Curso de Bacharelado em Engenharia de Software
Programação Orientada a Objetos
Profª Cristiana Pereira Bispo
Tarefa Individual

Encapsulamento

Encapsulamento é um dos pilares fundamentais da Programação Orientada a Objetos (POO), e é como colocamos proteção em nossas classes para controlar o acesso e modificação dos dados de um objeto.
Imagine que você tem uma classe Pessoa que armazena informações de uma pessoa, como o nome e a idade. Sem encapsulamento, qualquer parte do programa poderia acessar e modificar diretamente esses dados, o que poderia levar a problemas, como colocar uma idade inválida.
No encapsulamento, nós **"escondemos"** os atributos (dados) da classe, deixando-os privados, ou seja, inacessíveis diretamente de fora da classe. Para permitir que o resto do programa interaja com esses dados, usamos **métodos públicos**, conhecidos como *getters* (para acessar o valor de um atributo) e *setters* (para modificar o valor de um atributo), que podem ter regras de validação.

Aqui está um exemplo simples de como isso funciona em Java:

```
public class Pessoa {
   // Atributos privados, só podem ser acessados dentro da própria classe
   private String nome;
   private int idade;

   // Método getter para acessar o nome
   public String getNome() {
      return nome;
   }

   // Método setter para modificar o nome
   public void setNome(String nome) {
      this.nome = nome;
   }

   // Método getter para acessar a idade
   public int getIdade() {
      return idade;
   }

   // Método setter para modificar a idade com validação
   public void setIdade(int idade) {
      if (idade > 0) {
         this.idade = idade;
      } else {
         System.out.println("Idade inválida!");
```

```
    }
  }
}
```

Nesse exemplo:

- Os atributos nome e idade são privados, então só a classe Pessoa pode acessar diretamente.
- Usamos o método getNome() para **ler** o valor do nome, e o método setNome() para **modificar** o valor do nome.
- O método setIdade() inclui uma **regra de validação**, que não permite definir uma idade negativa.

Com o encapsulamento, você garante que os dados de um objeto sejam acessados e modificados de forma controlada, mantendo a segurança e integridade dos dados.

Agora é a sua vez!!!

Parte 1: Encapsulamento em Java - Cadastro de Funcionário

Descrição:
Crie uma classe chamada Funcionario que represente um funcionário de uma empresa. Essa classe deve ter os seguintes atributos privados:

- nome (String)
- salario (double)

Implemente métodos *getters* e *setters* para cada um dos atributos, com as seguintes regras:

1. O **nome** deve ser uma string com pelo menos 3 caracteres. Caso contrário, exibir uma mensagem de erro.
2. O **salário** deve ser um valor positivo. Se um valor inválido for passado, exibir uma mensagem de erro.

Em seguida, crie uma outra classe chamada Empresa com o método main que:

- Crie um objeto da classe Funcionario.
- Use os métodos *setters* para definir o nome e o salário do funcionário.
- Use os métodos *getters* para imprimir as informações do funcionário.

Dica: Não se esqueça de testar o que acontece quando você tenta definir um nome ou salário inválido!

Exemplo de Saída Esperada:

Nome do Funcionário: João Silva
Salário do Funcionário: 4500.00

Tentando definir um nome inválido...
Nome inválido! O nome deve ter pelo menos 3 caracteres.

Tentando definir um salário inválido...
Salário inválido! O valor deve ser positivo.

Parte 2: Encapsulamento em Java com Interface Gráfica (JFrame)
Descrição:
Agora, além de ter a classe Funcionario, você vai criar uma interface gráfica simples usando JFrame para cadastrar o nome e o salário do funcionário. O programa deve ter dois campos de texto e dois botões:

- **Campos de texto**:
  - Um para o nome do funcionário.
  - Um para o salário do funcionário.
- **Botões**:
  - Um botão para **salvar** as informações (quando pressionado, deve validar os dados e exibir uma mensagem de sucesso ou erro).
  - Um botão para **exibir** as informações (quando pressionado, deve mostrar o nome e o salário cadastrados).

Você deve criar duas classes:

1. **Classe Funcionario**: Contém os atributos nome e salario (como na parte 1), além dos métodos *getters* e *setters* com validação.
2. **Classe Empresa**: Cria a interface gráfica com JFrame, implementando as ações dos botões.

Objetivo:

1. Demonstrar o uso de encapsulamento em uma aplicação com interface gráfica.
2. Praticar a separação de lógica de negócios (classe Funcionario) e interface (classe **Empresa**).
3. Usar validações e interação entre a interface gráfica e o objeto Funcionario.

Desafio Extra:

- Implemente um terceiro botão chamado "Limpar" que limpe os campos de texto do nome e do salário.
- Adicione mais validações, como impedir que o usuário cadastre um funcionário sem preencher o nome ou o salário.