

18

Métodos e classes finais



Métodos “normais” vs. “finais”

19

- As classes, **concretas** ou **abstractas**, podem conter métodos normais, como os que temos usado, ou “finais”
- Um método diz-se **final** quando **não pode** ser redefinido numa sub-classe – existente, ou que venha a ser criada no futuro
 - A tentativa de redefinir um método final origina um erro do compilador

A palavra reservada `final`

20

- Em geral, devemos declarar os métodos invocados pelos construtores como finais
 - Isso impede que, por polimorfia, esses métodos tenham um comportamento inesperado
- Ocasionalmente, podemos querer definir uma **classe** como final para garantir que ela não pode ser redefinida
 - Útil, por exemplo, para garantir que os objectos dessa classe são imutáveis (**como acontece com as Strings**)

A palavra reservada `final`

21

- Recorde o modificador **`final`**, usado no contexto da definição de constantes
 - Tal como com os métodos, **`final`** indica que o elemento por ele afectado não pode ser alterado


```
public class Chess {  
    public static final int WHITE = 0; // Peças brancas  
    public static final int BLACK = 1; // Peças pretas  
    //...  
    public final int getFirstPlayer() {  
        return Chess.WHITE;  
    }  
    //...  
}
```

- Isto aplica-se a métodos, membros de dados, parâmetros, e mesmo a classes

Implementação de um método `final`

22

```
public class Chess {  
    public static final int WHITE = 0; // Peças brancas  
    public static final int BLACK = 1; // Peças pretas  
    //...  
    /**  
     * Retorna o primeiro jogador a mover peças, num jogo de xadrez.  
     * De acordo com as regras, começam SEMPRE as brancas.  
     * @return Começam as brancas.  
     */  
    public final int getFirstPlayer() {  
        return Chess.WHITE;  
    }  
    //...  
}
```



- Não queremos deixar que um dia mais tarde alguém viole as regras do xadrez, redefinindo este método! Assim, **temos a certeza de que começam sempre as brancas, nesta classe ou em qualquer sub-classe dela que venha a ser definida no futuro.**

A palavra reservada `final`

23

- O modificador `final`, exprime sempre limitações à modificação, mas aquilo a que se aplica varia com o contexto em que é usado
 - Nas constantes, `final` refere-se ao **valor** da “variável”
 - Nos métodos, `final` refere-se à possibilidade de redefinir o método
 - Nas classes, `final` refere-se à herança, i.e., à possibilidade de criar uma sub-classe
- No caso geral, o Java **não** fornece um mecanismo que garanta que um **objecto** é constante
 - Os programadores é que têm de garantir que uma classe é **imutável**, ao não fornecer operações que alteram o estado

Objectos imutáveis

24

- Um objecto imutável é um objecto que se mantém idêntico ao longo de todo o seu ciclo de vida
 - São relativamente mais simples de entender e usar do que os objectos mutáveis
 - São muito úteis como “peças” na construção de outros objectos mais complexos

Objectos imutáveis

25

- Como criar um objecto imutável?
 - Declarar todos os seus membros como **final**
 - Inicializar todos os campos no construtor
 - Não disponibilizar métodos modificadores
 - Mas podem-se disponibilizar métodos de consulta
 - Declarar a classe como **final**, para os seus membros não poderem ser redefinidos