POLO SANTA CRUZ - RIO DE JANEIRO - RJ/UNIVERSIDADE ESTÁCIO DE SÁ

Missão Prática | Nível 3 | Mundo 3

Curso: Desenvolvimento Full Stack
Disciplina Nível 2: RPG0015 – BackEnd sem banco não tem
Número da Turma: 2024.2
Semestre Letivo: Mundo-3
Aluno: Lukas Cauã Oliveira Xavier
Matrícula: 202305450556
URL GIT: https://github.com/Dev-Lukas2004/TrabalhoBancoN3

1º Título da Prática: BackEnd sem banco não tem

Desenvolvimento de um Sistema de Cadastro de Pessoas com Conexão ao Banco de Dados utilizando JDBC e Padrão DAO

2º Objetivo da Prática:

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

1º Procedimento | Mapeamento Objeto-Relacional e DAO

Imagem 1: Criar o projeto e configurar as bibliotecas necessárias:

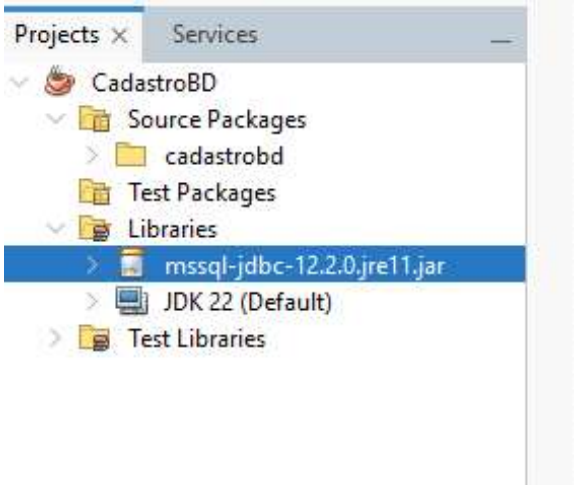


Imagem 2: Configurar o acesso ao banco pela aba de serviços do NetBeans:

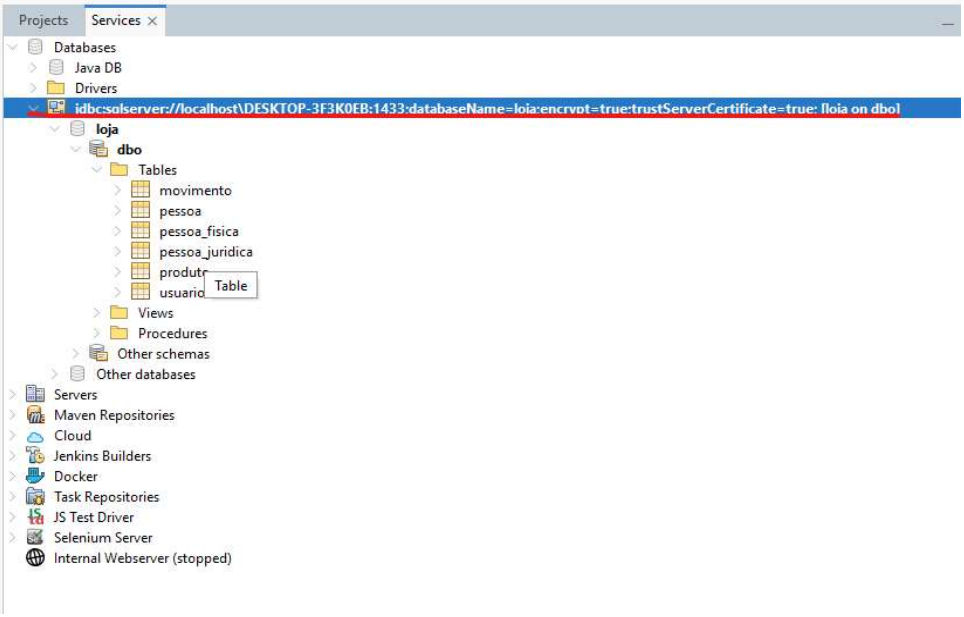


Imagem 3: Voltando ao projeto, criar o pacote **cadastrobd.model**, e nele criar as classes apresentadas a seguir:

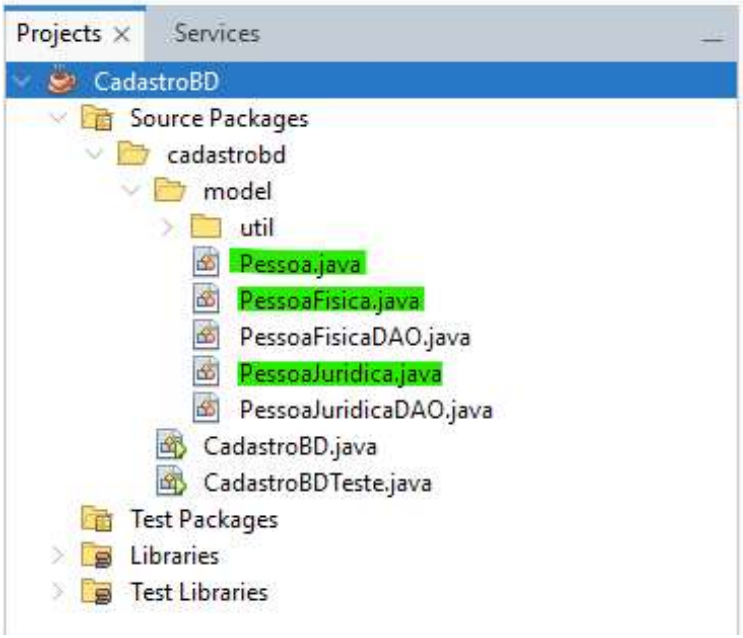


Imagem 4: Criar os **cadastro.model.util,** para inclusão das classes utilitárias que são apresentadas a seguir:

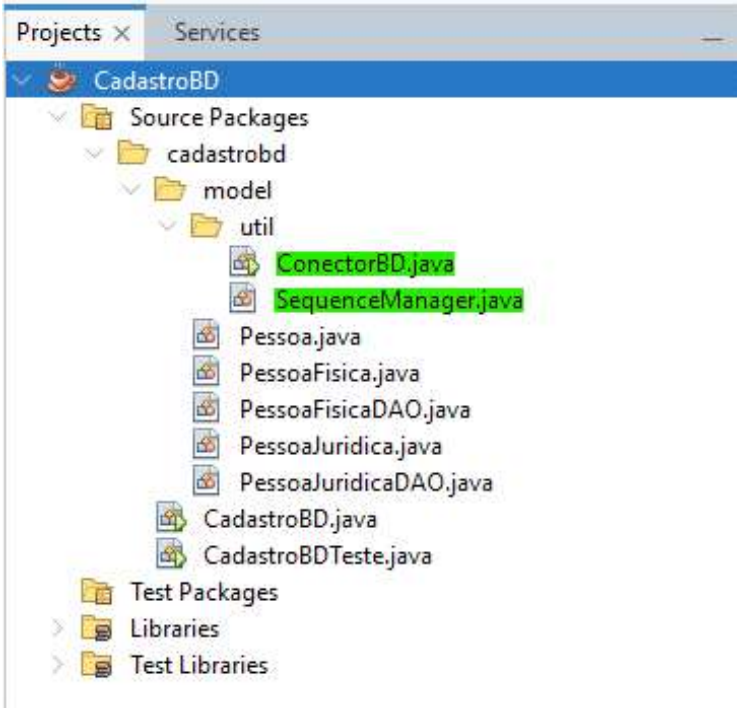


Imagem 5: Codificar as classes no padrão DAO, no pacote **cadastro.model:**

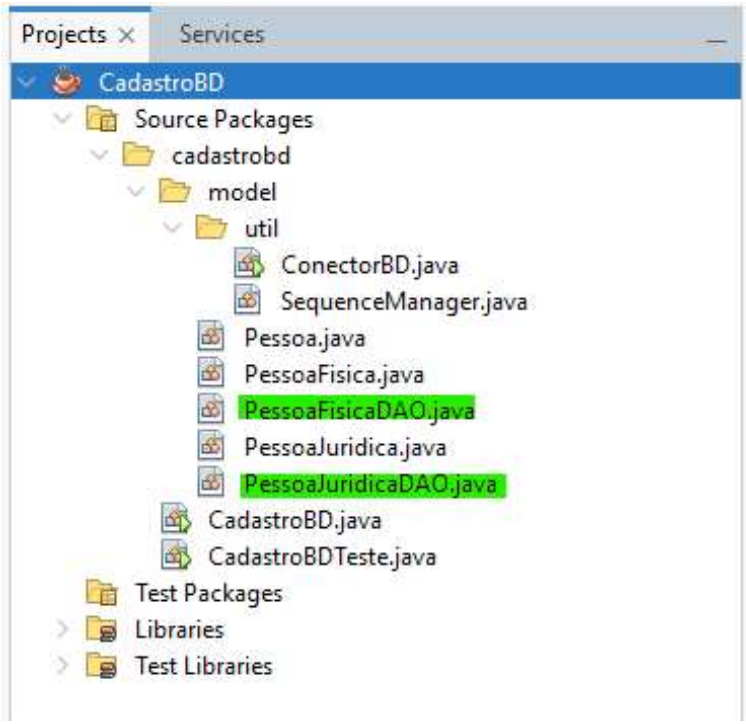
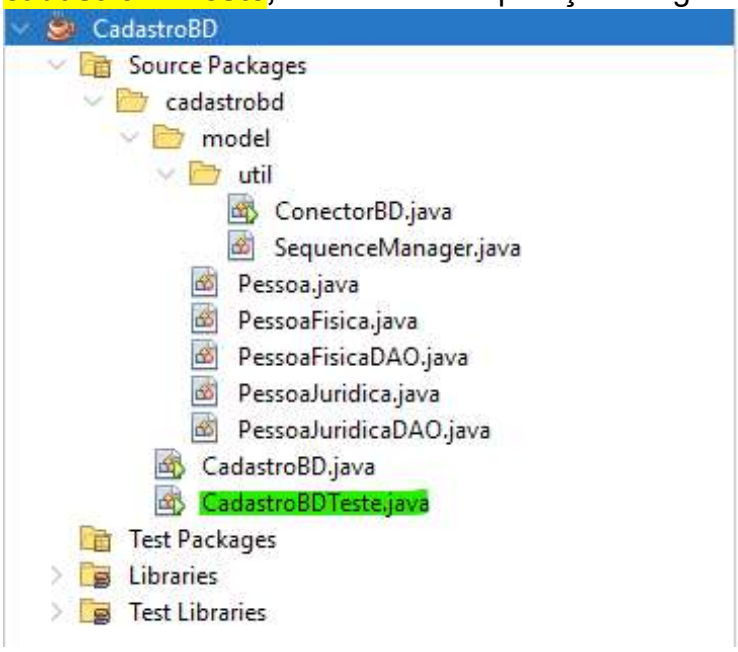


Imagem 6: Criar uma classe principal de testes com o nome **cadastroBDTeste,** efetuando as operações seguintes no método main:



A) INSERIR UMA PESSOA FÍSICA E PERSISTIR NO BANCO DE DADOS:

```
Escolha uma opcao:
1
(F) - Pessoa Fisica | (J) - Pessoa juridica
Escolha uma opcao:
F
Digite o nome para a pessoa fisica:
Lukas Caua
Digite o endereco:
rua cilon, 21
Digite a cidade:
Rio de Janeiro
Digite o estado:
RJ
Digite o telefone:
1212-1212
Digite o e-mail:
lukascaua10@gmail.com
Digite o CPF:
62586156632
## > Pessoa Fisica cadastrada com sucesso!
```

B) ALTERAR OS DADOS DA PESSOA FÍSICA NO BANCO:

```
Escolha uma opcao:
2
F - Alterar pessoa fisica | J - Alterar pessoa juridica
Escolha uma opcao:
f
Digite o ID da pessoa fisica que deseja alterar:
5
Digite o novo nome da pessoa fisica:
Caua Xavier
Digite o novo endereco:
Rua pedreira, 0
Digite a nova cidade:
Minas Gerais
Digite o novo estado:
MG
Digite o novo telefone:
1111-1111
Digite o novo e-mail:
cauaxavier00@yahoo@gmail.com
Digite o novo CPF:
12345678900
## > Pessoa fisica atualizada com sucesso.
```

C) CONSULTAR TODAS AS PESSOAS FÍSICAS DO BANCO DE DADOS E LISTAR NO CONSOLE:

```
##############################################
1 - Incluir pessoa
2 - Alterar pessoa
3 - Excluir pessoa
4 - Buscar pelo Id
5 - Listar todos
6 - Lista de pessoas fisicas
7 - Lista de pessoas juridicas
0 - Sair
##############################################
Escolha uma opcao:
6
ID: 3
Nome: Lukas Cau
Endereco: Rua teste, 21
Cidade: Rio de Janeiro
Estado: RJ
Telefone: 0000-0000
E-mail: lukascaua@gmail.com
CPF: 92809856743

ID: 5
Nome: Caua Xavier
Endereco: Rua pedreira, 0
Cidade: Minas Gerais
Estado: MG
Telefone: 1111-1111
E-mail: cauaxavier00@yahoo@gmail.com
CPF: 12345678900

##############################################
```

D) EXCLUIR A PESSOA FÍSICA CRIADA ANTERIORMENTE NO BANCO:

```
1 - Incluir pessoa
2 - Alterar pessoa
3 - Excluir pessoa
4 - Buscar pelo Id
5 - Listar todos
6 - Lista de pessoas fisicas
7 - Lista de pessoas juridicas
0 - Sair
############################################
Escolha uma opcao:
3
(F) - Excluir pessoa fisica | (J) - Excluir pessoa juridica
Escolha uma opcao:
f
Digite o ID da pessoa fisica para ser removida:
5
## > Pessoa fisica removida com sucesso.
```

E) INCLUIR UMA PESSOA JURÍDICA E PERSISTIR NO BANCO DE DADOS:

```
Escolha uma opcao:
1
(F) - Pessoa Fisica | (J) - Pessoa juridica
Escolha uma opcao:
J
Digite o nome para a pessoa juridica:
SuperCompras
Digite o endereco:
rua 20
Digite a cidade:
Rio de Janeiro
Digite o estado:
RJ
Digite o telefone:
0000-0000
Digite o e-mail:
super@gmail.com
Digite o CNPJ:
95972852000157
## > Pessoa Jur◆dica cadastrada com sucesso.
```

F) ALTERAR OS DADOS DA PESSOA JURÍDICA NO BANCO:

```
Escolha uma opcao:
2
F - Alterar pessoa fisica | J - Alterar pessoa juridica
Escolha uma opcao:
J
Digite o ID da pessoa juridica que deseja alterar:
9
Voce adicionou o ID: 9
Digite o novo nome da pessoa juridica:
Barbearia
Digite o novo endereco:
rua 30
Digite a nova cidade:
Fortaleza
Digite o novo estado:
CE
Digite o novo telefone:
1111-1111
Digite o novo email:
barbearia2@gmail.com
Digite o novo CNPJ:
17929329000134
## > ## >Pessoa juridica atualizada com sucesso.
```

G) CONSULTAR TODAS AS PESSOAS JURIDICAS DO BANCO E LISTAR NO CONSOLE:

```
5 - Listar todos
6 - Lista de pessoas fisicas
7 - Lista de pessoas juridicas
0 - Sair
#############################################
Escolha uma opcao:
7
ID: 4
Nome: SuperMarket
Endereco: Rua RE, 90 Bairro: S�o Joao
Cidade: Rio de Janeiro
Estado: RJ
Telefone: 2222-3333
E-mail: supermarkt@hotmail.com
CNPJ: 44012616000158

ID: 9
Nome: Barbearia
Endereco: rua 30
Cidade: Fortaleza
Estado: CE
Telefone: 1111-1111
E-mail: barbearia2@gmail.com
CNPJ: 17929329000134
```

H) EXCLUIR A PESSOA JURIDICA CRIADA ANTERIORMENTE NO BANCO:

```
##############################################
1 - Incluir pessoa
2 - Alterar pessoa
3 - Excluir pessoa
4 - Buscar pelo Id
5 - Listar todos
6 - Lista de pessoas fisicas
7 - Lista de pessoas juridicas
0 - Sair
##############################################
Escolha uma opcao:
3
(F) - Excluir pessoa fisica | (J) - Excluir pessoa juridica
Escolha uma opcao:
J
Digite o ID da pessoa juridica para ser removida:
9
## > Pessoa juridica excluida com sucesso.
```

Análise e Conclusão do 1º Procedimento:

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

Os componentes de middleware, como o JDBC (Java Database Connectivity), desempenham um papel crucial na arquitetura de sistemas de software, especialmente em aplicações empresariais e web. Aqui estão algumas das principais razões pelas quais esses componentes são importantes:

- **Abstração:** Esconde a complexidade da infraestrutura subjacente, permitindo que os desenvolvedores se concentrem na Lógica de negócio.

- **Reusabilidade:** Fornece componentes pré-construídos que podem ser reutilizados em diferentes projetos, acelerando o desenvolvimento.

- **Interoperabilidade:** Facilita a integração de diferentes sistemas e tecnologias, promovendo a heterogeneidade.

- **Escalabilidade:** Permite que as aplicações sejam escaladas de forma

mais fácil e eficiente.

- **Gerenciamento:** Oferece ferramentas para monitorar e gerenciar o desempenho das aplicações.

Beneficios específicos do JDBC:

- **Portabilidade:** Aplicações Java com JDBC podem ser executadas em diferentes plataformas e bancos de dados sem grandes modificações.

- **Produtividade:** Aumenta a produtividade dos desenvolvedores, pois eles não precisam escrever código SQL específico para cada banco de dados.

- **Segurança:** Oferece mecanismos para proteger os dados, como autenticação e autorização.

- **Transações:** Permite a realização de transações atômicas, garantindo a integridade dos dados.

Em resumo, o JDBC e outros componentes de middleware são fundamentais para criar aplicações que são escaláveis, seguras e fáceis de manter, ao mesmo tempo em que garantem uma comunicação eficiente e consistente com os sistemas de banco de dados.

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

Statement e **PreparedStatement** são duas interfaces do JDBC (Java Database Connectivity) utilizadas para executar comandos SQL em um banco de dados. Embora ambas sirvam para esse propósito, elas possuem características e aplicações distintas.

Statement

- **Criação:** A cada execução de um comando SQL, um novo objeto Statement é criado e enviado para o banco de dados.

- **Parâmetros:** Os parametros são concatenados diretamente à string do comando SQL, o que pode levar a problemas de segurança (injeção de SQL) e desempenho.

- **Compilação:** O comando SQL é compilado a cada execução.

PreparedStatement

- **Criação:** Um unico objeto PreparedStatement é criado para um determinado comando SQL, e este objeto pode ser reutilizado várias vezes com diferentes valores para os parâmetros.

- **Parâmetros:** Os parâmetros são passados como valores, não como parte da string do comando SQL, o que evita a injeção de SQL e melhora a segurança.

- **Compilação:** O comando SQL é compilado apenas uma vez, na criação do PreparedStatement, o que otimiza o desempenho, especialmente em consultas complexas ou que serão executadas várias vezes.

Resumo

- **Statement** é mais adequado para consultas SQL simples e situações em que a seguranga e o desempenho não são preocupações primárias.

- **PreparedStatement** é recomendado para consultas SQL complexas e para qualquer situação onde a segurança e o desempenho são importantes. Ele também facilita a manutenção do codigo e a gestão de parâmetros.

Em geral, o PreparedStatement é preferido na maioria das situações devido à sua segurança, desempenho e facilidade de uso.

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

O padrão DAO (Data Access Object) é uma ferramenta poderosa para melhorar a manutenibilidade do software, especialmente em aplicações que interagem com bancos de dados.

1 - Separação de Responsabilidades:

- **Lógica de Negócio vs Acesso a Dados:** O DAO encapsula todas as operações de acesso ao banco de dados em uma Única camada. Isso significa que a lógica de negócios da sua aplicação não precisa se preocupar com os detalhes de como os dados sao persistidos. Essa separaçãoo clara facilita a compreensão e a manutenção do código.

- **Facilita Testes:** Com a lógica de acesso a dados isolada, vocé pode testar a sua lógica de negócios de forma independente, utilizando mocks ou stubs para simular o comportamento do banco de dados.

2 - Abstração:

- **Independência do Banco de Dados:** O DAO fornece uma camada de abstração sobre o banco de dados específico. Isso significa que você pode trocar o banco de dados sem precisar modificar a lógica de negócios.

- **Facilidade de Mudanças:** Se você precisar mudar a estrutura do banco de dados ou a forma como os dados são armazenados, pode fazer isso alterando apenas a implementação do DAO, sem afetar o resto da aplicação.

3 - Reutilização:

- **Componentes Reutilizáveis:** Os DAOs podem ser reutilizados em diferentes partes da aplicação, reduzindo a duplicação de código e aumentando a consistência.

4 - Facilita a Manutenção:

- **Localização de Erros:** Ao isolar a lógica de acesso a dados em um único lugar, fica mais fácil encontrar e corrigir erros relacionados ao banco de dados.

- **Evita Mudanças Cascata:** Alterações na estrutura do banco de dados tendem a ter um impacto menor na aplicação como um todo, pois estão concentradas no DAO.

O padrão DAO melhora a manutenibilidade do software ao:

- **Aumentar a coesão:** Concentrando a lógica de acesso a dados em um único lugar.

Diminuir o acoplamento: Reduzindo a dependência entre a lógica de negócios e os detalhes de implementação do banco de dados.

Facilitar a testabilidade: Permitindo testar a lógica de negócios de forma isolada.

Aumentar a reutilização: Permitindo a reutilização de componentes.
Em resumo, o padrão DAO melhora a manutenibilidade do software ao fornecer uma estrutura clara e organizada para o acesso a dados, promovendo a separação de preocupações, encapsulamento, testabilidade, reutilização e facilidade de manutenção.

Em resumo, o padrão DAO melhora a manutenibilidade do software ao fornecer uma estrutura clara e organizada para o

acesso a dados, promovendo a separação de preocupações, encapsulamento, testabilidade, reutilização e facilidade de manutenção.

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

A herança, um conceito fundamental na programação orientada a objetos, não possui um mapeamento direto e nativo em bancos de dados relacionais. Isso ocorre porque bancos de dados relacionais são baseados em tabelas, que representam entidades planas e não hierárquicas como as classes em orientação a objetos.

Por que isso é um problema?

- **Modelagem de dados:** Muitas vezes, o mundo real possui hierarquias e relacionamentos de tipo-subtipo que a herança captura de forma natural.

- **Consultas**: Consultas que envolvem hierarquias podem se tornar complexas e ineficientes em um modelo puramente relacional.

Estratégias para Modelar Herança em Bancos Relacionais:
Diante desse desafio, diversas estratégias foram desenvolvidas para simular a herança em bancos de dados relacionais. Cada uma possui suas vantagens e desvantagens, e a escolha da melhor abordagem depende das características específicas da aplicação e do banco de dados utilizado.

1) Tabela por Subtipo:

- **Descrição:** Cada subtipo possui sua própria tabela, contendo os atributos específicos e uma chave estrangeira para a tabela da superclasse.

- **Vantagens:** Simplicidade e boa performance para consultas em subtipos específicos.

2) Tabela Única com Coluna Discriminadora:

- **Descrição:** Uma única tabela armazena todos os objetos, e uma coluna adicional (discriminadora) indica o tipo do objeto.

- **Vantagens:** Simplicidade e boa performance para consultas genéricas.

- **Desvantagens:** Pode levar a muitos valores nulos, especialmente se houver poucas diferenças entre os subtipos.

3) Tabela Principal + Tabelas Filhas:

- **Descrição:** Uma tabela principal armazena os atributos comuns, e tabelas filhas armazenam os atributos específicos, com uma chave estrangeira para a tabela principal.

- **Vantagens:** Boa normalização e flexibilidade para adicionar novos subtipos.

- **Desvantagens:** Consultas podem se tornar complexas devido aos joins.

4) Herança Total:

- **Descrição:** Cada subtipo possui sua própria tabela, contendo todos os atributos, inclusive os herdados.

- **Vantagens:** Simplicidade para consultas.

- **Desvantagens:** Redundância de dados e dificuldade de manter a consistência.

5) Mapeamento Objeto-Relacional (ORM):

- **Descrição:** Frameworks ORM, como Hibernate e Entity Framework, abstraem a complexidade do mapeamento entre objetos e bancos de dados, permitindo modelar a herança de forma mais natural.

- **Vantagens:** Facilidade de desenvolvimento e mapeamento transparente da herança.

- **Desvantagens:** Pode haver perda de desempenho em algumas situações.

Qual a melhor abordagem?

A escolha da melhor estratégia depende de diversos fatores, como:

- **Frequência de Consultas:** Se você precisa realizar muitas consultas que envolvem múltiplos subtipos, a tabela única com coluna discriminadora pode ser uma boa opção.
- **Número de Subtipos:** Se você tiver muitos subtipos, a tabela principal com tabelas filhas pode ser mais adequada.
- **Performance:** A performance das consultas pode variar significativamente entre as diferentes abordagens.
- **Complexidade da Hierarquia:** Hierarquias complexas podem exigir

combinações de diferentes estratégias.

Em Resumo:
A herança em bancos de dados relacionais é um desafio que exige um planejamento cuidadoso. A escolha da melhor estratégia depende das características específicas da aplicação e do banco de dados utilizado. Frameworks ORM podem simplificar o processo, mas é importante entender as implicações de cada abordagem para tomar a decisão mais adequada.

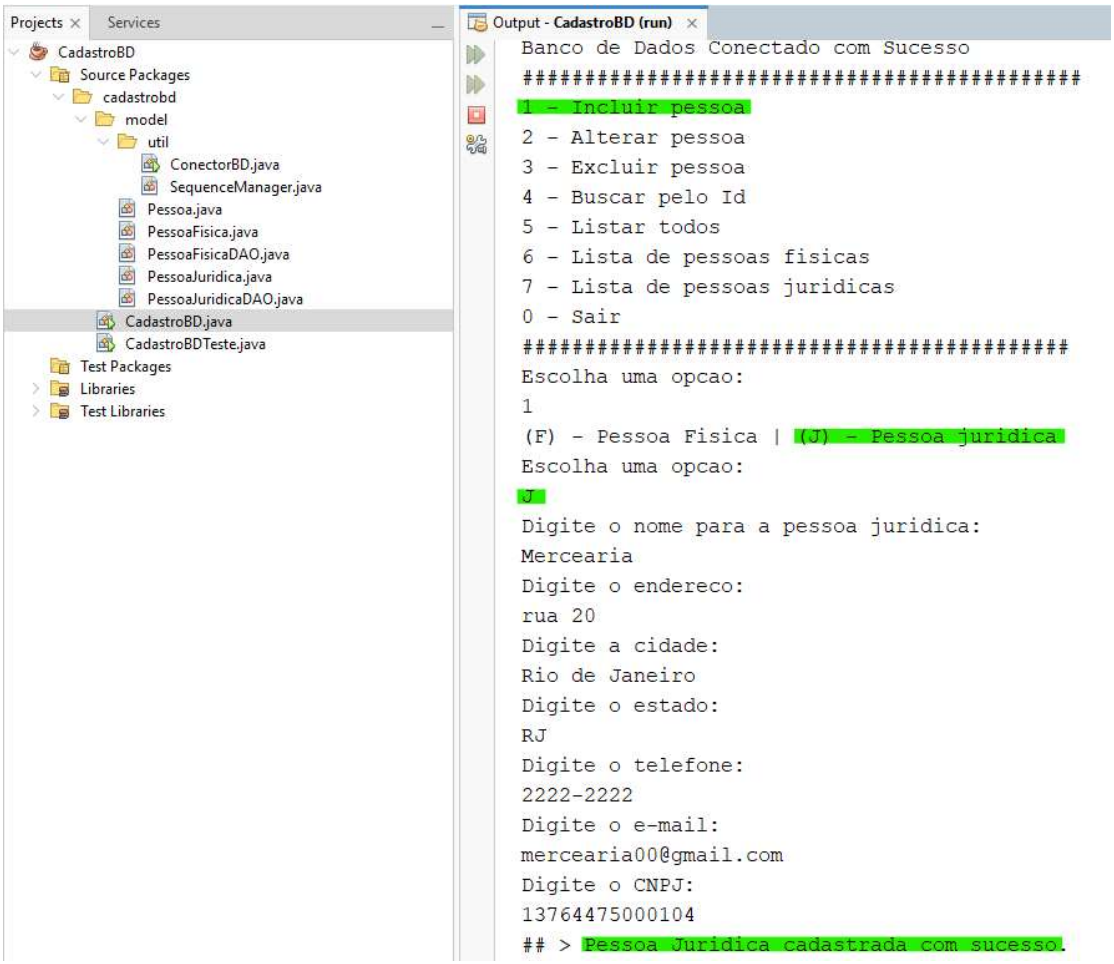
2º Procedimento | Alimentando a Base

- 1. Alterar o método **main** da classe principal do projeto, para implementação do cadastro em modo texto:

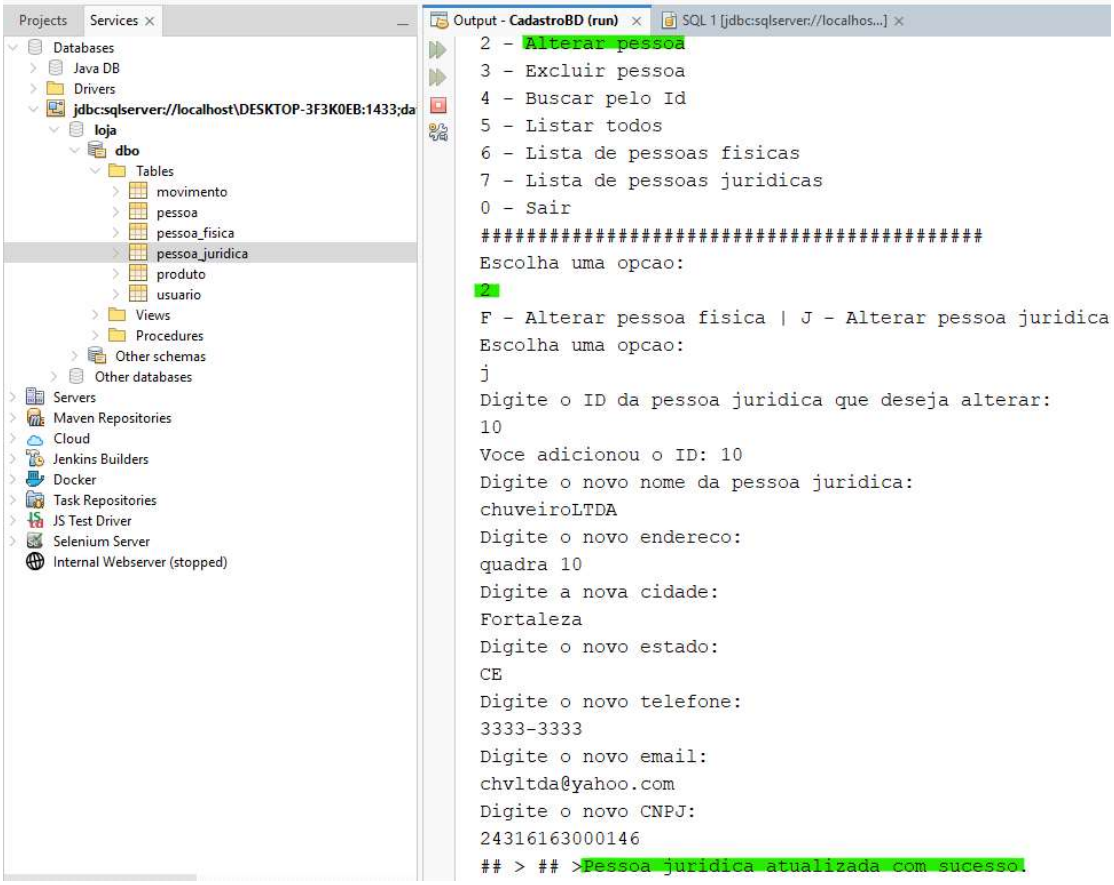
  - a) Apresentar as opções do programa para o usuário, sendo 1 para **Incluir**, 2 para **Alterar**, 3 para **Excluir**, 4 para **Exibir pelo id**, 5 para **Exibir todos** e 0 para **Finalizar a execução**.

```
##########################################
1 - Incluir pessoa
2 - Alterar pessoa
3 - Excluir pessoa
4 - Buscar pelo Id
5 - Listar todos
6 - Lista de pessoas fisicas
7 - Lista de pessoas juridicas
0 - Sair
##########################################
Escolha uma opcao:
```

  - b) Selecionada a opção **Incluir**, escolher o tipo de Pessoa (Física ou Jurídica), receber os dados a partir do teclado e adicionar no banco de dados através da classe DAO correta.



- c) Selecionada a opção **Alterar**, escolher o tipo de Pessoa (Física ou Jurídica), receber o id a partir do teclado e adicionar os atuais dados no Bnaco de dados através da classe DAO.



- d) Selecionada a opção **Excluir**, escolher o tipo de Pessoa (Física ou Jurídica), receber o id a partir do teclado e remover do banco de dados através do DAO.

```
##############################################
1 - Incluir pessoa
2 - Alterar pessoa
3 - Excluir pessoa
4 - Buscar pelo Id
5 - Listar todos
6 - Lista de pessoas fisicas
7 - Lista de pessoas juridicas
0 - Sair
##############################################
Escolha uma opcao:
3
(F) - Excluir pessoa fisica | (J) - Excluir pessoa juridica
Escolha uma opcao:
j
Digite o ID da pessoa juridica para ser removida:
10
## > Pessoa juridica excluida com sucesso.
```

- e) Selecionada a opção **Obter**, escolher o tipo de Pessoa (Física ou Jurídica), receber o id a partir do teclado e apresentar os dados atuais, recuperados do banco através do DAO.

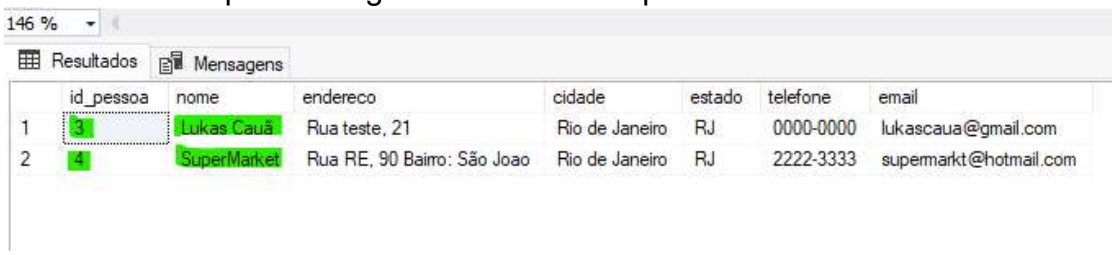
```
##############################################
1 - Incluir pessoa
2 - Alterar pessoa
3 - Excluir pessoa
4 - Buscar pelo Id
5 - Listar todos
6 - Lista de pessoas fisicas
7 - Lista de pessoas juridicas
0 - Sair
##############################################
Escolha uma opcao:
4
(F) - Buscar pessoa fisica | (J) - Buscar pessoa juridica
Escolha uma opcao:
f
Digite o ID da pessoa fisica:
3
Informacoes da pessoa fisica:
ID: 3
Nome: Lukas Cau
Endereco: Rua teste, 21
Cidade: Rio de Janeiro
Estado: RJ
Telefone: 0000-0000
E-mail: lukascaua@gmail.com
CPF: 92809856743
======>Busca feita com Sucesso.========
```

- f) Selecionada a opção **ObterTodos**, escolher o tipo de Pessoa (Física ou Jurídica), e apresentar os dados de todas as entidades presentes no banco de dados através do DAO.

```
##############################################
1 - Incluir pessoa
2 - Alterar pessoa
3 - Excluir pessoa
4 - Buscar pelo Id
5 - Listar todos
6 - Lista de pessoas fisicas
7 - Lista de pessoas juridicas
0 - Sair
##############################################
Escolha uma opcao:
5
Relacao de pessoas fisicas cadastradas:
ID: 3
Nome: Lukas Cau
Endereco: Rua teste, 21
Cidade: Rio de Janeiro
Estado: RJ
Telefone: 0000-0000
E-mail: lukascaua@gmail.com
CPF: 92809856743

Relacao de pessoas juridicas cadastradas:
ID: 4
Nome: SuperMarket
Endereco: Rua RE, 90 Bairro: S o Joao
Cidade: Rio de Janeiro
Estado: RJ
Telefone: 2222-3333
E-mail: supermarkt@hotmail.com
CNPJ: 44012616000158
==========>Exibicao Completa feita com Sucesso.==========
```

Observe: Exemplo dos registros no banco Sql Server.



	id_pessoa	nome	endereco	cidade	estado	telefone	email
1	3	Lukas Cauã	Rua teste, 21	Rio de Janeiro	RJ	0000-0000	lukascaua@gmail.com
2	4	SuperMarket	Rua RE, 90 Bairro: São Joao	Rio de Janeiro	RJ	2222-3333	supermarkt@hotmail.com

- g) Qualquer **exceção**,que possa ocorrer durante a execução do sistema deverá ser tratada.

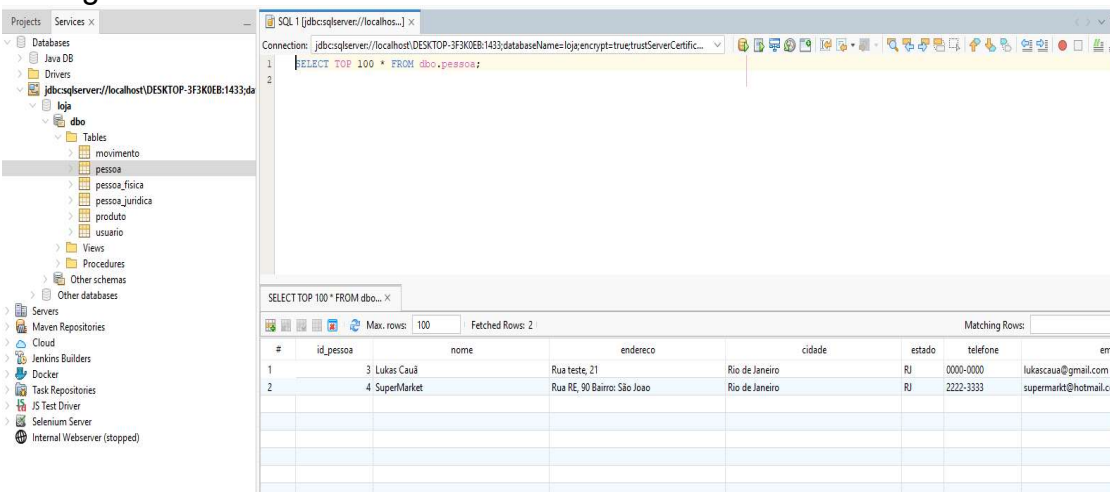
```
##############################################
1 - Incluir pessoa
2 - Alterar pessoa
3 - Excluir pessoa
4 - Buscar pelo Id
5 - Listar todos
6 - Lista de pessoas fisicas
7 - Lista de pessoas juridicas
0 - Sair
##############################################
Escolha uma opcao:
1
(F) - Pessoa Fisica | (J) - Pessoa juridica
Escolha uma opcao:
tste
## > A opcao escolhida invalida.
##############################################
1 - Incluir pessoa
2 - Alterar pessoa
3 - Excluir pessoa
4 - Buscar pelo Id
5 - Listar todos
6 - Lista de pessoas fisicas
7 - Lista de pessoas juridicas
0 - Sair
##############################################
Escolha uma opcao:
```

- **Exemplo de erro: (**uma exceção é lançada quando o usuário não informa o tipo de pessoa F ou J corretamente);

- h) Selecionada a opção de **Sair**, finalizar a execução do sistema.

```
##############################################
1 - Incluir pessoa
2 - Alterar pessoa
3 - Excluir pessoa
4 - Buscar pelo Id
5 - Listar todos
6 - Lista de pessoas fisicas
7 - Lista de pessoas juridicas
0 - Sair
##############################################
Escolha uma opcao:
0
## > Saindo....
BUILD SUCCESSFUL (total time: 14 minutes 47 seconds
```

2. Testar as funcionalidade do sistema:
Feitas as operações, verifique os dados no SQL Server utilizando a aba Services, na divisão Databases do NetBeans, ou através do SQL Server Management Studio:



Análise e Conclusão do 2º Procedimento:

- **Quais as diferenças entra persistência em arquivo e a presistência em bando de dados?**

A persistência em arquivos e a persistência em bancos de dados são duas abordagens distintas para armazenar e recuperar dados. Ambas têm seus usos, vantagens e desvantagens, e a escolha entre elas depende das necessidades específicas da aplicação. Segue abaixo as principais diferenças entre essas duas abordagens:

Persistência em Arquivo:

- Melhor para *Dados simples, pequenas aplicações e casos em que a flexibilidade e simplicidade são mais importantes.*
- *Menos suporte para operações complexas, escalabilidade e integridade dos dados.*

Persistência em Banco de Dados:

- Melhor para *Aplicações complexas, grandes volumes de dados e onde a integridade, segurança e desempenho são críticos.*
- *Oferece suporte avançado para consultas, transações, escalabilidade e administração.*

A escolha entre persistência em arquivo e persistência em banco de dados deve ser feita com base nas necessidades específicas da aplicação, no volume de dados, nos requisitos de desempenho e segurança, e na complexidade das operações necessárias.

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

No Java 8 e em versões subsequentes, o uso de operadores lambda trouxe uma série de melhorias na sintaxe e na expressividade do código, especialmente ao lidar com coleções e operações de fluxo de dados. Vou explicar como os operadores lambda simplificaram a impressão dos valores contidos nas entidades e como isso se reflete na prática.

Impressão de Valores em Coleções:
Antes do Java 8, a impressão de valores em uma coleção, como uma lista ou um conjunto, geralmente envolvia o uso de loops explícitos, como for ou foreach. Com a introdução das expressões lambda e das APIs de Streams no Java 8, a sintaxe para realizar operações em coleções, incluindo a impressão de valores, tornou-se mais concisa e legível.

Comparação das Abordagens:

- ***Código Antes do Java 8:*** *Requer um loop explícito e o gerenciamento manual da lógica de impressão. Pode ser mais verboso e propenso a erros quando a lógica é mais complexa.*

- ***Código Usando Lambda e Streams:*** *Simplifica o código ao permitir operações funcionais diretas sobre coleções. A sintaxe é mais declarativa e a operação é mais fácil de entender. Usar referências de método reduz ainda mais a necessidade de código adicional.*

Benefícios Adicionais

1. **Legibilidade:** O uso de lambda e streams melhora a legibilidade do código ao remover a necessidade de loops explícitos e ao tornar a intenção do código mais clara.

2. **Manutenção:** Com menos código e uma sintaxe mais clara, o código é mais fácil de manter e menos propenso a erros.

3. **Flexibilidade:** As APIs de Streams permitem operações complexas como filtragem, mapeamento e redução de dados de forma fluida e encadeada, o que pode ser combinado com a impressão para operações mais sofisticadas.

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

Em Java, o método main é o ponto de entrada para a execução de um programa. Ele precisa ser marcado como static porque é chamado pela Java Virtual Machine (JVM) para iniciar a aplicação, e isso ocorre sem a criação de uma instância da classe onde o método main está definido. Vou explicar em detalhes por que o método main, e por conseguinte, outros métodos acionados diretamente por ele, precisam ser estáticos:

1. Conceito de Métodos Estáticos

- ***Métodos Estáticos:*** *São métodos que pertencem à classe em vez de a uma instância específica dessa classe. Eles podem ser chamados diretamente usando o nome da classe, sem a necessidade de criar uma instância da classe.*
- ***Métodos de Instância:*** *São métodos que pertencem a uma instância específica de uma classe e requerem uma instância (objeto) da classe para serem chamados.*

2. Método main como Ponto de Entrada
O método main é definido como:

```
public static void main(String[] args) {
    // Código
}
```

Razões para o Método main ser Estático

1. Chamada Sem Instância: O método main é o ponto de entrada do programa e é chamado pela JVM quando o programa é iniciado. A JVM não cria uma instância da classe antes de chamar o método main. Portanto, o método main deve ser static para que a JVM possa invocá-lo diretamente, sem a necessidade de criar uma instância da classe.

2. Acesso a Métodos e Variáveis Estáticas: Dentro do método main, você pode acessar outros métodos e variáveis estáticas diretamente, sem criar uma instância da classe. Isso é útil para funções utilitárias ou operações que não dependem do estado de uma instância específica da classe.

3. Consistência e Simplicidade:
- Consistência: Marcar o método main como static é consistente com o fato de que ele deve ser executado independentemente de qualquer instância da classe. Isso evita a complexidade desnecessária de ter que criar uma instância da classe apenas para iniciar a execução do programa.

- Simplicidade: Permite que o programa comece sua execução de forma direta e simples. Sem a necessidade de instanciar um objeto, o ponto de entrada do programa pode ser facilmente acessado e iniciado pela JVM.

Em resumo: Métodos acionados diretamente pelo método main precisam ser marcados como static porque o método main é executado sem a necessidade de uma instância da classe. A marcação como static permite que métodos e variáveis sejam acessados diretamente sem criar um objeto, simplificando a execução inicial do programa.