DAMA (IN)VÁLIDAS

André Rato (45517) e Jorge Salsinha (45447)

Introdução

O jogo consiste em colocar 8 rainhas num tabuleiro de xadrez, de modo a que nenhuma seja atacada por outra. Para isso, diz-se que as rainhas estão numa posição válida. Caso contrário, se uma rainha estiver em posição de ser atacado tem-se uma configuração inválida.

JAVA é uma linguagem de programação de alto nível e que utiliza a programação orientada por objetos. Esta linguagem tem várias vantagens, tais como:

- a presença da filosofia "WORA" (*Write Once, Run Anywhere*), o que poupa tempo, esforço e linhas de código;
- desaloca automaticamente qualquer espaço de memória não utilizado sem que o programador se preocupe com isso;
- permite documentar em JAVADOC, o que torna fácil a sua análise e a consulta da sua documentação.

Apesar de ser uma linguagem que utiliza POO, foi possível verificar que os programas em JAVA demoram mais tempo a ser compilados e ocupam também mais espaço na memória.

Classes e seus componentes

Para a implementação dos dois programas pedidos foi necessário criar algumas classes, nomeadamente a classe abstrata Peca e suas subclasses, e a classe Tabuleiro. Foi também necessário criar uma interface, a interface Fila.

- **Classe Peca (*public abstract class Peca*):** classe utilizada para caracterizar cada uma das posições do tabuleiro. Esta classe utiliza métodos de outras classes, tais como os métodos da classe Tabuleiro.
  - **Variáveis de instância:**
    - **linha (*private int linha*):** guarda a linha onde a peça está;
    - **coluna (*private int coluna*):** guarda a coluna onde a peça está.
  - **Construtor (*public Peca(Tabuleiro tab, int linha, int coluna*)):** guarda nas variáveis de instância os valores dos argumentos.
  - **Métodos da classe:**
    - **linha() (*public int linha()*):** devolve a linha da peça;
    - **coluna() (*public int coluna()*):** devolve a coluna da peça;
    - **podeIrPara(int linha, int coluna) (*public abstract boolean(int linha, int coluna)*):** veri-fica os movimentos das peças;
    - **ataca(Peca vitima) (*public final boolean ataca(Peca vitima)*):** veri-fica se a peça pode, ou não;
    - **vazia() (*public boolean vazia()*):** verifica se, no tabuleiro, a peça existe ou não.

- **Classe Rainha (*public class Rainha extends Peca*):** classe utilizada para atribuir características a uma peça.

  - **Variáveis de instância:**

- **tamanho (*int tamanho*):** guarda o tamanho do tabuleiro em que a peça está.
- **Construtor (*public Rainha(Tabuleiro tab, int linha, int coluna*)):** utiliza o construtor da classe super (Peca) e guarda o tamanho do tabuleiro na variável *tamanho*.
- **Métodos da classe:**
  - **podeIrPara(int linha, int coluna) (*public boolean podeIrPara(int linha, int coluna)*):** verifica se o movimento da Rainha é possível.

- **Classe Nada (*public class Nada extends Peca*):** classe utilizada para atribuir características a uma peça.
  - **Construtor (*public Rainha(Tabuleiro tab, int linha, int coluna*)):** utiliza o construtor da classe super (Peca).
  - **Métodos da classe:**
    - **podeIrPara(int linha, int coluna) (*public boolean podeIrPara(int linha, int coluna)*):** devolve false pois quando uma Peca é Nada, não existe movimento.

- **Interface Fila (*public interface Fila*):** Interface utilizada para caracterizar alguns dos constituintes do Tabuleiro. As classes *public Linha*, *public Coluna*, *public DiagonalDescendente* e *public DiagonalAscendente* têm todas os métodos abaixo referidos, cada uma com o seu modo de tratamento dos dados (ver JAVADOC das classes).
  - **Métodos da interface:**
    - **comprimento() (*int comprimento()*):** devolve o comprimento da fila;
    - **pecas() (*int pecas()*):** devolve o número de peças na fila;
    - **peca(int pos) (*Peca peca(int pos)*):** devolve a peça que está na posição *pos* da fila.

- **Classe Tabuleiro (*public class Tabuleiro*):** classe utilizada para tratar o tabuleiro e executar todas as operações possíveis relacionadas ao mesmo.
  - **Variáveis de instância:**
    - **string (*private String string*):** guarda a configuração do tabuleiro em formato string;
    - **tamanho (*private int tamanho*):** guarda o tamanho do tabuleiro;
    - **tabuleiro (*private Peca[][] tabuleiro*):** array de arrays de peças que criam o tabuleiro.
  - **Construtor (*public Tabuleiro(String string*)):** recebe a configuração do tabuleiro, remove os espaços existentes nessa configuração e calcula o tamanho do tabuleiro utilizando a raiz quadrada; de seguida, são percorridas todas as posições do tabuleiro e são lhes associadas peças consoante o caractere em análise.
  - **Métodos da classe:**
    - **getTamanho() (*public int getTamanho()*):** devolve o tamanho do tabuleiro;
    - **toString() (*public String toString()*):** devolve o tabuleiro em forma de string;
    - **peca(int linha, int coluna) (*public Peca peca(int linha, int coluna)*):** devolve o tipo de peça que está presente na posição recebida pelo argumento;

- **ameacada(int linha, int coluna) (*public boolean ameacada(int linha, int coluna)*):** verifica se uma peça é ameaçada por outra (utiliza a interface Fila para realizar a verificação);
- **linha(int linha) (*public Linha linha(int linha)*):** devolve um novo objeto da classe Linha;
- **coluna(int coluna) (*public Coluna linha(int coluna)*):** devolve um novo objeto da classe Coluna;
- **DiagonalDescendente(int linha, int coluna) (*public DiagonalDescendente diagonalDescendente(int linha, int coluna)*):** devolve um novo objeto da classe DiagonalDescendente;
- **DiagonalAscendente(int linha, int coluna) (*public DiagonalAscendente diagonalAscendente(int linha, int coluna)*):** devolve um novo objeto da classe DiagonalAscendente.

- **Classe Gerador (*public class Gerador*):** classe utilizada para gerar configurações de tabuleiros, com certas restrições (número de rainhas, tabuleiros e tamanho dos mesmos).
  - **Métodos da classe:**
    - **random(m, q, n) (*static List<String> random(int m, int q, int n)*):** produz *n* configurações aleatórias de *q* rainhas em tabuleiros de dimensões *m***m*; utiliza a classe *Random* da biblioteca *java.util* para gerar novos números (posições).

Validador e suas vertentes

No programa Validador existem duas vertentes, o Validador Individual, que é executado quando *args.length == 0*, e o Filtro Validador, que é executado quando a*rgs[0].equals("filtro")*.

No primeiro modo, o utilizador insere uma configuração e o programa verifica, utilizando os métodos criados anteriormente, se a configuração do tabuleiro é válida ou não.

No segundo, o utilizador insere sucessivamente configurações, às quais o programa reescreve as válidas.

Gerador

No programa do Gerador, o funcionamento é bastante simples. O programa lê uma input em que a configuração da mesma é *string int int int*, associando a *string* ao comando (random) e os inteiros às variáveis *tamanho*, *damas* e *numeroTabuleiros*, pela ordem que são introduzidos pelo utilizador. A partir daí gera os *numeroTabuleiros* tabuleiros de dimensões *tamanho***tamanho*, com *damas* damas. Por fim dá print das configurações geradas.

Observações

Relativamente aos limites do programa, é possível verificar que, caso o utilizador não conheça bem os métodos, pode introduzir argumentos fora dos limites provocando exceções e fazendo com que o programa parasse.

Foram também tomadas algumas medidas de forma a permitir que fossem adicionadas mais peças. A adição dessas peças não foi possível, mas as classes criadas estão prontas a recebê-las. Isto é possível verificar-se no método *podeIrPara(linha, coluna)*.