



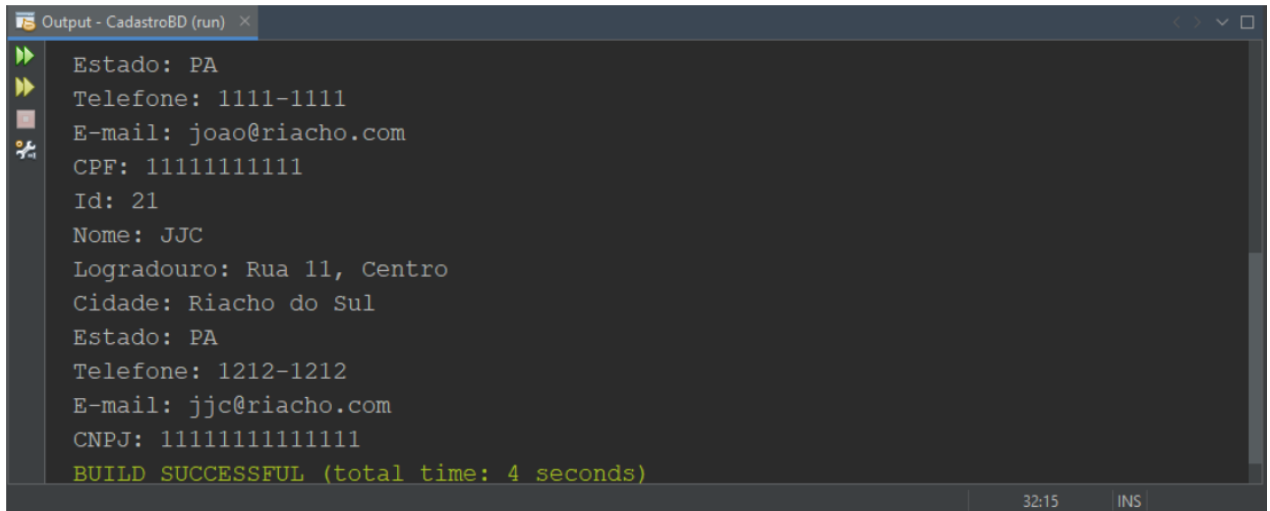
Estácio

UNIVERSIDADE ESTÁCIO DE SÁ POLO IBIRITÉ
CURSO: DESENVOLVIMENTO FULL STACK
DISCIPLINA: BACK-END SEM BANCO NÃO TEM

EVERTON GOMES COSTA
TURMA: 22.3
3º SEMESTRE

IBIRITÉ-MG
2023

1º Procedimento



```
Output - CadastroBD (run)
Estado: PA
Telefone: 1111-1111
E-mail: joao@riacho.com
CPF: 11111111111
Id: 21
Nome: JJC
Logradouro: Rua 11, Centro
Cidade: Riacho do Sul
Estado: PA
Telefone: 1212-1212
E-mail: jjc@riacho.com
CNPJ: 11111111111111
BUILD SUCCESSFUL (total time: 4 seconds)
```

Análise e Conclusão:

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

Os componentes de middleware, como o JDBC (Java Database Connectivity), são essenciais para facilitar a comunicação entre aplicativos Java e bancos de dados. O JDBC permite que aplicativos Java executem operações de banco de dados de maneira eficiente e padronizada. Sua importância reside na abstração da complexidade das interações com diferentes bancos de dados, fornecendo uma interface para comum aos desenvolvedores. Isso simplifica o desenvolvimento, promove a portabilidade e melhora a manutenibilidade dos aplicativos.

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

Ambos, Statement e PreparedStatement, são interfaces no JDBC usadas para executar consultas SQL, mas há diferenças cruciais:

Instrução: Geralmente usados para consultas SQL estáticas. As consultas são construídas como strings e podem apresentar vulnerabilidades de segurança, como injeção de SQL, se não forem tratadas corretamente.

PreparedStatement: Usado para consultas parametrizadas. As consultas são pré-compiladas, o que melhora o desempenho e evita a injeção de SQL, pois os parâmetros são tratados separadamente. Além disso, o uso de PreparedStatement é preferível para consultas que serão realizadas várias vezes, pois o plano de execução é reutilizado.

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

O padrão DAO (Data Access Object) melhorou a manutenibilidade do software separando a lógica de acesso dos dados da lógica de negócios. Isso é alcançado por meio da criação de interfaces ou classes específicas para manipulação de dados, fornecendo uma camada de abstração entre o código da aplicação e o banco de dados. As principais vantagens incluem:

Organização: As operações de acesso a dados são centralizadas em classes DAO, tornando o código mais organizado e fácil de entender.

Reutilização: A lógica de acesso aos dados pode ser reutilizada em várias partes do aplicativo, promovendo a modularidade.

Manutenção Facilitada: Mudanças no esquema do banco de dados podem ser isoladas na camada DAO, diminuindo o impacto nas outras partes do sistema.

Testabilidade: A separação entre a lógica de negócios e a lógica de acesso a dados facilita a realização de testes unitários e de integração.

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

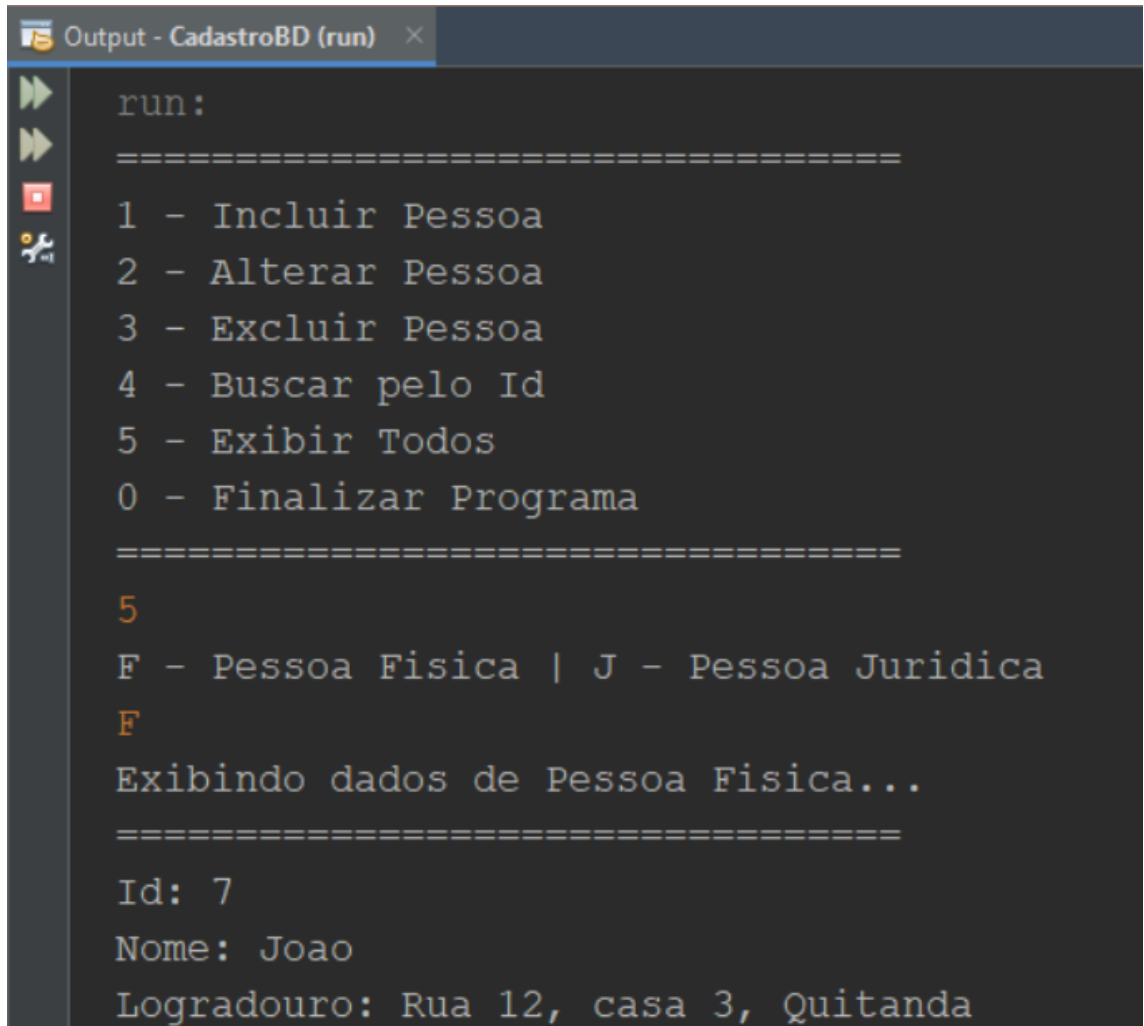
Em um modelo de prejuízo relacional, a herança é geralmente inovadora usando uma abordagem de tabela por classe (ou tabela por subtipo). Cada classe no modelo de objetos tem uma tabela correspondente no banco de dados. Existem duas abordagens principais:

Herança de Tabela Única: Todas as classes são mapeadas para uma única tabela no banco de dados. Colunas adicionais são usadas para representar atributos específicos de cada classe. Isso pode resultar em muitos campos nulos.

Herança de Tabela por Subtipo: Cada classe é mapeada para sua própria tabela no banco de dados. As tabelas são conectadas por chaves estrangeiras para representar relacionamentos entre as classes. Isso reduz a redundância de colunas nulas, mas pode exigir junções mais complexas.

Em ambas as abordagens, a herança é refletida nas relações entre as tabelas e na forma como os dados são armazenados e recuperados, mantendo a integridade referencial e a consistência dos dados.

2º procedimento



```
Output - CadastroBD (run) x
run:
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
=====
5
F - Pessoa Fisica | J - Pessoa Juridica
F
Exibindo dados de Pessoa Fisica...
=====
Id: 7
Nome: Joao
Logradouro: Rua 12, casa 3, Quitanda
```

Análise e Conclusão:

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

Armazenamento: Os dados são armazenados em arquivos no sistema de arquivos do sistema operacional.

Estrutura: Pode ser simples, como texto puro, ou estruturada usando formatos como JSON, XML, etc.

Desempenho: Geralmente é mais lento para grandes conjuntos de dados, pois requer leitura e gravação sequencial.

Consulta: Acesso direto aos dados pode ser solicitado; geralmente, é necessário percorrer todo o arquivo para encontrar informações específicas.

Escalabilidade: Limitada para grandes volumes de dados e acesso simultâneo.

Persistência em Banco de Dados:

Armazenamento: Os dados são armazenados em tabelas em um sistema de gerenciamento de banco de dados (SGBD).
Estrutura: Dados são organizados em esquemas relacionais, permitindo consultas complexas e relacionamentos.
Desempenho: geralmente mais eficiente para grandes conjuntos de dados; índices e otimizações do SGBD são aplicados.
Consulta: Suporta consultas SQL, facilitando a recuperação de dados específicos.
Escalabilidade: Pode lidar com grandes volumes de dados e suportar acesso simultâneo com técnicas como transações.

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

Antes do Java 8: A impressão de valores em listas ou coleções geralmente envolve loops tradicionais.
Com o Java 8 e Operadores Lambda: A introdução de expressões lambda e o novo pacote `java.util.streams` simplificaram a iteração e impressão de valores.
O uso `forEach` das expressões lambda torna o código mais conciso e legível.

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

O método `main` em Java é o ponto de entrada para um programa. Ele precisa ser estático (`static`) porque é invocado pela JVM (Java Virtual Machine) antes mesmo de criar uma instância da classe que contém o método `main`.

Métodos estáticos pertencem à classe e não a instâncias específicas. Isso significa que eles podem ser chamados diretamente na classe, sem a necessidade de criar um objeto. O método `main` deve ser estático para permitir que uma JVM ou chame sem criar uma instância da classe.

Uma marcação como `static` indica que o método está disponível no nível da classe e pode ser invocado sem a necessidade de criar uma instância da classe.