



Missão Prática | Nível 3 | Mundo 3

Filipe Alvim Santos - 202208291325

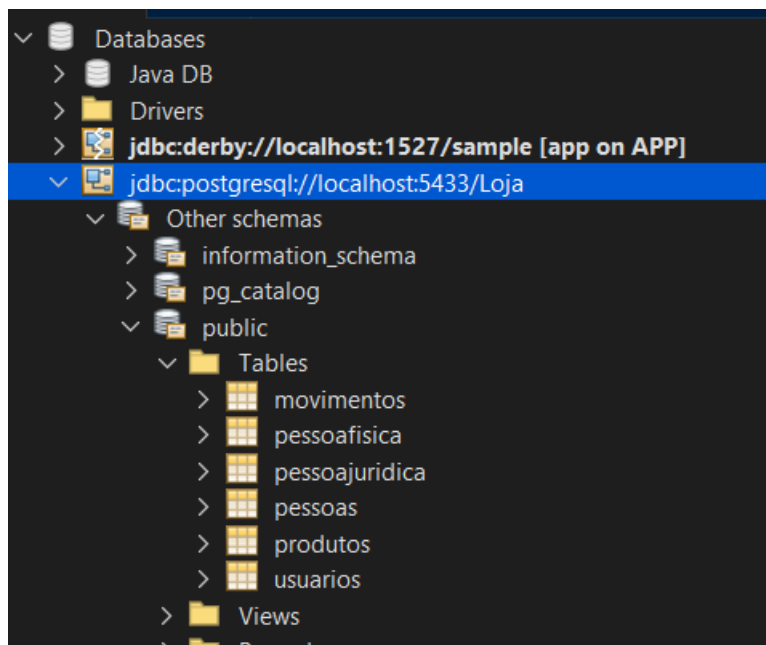
Campus Polo Sulacap – RJ / Desenvolvimento Full Stack
Nível 3: BackEnd sem banco não tem – Turma: 22.3 – 3º Semestre

Objetivo da Prática

Os objetivos da prática são: Implementar persistência com base no middleware JDBC, utilizando 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, terei criado um aplicativo cadastral com uso do PostgreSQL na persistência dos dados.

1º Procedimento | Mapeamento Objeto-Relacional e DAO



- Inserindo, alterando, excluindo e exibindo todos os resultados de Pessoa Física:

```
public class CadastroBD {
    public static void main(String[] args) {
        ConectorBD conectorBD = new ConectorBD();
        SequenceManager sequenceManager = new SequenceManager(conectorBD);

        // Instanciar uma PessoaFisica e persistir no banco de dados
        PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO(conectorBD, sequenceManager);
        PessoaFisica novaPessoaF = new PessoaFisica(0, cpf: "123.987.568-87", nome: "Filipe", logradouro: "Rua Manuel", cidade: "Rio de Janeiro", estado: "RJ", telefone: "12345-6789", email: "filipe@gmail.com");
        pessoaFisicaDAO.incluir(pessoaFisica: novaPessoaF);

        // Alterar os dados da pessoa fisica no banco
        // /int idPessoaParaAlterar = 19;
        // PessoaFisica pessoaAlterada = new PessoaFisica(idPessoaParaAlterar, "987.654.312-89", "Daniel", "Rua Nova", "São Paulo", "SP", "33333-3333", "daniel@gmail.com");
        // pessoaFisicaDAO.alterar(pessoaAlterada);

        // Excluir os dados da pessoa fisica no banco
        // /int idPessoaExcluir = 20;
        // pessoaFisicaDAO.excluir(idPessoaExcluir);
    }
}

CadastroBD.CadastroBD > main > novaPessoaF >
- CadastroBD (run) x
Logradouro: Rua 12, casa 3, Quitanda
Cidade: Riacho do Sul
Estado: PA
Telefone: 1111-1111
Email: joao@riacho.com
CPF: 11111111111
-----
ID: 4
Nome: Filipe
Logradouro: Rua Manuel
Cidade: Rio de Janeiro
Estado: RJ
Telefone: 12345-6789
Email: filipe@gmail.com
CPF: 123.987.568-87
-----
BUILD SUCCESSFUL (total time: 2 seconds)
```

```

public class CadastroBD {
    public static void main(String[] args) {
        ConectorBD conectorBD = new ConectorBD();
        SequenceManager sequenceManager = new SequenceManager(conectorBD);

        // Instanciar uma PessoaFisica e persistir no banco de dados
        PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO(conectorBD, sequenceManager);
        // PessoaFisica novaPessoaF = new PessoaFisica(0, "123.987.568-87", "Filipe", "Rua Manuel", "Rio de Janeiro", "RJ", "12345-6789", "filipe@gmail.com");
        // pessoaFisicaDAO.incluir(novaPessoaF);

        // Alterar os dados da pessoa fisica no banco
        int idPessoaParaAlterar = 4;
        PessoaFisica pessoaAlterada = new PessoaFisica(idPessoa, idPessoaParaAlterar, cpf="987.654.312-89", nome: "Daniel", logradouro: "Rua Nova", cidade: "São Paulo", estado: "SP", telefone: "33333-3333", email: "daniel@gmail.com");
        pessoaFisicaDAO.alterar(pessoaAlterada);

        // Excluir os dados da pessoa fisica no banco
        // int idPessoaExcluir = 20;
        // pessoaFisicaDAO.excluir(idPessoaExcluir);
    }
}

```

dstrobd.CadastroBD > main >

- CadastroBD (run) X

Cidade: Riacho do Sul
Estado: PA
Telefone: 1111-1111
Email: joao@riacho.com
CPF: 11111111111

ID: 4
Nome: Daniel
Logradouro: Rua Nova
Cidade: São Paulo
Estado: SP
Telefone: 33333-3333
Email: daniel@gmail.com
CPF: 987.654.312-89

BUILD SUCCESSFUL (total time: 2 seconds)

```

// Excluir os dados da pessoa fisica no banco
int idPessoaExcluir = 4;
pessoaFisicaDAO.excluir(id: idPessoaExcluir);

// Consultar todas as pessoas físicas do banco de dados e listar no console
List<PessoaFisica> pessoas = pessoaFisicaDAO.getPessoas();
for (PessoaFisica pessoa : pessoas) {
    System.out.println("ID: " + pessoa.getId());
    System.out.println("Nome: " + pessoa.getNome());
    System.out.println("Logradouro: " + pessoa.getLogradouro());
    System.out.println("Cidade: " + pessoa.getCidade());
    System.out.println("Estado: " + pessoa.getEstado());
    System.out.println("Telefone: " + pessoa.getTelefone());
    System.out.println("Email: " + pessoa.getEmail());
    System.out.println("CPF: " + pessoa.getCpf());
    System.out.println(x: "-----");
}

```

dstrobd.CadastroBD > main >

- CadastroBD (run) X

run:
ID: 1
Nome: Joao
Logradouro: Rua 12, casa 3, Quitanda
Cidade: Riacho do Sul
Estado: PA
Telefone: 1111-1111
Email: joao@riacho.com
CPF: 11111111111

BUILD SUCCESSFUL (total time: 2 seconds)

- Inserindo, alterando, excluindo e exibindo todos os resultados de Pessoa Jurídica:

```
//Instanciar uma PessoaJuridica e persistir no banco de dados
PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO(conectorBD, sequenceManager);
PessoaJuridica novaPessoaJ = new PessoaJuridica(id_pessoa: 0, cnpj: "12.987.548/0001-87", nome: "Startup", logradouro: "Rua do Amanhã", cidade: "Rio de Janeiro", estado: "RJ", telefone: "12345-6789", email: "startup@gmail.com");
pessoaJuridicaDAO.incluir(pessoaJuridica: novaPessoaJ);
```

cadastrobd.CadastroBD > main > novaPessoaJ >

CadastroBD (run) x

run:

ID: 2
Nome: JJC
Logradouro: Rua 11, Centro
Cidade: Riacho do Norte
Estado: PA
Telefone: 1212-1212
Email: jc@riacho.com
CNPJ: 222222222222

ID: 5
Nome: Startup
Logradouro: Rua do Amanhã
Cidade: Rio de Janeiro
Estado: RJ
Telefone: 12345-6789
Email: startup@gmail.com
CNPJ: 12.987.548/0001-87

BUILD SUCCESSFUL (total time: 2 seconds)

```
//Alterar os dados da pessoa jurídica no banco
int idPessoaParaAlterar = 5;
PessoaJuridica pessoaAlterada = new PessoaJuridica(idPessoaParaAlterar, cnpj: "54.857.158/0002-12", nome: "TTT", logradouro: "Rua do Futuro", cidade: "São Paulo", estado: "SP", telefone: "33333-3333", email: "ttt@gmail.com");
pessoaJuridicaDAO.alterar(pessoaJuridica pessoaAlterada);

CadastroBD.CadastroBD > main > idPessoaParaAlterar >
CadastroBD (run) x
run:
ID: 2
Nome: JJC
Logradouro: Rua 11, Centro
Cidade: Riacho do Norte
Estado: PA
Telefone: 1212-1212
Email: jjc@riacho.com
CNPJ: 222222222222
BUILD SUCCESSFUL (total time: 2 seconds)

//Excluir os dados da pessoa jurídica no banco
int idPessoaExcluir = 5;
pessoaJuridicaDAO.excluir(id: idPessoaExcluir);

//Consultar todas as pessoas físicas do banco de dados e listar no console
List<PessoaJuridica> pessoas = pessoaJuridicaDAO.getPessoas();
for (PessoaJuridica pessoa : pessoas) {
    System.out.println("ID: " + pessoa.getId());
    System.out.println("Nome: " + pessoa.getNome());
    System.out.println("Logradouro: " + pessoa.getLogradouro());
    System.out.println("Cidade: " + pessoa.getCidade());
    System.out.println("Estado: " + pessoa.getEstado());
    System.out.println("Telefone: " + pessoa.getTelefone());
    System.out.println("Email: " + pessoa.getEmail());
    System.out.println("CNPJ: " + pessoa.getCnpj());
    System.out.println(x: "-----");
}
}

CadastroBD.CadastroBD > main > idPessoaExcluir >
CadastroBD (run) x
run:
ID: 2
Nome: JJC
Logradouro: Rua 11, Centro
Cidade: Riacho do Norte
Estado: PA
Telefone: 1212-1212
Email: jjc@riacho.com
CNPJ: 222222222222
-----
BUILD SUCCESSFUL (total time: 2 seconds)
```

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

Resposta: Os componentes de middleware, como o JDBC (Java Database Connectivity), são essenciais para a comunicação entre diferentes sistemas e

aplicativos. Eles fornecem uma camada de abstração que permite que os aplicativos se comuniquem com bancos de dados e outros recursos, independentemente dos detalhes específicos desses sistemas. Isso facilita a interoperabilidade e a portabilidade, permitindo que os desenvolvedores se concentrem na lógica do aplicativo em vez de nos detalhes de baixo nível da comunicação entre sistemas. Além disso, o middleware pode oferecer recursos adicionais, como gerenciamento de transações, segurança e escalabilidade.

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

Resposta: A principal diferença entre *Statement* e *PreparedStatement* reside na forma como os dados são manipulados e na eficiência.

O *Statement* é usado para executar consultas SQL simples sem parâmetros. Cada vez que um *Statement* é executado, ele precisa ser compilado novamente pelo banco de dados, o que pode levar a uma sobrecarga significativa se a mesma consulta for executada repetidamente.

Por outro lado, *PreparedStatement* é usado para consultas SQL que têm parâmetros. Ele é pré-compilado no banco de dados e pode ser usado repetidamente, o que o torna mais eficiente para consultas que são executadas várias vezes. Além disso, *PreparedStatement* ajuda a prevenir ataques de injeção SQL, pois os parâmetros são automaticamente escapados.

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

Resposta: O padrão DAO (Data Access Object) melhora a manutenibilidade do software ao abstrair e encapsular todas as operações de acesso a dados em um objeto DAO. Isso separa a lógica de negócios da lógica de acesso a dados, permitindo que cada uma seja modificada independentemente da outra. Além disso, como o DAO fornece uma interface consistente para o acesso a dados, qualquer mudança no armazenamento de dados terá impacto mínimo no código do aplicativo. Isso facilita a manutenção e a escalabilidade do software.

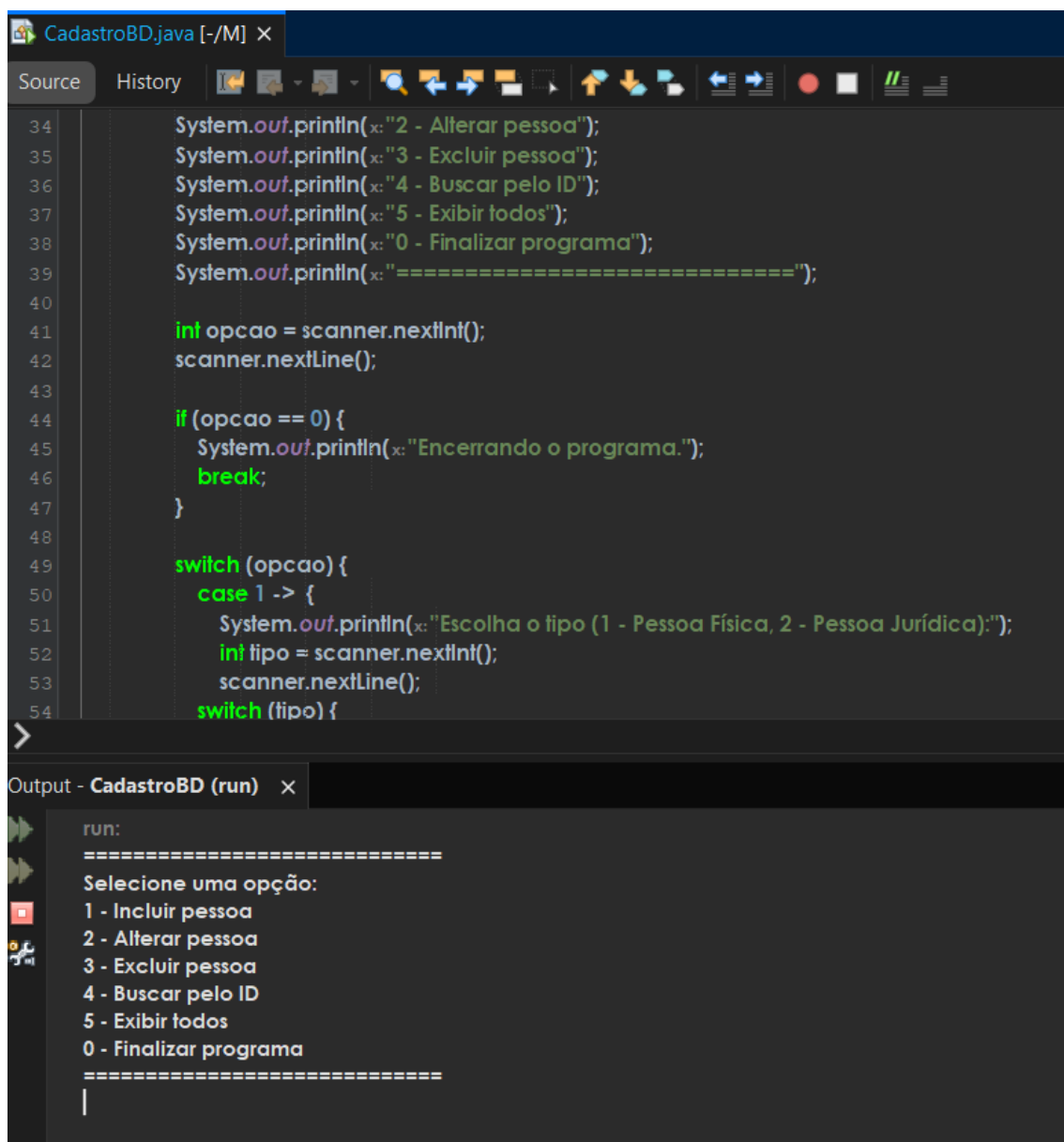
d) Como a herança é refletida no banco de dados, quando lidamos com um modelo estritamente relacional?

Resposta: A herança em um modelo de banco de dados relacional estrito pode ser refletida de várias maneiras, mas as três abordagens

mais comuns são a estratégia de tabela única, a estratégia de tabela por hierarquia de classe e a estratégia de tabela por subclasse concreta.

Cada uma dessas estratégias tem suas próprias vantagens e desvantagens em termos de complexidade, desempenho e flexibilidade, e a escolha entre elas geralmente depende das necessidades específicas do aplicativo.

2º Procedimento | Alimentando a Base



The screenshot shows an IDE window titled 'CadastroBD.java [-/M] x'. The 'Source' tab is active, displaying Java code. The code defines a menu with options: 2 - Alterar pessoa, 3 - Excluir pessoa, 4 - Buscar pelo ID, 5 - Exibir todos, and 0 - Finalizar programa. It uses a Scanner to read user input. An if statement checks for option 0 to break the loop. A switch statement handles option 1, prompting the user to choose between 'Pessoa Física' (1) and 'Pessoa Jurídica' (2). The 'Output' tab shows the program's execution, displaying the menu and the prompt 'Selecione uma opção:'.

```
34      System.out.println(x: "2 - Alterar pessoa");
35      System.out.println(x: "3 - Excluir pessoa");
36      System.out.println(x: "4 - Buscar pelo ID");
37      System.out.println(x: "5 - Exibir todos");
38      System.out.println(x: "0 - Finalizar programa");
39      System.out.println(x: "=====");
40
41      int opcao = scanner.nextInt();
42      scanner.nextLine();
43
44      if (opcao == 0) {
45          System.out.println(x: "Encerrando o programa.");
46          break;
47      }
48
49      switch (opcao) {
50          case 1 -> {
51              System.out.println(x: "Escolha o tipo (1 - Pessoa Física, 2 - Pessoa Jurídica):");
52              int tipo = scanner.nextInt();
53              scanner.nextLine();
54              switch (tipo) {
```

Output - CadastroBD (run) x

```
run:
=====
Selecione uma opção:
1 - Incluir pessoa
2 - Alterar pessoa
3 - Excluir pessoa
4 - Buscar pelo ID
5 - Exibir todos
0 - Finalizar programa
=====
|
```

```
=====
Selecione uma opção:
1 - Incluir pessoa
2 - Alterar pessoa
3 - Excluir pessoa
4 - Buscar pelo ID
5 - Exibir todos
0 - Finalizar programa
=====
5
Escolha o tipo (1 - Pessoa Física, 2 - Pessoa Jurídica):
1
ID: 1
Nome: Joao
Logradouro: Rua 12, casa 3, Quitanda
Cidade: Riacho do Sul
Estado: PA
Telefone: 1111-1111
Email: joao@riacho.com
-----
=====
```

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

Resposta: A persistência em arquivo e a persistência em banco de dados têm diferenças significativas:

A persistência em arquivo envolve o armazenamento de dados em um sistema de arquivos. É simples de implementar e não requer um sistema de gerenciamento de banco de dados. No entanto, a recuperação e a manipulação de dados podem ser mais lentas e mais complexas, especialmente para grandes volumes de dados. Além disso, a consistência e a integridade dos dados podem ser um desafio.

Por outro lado, a persistência em banco de dados envolve o armazenamento de dados em um banco de dados gerenciado por um sistema de gerenciamento de banco de dados (DBMS). Isso permite consultas complexas, recuperação rápida de dados e forte consistência e integridade dos dados. No entanto, requer mais recursos e complexidade para configurar e manter o DBMS. Além disso, os bancos de dados geralmente exigem um esquema predefinido, o que pode limitar a flexibilidade.

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

Resposta: O uso de operadores *lambda* no Java simplificou a impressão dos valores contidos nas entidades de várias maneiras. Antes do Java 8, você teria que usar loops explícitos para iterar sobre coleções e imprimir valores. Com a introdução de operadores *lambda*, você pode agora usar o método `forEach` juntamente com uma expressão *lambda* para imprimir valores de uma maneira muito mais concisa e legível.

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

Resposta: Métodos acionados diretamente pelo método `main` precisam ser marcados como *static* porque o método `main` é um método estático e só pode chamar diretamente outros métodos estáticos. Métodos estáticos pertencem à classe em si, e não a uma instância (objeto) da classe. Portanto, eles podem ser chamados sem a necessidade de criar um objeto da classe. Isso é útil para operações que não dependem do estado de um objeto específico, ou para métodos utilitários que não precisam acessar variáveis de instância.

Conclusão

Implementei um sistema simples de cadastro de pessoa físicas ou jurídicas com persistência em um Banco de Dados PostgreSQL. Achei bastante usual o trabalho, mesmo sendo simples.