

Campus: Santa Cruz da Serra - Duque de Caxias/RJ

Curso: Desenvolvimento full stack

Disciplina: RPG0016 - BackEnd sem banco não tem

Número da Turma: 9001

Semestre letivo: Mundo 3

Integrantes: Felipe Freaza Fidalgo

## BackEnd sem banco não tem (CadastroBD)

```
Saida - CadastroBD (run)
run:
Id: 31
Nome: Lucas Oliveira
Logradouro: Avenida das Flores
Cidade: Santa Clara
Estado: MG
Telefone: 8765-4321
E-mail: lucas.oliveira@example.com
CPF: 83527194682
Id: 32
Nome: LO
Logradouro: Avenida das Flores
Cidade: Santa Clara
Estado: MG
Telefone: 8765-1234
E-mail: lo@example.com
CNPJ: 62831745926381
CONSTRUÍDO COM SUCESSO (tempo total: 2 segundos)
```

### Análise e Conclusão:

## 1º Procedimento | Mapeamento Objeto-Relacional e DAO

### 1.Qual a importância dos componentes de middleware, como o JDBC?

Componentes de middleware, como o JDBC, são cruciais para a interação entre aplicativos e bancos de dados. Eles abstraem detalhes do banco de dados, promovem portabilidade, segurança e gerenciamento de conexões. Oferecem desempenho

otimizado, recursos avançados e integração com diversas fontes de dados. Além disso, simplificam a manutenção ao separar a lógica de aplicativo da lógica de acesso a dados, aprimorando a eficiência e segurança do desenvolvimento de software.

## **2.Qual a diferença no uso de Statement ou PreparedStatement para a manipulação de dados?**

`Statement` e `PreparedStatement` são usados em Java para consultas SQL, mas diferem em segurança e eficiência. O `Statement` lida com consultas estáticas, permitindo inserção direta de valores, mas é vulnerável a ataques de injeção SQL e ineficiente para consultas repetidas. O `PreparedStatement` é preferível, pois usa placeholders e é pré-compilado, reduzindo riscos de segurança e melhorando o desempenho ao executar consultas várias vezes. É uma prática recomendada usar `PreparedStatement` para consultas parametrizadas.

## **3.Como o padrão DAO melhora a manutenibilidade do software?**

O padrão DAO (Data Access Object) melhora a manutenibilidade do software ao separar a lógica de acesso a dados da lógica de negócios. Isso promove reusabilidade, flexibilidade tecnológica e testabilidade, facilitando a aplicação de mudanças no esquema do banco de dados. Além disso, aumenta a compreensão do código, simplifica a identificação de problemas e permite foco nas necessidades do negócio. Em conjunto, esses benefícios contribuem para um software mais organizado e de fácil manutenção ao longo do tempo.

## **4.Como a herança é refletida no banco de dados, quando lidamos com um modelo estritamente relacional?**

No modelo estritamente relacional, a herança é representada através de estratégias de mapeamento. A abordagem "Tabela por Subtipo" mapeia subtipos para tabelas separadas, mantendo atributos específicos e comuns. A abordagem "Tabela por Superclasse" mapeia todos os tipos para uma única tabela, com colunas nulas para atributos não aplicáveis. A primeira é mais normalizada, mas pode requerer joins

complexos. A segunda é mais simples para consultas, mas pode gerar valores nulos. Ambas as abordagens têm prós e contras na modelagem de herança em bancos de dados relacionais.

## **2º Procedimento – Alimentando a Base**

### **1.Quais as diferenças entre a persistência em arquivo e a persistência em banco de dados?**

A persistência em arquivo envolve armazenar dados em formatos diversos, sendo flexível, mas demandando gerenciamento manual. Em contraste, a persistência em banco de dados utiliza tabelas relacionais, com acesso via SQL, gerenciamento automatizado, segurança e escalabilidade melhor controlados. A persistência em arquivo oferece flexibilidade, mas é menos eficiente e segura em comparação com a estrutura organizada e recursos avançados de um banco de dados. A escolha entre ambas depende das necessidades do projeto, escalabilidade e segurança desejadas.

### **2.Como o uso de operador lambda simplificou a impressão dos valores contidos nas entidades, nas versões mais recentes do Java?**

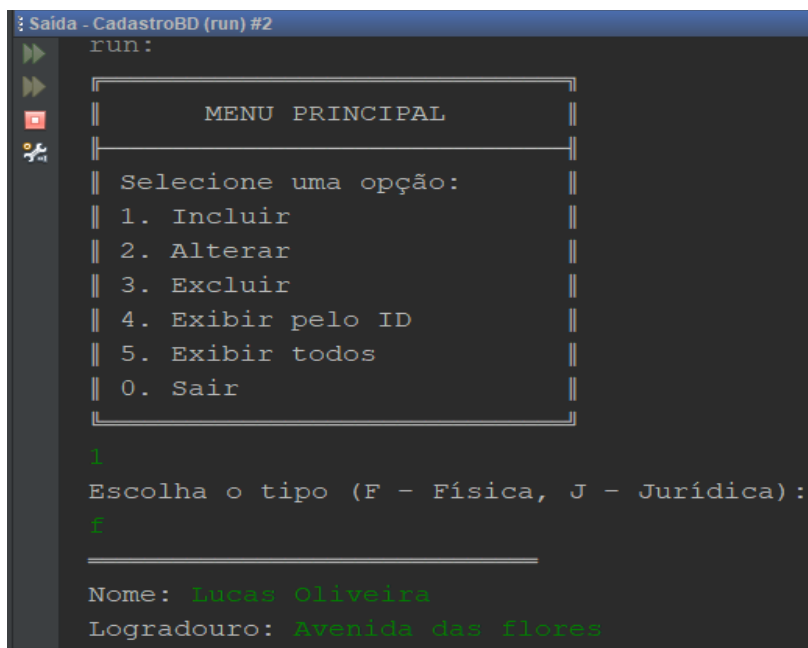
A introdução dos operadores lambda nas versões recentes do Java simplificou a impressão de valores em entidades. Anteriormente, eram necessários loops e estruturas de controle extensas. Com os lambdas, é possível usar métodos como `forEach` em coleções ou streams para iterar e imprimir de forma mais concisa e legível. Isso reduz o código boilerplate, melhora a eficiência e facilita a manutenção do código. Em resumo, os operadores lambda simplificam a impressão de valores em entidades, tornando o código mais claro e funcional.

### **3.Por que métodos acionados diretamente pelo método main, sem o uso de um objeto, precisam ser marcados como static?**

Na álgebra relacional, os operadores incluem seleção, projeção, união, interseção, diferença, produto cartesiano e junção. No cálculo relacional, há o cálculo de tupla, usando variáveis para expressar tuplas desejadas, e o cálculo de domínio, usando variáveis quantificadas para definir conjuntos de valores desejados. Ambos os modelos servem para expressar consultas em bancos de dados relacionais, sendo a escolha entre eles baseada na preferência do usuário e na complexidade da consulta.

## Conclusão:

O projeto evoluiu consideravelmente com a introdução do banco de dados, padrão DAO e separação em pacotes. A estruturação demonstra uma organização mais profissional, facilitando a manutenção e escalabilidade. A utilização do padrão DAO desacopla as operações do banco das classes de entidade, melhorando a separação de responsabilidades.



```
Saida - CadastroBD (run) #2
run:

|-----|
| MENU PRINCIPAL |
|-----|
| Selecione uma opção: |
| 1. Incluir |
| 2. Alterar |
| 3. Excluir |
| 4. Exibir pelo ID |
| 5. Exibir todos |
| 0. Sair |
|-----|

1
Escolha o tipo (F - Física, J - Jurídica):
F

Nome: Lucas Oliveira
Logradouro: Avenida das Flores
```