09

Encapsulamento de campos e hierarquia de classes

Nesse ponto do projeto veremos algumas técnicas de refatoração que podem nos ajudar durante a criação de nossas classes modelos que possuem tanto atributos como também uma hierarquia entre super ou sub classes.

Portanto, crie a classe `Gerente` no pacote **br.com.alura.model**, adicione os seguintes atributos e faça a extensão para a classe `Funcionario`, afinal um `Gerente` também é um funcionário.

> Também é possível criar subclasses usando as sugestões do IntelliJ (**Alt + Enter**), ou seja, dentro do classe `Funcionario` utilize as sugestões do IntelliJ e use a opção **Create subclasse**. Em muitas das situações costuma ser mais objetivo!

Note que é necessário realizar a implementação do construtor personalizado.

Em seguida, adicione os seguintes atributos para a classe `Gerente`, pois nesse instante se tratam de atributos que só o gerente possui:

- `String` usuario
- `String` senha

Observe que precisamos encapsular esses membros e já vimos uma maneira de fazer isso usando o **Column selection** junto do **Generate**, porém, faremos de uma maneira mais obtiva dessa vez.

Refatoração com o encapsulamento de campos

Dentro da classe `Gerente`, após ter adicionado os atributos **usuario** e **senha**, utilize o **Refactor This** (**Ctrl + Shift + Alt + T / CMD + T**), então, use a opção **Encapsulate Fields...**. Nela, selecione todos os atributos que deseja encapsular, no caso tanto o `usuario` como a `senha`. Depois clique em **Refactor**.

> **Observações:** Repara que por padrão o processo de encapsulamento vai deixar os atributos privados e vai criar tanto os getters como os setters públicos. Entretanto, é possível ajustar na mesma janela que faz a seleção dos atributos o que desejar gerar.

Se analisarmos bem, usuário e senha é algo comum para qualquer funcionário dentro da empresa, afinal, eles precisam acessar o sistema de alguma forma, portanto, faz todo o sentido mover tais campos para a superclasse `Funcionario`. Podemos fazer manualmente, porém, o IntelliJ também nos ajuda nisso.

Movendo membros entre a hierarquia de classes

Para movermos membros entre a hierarquia de classes, isto é, tanto da subclass para a superclasse ou da superclasse para a subclasse, podemos usar o **Refactor This** também.

Movendo para superclasse ou interface

Sendo assim, utilize o **Refactor This** na classe `Gerente` e escolha a opção **Pull Members Up...**.

Então selecione os atributos `usuario` e `senha`, como também os seus **getters** e **setters** e clique em **Refactor**. Veja que agora a classe `Gerente` não implementa mais esses atributos, porém, ainda tem acesso aos getters e setters que estão dentro da classe `Funcionario`

Movendo para subclasses

Caso, amanhã ou outro dia, for decidido que apenas o gerente tem que ter **usuário** e **senha** basta apenas usar o **Refactor This** novamente e usar a opção **Push Members Down...** que os membros serão enviados para as classes que herdam da superclass diretamente.

Cuidados a serem tomados: Quando utilizar o **Push Members Down...** lembre-se que os membros movidos serão enviados para todas as **classes que herdam da superclass diretamente da superclasse**, ou seja, se tiver um gerente, atendente, vendedor ou qualquer outro funcionário que também herda de funcionário e tentar enviar os membros com o intuito de só mandar para o gerente, todos os outros também receberão os membros enviados.

Uma opção bem válida antes de usar o **Pull Members Down..."** é considerar o uso do **Hierarchy** (**Ctrl + H**) conforme vimos na aula.