

Abstração

Programando na orientação à objeto

Abstração

- Mostra o geral e esconde o específico
- Útil para quando se conhece partes da implementação mas não o todo
- Ajuda muito na reusabilidade de código
- Flexibilidade do sistema

Exemplo

Implemente uma classe animal e crie cada som dos animais

```
1 package faunaBrasil;
2
3 public abstract class Animal {
4
5     public String nome;
6
7     public abstract String som();
8
9 }
```

```
1 package faunaBrasil;
```

Define a classe abstrata



```
2  
3 public abstract class Animal {
```

```
4  
5     public String nome;
```

Define método abstrato



```
6  
7     public abstract String som();  
8  
9 }
```

Abstração


Uma classe abstrata possui, no mínimo, um método abstrato

Classes abstratas não podem ser instanciadas

É necessário implementar o método abstrato em classes filhas

```
1 package faunaBrasil;
2
3 public class Cachorro extends Animal{
4
5     @Override
6     public String som() {
7         return "Que cachorro o que, eu não sou cachorro não";
8     }
9
10 }
```

```
1 package faunaBrasil;
2
3 public class Cachorro extends Animal{
4
5     @Override
6     public String som() {
7         return "Que cachorro o que, eu não sou cachorro não";
8     }
9
10 }
```



Implementação do método

Exercícios

Imagine que você está desenvolvendo um sistema de controle de pagamento de funcionários para uma empresa. A empresa possui diferentes tipos de funcionários, como funcionários assalariados, por hora e comissionados. Todos os funcionários têm algumas características comuns, como nome, número de identificação e salário. No entanto, a maneira como o salário é calculado varia de acordo com o tipo de funcionário.

Implemente um programa em Java que modele essa situação.

Passos para a solução

1. **Crie uma classe abstrata `Funcionario`** com as seguintes características:
 - Atributos: `nome` (String), `id` (int).
 - Métodos:
 - Um método abstrato `calcularSalario()`, que será implementado por cada tipo específico de funcionário.
 - Métodos concretos para `getNome()`, `getId()` e um construtor para inicializar os atributos.
2. **Crie três classes que herdam de `Funcionario`:**
 - **`FuncionarioAssalariado`:**
 - Atributo específico: `salarioFixo` (double).
 - Implemente o método `calcularSalario()`, que retorna o salário fixo.
 - **`FuncionarioPorHora`:**
 - Atributos específicos: `horasTrabalhadas` (int), `salarioPorHora` (double).
 - Implemente o método `calcularSalario()`, que retorna o produto das horas trabalhadas pelo salário por hora.
 - **`FuncionarioComissionado`:**
 - Atributos específicos: `vendas` (double), `percentualComissao` (double).
 - Implemente o método `calcularSalario()`, que retorna o produto das vendas pelo percentual de comissão.
3. **Crie uma classe `FolhaDePagamento`** que contenha um método `imprimirSalario(Funcionario funcionario)` para exibir o nome do funcionário e o salário calculado