTrials() que permite ao jogador obter o número de tentativas realizadas na atual ronda do jogo

Code bestTrial () que devolve o código para esta ronda com melhor pontuação, isto é, com o maior valor de **a** e em caso de empate o que tem melhor valor de **b** e se tiverem ambas os mesmos valores a que aparece primeiro na ordenação lexicográfica das strings que representam os códigos.

int score() que indica qual a pontuação do jogo;

boolean wasSecretRevealed () que indica se o segredo da ronda foi desvendado.

4. Classe abstrata AbstractMastermindGame que implementa a interface MastermindGame. O construtor e os métodos abstratos são:

   AbstractMastermindGame(**int** seed, **int** size, Colour[] colours) que constroi um jogo Mastermind abstrato em que o tamanho do código secreto é dado por size, as cores a usar para a construção dos códigos é dado por colours e a semente seed do gerador de aleatórios usado para gerar os segredos e as ajudas. De notar que para guardar as tentativas e informação necessária para a representação textual será necessário usar outros atributos que devem também ser inicializados pelo construtor.

   abstract int score(); que há-de dizer o score atual do jogo

   abstract boolean isRoundFinished(); que indicará se a ronda está terminada;

   abstract boolean updateScore(); que sempre que uma ronda termine, atualizará o socre do jogo;

   e re-implementa o método

   String toString() que devolve uma representação textual deste Jogo, imprimindo o número de tentativas, o score, e o tabuleiro. Na representação do tabuleiro, se o segredo ainda não foi revelado, devem aparecer “?” nas cores e se já foi, o código secreto. Devem ainda imprimir as últimas 10 tentativas por ordem de tentativa (com a atualização devida caso já tenha sido experimentada);

5. Classe concreta BullsAndCows que implementa a classe AbstractMastermindGame que tem um construtor:

   public BullsAndCows(**int** seed, **int** size, Colour[] colours) que constroi um jogo Mastermind com cores binárias.

   e, tem redefinidos os seguintes métodos:

   public int score() que devolve o score do jogo num dado momento.

   public boolean updateScore(); que atualiza o score quando uma ronda é terminada acrescentando 2000 pontos.

   public boolean isRoundFinished(); que indica se a ronda está terminada;

   e re-implementa o método

   Colour hint() que para além do que está definido na classe abstrata atualiza o score para metade do valor atual.

6. Classe concreta MultiColourMastermindGame que implementa a classe AbstractMastermindGame que tem um construtor:

   MultiColourMastermindGame(**int** seed, **int** size, Colour[] colours) que constroi um jogo Mastermind com cores binárias.

   e, tem redefinidos os seguintes métodos:

   int score() que devolve o score do jogo num dado momento.

   boolean updateScore(); que atualiza o score quando uma ronda é terminada adicionando 100 pontos quando o código é descoberto com no máximo duas tentativas, 50 se o número de tentativas não for além 5 e 20 caso contrário. Estes valores são divididos pelo número de ajudas usadas +1.

   boolean isRoundFinished(); que indica se a ronda está terminada;

   e re-implementa o método

   Colour hint() que para além do que está definido na classe abstrata deve ser responsável por contabilizar quantas ajudas foram dadas em cada ronda.

7. Classe Main com um main de criação da total responsabilidade de cada grupo que exercite todas as funcionalidades do jogo que acabaram de criar.

8. Um documento com a descrição em UML da estrutura da solução apresentada.

Material Fornecido

São fornecidos: A interface Colour e os enumerados BinaryColour, MultiColour que implementam Colour para serem usados na construção dos códigos. De notar que as classes não estão completas no que diz respeito aos métodos necessários.

Observações durante a realização do projeto

- Algumas das especificações são deliberadamente deixadas laxas para dar oportunidade ao alunos de apresentarem soluções originais;
- Durante a realização do projeto é natural que surjam alguns problemas com algumas implementações que eu não antecipei. Aí devem falar comigo para esclarecer,
- Devem inferir a formação dos métodos toString dos inputs e outputs to testes;
- Os testes fornecidos não são exaustivos havendo situações limite que poderão não estar testadas com o intuito de serem avaliadas durante a prova oral;
- A classe Main não está implementada. Os alunos deverão usar a classe Scanner para permitir ao utilizador jogar o jogo.

Atenção

Antes de submeter o trabalho, certificando-se que **TODOS** os passos descritos no documento são cumpridos, que entrega todas as classes incluindo a classe Main bem como o documento com a descrição do UML.

O que entregar

Deve criar o ficheiro **projectoGrupoX.zip**, contendo todos os ficheiros pedidos. Cada grupo deve fazer apenas uma submissão (ou seja só um dos elementos do grupo é que deve submeter o trabalho, não precisam de submeter todos). Devem fazer as submissões até dia 30 de Novembro às 23h59.

Avaliação

Na avaliação do projecto serão tidos em conta, entre outras coisas, a qualidade e complexidade do código, a documentação, a correção da solução, desempenho e contributo de cada membro do grupo. Não se esqueçam que a nota é individual.