

**TECNÓLOGO EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS**

**Larissa Lopes Oliveira - 552628**

**Lucas Alcântara Carvalho - 95111**

**Luna Faustino Lima - 552473**

**Murillo Ferreira Ramos - 553315**

**Pedro Luiz Prado - 553874**

**Checkpoint - Advanced Java**

**FIAP - SP**

**2024**

# Pacotes e Classes

* Pacote annotation onde teremos as interfaces Coluna e Tabela.
* Pacote dao com a classe JpaDao, onde teremos o comando SQL.
* Pacote model com a classe mãe Funcionario e as filhas FuncionarioJunior, FuncionarioPleno, FuncionarioSenior. Na classe Funcionario temos os atributos e os métodos imprimirInformacoes e calcularSalario.
* Pacote view, onde temos a classe de execução Main.

Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

# CÓDIGOS DE CADA CLASSE

* **Coluna**

Aqui começamos com as anotações @Retention e @Target. A @Retention define o tempo por quanto tempo a anotação @Coluna estará disponível, definimos como RUNTIME, ou melhor, enquanto o programa estiver rodando, ela estará disponível.

@Target especifica onde a anotação @Coluna poderá ser aplicada, usamos FIEL e METHOD, ou melhor, essa anotação pode ser aplicada a campos e métodos.

Usamos o @interface para criar uma anotação personalizada, e então definimos os elementos nome, sendo obrigatório e os horasTrabalhadas e valorPagoHoras, que por padrão tem o valor de 0.0.

Texto

Descrição gerada automaticamente

* **Tabela**

O que fizemos de diferente aqui foi o uso de TYPE, mostrando que essa anotação pode ser usada em classes.

Texto

Descrição gerada automaticamente

* **JpaDao**

Nessa classe criamos um método que recebe um objeto genérico como parâmetro. O método utiliza de reflexão para obter a classe que utilizou a anotação @Tabela. Então imprime uma query SQL concatenando o nome da classe que foi pego na linha anterior.

Texto

Descrição gerada automaticamente

* Funcionario

Usamos a anotação @Tabela para associar essa classe a uma tabela no banco de dados, onde o nome será TAB\_FUNCIONARIO. Também usamos a anotação @Coluna para mapear cada atributo a uma coluna diferente dessa tabela.  
  
Criamos o construtor e os getters e setters.

O método imprimirInformacao imprime os dados inseridos nos atributos. O método calcularSalario retorna a quantidade de horas trabalhadas multiplicadas pelo valor pago por hora.

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

* FuncionarioJunior

Usamos a anotação @Tabela para dar um nome diferente para a tabela, mas não precisamos definir o @Coluna, já que os campos já foram mapeados na classe mãe.

Essa classe é filha da classe Funcionario. Definimos que o bônus dessa classe é de 10% e criamos o construtor dela.

Fizemos um @Override dos dois métodos que ela herda. Onde personalizamos a primeira linha da impressão para mostrar que é um funcionário Júnior e na última linha imprimimos o valor total do salário levando em conta o resultado do outro Override.

No método calcularSalario primeiro retornamos o resultado do método padrão, que seria horasTrabalhadas \* valorPagoHoras. Dividimos a quantidade de horas trabalhadas por 10 e então multiplica por (10 \* valorPagoHoras \* bonusPercentual), assim o acréscimo só é feito se o funcionário teve de 10 para cima, abaixo de 10 o acréscimo não é aplicado. Por último retornamos o salário-base + o bonus da linha anterior.

Texto

Descrição gerada automaticamente

* FuncionarioPleno

Praticamente a mesma coisa da outra classe filha. Personalizamos o nome da tabela, o percentual do bônus para 20% e a primeira linha do imprimirInformacao.

Texto

Descrição gerada automaticamente

* FuncionacioSenior

Personalizamos o nome da tabela, o percentual do bônus para 30% e a primeira linha do imprimirInformacao.

Texto

Descrição gerada automaticamente

* Main

Instanciamos as classes, definimos os atributos e chamados os métodos usando os objetos.

Instanciamos a Funcionario duas vezes para mostrar que o acréscimo do bônus só está sendo aplicado em casos que as horasTrabalhadas são iguais ou maiores que 10.

Texto

Descrição gerada automaticamente

No terminal temos a impressão do querySQL levando em conta o nome da classe e o método imprimirInformacoes. Como podemos ver no segundo funcionário Júnior, não foi aplicado o bônus de 10%, já que trabalhou somente 8 horas.

Texto

Descrição gerada automaticamente