INSTITUTO SUPERIOR POLITECNICO DE TECNOLOGIAS E CIENCIAS

1º Semestre

2025/26

ISPTEC

Sumário

- A **encapsulação**, é o processo de empacotamento dos ***dados*** e ***métodos*** **do objecto**. Um dos *benefícios mais poderosos da encapsulação*, é poder esconder os detalhes dos outros objectos.

- Isto significa que a **parte interna de** um **objecto dá uma visibilidade limitada ao exterior**. Esta organização, resulta na salvaguarda da parte interna, contra o acesso externo indesejado.

- A parte **externa de um objecto** é normalmente designada como a ***interface*** **do objecto**, porque actua como uma **interface** com o resto do programa.

- Para descobrir o que um objeto pode fazer, basta olhar para **as assinaturas dos métodos públicos definidos na classe desse objeto**. A assinatura de um método é composta pelo seu nome e seus parâmetros. As assinaturas dos métodos públicos de um objeto formam a sua **interface de uso**.

- **Os dados de um objecto** só podem ser acedidos pelas **suas operações**.
- **Estado** é acessível apenas através de **operações** (invocadas por mensagens).
- O acesso aos **atributos** da classe só pode ser efetuado através dos seus **Métodos / Funções**
- **Esconde** do “mundo exterior” o processamento interno da Classe
- Combina **dados** e **métodos** numa única “cápsula”

- Encapsulamento
  - Conceitos gerais
  - Getters e Setters
  - Exercícios de Fixação
  - Modificadores de acesso
  - Porque encapsular?
  - Implementação e Interface

Agora que já sabemos mais sobre o conceito de **Encapsulamento** como aplicar?

Getters e Setters

- Os métodos chamados **getter** e **setter** são métodos que permitem obter e definir atributos da classe respectivamente.
- Usar estes **métodos especiais** permite o **encapsulamento** do objeto
- Por regra, as variáveis **de gestão de estado da Classe** são mantidas privadas (**private**) e utilizam-se os **Getters** e **Setters** para obter e modificar **o estado do Objeto**.
- Não é obrigatório criar **Getters** e **Setters** para todas as variáveis de gestão de estado da Classe.
- Apenas devem ser criados **Getters** e **Setters** que façam sentido.

Sumário

- Definir uma classe **Empregado**, que define o nome, cargo e salario definidos no seu constructor.

```
1  public class Empregado {
2
3      String nome;
4      String cargo;
5      double salario;
6
7      Empregado(String nome, String cargo, double salario) {
8
9          this.nome = nome;
10         this.cargo = cargo;
11         this.salario = salario;
12     }
13 }
14
```

Definir os getters e setters da classe Empregado.

```
3   public class Empregado {
4
5       private String nome;
6       private String cargo;
7       private double salario;
8
9       public String getNome() {
10          return nome;
11      }
12
13      public void setNome(String nome) {
14          this.nome = nome;
15      }
16
17      public String getCargo() {
18          return cargo;
19      }
20
21      public void setCargo(String cargo) {
22          this.cargo = cargo;
23      }
24
25      public double getSalario() {...3 lines }
28
29      public void setSalario(double salario) {...3 lines }
32
33      Empregado(String nome, String cargo, double salario) {
34
35          this.nome = nome;
36          this.cargo = cargo;
37          this.salario = salario;
38      }
39  }
40
```

Sumário

- Encapsulamento
  - Conceitos gerais
  - Getters e Setters
  - Exercícios de Fixação
  - Modificadores de acesso
  - Porque encapsular?
  - Implementação e Interface

- Em Java, **modificadores de acesso** determinam se outras classes podem ou não usar um determinado atributo, ou invocar um método particular.

Modificador public

- O ***modificador public***, dá às variáveis e aos métodos toda a visibilidade.
- As variáveis e os métodos declarados com o ***modificador public*** podem ser acedidos por qualquer classe.
- Se uma variável for declarada como ***public***, qualquer objeto pode referenciar essa variável, recolher ou alterar a informação nela contida.
- Do mesmo modo, se um **método** for declarado como ***public***, pode ser invocado a partir de qualquer outro objecto.

Modificador private

- Uma variável ou um método declarado como ***private***, apenas pode ser acedido a partir da própria classe.

- Se algum código fora da classe onde foram definidos as variáveis e os métodos tentar acessar ou alterar o valor do atributo **privado salari**o, um erro de compilação será gerado

- Definir **todos os atributos como privado e** métodos para implementar as lógicas de acesso e alteração é quase uma **regra da orientação a objetos**.

Modificador protected

- Uma variável ou um método declarado como ***protected***, apenas pode ser acedido a partir da própria **classe** ou pelas **subclasses da classe**.

- **Falaremos nesse modificador com mais atenção quando vermos as heranças**

- Encapsulamento
  - Conceitos gerais
  - Getters e Setters
  - Exercícios de Fixação
  - Modificadores de acesso
  - Porque encapsular?
  - Implementação e Interface

Porque encapsular

- Uma das ideias mais importantes da orientação a objetos é o encapsulamento. Encapsular significa esconder a **implementação** dos objetos.

- O encapsulamento favorece principalmente dois aspectos de um sistema: a **manutenção** e o **desenvolvimento**.

  - A **manutenção** é favorecida pois, uma vez aplicado o encapsulamento, quando o funcionamento de um objeto deve ser alterado, em geral, basta modificar a classe do mesmo.

  - O **desenvolvimento** é favorecido pois, uma vez aplicado o encapsulamento, conseguimos determinar precisamente as responsabilidades de cada classe da aplicação.

- Encapsulamento
  - Conceitos gerais
  - Getters e Setters
  - Exercícios de Fixação
  - Modificadores de acesso
  - Porque encapsular?
  - Implementação e Interface

- Dentro de um sistema orientado a objetos, cada objeto realiza um conjunto de tarefas de acordo com as suas responsabilidades.

  - Exemplo
    - Os objectos da classe **Conta** realizam as operações de **levantar**, **depositar**, **transferir** e **geração de extracto**.

- Para descobrir o que um **objeto pode fazer**, basta olhar para as assinaturas dos métodos públicos definidos na classe desse objeto.

- **A assinatura de um método** é composta pelo seu **nome** e seus **parâmetros**. As assinaturas dos métodos públicos de um objeto formam a sua **interface de uso**.

- Para descobrir **como** um objeto da classe **Conta** realiza as suas operações, devemos observar o corpo de cada um dos métodos dessa classe. Os corpos dos métodos constituem a **implementação** das operações dos objetos.

Dúvidas