Fundamentos de Java 10

Avaliação

1) Quem pode acessar e alterar diretamente o valor do atributo `numQuarto` no código abaixo?

```
1 package com.minhaempresa;
2
3 public class Hotel {
4   private int numQuarto = 122;
5 }
```

- ○ Qualquer classe. **(alternativa incorreta)**
- ○ Qualquer classe que pertença ao pacote `com.minhaempresa`. **(alternativa incorreta)**
- ◉ Apenas a classe `Hotel`. **(alternativa correta)**
- ○ Qualquer classe que estenda a classe `Hotel`. **(alternativa incorreta)**

Correto

Resposta correta!
Quando um atributo é declarado como privado (`private`), apenas a própria classe pode acessá-lo e alterar o seu valor. A mesma regra vale para construtores e métodos.

---

2) Quais das afirmações abaixo são corretas?
*** Marque todas as alternativas que respondem o enunciado da questão**

- ☐ O encapsulamento é utilizado para fazer com que programas rodem mais rápido. **(alternativa incorreta)**
- ☐ Uma classe fortemente encapsulada é sempre imutável. **(alternativa incorreta)**
- ☐ Encapsulamento é uma forma de esconder dados. **(alternativa correta)**
- ☑ O encapsulamento permite mudanças internas na classe enquanto a interface (isto é, o que é visível para quem chama a classe) permanece inalterada. **(alternativa correta)**
- ☑ O encapsulamento ajuda a evitar que os dados da classe sejam corrompidos. **(alternativa correta)**

Incorreto

Resposta incorreta!
Uma classe fortemente encapsulada não permite acesso externo ao modelo interno da classe. Ao invés disso, o acesso é permitido apenas através de métodos getters e setters. O tempo adicional necessário para que a chamada passe pelos métodos getters e setters normalmente torna a execução mais lenta (muito pouco, é verdade). O encapsulamento é uma forma de esconder dados. Como ele não permite acesso externo a estruturas internas da classe, a possibilidade de que acessos externos corrompam os dados diminui drasticamente. Além disso, métodos setters podem validar os dados que estão sendo fornecidos. Nem toda classe fortemente encapsulada é imutável.

---

3) Qual das alternativas abaixo é falsa?

- ○ Um método de classe não está associado a uma instância em particular de uma classe. **(alternativa incorreta)**
- ○ Um método declarado como `static` é também conhecido como método de classe. **(alternativa incorreta)**
- ○ Um método que não é estático é conhecido como um método de instância. **(alternativa incorreta)**
- ◉ A palavra-chave `this` pode ser utilizada dentro de um método `static`. **(alternativa correta)**
- ○ A palavra-chave `super` não pode ser utilizada dentro de um método `static`. **(alternativa incorreta)**

Correto

Resposta correta!
A palavra-chave `this` se refere à instância do objeto na qual o método foi invocado. Um método estático (também conhecido como método de classe) não é invocado numa instância em particular da classe, mas sim diretamente na classe. Como um método estático não está associado com uma instância, a tentativa de utilizar a palavra-chave `this` no método resultará em um erro de compilação. O mesmo ocorre com a palavra-chave `super`.

---

4) Qual alternativa pode ser considerada falsa com relação aos construtores?

- A chamada `this()` pode ser usada para invocar outros construtores presentes na mesma classe. **(alternativa incorreta)**
- Quando a classe possui algum construtor criado explicitamente, o Java não gera um construtor padrão. **(alternativa incorreta)**
- Construtores podem ser sobrecarregados. **(alternativa incorreta)**
- Existem casos onde uma classe não tem qualquer construtor. **(alternativa correta)**
- As únicas formas possíveis de invocar um construtor de uma classe são através das chamadas `this()` e da invocação do operador `new` para construir objetos da classe. **(alternativa incorreta)**

Correto

Resposta correta!
Todas as alternativas são corretas, exceto a que diz que existem casos onde uma classe não tem qualquer construtor. Quando o programador não declara um construtor explicitamente, o Java gera um construtor padrão. Este construtor não recebe quaisquer parâmetros.

5) Uma classe não pode ser considerada altamente encapsulada a menos que:

- Todos os seus métodos sejam declarados como privados. **(alternativa incorreta)**
- A classe seja uma subclasse direta de `Object`. **(alternativa incorreta)**
- Todas as suas variáveis locais sejam declaradas como `final`. **(alternativa incorreta)**
- Nenhuma das alternativas. **(alternativa incorreta)**
- Métodos setters sejam utilizados para prevenir que dados inválidos sejam fornecidos à classe, juntamente com atributos de visibilidade privada. **(alternativa correta)**

Correto

Resposta correta!
O princípio do encapsulamento significa esconder, de quem acessa o objeto externamente, detalhes internos de funcionamento do mesmo. Logo, atributos não podem ser modificados diretamente. O correto é que existam métodos setters que realizam estas modificações, fazendo antes a validação dos dados fornecidos. E os atributos devem ser privados.

6) Qual a forma correta de declarar e inicializar uma constante pública, sem que haja a necessidade de instanciar um objeto da classe onde a constante está declarada?

- `public static int CONSTANTE = 10;` **(alternativa incorreta)**
- `public static final int CONSTANTE = 10;` **(alternativa correta)**
- `public static final int CONSTANTE;` **(alternativa incorreta)**
- `public final int CONSTANTE = 10;` **(alternativa incorreta)**
- `private static final int CONSTANTE = 10;` **(alternativa incorreta)**

Correto

Resposta correta!
Por ser constante, o modificador `final` deve ser utilizado. Por ser pública, o modificador `public` deve ser utilizado. Como não é necessário instanciar um objeto da classe, o atributo deve estar ligado à classe e não ao objeto, o que implica na utilização do modificador `static`. E sempre que o modificador `final` é utilizado, é necessário que seja especificado um valor de inicialização para o atributo.

.: Retornar para a página do curso