



Campus: Polo Ingleses

Curso: Desenvolvimento Full Stack

Disciplina: Nível 3 - Back-end Sem Banco Não Tem

Turma: 9001

Semestre: 23.3

Aluna: Alexandre Henrique Fernandes Nolla

1º Procedimento | Criação das Entidades e Sistema de Persistência

Objetivo da prática: Testar a conexão do Driver JDBC através de uma carga inicial no banco de dados implementada na classe de testes.

Resultados da execução dos códigos:

```
alexandrenolla@Alexandres-MBP CadastroDB % /usr/bin/env /Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java @/var/folders/tq/5zp30cl12wd61jpknd6qcl40000gn/T/cp_cbb89h8dhajic7huyhsxyc0h3.argfile com.CadastroDB.CadastroDbApplicationTests
Nome: Joao
Logradouro: Rua 11, Centro
ESTADO: PA
Telefone: 1111-1111
Email: joao@riacho.com
CPF: 111111111
Id: 21
Nome: JJC
Logradouro: Rua 11, Centro
Estado: PA
Telefone: 1212-1212
Email: jjc@riacho.com
CNPJ: 111111111
alexandrenolla@Alexandres-MBP CadastroDB %
```

Análise e Conclusão:

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

Os componentes de middleware, exemplificados pelo JDBC (Java Database Connectivity), desempenham um papel fundamental no desenvolvimento de aplicações, atuando como uma camada intermediária entre o código da aplicação e os recursos subjacentes, como bancos de dados. O JDBC, uma interface específica para a comunicação entre aplicações Java e bancos de dados, simplifica a complexidade associada ao acesso e manipulação de dados.

A relevância do JDBC e de componentes de middleware semelhantes reside na simplificação e abstração das operações relacionadas a dados. Eles estabelecem uma conexão uniforme com diferentes tipos de bancos de dados, permitindo que os desenvolvedores foquem na

lógica da aplicação, sem a necessidade de lidar diretamente com os detalhes específicos de cada banco de dados.

Adicionalmente, o JDBC contribui para a portabilidade das aplicações, possibilitando o uso do mesmo código Java com diferentes bancos de dados, contanto que haja suporte JDBC para esses sistemas. Isso promove flexibilidade e facilita a manutenção do código ao longo do tempo.

Em resumo, os componentes de middleware, como o JDBC, desempenham um papel essencial ao simplificar a interação entre aplicações e recursos, promovendo a modularidade, portabilidade e eficiência no desenvolvimento de software.

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

A diferença principal entre o uso de Statement e PreparedStatement na manipulação de dados está na forma como lidam com consultas SQL e parâmetros. O Statement é utilizado para executar consultas SQL simples, mas não lida eficientemente com parâmetros, o que pode resultar em vulnerabilidades de segurança como injeção de SQL. Por outro lado, o PreparedStatement é mais seguro e eficiente quando se trabalha com consultas parametrizadas, pois permite a definição de parâmetros de forma segura, evitando a injeção de SQL e melhorando o desempenho por meio do uso de consultas preparadas, que podem ser reutilizadas com diferentes valores de parâmetros.

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

O padrão DAO (Data Access Object) é um padrão de projeto de software que fornece uma camada de abstração entre a lógica de negócios da aplicação e o acesso aos dados. Isso permite que a lógica de negócios da aplicação seja isolada das mudanças no armazenamento de dados, tornando o software mais manutenível.

O DAO encapsula as operações de acesso a dados em objetos específicos. Isso significa que a lógica de negócios da aplicação não precisa lidar diretamente com a API do banco de dados. Em vez disso, a aplicação pode simplesmente chamar os métodos do DAO para acessar os dados.

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

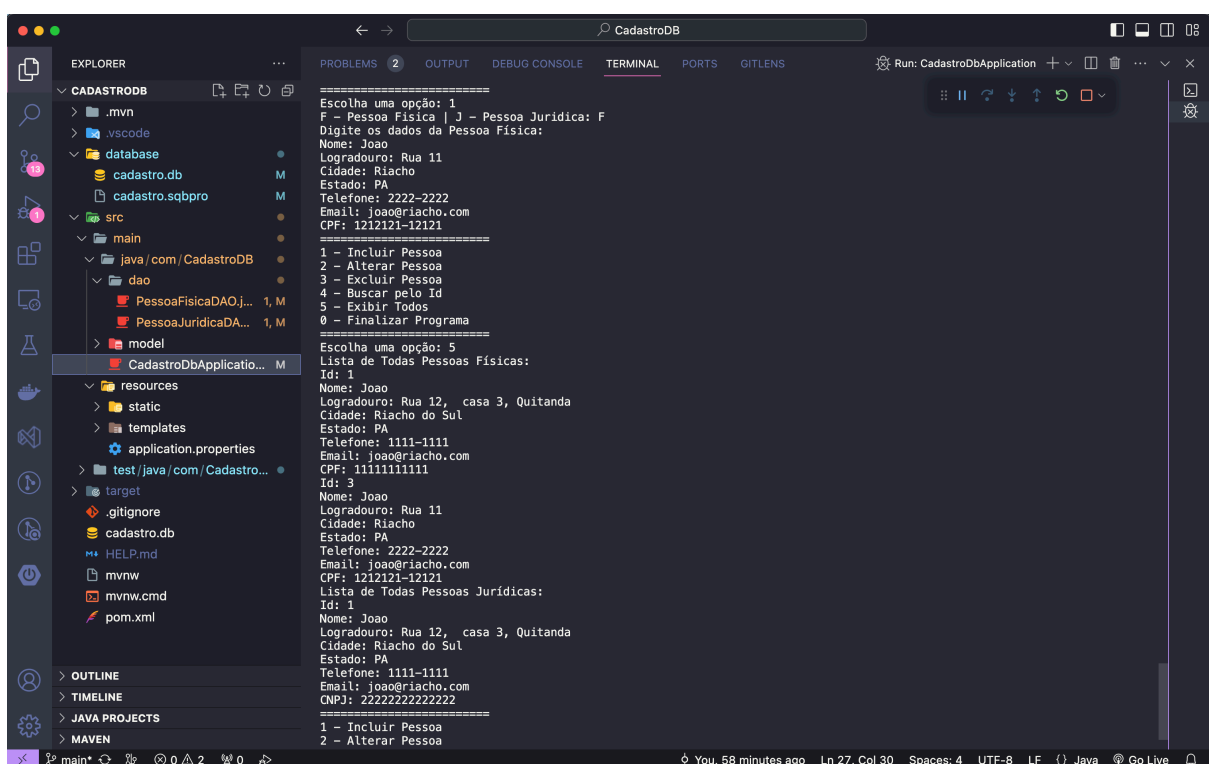
Quando lidamos com um modelo estritamente relacional no banco de dados e utilizamos herança em nossa modelagem, ela é frequentemente refletida através de tabelas. Cada

classe na hierarquia de herança é representada por uma tabela separada, e as relações entre essas tabelas espelham a estrutura de herança no modelo de dados.

2º Procedimento | Alimentando a Base

Objetivo da prática: Popular o Banco de dados através de um menu interativo no console que simula um CRUD renderizado no front-end.

Resultados da execução dos códigos:



```
=====
Escolha uma opção: 1
F - Pessoa Física | J - Pessoa Jurídica: F
Digite os dados da Pessoa Física:
Nome: Joao
Logradouro: Rua 11
Cidade: Riacho
Estado: PA
Telefone: 2222-2222
Email: joao@riacho.com
CPF: 1212121-12121
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
=====
Escolha uma opção: 5
Lista de Todas Pessoas Físicas:
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
Id: 3
Nome: Joao
Logradouro: Rua 11
Cidade: Riacho
Estado: PA
Telefone: 2222-2222
Email: joao@riacho.com
CPF: 1212121-12121
Lista de Todas Pessoas Jurídicas:
Id: 1
Nome: Joao
Logradouro: Rua 12, casa 3, Quitanda
Cidade: Riacho do Sul
Estado: PA
Telefone: 1111-1111
Email: joao@riacho.com
CNPJ: 22222222222222
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
```

Análise e Conclusão:

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

A persistência em arquivo e a persistência em banco de dados são duas abordagens distintas para armazenar e recuperar dados em um sistema. A principal diferença reside no local onde as informações são mantidas e na estrutura utilizada para organizá-las.

Na persistência em arquivo, os dados são armazenados em arquivos do sistema de arquivos. Cada entidade ou conjunto de dados pode ser representado por um arquivo separado, e o acesso aos dados é feito diretamente nos arquivos. Essa abordagem é simples e adequada

para volumes pequenos de dados, mas pode tornar-se menos eficiente e mais suscetível a problemas de concorrência em escalas maiores.

Por outro lado, na persistência em banco de dados, os dados são organizados em tabelas dentro de um sistema de gerenciamento de banco de dados (SGBD). O acesso aos dados é realizado por meio de consultas SQL, e a estrutura de banco de dados oferece uma maneira mais organizada e eficiente de gerenciar grandes conjuntos de informações. Além disso, os SGBDs proporcionam funcionalidades como transações, controle de concorrência e integridade referencial.

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 nas versões mais recentes do Java simplificou significativamente a tarefa de imprimir os valores contidos nas entidades. Anteriormente, para realizar essa operação, era necessário escrever um bloco de código mais extenso utilizando as versões tradicionais do Java. Com a introdução de operadores lambda, a sintaxe tornou-se mais concisa, permitindo expressar a lógica de impressão de forma mais direta e compacta.

Esses operadores possibilitam a definição de funções anônimas de maneira mais sucinta, eliminando a necessidade de criar classes ou métodos adicionais apenas para tarefas simples como a impressão de valores. Ao adotar operadores lambda, o código torna-se mais legível e eficiente, proporcionando uma abordagem mais moderna e expressiva para operações como a impressão de dados contidos nas entidades em Java.

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

Métodos acionados diretamente pelo método main, sem o uso de um objeto, precisam ser marcados como static devido à natureza estática do método main em Java. O método main é o ponto de entrada da execução do programa e pertence à classe, não a instâncias específicas da classe. Portanto, para que métodos possam ser chamados diretamente a partir do método main, sem a criação de uma instância da classe, eles devem ser declarados como static.

A marcação como static indica que o método pertence à classe em si e não requer uma instância específica para ser invocado. Isso é essencial para garantir que o método main possa ser chamado sem a necessidade de criar um objeto da classe principal.

