



Estácio

Universidade Estácio de Sá

3º período em Desenvolvimento Full stack

Matrícula: 202302891292

Aluno: Luiz Fabrício Mello Ferreira

Backend sem banco não tem

1º Procedimento | Mapeamento Objeto-Relacional e DAO

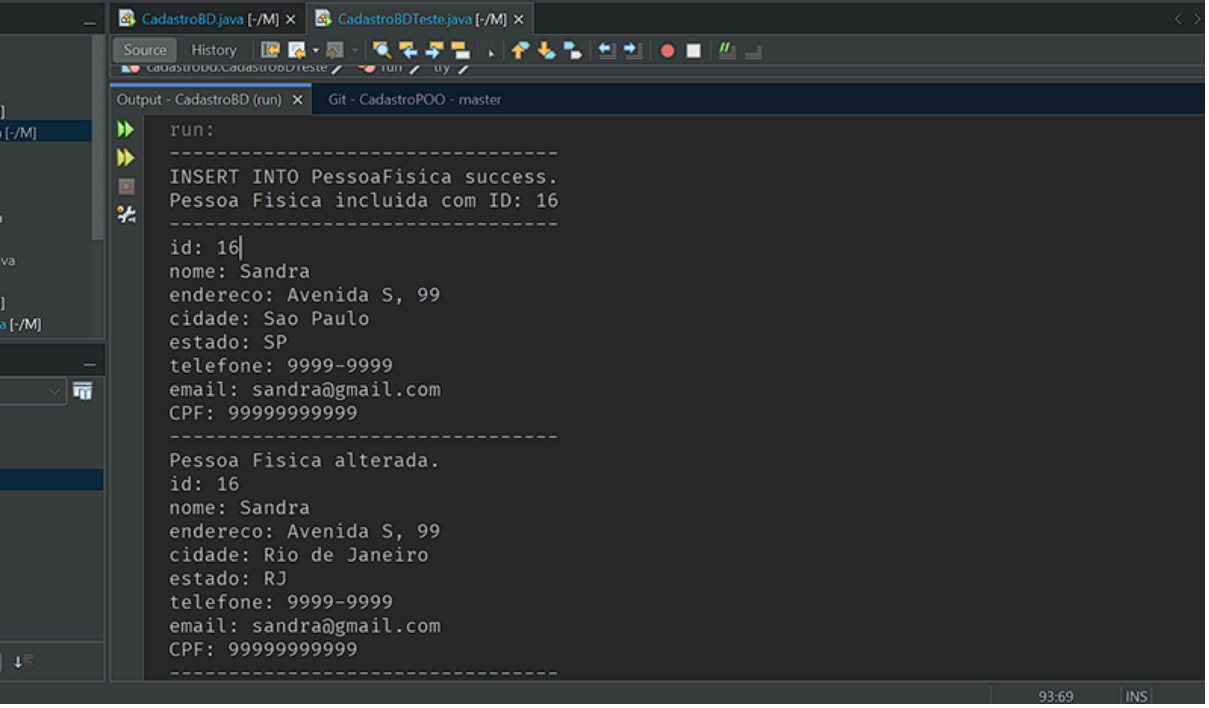
Objetivo da Prática

- Implementar persistência com base no middleware JDBC.
- Utilizar o padrão DAO (Data Access Object) no manuseio de dados.
- Implementar o mapeamento objeto-relacional em sistemas Java.
- Criar sistemas cadastrais com persistência em banco relacional.
- No final do exercício, o aluno terá criado um aplicativo cadastral com uso do SQL Server na persistência de dados.

Códigos solicitados estão no repositório:

<https://github.com/luizfabriciorex/UNESA/tree/main/terceiroPeriodo/pratica-3>

Resultado da execução dos códigos:



```
run:
-----
INSERT INTO PessoaFisica success.
Pessoa Fisica incluída com ID: 16
-----
id: 16|
nome: Sandra
endereço: Avenida S, 99
cidade: Sao Paulo
estado: SP
telefone: 9999-9999
email: sandra@gmail.com
CPF: 99999999999
-----
Pessoa Fisica alterada.
id: 16
nome: Sandra
endereço: Avenida S, 99
cidade: Rio de Janeiro
estado: RJ
telefone: 9999-9999
email: sandra@gmail.com
CPF: 99999999999
-----
```

Figura 1. Execução da classe CadastroBDTeste: inserção e alteração de Pessoa Física.

```
-----
Pessoas Fisica excluida.
-----
INSERT INTO PessoaJuridica success.
Pessoa Juridica incluida com ID: 19
-----
id: 19
nome: Fabrica Fox
endereço: Avenida F, 88
cidade: Goias
estado: GO
telefone: 8888-8888
email: fox@gmail.com
CNPJ: 8888888888888888
-----
Pessoa Juridica alterada.
-----
Exibir todas as pessoas juridicas:
-----
id: 4
nome: Distribuidora Delta
endereço: Avenida D, 44
cidade: Brasilia
estado: DF
-----
```

Figura 2. Execução da classe CadastroBDTeste: exclusão de Pessoa Física, inserção e alteração de Pessoa Jurídica.

```
-----
CNPJ: 4444444444444444
-----
id: 5
nome: Empresa Echo
endereço: Avenida E, 55
cidade: Vitoria
estado: ES
telefone: 5555-5555
email: echo@gmail.com
CNPJ: 5555555555555555
-----
id: 19
nome: Fabrica Fox
endereço: Avenida F, 88
cidade: Belo Horizonte
estado: MG
telefone: 8888-8888
email: fox@gmail.com
CNPJ: 8888888888888888
-----
Pessoas Juridica excluida.
BUILD SUCCESSFUL (total time: 1 second)
```

Figura 3. Execução da classe CadastroBDTeste: exclusão de Pessoa Jurídica.

Análise e Conclusão

a) Qual a importância dos componentes de middleware, como o JDBC?

R: JDBC (Java Database Connectivity) é uma interface de programação de aplicações (API Application Program Interface) que fornece acesso universal a fonte de dados em Java. Com JDBC é possível interligar aplicações Java a bancos de dados relacionais SQL, planilhas Excel e até arquivos textos [1]. Sua importância consiste em agir como intermediário entre a aplicação e a fonte de dados, de modo a permitir a comunicação e o intercâmbio de dados, de maneira padronizada, independente do banco de dados; basta que haja um driver adequado do banco de dados, configurado na aplicação e que este BD seja relacional, padrão SQL. Como consequência do uso de JDBC, há a maior portabilidade de aplicações Java entre diferentes plataformas, a simplificação do desenvolvimento com comandos padronizados e a interoperabilidade entre sistemas heterogêneos.

b) Qual a diferença no uso de Statement ou PreparedStatement para a manipulação de dados?

Ambas as classes Statement e PreparedStatement são utilizadas para executar comandos SQL emJava; entretanto, diferem em características e desempenho [2]:

- Statement é utilizada para executar comandos SQL baseados em String.
- PreparedStatement é utilizada para executar comandos SQL parametrizados.

Em relação à segurança, a classe PreparedStatement é preferencial quando se trata de prevenir ataques de injeção SQL, pois permite definir parâmetros de consulta que são tratados de forma segura pelo banco de dados. Isso significa que os valores dos parâmetros são tratados como dados e não como parte da consulta SQL em si, reduzindo significativamente o risco de injeção de código malicioso.

Em termos de desempenho, PreparedStatement é geralmente mais eficiente do que Statement, especialmente quando a mesma consulta é executada várias vezes com valores diferentes. Isso ocorre porque o banco de dados pode compilar a consulta apenas uma vez e reutilizá-la com diferentes parâmetros.

Em relação à legibilidade e manutenção do código, `PreparedStatement` geralmente torna o código mais legível e mais fácil de manter, especialmente quando se lida com consultas complexas, com muitos parâmetros. Isso ocorre porque os parâmetros podem ser definidos de forma clara e separada da consulta SQL, o que torna mais fácil entender o que a consulta faz.

Em relação aos tipos de dados, com `PreparedStatement`, o JDBC pode inferir automaticamente o tipo de dados dos parâmetros, o que simplifica o código. Com `Statement`, é necessário lidar manualmente com a conversão dos tipos de dados.

Ainda, alguns bancos de dados podem armazenar cache de consultas, o que pode melhorar ainda mais o desempenho em consultas subsequentes com `PreparedStatement`.

c) Como o padrão DAO melhora a manutenibilidade do software?

O padrão DAO (`Data Access Object`) melhora a manutenibilidade do software, ao separar ou isolar o código responsável pelo acesso e manipulação dos dados, do restante da lógica de negócios da aplicação. Isso facilita a manutenção do código, uma vez que as alterações nos requisitos de acesso aos dados podem ser realizadas de maneira separada, sem afetar diretamente outras partes do sistema.

O padrão DAO também promove a reutilização de código ao encapsular a lógica de acesso a dados em classes específicas. Assim, se houver a necessidade de modificar a forma como os dados são acessados (por exemplo, mudança de banco de dados relacional para um banco de dados não-relacional), apenas as classes DAO precisam ser modificadas, de modo a manter o restante do código inalterado.

Outro benefício é que o DAO torna o código mais legível e compreensível, pois fornece uma abstração clara e consistente para as operações de acesso a dados em todo o sistema. Isso facilita a colaboração entre os membros da equipe e a identificação e correção de problemas relacionados ao acesso a dados.

d) Como a herança é refletida no banco de dados, quando lidamos com um modelo estritamente relacional?

Quando se lida com um modelo estritamente relacional em banco de dados, a herança geralmente é refletida com uso de uma técnica conhecida como "tabelas de junção" (ou "tabelas de associação"). Esta abordagem é chamada de modelagem de herança de tabela única, ou modelagem de herança de tabela por classe. Eis como funciona:

- **Tabela Base (ou Superclasse):** Uma tabela é criada para representar a classe base ou superclasse. Esta tabela contém os atributos comuns a todas as subclasses.
- **Tabelas de Subclasse:** Para cada subclasse, é criada uma tabela separada contendo apenas os atributos específicos daquela subclasse. Essas tabelas também terão uma chave estrangeira que referencia a tabela base.
- **Chave Estrangeira:** A chave primária da tabela base é usada como chave estrangeira nas tabelas de subclasse para estabelecer a relação entre elas.
- **Junção de Tabelas:** Quando uma consulta é feita para recuperar dados de uma hierarquia de herança, é necessário fazer uma junção (JOIN) entre a tabela base e as tabelas de subclasse usando as chaves primárias e estrangeiras correspondentes.

Essas abordagens permitem que a hierarquia de herança seja representada de forma eficiente em um modelo relacional, de modo a manter a integridade referencial entre as tabelas e permitir consultas que recuperam dados de todas as classes relacionadas na hierarquia de herança.

2º Procedimento | Alimentando a Base

Objetivo da Prática

- Identificar os requisitos de um sistema e transformá-los no modelo adequado.
- Utilizar ferramentas de modelagem para bases de dados relacionais.
- Explorar a sintaxe SQL na criação das estruturas do banco (DDL).
- Explorar a sintaxe SQL na consulta e manipulação de dados (DML)
- No final do exercício, o aluno terá vivenciado a experiência de modelar a base de dados para um sistema simples, além de implementá-la, através da sintaxe SQL, na plataforma do SQL Server.

Códigos solicitados estão no repositório:

<https://github.com/luizfabriorex/UNESA/tree/main/terceiroPeriodo/pratica-3>

Resultado da execução dos códigos:

```

run:|

=====
1 - Incluir
2 - Alterar
3 - Excluir
4 - Buscar pelo ID
5 - Exibir todos
0 - Sair
=====
ESCOLHA: 1
F - Pessoa Fisica | J - Pessoa Juridica
TIPO DE PESSOA: F
Informe o nome: Hugo
Informe o CPF: 88888888888
Informe o endereco: Habitat Hectare
Informe a cidade: Higienopolis
Informe o estado: SP
Informe o telefone: 8888-8888
Informe o email: hugo@gmail.com
INSERT INTO PessoaFisica success.
=====
1 - Incluir

```

Figura 4. Execução da classe CadastroBD: opção 1- Incluir.

```
Output - CadastroBD (run) X Git - CadastroPOO - master

endereco: Avenida C, 33
cidade: Fortaleza
estado: CE
telefone: 3333-3333
email: carlos@gmail.com
CPF: 33333333333

-----

id: 20
nome: Hugo
endereco: Habitat Hectare
cidade: Higienopolis
estado: SP
telefone: 8888-8888
email: hugo@gmail.com
CPF: 88888888888

-----

id: 4
nome: Distribuidora Delta
endereco: Avenida D, 44
cidade: Brasilia
estado: DF
telefone: 4444-4444
email: delta@gmail.com
CNPJ: 444444444444444444

CadastroBD (run) running... 1:1 INS
```

Figura 5. Execução da classe CadastroBD. Pessoa incluída com sucesso.

```
Output - CadastroBD (run) X Git - CadastroPOO - master

=====
1 - Incluir
2 - Alterar
3 - Excluir
4 - Buscar pelo ID
5 - Exibir todos
0 - Sair
=====

ESCOLHA: 2
F - Pessoa Fisica | J - Pessoa Juridica
TIPO DE PESSOA: F
Informe o ID da Pessoa Fisica: 20
Informe o nome: Gisele
Informe o CPF: 88888888888
Informe o endereco: Habitat Hectare
Informe a cidade: Governador Valadares
Informe o estado: MG
Informe o telefone: 8888-8888
Informe o email: gisele@hotmail.com

=====
1 - Incluir
2 - Alterar
```

Figura 6. Execução da classe CadastroBD: opção 2- Alterar

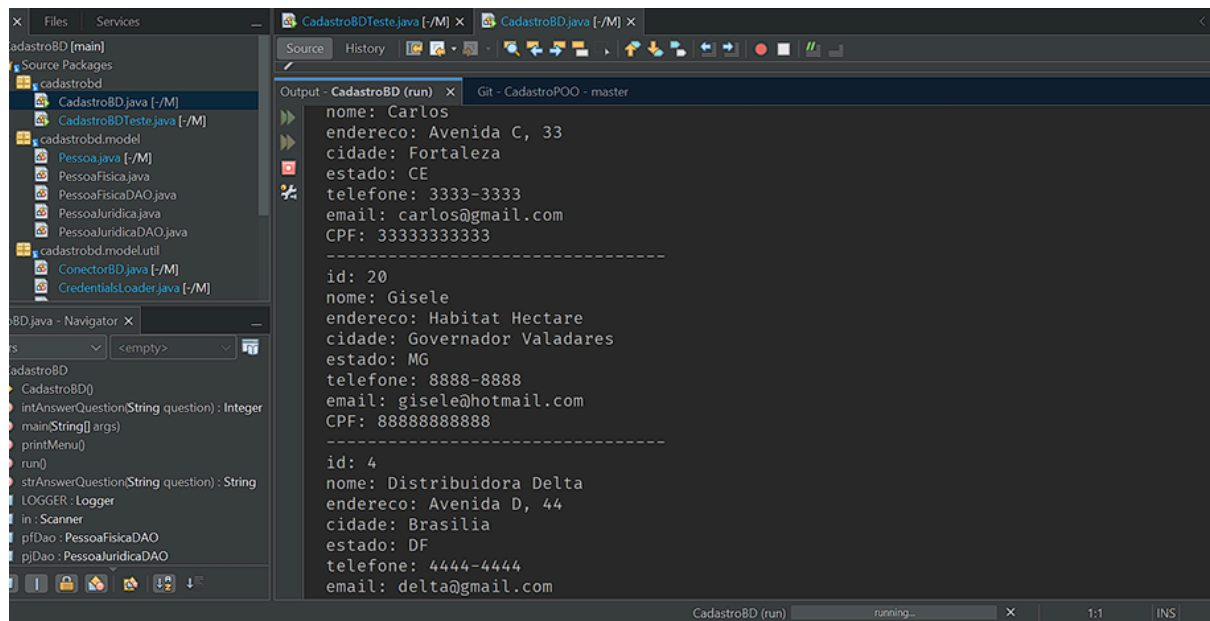


Figura 7. Execução da classe CadastroBD. Pessoa alterada com sucesso.

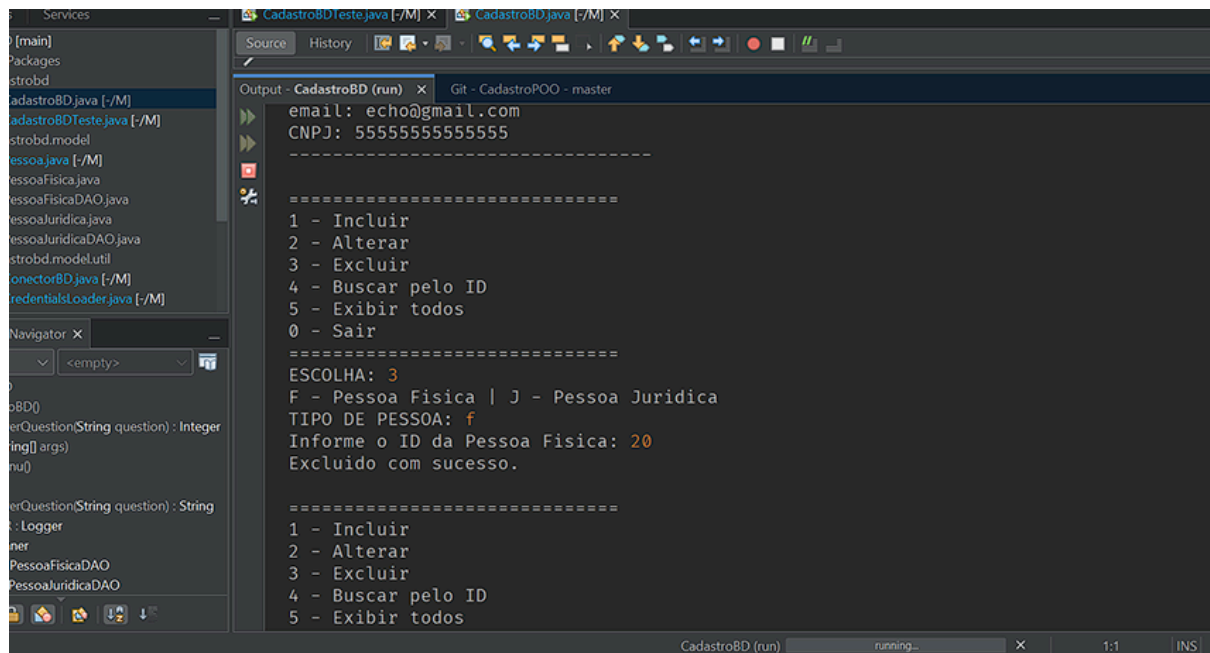


Figura 8. Execução da classe CadastroBD. Opção 3- Excluir.

```
=====  
1 - Incluir  
2 - Alterar  
3 - Excluir  
4 - Buscar pelo ID  
5 - Exibir todos  
0 - Sair  
=====  
ESCOLHA: 4  
F - Pessoa Fisica | J - Pessoa Juridica  
TIPO DE PESSOA: j  
Informe o ID da Pessoa Juridica: 5  
id: 5  
nome: Empresa Echo  
endereco: Avenida E, 55  
cidade: Vitoria  
estado: ES  
telefone: 5555-5555  
email: echo@gmail.com  
CNPJ: 55555555555555  
=====  
1 - Incluir
```

Figura 9. Execução da classe CadastroBD. Opção 4- Buscar pelo ID.

```
=====  
1 - Incluir  
2 - Alterar  
3 - Excluir  
4 - Buscar pelo ID  
5 - Exibir todos  
0 - Sair  
=====  
ESCOLHA: 5  
-----  
id: 1  
nome: Andrea  
endereco: Avenida A, 11  
cidade: Rio Branco  
estado: AC  
telefone: 1111-1111  
email: andrea@gmail.com  
CPF: 111111111111  
-----  
id: 2  
nome: Bruna  
endereco: Avenida B, 22  
cidade: Salvador
```

Figura 10. Execução da classe CadastroBD. Opção 5- Exibir todos.

Análise e Conclusão

a) Quais as diferenças entre a persistência em arquivo e a persistência em banco de dados?

A persistência em arquivos consiste no armazenamento de dados em um sistema de arquivos em formatos binário, por exemplo, arquivos com extensão .dat, .bin, .xls, .xlsx; ou arquivos texto, por exemplo, com extensões .csv, .json, .xml, são reconhecidamente arquivos de dados. É uma forma simples e direta de salvar dados, mas não é facilmente escalável, inicialmente segura ou eficiente, para consultas complexas e manipulação de dados quando comparada à persistência em banco de dados.

Por outro lado, a persistência em bancos de dados utiliza sistemas de gerenciamento de banco de dados (SGBD) para armazenar, recuperar e gerenciar dados de maneira estruturada, com uso de tabelas, índices, suporta operações complexas e transações. Também oferece recursos avançados como atomicidade, consistência, isolamento, durabilidade (ACID), além de segurança, backup e otimizações para acessos simultâneos.

b) Como o uso de operador lambda simplificou a impressão dos valores contidos nas entidades, nas versões mais recentes do Java?

O uso de operadores lambda, a partir do Java 8, simplificou a impressão de valores contidos em entidades, ao permitir que desenvolvedores escrevam código mais legível e conciso, sem a necessidade de criar classes anônimas para operações simples. Com operadores lambda, é possível passar comportamentos de forma direta e declarativa.

Por exemplo, para imprimir os valores de uma lista, antes do Java 8, seria necessário criar um loop explícito. Com lambdas, isso pode ser feito de maneira simplificada usando métodos de stream e operações como **forEach**

```
lista.forEach(elemento-> System.out.println(elemento));
```

Esse código substitui várias linhas de um loop for ou loop aprimorado for-each, tradicionais, ao mesmo tempo que torna o código mais expressivo, focado no que realmente se deseja realizar (a ação de imprimir), e não em como realizar o loop ou controle de fluxo.

c) Por que métodos acionados diretamente pelo método main, sem o uso de um objeto, precisam ser marcados como static?

O método main é o primeiro método a ser executado quando uma classe Java é compilada e executada, caso, obviamente, a classe contenha um método main. A principal razão pela qual o método main é definido como estático, é justamente para conceder a possibilidade de ser diretamente executado sem necessidade de instanciar um objeto do tipo da classe, para depois invocar o método main, ou seja, uma comodidade para evitar código desnecessário.

Por definição, atributos ou métodos estáticos pertencem à classe onde são definidos, e não ao objeto instanciado a partir dessa classe. Assim, se o atributo ou método forem públicos, podem ser referenciados externamente por outras classes, através somente do nome da classe e não por objeto. Contudo, métodos estáticos, em suas implementações, ao tentar invocar outros métodos sem uso de objetos de referência, somente podem invocar diretamente outros métodos estáticos. Logo, ambos métodos precisam ser estáticos, tanto o método que invoca o outro método, como o método invocado.