



Universidade Estácio de Sá

- DESENVOLVIMENTO FULL STACK

- Disciplina: RPG0014 - Iniciando o caminho pelo Java

- Semestre Letivo: 2024.3

- Repositorio Git:

<https://github.com/DarciFalcao/Estaciomissao1mundo3.git>

DARCI RODRIGUES FALCÃO NETO

Missão Prática | Nível 1 | Mundo 3

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

Procedimento 1: Criação das Entidades e Sistema de Persistência

Procedimento 2: Criação do Cadastro em Modo Texto

Objetivos da Prática

- Objetivos da Prática
- Aplicar herança e polimorfismo na criação de entidades.
- Implementar a persistência de objetos em arquivos binários.
- Desenvolver uma interface de cadastro em modo texto.
- Utilizar o controle de exceções na plataforma Java.
- Ao final do projeto, o aluno terá criado um sistema de cadastro em Java, explorando os conceitos de programação orientada a objetos e a persistência de dados em arquivos binários.

Entidades:

- Classe Pessoa

```
package model;

/**
 *
 * @author grego
 */

import java.io.Serializable;

public class Pessoa implements Serializable{
    private int id;
    private String nome;

    // construtor
    public Pessoa(int id, String nome) {
        this.id = id;
        this.nome = nome;
    }

    public int getId() {
        return id;
    }

    public void setId(int id){
        this.id = id;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    //Método exibir
    public void exibir(){
        System.out.print("id: "+this.id + "\n" + "Nome: " + this.nome + "\n");
    }
}
```

```
}
```

• Classe PessoaFisica

```
package model;

/**
 *
 * @author grego
 */

import java.io.Serializable;

public class PessoaFisica extends Pessoa implements Serializable {
    private String cpf;
    private int idade;

    //Constutor
    public PessoaFisica(int id, String nome, String cpf, int idade){
        super(id, nome);
        this.cpf = cpf;
        this.idade = idade;
    }

    public String getCpf() {
        return cpf;
    }

    public void setCpf(String cpf) {
        this.cpf = cpf;
    }

    public int getIdade(){
        return idade;
    }

    public void setIdade(int idade){
        this.idade = idade;
    }

    //Método exibir
    public void exibir(){
        System.out.print("\n"+ "id: "+getId()+"\nNome: "+getNome()+ " \nCPF: "
            +this.cpf + "\n" + "Idade: " + this.idade + "\n");
    }
}
```

• Classe PessoaJuridica

```
package model;

/**
 *
 * @author grego
 */

import java.io.Serializable;
```

```

public class PessoaJuridica extends Pessoa implements Serializable{
    private String cnpj;

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

    public String getCnpj() {
        return cnpj;
    }

    public void setCnpj(String cnpj) {
        this.cnpj = cnpj;
    }

    public void exibir(){
        System.out.print("\n" + "id: "+getId()+ "\nNome: "+getNome()+"\nCNPJ: "+this.cnpj+"\n");
    }
}

```

Gerenciadores:

- Classe PessoaFisicaRepo

```

package model;
import java.util.ArrayList;
import java.util.NoSuchElementException;
import java.util.Optional;
import java.io.*;

/**
 *
 * @author grego
 */
public class PessoaFisicaRepo {

    private ArrayList<PessoaFisica> listaPessoasFisicas = new ArrayList<>();

    public void inserir(PessoaFisica pessoaFisica){
        listaPessoasFisicas.add(pessoaFisica);
    }

    public void alterar(PessoaFisica pessoaFisica, String novoNome, String novoCpf, int novaIdade) {
        pessoaFisica.setNome(novoNome);
        pessoaFisica.setCpf(novoCpf);
        pessoaFisica.setIdade(novaIdade);
    }

    public void excluir(int id) {
        try{
            listaPessoasFisicas.remove(obter(id));
        }
    }
}

```

```

    }catch(NoSuchElementException e){
        System.out.println("erro");
    }
}

    public PessoaFisica obter(int id) {
        Optional<PessoaFisica> pessoaFisicaLocalizada =
        listaPessoasFisicas.stream().
        filter(pessoaFisica -> pessoaFisica.getId() ==
        id).findFirst();
        if (pessoaFisicaLocalizada.isPresent()) {
            return pessoaFisicaLocalizada.get();
        } else {
            return null;
        }
    }

    public ArrayList<PessoaFisica> obterTodos(){
        return listaPessoasFisicas;
    }

    public void persistir(String arquivo)throws IOException {
        ObjectOutputStream arquivoSaida = new ObjectOutputStream(new
        FileOutputStream(arquivo));
        arquivoSaida.writeObject(listaPessoasFisicas);
        arquivoSaida.close();
        System.out.println("Dados de pessoas fisicas armazenados."); }

    public void recuperar(String arquivo) throws IOException,
    ClassNotFoundException {
        ObjectInputStream arquivoEntrada = new ObjectInputStream(new
        FileInputStream(arquivo));
        listaPessoasFisicas = (ArrayList<PessoaFisica>)
        arquivoEntrada.readObject();
        arquivoEntrada.close();
        System.out.println("Dados de pessoas fisicas recuperados."); }
}

```

• Classe PessoaJuridicaRepo

```

package model;
import java.util.ArrayList;
import java.util.Optional;
import java.io.*;

/**
 *
 * @author grego
 */
public class PessoaJuridicaRepo {

    private ArrayList<PessoaJuridica> listaPessoasJuridicas = new
    ArrayList<>();

    public void inserir(PessoaJuridica pessoaJuridica){
        listaPessoasJuridicas.add(pessoaJuridica);
    }
}

```

```

    }

    public void alterar(PessoaJuridica pessoaJuridica, String novoNome,
String novoCnpj) {
        pessoaJuridica.setNome(novoNome);
        pessoaJuridica.setCnpj(novoCnpj);
    }

    public void excluir(int id){
        listaPessoasJuridicas.remove(obter(id));
    }

    public PessoaJuridica obter(int id) {
        Optional<PessoaJuridica> pessoaJuridicaLocalizada =
        listaPessoasJuridicas.stream().
        filter(pessoaJuridica -> pessoaJuridica.getId() ==
id).findFirst();
        if (pessoaJuridicaLocalizada.isPresent()) {
            return pessoaJuridicaLocalizada.get();
        } else {
            return null;
        }
    }

    public ArrayList<PessoaJuridica> obterTodos(){
        return listaPessoasJuridicas;
    }

    public void persistir(String arquivo)throws IOException {
        ObjectOutputStream arquivoSaida = new ObjectOutputStream(new
        FileOutputStream(arquivo));
        arquivoSaida.writeObject(listaPessoasJuridicas);
        arquivoSaida.close();
        System.out.println("\nDados das pessoas juridicas armazenados.");    }

    public void recuperar(String arquivo)throws IOException,
        ClassNotFoundException {
        ObjectInputStream arquivoEntrada = new ObjectInputStream(new
        FileInputStream(arquivo));
        listaPessoasJuridicas = (ArrayList<PessoaJuridica>)
        arquivoEntrada.readObject();
        arquivoEntrada.close();
        System.out.println("Dados de pessoas juridicas recuperados.");    }
}

```

Aplicação:

- Classe CadastroPOO - Referente ao Procedimento 1

```

package model;
import java.io.IOException;

/**
 *
 * @author grego
 */

```

```

public class CadastroPOO {
    public static void main(String[] args) {

        PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
        PessoaFisica pessoaFisica1 = new PessoaFisica(1, "Ana",
"11111111111", 25);
        PessoaFisica pessoaFisica2 = new PessoaFisica(2, "Carlos Jose",
"22222222222", 52);
        repo1.inserir(pessoaFisica1);
        repo1.inserir(pessoaFisica2);
        try {
            repo1.persistir("listaPessoasFisicas.bin");
        } catch (IOException erro) {
            System.out.println("Erro ao persistir os dados: " +
erro.getMessage());
        }

        PessoaFisicaRepo repo2 = new PessoaFisicaRepo();

        try {
            repo2.recuperar("listaPessoasFisicas.bin");
            repo2.obterTodos()
                .forEach(pessoaFisica -> {
                    pessoaFisica.exibir();
                });

        } catch (IOException | ClassNotFoundException erro) {
            System.out.println("Erro ao recuperar os dados: " +
erro.getMessage());
        }

        PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
        PessoaJuridica pessoaJuridica1 = new PessoaJuridica(3, "XPTO Sales", "33333333333333");
        PessoaJuridica pessoaJuridica2 = new PessoaJuridica(4, "XPTO
Solutions", "4444444444444444");
        repo3.inserir(pessoaJuridica1);
        repo3.inserir(pessoaJuridica2);
        try {
            repo3.persistir("listaPessoasJuridicas.bin");
        } catch (IOException erro) {
            System.out.println("Erro ao persistir os dados: " +
erro.getMessage());
        }

        PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
        try {
            repo4.recuperar("listaPessoasJuridicas.bin");
            repo4.obterTodos()
                .forEach(pessoaJuridica -> {
                    pessoaJuridica.exibir();
                });
        } catch (IOException | ClassNotFoundException erro) {
            System.out.println("Erro ao recuperar os dados: " +
erro.getMessage());
        }
    }
}

```


• Classe CadastroPOO2 - Referente ao Procedimento 2

```
package model;
import java.io.IOException;
import java.util.Scanner;

/**
 *
 * @author grego
 */
public class CadastroPOO2 {

    public static void main(String[] args) {

        PessoaFisicaRepo pfRepo = new PessoaFisicaRepo();
        PessoaJuridicaRepo pjRepo = new PessoaJuridicaRepo();

        Scanner scan = new Scanner(System.in);
        String escolha;

        do {
            System.out.println("=====");
            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 - Persistir/Salvar Dados");
            System.out.println("7 - Recuperar/Carregar Dados");
            System.out.println("0 - Finalizar Programa");
            System.out.println("====="); escolha =

scan.next();

        switch (escolha) {

            // Incluir
            case "1":
                do {
                    System.out.println("====="); System.out.println("F
- Pessoa Fisica | J - Pessoa Juridica | M - Menu");

                    escolha = scan.next();
                    scan.nextLine();

                    switch (escolha.toUpperCase()) {

                        case "F":
                            System.out.print("Digite o id da pessoa: "); int idInformado =
scan.nextInt(); System.out.println("Insira os dados... ");
scan.nextLine();
                            System.out.print("Nome: "); String nome = scan.nextLine();
                            System.out.print("CPF: "); String cpf = scan.nextLine();
                            System.out.print("Idade: "); int idade = scan.nextInt();
                                int pfRepoSize = pfRepo.obterTodos().size();
```

```

        PessoaFisica pessoaFisica = new
PessoaFisica(idInformado, nome, cpf, idade);
pfRepo.inserir(pessoaFisica);

System.out.println("Inclusao realizada com sucesso!");
pessoaFisica.exibir();
break;

case "J":
System.out.print("Digite o id da pessoa: "); int idInformado2 =
scan.nextInt(); scan.nextLine();
System.out.print("Nome: "); nome = scan.nextLine();
System.out.print("CNPJ: "); String cnpj = scan.nextLine();

int pjRepoSize = pjRepo.obterTodos().size();
PessoaJuridica pessoaJuridica = new PessoaJuridica(idInformado2,
nome, cnpj);
pjRepo.inserir(pessoaJuridica);

System.out.println("Inclusao realizada com sucesso!");
pessoaJuridica.exibir();
break;

case "M":
break;

default:
System.out.println("Opcao invalida."); break;
}
} while (!escolha.equalsIgnoreCase("M")); break;

// Alterar
case "2":
do {
System.out.println("====="); System.out.println("F
- Pessoa Fisica | J - Pessoa Juridica | M - Menu");
escolha = scan.next();
scan.nextLine();

switch (escolha.toUpperCase()) {

case "F":
System.out.println("Digite o ID da pessoa: ");
int idPessoaFisica = scan.nextInt(); scan.nextLine();

PessoaFisica pessoaFisicaLocalizada =
pfRepo.obter(idPessoaFisica);

if (pessoaFisicaLocalizada != null) {
pessoaFisicaLocalizada.exibir();

System.out.println("Nome atual: " + pessoaFisicaLocalizada.getNome());
System.out.print("Novo nome: "); String novoNome = scan.nextLine();

System.out.println("CPF atual: " + pessoaFisicaLocalizada.getCpf());
System.out.print("Novo CPF: "); String novoCPF = scan.nextLine();

```

```

System.out.println("Idade atual: " + pessoaFisicaLocalizada.getCpf());
System.out.print("Nova Idade: "); int novaIdade = scan.nextInt();

pfRepo.alterar(pessoaFisicaLocalizada, novoNome, novoCPF, novaIdade);

System.out.println("Pessoa alterada com sucesso!");
} else
System.out.println("Pessoa nao localizada! ");
break;

case "J":
System.out.println("Digite o ID da pessoa: ");
int idPessoaJuridica = scan.nextInt(); scan.nextLine();

PessoaJuridica pessoaJuridicaLocalizada =
pjRepo.obter(idPessoaJuridica);

if (pessoaJuridicaLocalizada != null) {
pessoaJuridicaLocalizada.exibir();

System.out.println("Nome atual: " +
pessoaJuridicaLocalizada.getNome());
System.out.println("Novo nome: "); String novoNome =
scan.nextLine();
System.out.println("CNPJ atual: " +
pessoaJuridicaLocalizada.getCnpj());
System.out.println("Novo CNPJ: "); String novoCNPJ =
scan.nextLine();

pjRepo.alterar(pessoaJuridicaLocalizada, novoNome, novoCNPJ);

System.out.println("Pessoa alterada com sucesso!");
} else
System.out.println("Pessoa nao localizada!");
break;

case "M":
break;

default:
System.out.println("Opcao invalida."); break;
}
} while (!escolha.equalsIgnoreCase("M")); break;

// EXCLUIR
case "3":
do {
System.out.println("====="); System.out.println("F
- Pessoa Fisica | J - Pessoa Juridica | M - Menu");

escolha = scan.next();
scan.nextLine();

switch (escolha.toUpperCase()) {

case "F":
System.out.println("Digite o ID da pessoa: ");

```

```

int idPessoaFisica = scan.nextInt();

PessoaFisica pessoaFisicaLocalizada = pfRepo.obter(idPessoaFisica);

if (pessoaFisicaLocalizada != null) {
pessoaFisicaLocalizada.exibir(); pfRepo.excluir(idPessoaFisica);

System.out.println("Pessoa excluida com sucesso!");
} else
System.out.println("Pessoa nao localizada!");
break;

case "J":
System.out.println("Digite o ID da pessoa: ");
int idPessoaJuridica = scan.nextInt();

PessoaJuridica pessoaJuridicaLocalizada =
pjRepo.obter(idPessoaJuridica);

if (pessoaJuridicaLocalizada != null) {
pessoaJuridicaLocalizada.exibir();

pjRepo.excluir(idPessoaJuridica);

System.out.println("Pessoa excluida com sucesso!");
} else
System.out.println("Pessoa nao localizada!");
break;

case "M":

break;

default:
System.out.println("Opcao invalida."); break;
}

} while (!escolha.equalsIgnoreCase("M")); break;

// obterId
case "4":
do {
System.out.println("====="); System.out.println("F
- Pessoa Fisica | J - Pessoa Juridica | M - Menu");

escolha = scan.next();
scan.nextLine();

switch (escolha.toUpperCase()) {

case "F":
System.out.println("Digite o ID da pessoa: ");
int idPessoaFisica = scan.nextInt();

PessoaFisica pessoaFisicaLocalizada = pfRepo.obter(idPessoaFisica);

if (pessoaFisicaLocalizada != null) { System.out.println("Pessoa
localizada!"); pessoaFisicaLocalizada.exibir(); } else

```

```

System.out.println("Pessoa nao localizada!");
break;

case "J":
System.out.println("Digite o ID da pessoa: ");
int idPessoaJuridica = scan.nextInt();

PessoaJuridica pessoaJuridicaLocalizada =
pjRepo.obter(idPessoaJuridica);

if (pessoaJuridicaLocalizada != null) { System.out.println("Pessoa
localizada!");

pessoaJuridicaLocalizada.exibir(); } else
System.out.println("Pessoa nao localizada!");
break;

case "M":
break;

default:
System.out.println("Opcao invalida."); break;
}

} while (!escolha.equalsIgnoreCase("M")); break;

//obterTodos
case "5":
do {
System.out.println("====="); System.out.println("F
- Pessoa Fisica | J - Pessoa Juridica | M - Menu");

escolha = scan.next();
scan.nextLine();

switch (escolha.toUpperCase()) {

case "F":
System.out.println("Lista de pessoas Fisicas: ");
pfRepo.obterTodos()
.forEach(pessoaFisica -> { pessoaFisica.exibir();
System.out.println(); });
break;

case "J":
System.out.println("Lista de pessoas juridicas: ");
pjRepo.obterTodos()
.forEach(pessoaJuridica -> { pessoaJuridica.exibir();
System.out.println(); });
break;
case "M":
break;

default:
System.out.println("Opcao invalida"); break;
}

} while (!escolha.equalsIgnoreCase("M")); break;

```

```

// Persistir/Salvar
case "6":
    System.out.println("Escolha o nome do arquivo"); escolha =
    scan.next();
    scan.nextLine();
    try {
        pfRepo.persistir(escolha+".fisica.bin");
        pjRepo.persistir(escolha+".juridica.bin"); } catch (IOException
    erro) {
        System.out.println("Erro ao persistir/salvar os dados: " +
        erro.getMessage());
    }
    break;

//Recuperar/Carregar
case "7":
    System.out.println("Informe o nome do arquivo salvo"); escolha =
    scan.next();
    scan.nextLine();
    try {
        pfRepo.recuperar(escolha+".fisica.bin");
        pjRepo.recuperar(escolha+".juridica.bin"); } catch (ClassNotFoundException
    | IOException erro) { System.out.println("Erro ao recuperar os dados: " +
    erro.getMessage());
    }
        break;

case "0":
    System.out.println("Sistema Finalizado com sucesso."); break;

default:
    System.out.println("Opcao invalida"); break;
}
} while (!escolha.equals("0"));
scan.close();
}
}

```

Resultados:

?? Procedimento 1:

```
Output - CadastroPOO (run) X
run:
Dados de pessoas fisicas armazenados.
Dados de pessoas fisicas recuperados.

id: 1
Nome: Ana
CPF: 11111111111
Idade: 25

id: 2
Nome: Carlos Jose
CPF: 22222222222
Idade: 52

Dados das pessoas juridicas armazenados.
Dados de pessoas juridicas recuperados.

id: 3
Nome: XPTO Sales
CNPJ: 33333333333333

id: 4
Nome: XPTO Solutions
CNPJ: 4444444444444444
BUILD SUCCESSFUL (total time: 0 seconds)
```

Análise e Conclusão

- Quais as vantagens e desvantagens do uso de herança?

Vantagens:

1. Classes podem herdar características(métodos e atributos) de outras classes situadas acima ou transmitir suas características às classes abaixo;
2. Evita repetir o mesmo código várias vezes;
3. Caso uma alteração seja necessária, ela só precisará ser feita na classe pai, e será automaticamente propagada para as subclasses.

Desvantagens:

1. Fraco encapsulamento entre classes e subclasses e o forte acoplamento entre elas onde ao mudar uma superclasse pode afetar todas as subclasses;
2. Quando um objeto precisa ser de uma classe diferente em momentos diferentes e não é possível com a herança.

- Por que a interface Serializable é necessária ao efetuar persistência em arquivos binários?

Essa interface permite que os objetos sejam serializados(convertidos em uma sequência de bytes) e desserializados com a conversão de volta à um objeto.

- Como o paradigma funcional é utilizado pela API stream no Java?

A API stream é usada para manipular coleções (Collections) de uma maneira mais eficiente, utilizando funções. Ela possibilita uma iteração sobre essas coleções de objetos e, a cada elemento, realizar alguma ação, seja ela de filtragem, mapeamento, transformação, etc.

- Quando trabalhamos com Java, qual padrão de desenvolvimento é adotado na persistência de dados em arquivos?

Nesse projeto foram utilizadas a classe ObjectOutputStream para escrever objetos em um arquivo "[prefixo].fisica.bin" e [prefixo].juridica.bin" e a classe ObjectInputStream para ler os objetos dos mesmos arquivos.

- O que são elementos estáticos e qual o motivo para o método main adotar esse modificador?

Os elementos estáticos são elementos que "existem", ou seja, estão disponíveis para uso, sem a necessidade de serem instanciados. Podem ser utilizados em código sem a necessidade de existirem objetos produzidos (sem a necessidade de um comando "new Classe(")).

- Para que serve a classe Scanner ?

Para leitura de dados de entrada (inteiros, boolean, string, etc) inseridos pelo usuário através do teclado.

- Como o uso de classes de repositório impactou na organização do código?

As classes de repositórios serviram para gerenciar,centralizar e organizar as atividades de inserir, excluir, alterar, localizar, recuperar e salvar os dados de pessoa física e jurídica.