Trabalho Prático | DGT2827 PROGRAMAÇÃO BACK-END COM JAVA

Felipe Eduardo Oliveira de Melo - 202411164421

Polo Votuporanga

Desenvolvimento Full Stack – 9001 3AA – 2025.4

Objetivo da Prática

Implementação de um cadastro de clientes em modo texto, com persistência em arquivos, baseado na tecnologia Java.

1º Procedimento | Criação das Entidades e Sistema de Persistência

Entidades:

```
package model;

import java.io.Serializable;

public class Pessoa implements Serializable {

 public int id;

public String nome;

public Pessoa(int id, String nome) {
    this.id=id;
    this.nome=nome;
}
public void exibir() {
  System.out.println("id: " + this.id);
  System.out.println("Nome: " + this.nome);
}
/... getters e setters

package model;

public class PessoaFisica extends Pessoa {
```

```
public String cpf;

public int idade;

public PessoaFisica(int id, String nome, String cpf, int idade) {
  super(id, nome);
  this.cpf = cpf;
  this.idade = idade;
}
@Override
public void exibir() {
  super.exibir();
  System.out.println("CPF: " + this.cpf);
  System.out.println("Idade: " + this.idade);
}
/... getters e setters



package model;

public class PessoaJuridica extends Pessoa {

public String cnpj;

public PessoaJuridica(int id, String nome, String cnpj) {
  super(id, nome);
  this.cnpj = cnpj;
}



@Override
public void exibir() {
  super.exibir();
  System.out.println("cnpj: " + this.cnpj);
}
/... getters e setters

Repositorios:

package model;

import java.io.File; import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.ObjectInputStream;
```

```
import java.io.ObjectOutputStream;

import java.util.ArrayList;

public class PessoaFisicaRepo { private ArrayList listPessoaFisica =
new ArrayList<>();public void inserir(PessoaFisica pessoaFisica) {
  listPessoaFisica.add(pessoaFisica);
}

public void alterar(PessoaFisica pessoaFisica) {
  for (int i = 0; i < listPessoaFisica.size(); i++) {
     PessoaFisica p = listPessoaFisica.get(i);

     if (p.getId() == pessoaFisica.getId()) {
        listPessoaFisica.set(i, pessoaFisica);
        return;
     }
  }
}

public void excluir(int id) {
  for (int i = 0; i < listPessoaFisica.size(); i++) {
     PessoaFisica p = listPessoaFisica.get(i);

     if(p.getId() == id) {
        listPessoaFisica.remove(i);
        break;
     }
  }
}
public PessoaFisica obter(int id) {
  for (PessoaFisica p : listPessoaFisica){
     if(p.getId() == id) {
        return p;
     }
  }
  return null;
}
public ArrayList<PessoaFisica> obterTodos() {
  return listPessoaFisica;
}

public void persistir(String nomeArquivo) throws IOException {
  try (FileOutputStream fileOut = new FileOutputStream(nomeArquivo);
       ObjectOutputStream objectOut = new ObjectOutputStream(fileOut)) {

     objectOut.writeObject(listPessoaFisica);
     System.out.println("Dados de Pessoa Fisica armazenados.");
  }
```

```
}

public void recuperar(String nomeArquivo) throws IOException, ClassNotFound
Exception {
  File arquivo = new File(nomeArquivo);

  if(!arquivo.exists()) {
     System.out.println("Arquivo não encontrado. Iniciando com lista vazia.");
     return;
  }

  try (FileInputStream fileIn = new FileInputStream(nomeArquivo);
        ObjectInputStream objectIn = new ObjectInputStream(fileIn)) {
     this.listPessoaFisica = (ArrayList<PessoaFisica>) objectIn.readObject();
     System.out.println("Dados de Pessoa Fisica recuperados.");
  }
}


}

package model;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.util.ArrayList;

public class PessoaJuridicaRepo { private ArrayList listPessoaJuridica =
new ArrayList<>();public voidinserir(PessoaJuridica pessoaJuridica) {
  listPessoaJuridica.add(pessoaJuridica);
}

public void alterar(PessoaJuridica pessoaJuridica) {
  for (int i = 0; i < listPessoaJuridica.size(); i++) {
     PessoaJuridica p = listPessoaJuridica.get(i);

     if (p.getId() == pessoaJuridica.getId()) {
        listPessoaJuridica.set(i, pessoaJuridica);
        return;
     }
  }
```

```
}

public void excluir(int id) {
  for (int i = 0; i < listPessoaJuridica.size(); i++) {
     PessoaJuridica p = listPessoaJuridica.get(i);

     if (p.getId() == id){
        listPessoaJuridica.remove(i);
        break;
     }
  }
}

public PessoaJuridica obter(int id) {
  for (PessoaJuridica p : listPessoaJuridica) {
     if (p.getId() == id){
        return p;
     }
  }
  return null;
}
public ArrayList<PessoaJuridica> obterTodos() {
  return listPessoaJuridica;
}

public void persistir(String nomeArquivo) throws IOException {
  try (FileOutputStream fileOut = new FileOutputStream(nomeArquivo);
        ObjectOutputStream objectOut = new ObjectOutputStream(fileOut)) {
     objectOut.writeObject(listPessoaJuridica);
     System.out.println("Dados de Pessoa Juridica armazenados.");
  }
}

public void recuperar(String nomeArquivo) throws IOException, ClassNotFound
Exception {
  File arquivo = new File(nomeArquivo);

  if (!arquivo.exists()) {
     System.out.println("Arquivo não encontrado. Iniciando com lista vazia.");
     return;
  }

  try (FileInputStream fileIn = new FileInputStream(nomeArquivo);
        ObjectInputStream objectIn = new ObjectInputStream(fileIn)) {
     this.listPessoaJuridica = (ArrayList<PessoaJuridica>) objectIn.readObject();
     System.out.println("Dados de Pessoa Juridica recuperados.");
  }
```

```
}


}

package cadastropoo;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Scanner;
import model.PessoaFisica;
import model.PessoaFisicaRepo;
import model.PessoaJuridica;
import model.PessoaJuridicaRepo;

/**
 *
 * @author Felipe
 */
public class CadastroPOO {

   /**
    * @param args the command line arguments
    */
   public static void main(String[] args) {
      PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
      repo1.inserir(new PessoaFisica(1, "Ana", "11111111111", 25));
      repo1.inserir(new PessoaFisica(2, "Carlos", "22222222222", 52));
      String nomeFixoArquivoFisica = "PessoasFisica.bin";
      try {
         repo1.persistir(nomeFixoArquivoFisica);
      } catch (IOException e) {
         System.out.println("Erro grave: Não consegui salvar no disco!");
         e.printStackTrace();
      }

      PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
      try {
         repo2.recuperar(nomeFixoArquivoFisica);
      } catch (IOException | ClassNotFoundException e) {
         System.out.println("Erro ao carregar dados: " + e.getMessage());
      }
      //Exibir dados recuperados
      ArrayList<PessoaFisica> listPessoaFisica = repo2.obterTodos();
      for (PessoaFisica pessoa : listPessoaFisica) {
         pessoa.exibir();
      }

      PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
```

```
        repo3.inserir(new PessoaJuridica(3, "XPTO Sales", "33333333333333"));
        repo3.inserir(new PessoaJuridica(4, "XPTO Solutions",
"44444444444444"));
        String nomeFixoArquivoJuridica = "PessoasJuridica.bin";
        try {
           repo3.persistir(nomeFixoArquivoJuridica);
        } catch (IOException e) {
           System.out.println("Erro grave: Não consegui salvar no disco!");
           e.printStackTrace();
        }

        PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
        try {
           repo4.recuperar(nomeFixoArquivoJuridica);
        }  catch (IOException | ClassNotFoundException e) {
           System.out.println("Erro ao carregar dados: " + e.getMessage());
        }
        //Exibir dados recuperados
        ArrayList<PessoaJuridica> listPessoaJuridica = repo4.obterTodos();
        for (PessoaJuridica pessoa : listPessoaJuridica) {
           pessoa.exibir();
        }
```

a. *Quais as vantagens e desvantagens do uso de herança?*
   As vantagens são a organização do código, facilidade de manutenção e permite a reusabilidade do código.

b. *Por que a interface Serializable é necessária ao efetuar persistência em arquivos binários?*
   Porque ela pode converte a classe em sequência de bytes.

c. *Como o paradigma funcional é utilizado pela API stream no Java?*
   Combinado com as expressões lambda, que facilita a manutenção do código.

d. *Quando trabalhamos com Java, qual padrão de desenvolvimento é adotado na persistência de dados em arquivos?*
   DAO – Data Acess Object

2º Procedimento | Criação do Cadastro em Modo Texto

```
package cadastropoo;

import java.io.IOException;
```

```
import java.util.ArrayList;
import java.util.Scanner;
import model.PessoaFisica;
import model.PessoaFisicaRepo;
import model.PessoaJuridica;
import model.PessoaJuridicaRepo;

/**
 *
 * @author Felipe
 */
public class CadastroPOO {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        PessoaFisicaRepo pessoaFisicaRepo = new PessoaFisicaRepo();
        PessoaJuridicaRepo pessoaJuridicaRepo = new PessoaJuridicaRepo();
        Scanner scanner = new Scanner(System.in);
        int opcao = 0;

        do {
            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("6 - Registrar dados");
            System.out.println("7 - Recuper dados");
            System.out.println("0 - Finalizar programa");

            opcao = scanner.nextInt();

            switch (opcao) {
                case 1:
                    System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
                    char opcaoCase1 = scanner.next().toUpperCase().charAt(0);

                    switch (opcaoCase1) {
                        case 'F':
                            System.out.println("Digite o id da pessoa");
                            int idFisica = scanner.nextInt();
                            System.out.println("Digite o nome:");
                            scanner.nextLine();
                            String nomeFisica = scanner.nextLine();
                            System.out.println("Digite o CPF:");
                            String cpfFisica = scanner.nextLine();
                            System.out.println("Digite a idade:");
```

```
                    int idadeFisica = scanner.nextInt();
                    pessoaFisicaRepo.inserir(new PessoaFisica(idFisica,
nomeFisica, cpfFisica, idadeFisica));
                    break;
                case 'J':
                    System.out.println("Digite o id da pessoa");
                    int idJuridica = scanner.nextInt();
                    System.out.println("Digite o nome:");
                    scanner.nextLine();
                    String nomeJuridica = scanner.nextLine();
                    System.out.println("Digite o CNPJ:");
                    String cnpjJuridica = scanner.nextLine();
                    pessoaJuridicaRepo.inserir(new PessoaJuridica(idJuridica,
nomeJuridica, cnpjJuridica));
                    break;
                default:
                    System.out.println("Opção inválida! Digite F ou J.");
                    break;
            }
            break;
        case 2:
            System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
            char opcaoCase2 = scanner.next().toUpperCase().charAt(0);

            switch (opcaoCase2) {
                case 'F':
                    System.out.println("Informe o id da pessoa que deseja
alterar");
                    int idObterFisica = scanner.nextInt();
                    PessoaFisica pessoaFisica =
pessoaFisicaRepo.obter(idObterFisica);
                    pessoaFisica.exibir();
                    System.out.println("Digite o id da pessoa");
                    int idFisica = scanner.nextInt();
                    System.out.println("Digite o nome:");
                    scanner.nextLine();
                    String nomeFisica = scanner.nextLine();
                    System.out.println("Digite o CPF:");
                    String cpfFisica = scanner.nextLine();
                    System.out.println("Digite a idade:");
                    int idadeFisica = scanner.nextInt();
                    pessoaFisicaRepo.alterar(new PessoaFisica(idFisica,
nomeFisica, cpfFisica, idadeFisica));
                    break;
                case 'J':
                    System.out.println("Informe o id da pessoa que deseja
alterar");
                    int idObterJuridica = scanner.nextInt();
                    PessoaJuridica pessoaJuridica =
```

```
pessoaJuridicaRepo.obter(idObterJuridica);
                    pessoaJuridica.exibir();
                    System.out.println("Digite o id da pessoa");
                    int idJuridica = scanner.nextInt();
                    System.out.println("Digite o nome:");
                    scanner.nextLine();
                    String nomeJuridica = scanner.nextLine();
                    System.out.println("Digite o CNPJ:");
                    String cnpjJuridica = scanner.nextLine();
                    pessoaJuridicaRepo.alterar(new PessoaJuridica(idJuridica,
nomeJuridica, cnpjJuridica));
                    break;
                default:
                    System.out.println("Opção inválida! Digite F ou J.");
                    break;
            }
            break;
        case 3:
            System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
            char opcaoCase3 = scanner.next().toUpperCase().charAt(0);

            switch (opcaoCase3) {
                case 'F':
                    System.out.println("Informe o id da pessoa que deseja
excluir");
                    int idExcluirFisica = scanner.nextInt();
                    pessoaFisicaRepo.excluir(idExcluirFisica);
                    System.out.println("Sucesso");
                    break;
                case 'J':
                    System.out.println("Informe o id da pessoa que deseja
excluir");
                    int idExcluirJuridica = scanner.nextInt();
                    pessoaJuridicaRepo.excluir(idExcluirJuridica);
                    System.out.println("Sucesso");
                    break;
                default:
                    System.out.println("Opção inválida! Digite F ou J.");
                    break;
            }
            break;
        case 4:
            System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
            char opcaoCase4 = scanner.next().toUpperCase().charAt(0);

            switch (opcaoCase4) {
                case 'F':
                    System.out.println("Informe o id da pessoa que deseja obter
os dados");
```

```
                    int idObterPFisica = scanner.nextInt();
                    PessoaFisica pf = pessoaFisicaRepo.obter(idObterPFisica);
                    pf.exibir();
                    break;
                case 'J':
                    System.out.println("Informe o id da pessoa que deseja obter
os dados");
                    int idObterPJuridica = scanner.nextInt();
                    PessoaJuridica pj =
pessoaJuridicaRepo.obter(idObterPJuridica);
                    pj.exibir();
                    break;
                default:
                    System.out.println("Opção inválida! Digite F ou J.");
                    break;
            }
            break;
        case 5:
            System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
            char opcaoCase5 = scanner.next().toUpperCase().charAt(0);

            switch (opcaoCase5) {
                case 'F':
                    ArrayList<PessoaFisica> pessoasFisica =
pessoaFisicaRepo.obterTodos();
                    for (PessoaFisica pessoa : pessoasFisica) {
                        pessoa.exibir();
                    }
                    break;
                case 'J':
                    ArrayList<PessoaJuridica> pessoasJuridica =
pessoaJuridicaRepo.obterTodos();
                    for (PessoaJuridica pessoa : pessoasJuridica) {
                        pessoa.exibir();
                    }
                    break;
                default:
                    System.out.println("Opção inválida! Digite F ou J.");
                    break;
            }
            break;
        case 6:
            scanner.nextLine();
            System.out.println("Digite o prefixo dos arquivos");
            String caminhoArquivoPersistir = scanner.nextLine();
            try {
                pessoaFisicaRepo.persistir(caminhoArquivoPersistir +
"fisica.bin");
                pessoaJuridicaRepo.persistir(caminhoArquivoPersistir +
```

```
"juridica.bin");
                } catch (IOException e) {
                    System.out.println("Erro grave: Não consegui salvar no
disco!");
                    e.printStackTrace();
                }
                break;
            case 7:
                scanner.nextLine();
                System.out.println("Digite o prefixo dos arquivos");
                String caminhoArquivoRecuperar = scanner.nextLine();
                try {
                    pessoaFisicaRepo.recuperar(caminhoArquivoRecuperar +
"fisica.bin");
                    pessoaJuridicaRepo.recuperar(caminhoArquivoRecuperar +
"juridica.bin");
                } catch (IOException | ClassNotFoundException e) {
                    System.out.println("Erro ao carregar dados: " +
e.getMessage());
                }
                break;
            case 0:
                System.out.println("Finalizando o sistema...");
                break;
            default:
                System.out.println("\nOpção inválida!");
                System.out.println("Por favor, digite um número entre 0 e 7.");
                break;
        }
    } while (opcao != 0);

  }

}
```

a. *que são elementos estáticos e qual o motivo para o método main adotar esse modificador?*
   elementos estáticos pertence a classe em si, e não ao objeto.

b. *Para que serve a classe Scanner?*
   Para receber dados do teclado por linha de comando.

c. *Como o uso de classes de repositório impactou na organização do código?*
   Podendo centralizar a parte de persistência separa da regra de negócio.

Conclusão

O projeto desenvolvido teve como objetivo consolidar os fundamentos da programação orientada a objetos em Java, focando em utilizar herança, polimorfismo e persistência de dados através da serialização de objetos. Foi utilizado o padrão de desenvolvimento MVC.