Estácio

2º Procedimento | Alimentando a Base

Segue código da classe CadastroBDTeste.java refatorado:

```
package cadastrobd;

import cadastrobd.model.*;
import cadastrobd.model.PessoaFisicaDAO;
import cadastrobd.model.PessoaJuridicaDAO;

import java.util.ArrayList;
import java.util.InputMismatchException;
import java.util.Scanner;

public class CadastroBDTeste {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        PessoaFisicaDAO pfDAO = new PessoaFisicaDAO();
        PessoaJuridicaDAO pjDAO = new PessoaJuridicaDAO();

        int opcao = -1;
        try {
            while (opcao != 0) {
                System.out.println("\n=== MENU ===");
                System.out.println("1 - Incluir Pessoa");
                System.out.println("2 - Alterar Pessoa");
                System.out.println("3 - Excluir Pessoa");
                System.out.println("4 - Buscar pelo ID");
                System.out.println("5 - Exibir todos");
                System.out.println("0 - Finalizar Programa");
                System.out.print("Informe a opção desejada: ");
                opcao = scanner.nextInt();
                scanner.nextLine();

                switch (opcao) {
                    case 1: // Incluir
                        int tipoIncluir = lerTipoPessoa(scanner);
                        if (tipoIncluir == 1) {
                            PessoaFisica pf = lerDadosPessoaFisica(scanner, null);
                            pfDAO.incluir(pf);
                            System.out.println("Pessoa Física incluída:");
                            pf.exibir();
                        } else if (tipoIncluir == 2) {
                            PessoaJuridica pj = lerDadosPessoaJuridica(scanner, null);
                            pjDAO.incluir(pj);
                            System.out.println("Pessoa Jurídica incluída:");
```

```
            pj.exibir();
        }
        break;

    case 2: // Alterar
        int tipoAlterar = lerTipoPessoa(scanner);
        System.out.print("Informe o ID para alterar: ");
        int idAlt = scanner.nextInt();
        scanner.nextLine();

        if (tipoAlterar == 1) {
            PessoaFisica pfAlt = (PessoaFisica) pfDAO.getPessoa(idAlt);
            if (pfAlt == null) {
                System.out.println("Pessoa Física não encontrada.");
                break;
            }
            System.out.println("Dados atuais:");
            pfAlt.exibir();
            PessoaFisica pfNovos = lerDadosPessoaFisica(scanner, pfAlt);
            pfDAO.alterar(pfNovos);
            System.out.println("Pessoa Física alterada:");
            pfNovos.exibir();
        } else if (tipoAlterar == 2) {
            PessoaJuridica pjAlt = (PessoaJuridica) pjDAO.getPessoa(idAlt);
            if (pjAlt == null) {
                System.out.println("Pessoa Jurídica não encontrada.");
                break;
            }
            System.out.println("Dados atuais:");
            pjAlt.exibir();
            PessoaJuridica pjNovos = lerDadosPessoaJuridica(scanner, pjAlt);
            pjDAO.alterar(pjNovos);
            System.out.println("Pessoa Jurídica alterada:");
            pjNovos.exibir();
        }
        break;

    case 3: // Excluir
        int tipoExcluir = lerTipoPessoa(scanner);
        System.out.print("Informe o ID para excluir: ");
        int idExc = scanner.nextInt();
        scanner.nextLine();

        if (tipoExcluir == 1) {
            pfDAO.excluir(idExc);
            System.out.println("Pessoa Física excluída: ID " + idExc);
        } else if (tipoExcluir == 2) {
            pjDAO.excluir(idExc);
            System.out.println("Pessoa Jurídica excluída: ID " + idExc);
        }
        break;
```

```
case 4: // Exibir pelo ID
    int tipoExibir = lerTipoPessoa(scanner);
    System.out.print("Informe o ID para exibir: ");
    int idEx = scanner.nextInt();
    scanner.nextLine();

    if (tipoExibir == 1) {
        PessoaFisica pfEx = (PessoaFisica) pfDAO.getPessoa(idEx);
        if (pfEx != null) {
            pfEx.exibir();
        } else {
            System.out.println("Pessoa Física não encontrada.");
        }
    } else if (tipoExibir == 2) {
        PessoaJuridica pjEx = (PessoaJuridica) pjDAO.getPessoa(idEx);
        if (pjEx != null) {
            pjEx.exibir();
        } else {
            System.out.println("Pessoa Jurídica não encontrada.");
        }
    }
    break;

case 5: // Exibir todos
    int tipoExibirTodos = lerTipoPessoa(scanner);

    if (tipoExibirTodos == 1) {
        ArrayList<Pessoa> listaPF = pfDAO.getPessoas();
        if (listaPF.isEmpty()) {
            System.out.println("Nenhuma Pessoa Física cadastrada.");
        } else {
            for (Pessoa p : listaPF) {
                ((PessoaFisica) p).exibir();
            }
        }
    } else if (tipoExibirTodos == 2) {
        ArrayList<Pessoa> listaPJ = pjDAO.getPessoas();
        if (listaPJ.isEmpty()) {
            System.out.println("Nenhuma Pessoa Jurídica cadastrada.");
        } else {
            for (Pessoa p : listaPJ) {
                ((PessoaJuridica) p).exibir();
            }
        }
    }
    break;

case 0:
    System.out.println("Finalizando o programa.");
    break;

default:
```

```
                System.out.println("Opção inválida. Tente novamente.");
                break;
            }
        }
    } catch (InputMismatchException e) {
        System.out.println("Entrada inválida, por favor informe um número.");
    } catch (Exception e) {
        System.out.println("Erro inesperado: " + e.getMessage());
    } finally {
        scanner.close();
    }
}

private static int lerTipoPessoa(Scanner scanner) {
    int tipo = -1;
    while (tipo != 1 && tipo != 2) {
        System.out.print("Tipo de pessoa (1 - Física, 2 - Jurídica): ");
        tipo = scanner.nextInt();
        scanner.nextLine();
        if (tipo != 1 && tipo != 2) {
            System.out.println("Tipo inválido, escolha 1 ou 2.");
        }
    }
    return tipo;
}

private static PessoaFisica lerDadosPessoaFisica(Scanner scanner, PessoaFisica pfExistente) {
    PessoaFisica pf = (pfExistente != null) ? pfExistente : new PessoaFisica();

    System.out.print("Nome: ");
    pf.setNome(scanner.nextLine());

    System.out.print("Endereço: ");
    pf.setEndereco(scanner.nextLine());

    System.out.print("Telefone: ");
    pf.setTelefone(scanner.nextLine());

    System.out.print("Email: ");
    pf.setEmail(scanner.nextLine());

    System.out.print("CPF: ");
    pf.setCpf(scanner.nextLine());

    return pf;
}

private static PessoaJuridica lerDadosPessoaJuridica(Scanner scanner, PessoaJuridica
pjExistente) {
    PessoaJuridica pj = (pjExistente != null) ? pjExistente : new PessoaJuridica();

    System.out.print("Nome: ");
```

```
        pj.setNome(scanner.nextLine());

        System.out.print("Endereço: ");
        pj.setEndereco(scanner.nextLine());

        System.out.print("Telefone: ");
        pj.setTelefone(scanner.nextLine());

        System.out.print("Email: ");
        pj.setEmail(scanner.nextLine());

        System.out.print("CNPJ: ");
        pj.setCnpj(scanner.nextLine());

        return pj;
    }
}
```

Segue resultado da execução do código:

```
run:

=== MENU ===
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir todos
0 - Finalizar Programa
Informe a opção desejada: 4
Tipo de pessoa (1 - Física, 2 - Jurídica): 1
Informe o ID para exibir: 1
ID: 1
Nome: Maria Silva
Endereço: Rua A, 123
Telefone: 11999999999
Email: maria@email.com
Tipo: F
CPF: 123.456.789-00

=== MENU ===
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir todos
0 - Finalizar Programa
Informe a opção desejada: 0
Finalizando o programa.
BUILD SUCCESSFUL (total time: 5 minutes 12 seconds)
```

Análise e Conclusão

1. Diferenças entre persistência em arquivo e persistência em banco de dados?

Aspecto	Persistência em Arquivo	Persistência em Banco de Dados
Estrutura e organização	Dados geralmente armazenados em arquivos texto, CSV, binários, XML, JSON, etc.	Dados organizados em tabelas relacionais, com suporte a consultas complexas.
Consulta e manipulação	Consulta e atualização mais simples ou limitada, geralmente sequencial.	Consultas complexas com SQL, índices, relacionamentos, transações.
Desempenho	Pode ser lento para grandes volumes, sem suporte a índices.	Otimizado para grandes volumes e alta concorrência.
Concorrência	Geralmente não suporta múltiplos acessos concorrentes de forma eficiente.	Suporta múltiplas conexões simultâneas, com controle de concorrência e transações.
Integridade e segurança	Difícil garantir integridade e segurança (backup, recuperação, bloqueios).	Recursos avançados para integridade referencial, segurança e recuperação.
Escalabilidade	Limitada, difícil de gerenciar conforme o volume cresce.	Alta escalabilidade, com replicação e particionamento.
Ferramentas e suporte	Simples, poucas ferramentas específicas.	Amplo suporte, ferramentas para gerenciamento, BI, etc.

2. Como o uso de lambda simplificou a impressão em Java?
O lambda permite escrever código mais curto e legível para iterar coleções, substituindo loops tradicionais por chamadas simples como lista.forEach(p -> p.exibir()) ou lista.forEach(Pessoa::exibir).

3. Por que métodos chamados diretamente pelo main precisam ser static?
Porque o método main é estático e não cria um objeto da classe. Métodos estáticos podem ser chamados sem instanciar a classe, enquanto métodos não estáticos precisam de um objeto para serem usados.