Campus: Polo Jacarepaguá, Rio de Janeiro/RJ

Curso: Desenvolvimento Full Stack

Disciplina: RPG0016 – Back-end Sem Banco Não Tem

Semestre Letivo: 2023.1 FLEX

Integrante: Tatiana Mara Vieira Pinto **Matrícula:** 202304161828

Github: https://github.com/TatianaMaraVieira/Mundo03_Nivel03.git

Objetivo da Prática

Cadastro de pessoa física:

```
ID: 4
Nome: Julio Marques
Endereco: Rua Eugenio Mussoi
Cidade: Santa Maria
Estado: RS
Telefone: 55996200959
E-mail: juliomarquesjr@yahoo.com.br
CPF: 91231954312

ID: 6
Nome: Julio Marques
Endereco: Rua XX, Centro
Cidade: Santa Maria
Estado: RS
Telefone: 9529341234
E-mail: email@email.com
CPF: 9012912412
```

Cadastro de pessoa jurídica:

```
ID: 5
Nome: Maxim Web Sites
Endereco: Rua Eugenio Mussoi
Cidade: Santa Maria
Estado: RS
Telefone: 9981921312
E-mail: contato@maximweb.com.br
CNPJ: 99918832099534
```

Pessoa Editada:

```
ID: 4
Nome: Julio Cesar Marques Jr
Endereco: Rua XXX, Centro
Cidade: Santa Maria
Estado: RS
Telefone: 99123123410
E-mail: julio@email.com
CPF: 99912312341
```

1º Procedimento | Mapeamento Objeto-Relacional e DAO

Análise e Conclusão:

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

   Eles proporcionam uma camada de abstração que permite aos desenvolvedores escrever código independente de banco de dados, aumentando a portabilidade e a reutilização do código. O JDBC, em particular, permite conexões flexíveis e seguras a uma ampla variedade de bancos de dados, suporta transações para garantir a integridade dos dados, e oferece recursos como pooling de conexões para melhorar o desempenho e a escalabilidade das aplicações. Além disso, esses componentes ajudam a proteger as aplicações contra-ataques comuns, como a injeção de SQL, através de mecanismos robustos de segurança.

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

   A principal diferença entre elas é que Statement é usado para consultas estáticas sem parâmetros, enquanto PreparedStatement é usado para consultas pré-compiladas com parâmetros dinâmicos. PreparedStatement oferece melhores desempenhos por ser pré-compilado, maior segurança contra injeção de SQL através da parametrização segura de entradas, e maior facilidade de uso ao inserir valores dinâmicos. Portanto, PreparedStatement é geralmente a escolha preferida para consultas repetidas e para a manipulação segura de dados dinâmicos.

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

   O padrão DAO contribui significativamente para a manutenibilidade do software ao promover uma clara separação entre a lógica de acesso a dados e o restante da aplicação. Isso facilita a manutenção, pois as alterações nos métodos de acesso ou nas fontes de dados ficam confinadas aos DAOs, sem afetar a lógica de negócios. Além disso, centraliza o acesso a dados, aumentando a reusabilidade e reduzindo a duplicação de código. O padrão também oferece flexibilidade, permitindo mudanças de fontes de dados com impacto mínimo sobre o código existente, e melhora a testabilidade, facilitando a criação de testes unitários pela possibilidade de mockar os DAOs. Em suma, o padrão DAO torna o software mais organizado, adaptável e fácil de testar e manter.

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

   Na modelagem de bancos de dados relacionais, a herança de um sistema orientado a objetos pode ser mapeada de três formas principais, sem uma distinção clara em itens. Primeiramente, pode-se optar por uma abordagem de tabela única, onde uma única tabela armazena os atributos de todas as classes na hierarquia, resultando frequentemente em muitas colunas nulas para atributos que não se aplicam a todas as instâncias. Alternativamente, a estratégia de tabela por classe concreta cria uma tabela separada para cada classe concreta, incluindo tanto seus atributos específicos quanto os herdados, o que pode levar à duplicação de dados herdados em múltiplas tabelas. Por fim, a abordagem de tabela por hierarquia de classe designa uma tabela para cada classe na hierarquia, com tabelas de subclasses contendo apenas atributos únicos àquela classe e uma referência à tabela da classe base. A escolha entre essas metodologias depende de um equilíbrio entre a necessidade de eficiência nas consultas, facilidade de manutenção e o desejo de evitar redundância de dados e colunas nulas, ajustando-se às especificidades e requisitos do projeto em questão.

2º Procedimento | Alimentando a Base

```
##########################################
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:
```

Incluir pessoa:

```
(F) - Pessoa Fisica | (J) - Pessoa juridica
Escolha uma opcao:
f
Digite o nome para a pessoa fisica:
Julio Marques
Digite o endereco:
Rua XX, Centro
Digite a cidade:
Santa Maria
Digite o estado:
RS
Digite o telefone:
9529341234
Digite o e-mail:
email@email.com
Digite o CPF:
9012912412
```

Alterar pessoa:

```
F - Alterar pessoa fisica | J - Alterar pessoa juridica
Escolha uma opcao:
F
Digite o ID da pessoa fisica que deseja alterar:
4
Digite o novo nome da pessoa fisica:
Julio Cesar Marques Jr
Digite o novo endereco:
Rua XXX, Centro
Digite a nova cidade:
Santa Maria
Digite o novo estado:
RS
Digite o novo telefone:
99123123410
Digite o novo e-mail:
julio@email.com
Digite o novo CPF:
99912312341
```

Exibir todos:

```
5
Relacao de pessoas fisicas cadastradas:
ID: 4
Nome: Julio Cesar Marques Jr
Endereco: Rua XXX, Centro
Cidade: Santa Maria
Estado: RS
Telefone: 99123123410
E-mail: julio@email.com
CPF: 99912312341

Relacao de pessoas juridicas cadastradas:
ID: 5
Nome: Maxim Web Sites
Endereco: Rua Eugenio Mussoi
Cidade: Santa Maria
Estado: RS
Telefone: 9981921312
E-mail: contato@maximweb.com.br
CNPJ: 99918832099534
```

Conclusão:

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

   A escolha entre persistência em arquivo e em banco de dados depende fundamentalmente das necessidades específicas de cada projeto. Enquanto a persistência em arquivo se destaca pela sua simplicidade e facilidade de implementação, sendo adequada para projetos menores ou com requisitos de armazenamento simples, ela pode encontrar limitações à medida que a complexidade dos dados e as necessidades de segurança aumentam. Por outro lado, a persistência em banco de dados oferece uma solução mais robusta, capaz de lidar com grandes volumes de dados, complexidade de consultas, controle de acesso concorrente e requisitos rigorosos de segurança e integridade dos dados. Embora mais complexa em termos de configuração e uso, a persistência em banco de dados é preferível para aplicações que demandam funcionalidades avançadas de gerenciamento de dados. A decisão entre as duas abordagens deve, portanto, considerar o equilíbrio entre simplicidade e necessidade de funcionalidades mais complexas, baseando-se no volume de dados, complexidade das operações e requisitos de segurança.

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 no Java 8 transformou significativamente a maneira de manipular coleções, tornando o código para impressão e outras operações sobre dados mais conciso e legível. Antes, loops como for ou for-each eram necessários para tarefas simples, como imprimir elementos de uma lista. Com os lambdas e o Stream API, essas operações se tornam mais diretas, permitindo expressões compactas e claras, como lista.forEach(System.out::println). Essa mudança não apenas simplifica o código, mas também facilita a implementação de funcionalidades mais complexas e a execução de operações em paralelo, otimizando o desenvolvimento e o aproveitamento dos recursos do sistema. Em resumo, os operadores lambda representam um avanço na programação Java, promovendo um estilo mais funcional e declarativo.

3. 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 para que sejam acessíveis dentro do contexto estático inicial do programa. Isso permite que a JVM execute o programa sem a necessidade de instanciar objetos, seguindo o fluxo de execução definido a partir do ponto de entrada estático do programa.