Faculdade Estácio - POLO FREGUESIA - RIO DE JANEIRO - RJ

Curso: Desenvolvimento Full Stack
Disciplina: Iniciando O Caminho Pelo Java
Número da Turma: 9001
Semestre Letivo: 2025.1
Integrante: Gabriel Galvão Reis
Repositório: https://github.com/galvao-reis/estacio-missao-pratica-3-1

Iniciando o Caminho pelo Java

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

Objetivo da Prática

1. Utilizar herança e polimorfismo na definição de entidades.
2. Utilizar persistência de objetos em arquivos binários.
3. Implementar uma interface cadastral em modo texto.
4. Utilizar o controle de exceções da plataforma Java.
5. No final do projeto, o aluno terá implementado um sistema cadastral em Java, utilizando os recursos da programação orientada a objetos e a persistência em arquivos binários.

Sumário

Arquivos desenvolvidos pela Prática

1. cadastroPOO.java

```
package cadastropoo;

import model.*;
import java.io.*;
import java.util.Scanner;
import java.util.ArrayList;

public class CadastroPOO {

	private static final PessoaFisicaRepo pessoasFisicas = new
PessoaFisicaRepo();
	private static final PessoaJuridicaRepo pessoasJuridicas = new
PessoaJuridicaRepo();
	private static final Scanner input = new Scanner(System.in);
	/**
	* @param args the command line arguments
	*/
	public static void main(String[] args) {



	while (true){
	System.out.println(
		"""

================================================================
===
		Por favor, escolha um numero para continuar (Apenas números):
		1 -> Incluir Pessoa.
		2 -> Alterar Pessoa.
		3 -> Excluir Pessoa.
		4 -> Buscar pelo ID.
		5 -> Exibir Todos.
		6 -> Persistir Dados.
		7 -> Recuperar Dados.
		0 -> Finalizar Programa.

================================================================
==="""
	);
	int opcao = -1;
	String resposta = input.nextLine();
	try{
	opcao = Integer.parseInt(resposta);
	}
	catch(NumberFormatException exception){
	//System.out.println(exception.toString());
	System.out.println("As opções apenas aceitam números.");
	}
```

```
        switch(opcao){
        case 1->{
                cadastroPessoa();
        }
        case 2->{
                alterarCadastro();
        }
        case 3->{
                excluirCadastro();
        }
        case 4->{
                procurarID();
        }
        case 5->{
                exibirTodos();
        }
        case 6->{
                salvarArquivo();
        }
        case 7->{
                recuperarArquivo();
        }
        case 0->{
                System.exit(0);
        }
        }
        }
        }

        private static int verificarTipoPessoa(){
        int opcao = -1;
        while (opcao < 0 | opcao > 2){
        System.out.println("""
                        Qual o tipo? (Apenas Numeros)
                        1 - Fisica
                        2 - Juridica
                        0 - Retornar""");
        String resposta = input.nextLine();
        try{
                opcao = Integer.parseInt(resposta);
                if (opcao < 0){
                System.out.println( "Por favor insira um numero posivito." );
                }
        }
        catch(NumberFormatException exception){
                System.out.println("A resposta nao e um numero, tente
novamente");
        }
        }
        return opcao;
        }

        private static void cadastroPessoa(){
        //opcao só pode ser 1,2 ou 0;
```

```
        int opcao = verificarTipoPessoa();
        switch (opcao){
        case 1 -> {
                cadastroPessoaFisica();
        }
        case 2 -> {
                cadastroPessoaJuridica();
        }
        default ->{
                System.out.println("Esta opcao não é reconhecida. Retornando ao
menu.");
        }
        }
        }

        private static int pedirId(){
        String resposta;
        int id = -1;
        while (id < 0){
        System.out.println("Insira o ID desejado: (Apenas Numeros)");
        System.out.println("Para retornar digite 0");
        resposta = input.nextLine();
        try {
                id = (Integer.parseInt(resposta));
        }
        catch(NumberFormatException exception){
                System.out.println("Este ID nao e valido!");
        }
        }
        return id;
        }

        private static String pedirNomePessoa(){
        System.out.println("Insira o nome da pessoa:");
        return input.nextLine();
        }

        private static String pedirNomeEmpresa(){
        System.out.println("Insira o nome da empresa:");
        return input.nextLine();
        }

        private static String pedirCpf(){
        System.out.println("Insira o CPF da pessoa:");
        return input.nextLine();
        }

        private static String pedirCnpj(){
        System.out.println("Insira o CNPJ da empresa:");
        return input.nextLine();

        }
        private static int pedirIdade(){
        int idade = -1;
```

```
        while (idade < 0){
        System.out.println("Insira a idade da pessoa: (Apenas Numeros)");
        String resposta = input.nextLine();
        try {
                idade = (Integer.parseInt(resposta));
                if ( idade < 0 ){
                System.out.println("Idade deve ser um numero positivo.");
                }
        }
        catch(NumberFormatException exception){
                System.out.println("A idade inserida não é um número! Tente
novamente");
                cadastroPessoaFisica();
        }
        }
        return idade;
        }

        private static String verificarInformacoes(){
        String resposta = "";

        while (!(resposta.equalsIgnoreCase("S") ||
resposta.equalsIgnoreCase("N"))){
        System.out.println("Estas informações estão corretas? (S/N)");
        resposta = input.nextLine();
        if (!(resposta.equalsIgnoreCase("S") || resposta.equalsIgnoreCase("N"))){
        System.out.println("A resposta deve ser S ou N, apenas.");
        }
        }
        return resposta;
        }

        private static void cadastroPessoaFisica(){
        PessoaFisica pFisica = new PessoaFisica(-1,"","",-1);
        boolean finished = false;
        while (!finished){
        int id = -1;
        while (id < 0){
                id = pedirId();
                if ( id == 0 ){
                return;
                }
                if (verificarIdExiste(id)){
                id = -1;
                System.out.println("Esse ID ja existe, insira um novo numero.");
                }
                else{
                pFisica.setId(id);
                }
        }

        pFisica.setNome(pedirNomePessoa());

        pFisica.setCpf(pedirCpf());
```

```
pFisica.setIdade(pedirIdade());

System.out.println("----------------------");
pFisica.exibir();
System.out.println("----------------------");

String resposta = verificarInformacoes();
if (resposta.equalsIgnoreCase("S")){
        finished = true;
        pessoasFisicas.inserir(pFisica);
}
else if(resposta.equalsIgnoreCase("N")){
        System.out.println("Tente novamente.");
}
}
}
private static void cadastroPessoaJuridica(){
PessoaJuridica pJuridica = new PessoaJuridica(-1,"","");
boolean finished = false;
while (!finished){
int id = -1;
while (id < 0){
        id = pedirId();
        if ( id == 0 ){
        return;
        }
        if (verificarIdExiste(id)){
        id = -1;
        System.out.println("Esse ID ja existe, insira um novo numero.");
        }
        else{
        pJuridica.setId(id);
        }
}


pJuridica.setNome(pedirNomeEmpresa());

pJuridica.setCnpj(pedirCnpj());

System.out.println("----------------------");
pJuridica.exibir();
System.out.println("----------------------");
String resposta = verificarInformacoes();

if (resposta.equalsIgnoreCase("S")){
        finished = true;
        pessoasJuridicas.inserir(pJuridica);
}
else if(resposta.equalsIgnoreCase("N")){
        System.out.println("Tente novamente.");
}
```

```
        }

        }

        private static void alterarCadastro(){
        int opcao = verificarTipoPessoa();
        switch (opcao){
        case 1 -> {
                alterarPessoaFisica();
        }
        case 2 -> {
                alterarPessoaJuridica();
        }
        default ->{
                System.out.println("Retornando ao menu.");
        }
        }
        }

        private static void alterarPessoaFisica(){
        int id = -1;
        while (id <0 | !verificarIdFisicoExiste(id)){
        System.out.println( "Insira o ID da pessoa que deseja alterar:" );
        String resposta = input.nextLine();
        try{
                id = Integer.parseInt(resposta);
        }
        catch(NumberFormatException exception){
                System.out.println("O ID deve ser um numero. Tente novamente.");
        }
        if (!verificarIdFisicoExiste(id)){
                System.out.println("O ID inserido não existe ou não é de uma
Pessoa Fisica.");
        }
        }
        System.out.println("Os dados atuais para este id são:");
        PessoaFisica pFisica = pessoasFisicas.obter(id);
        System.out.println("-----------------");
        pFisica.exibir();
        System.out.println("-----------------");
        System.out.println("Entre com os dados novos:");
        pFisica.setNome(pedirNomePessoa());
        pFisica.setCpf(pedirCpf());
        pFisica.setIdade(pedirIdade());

        System.out.println("As novas informações são:");
        System.out.println("--------------------");
        pFisica.exibir();
        System.out.println("--------------------");

        String resposta = verificarInformacoes();
        if (resposta.equalsIgnoreCase("S")){
        pessoasFisicas.alterar(pFisica);
        }
```

```
        else if (resposta.equalsIgnoreCase("N")){
        System.out.println("Alteração cancelada. Retornando ao menu.");
        }



        }
        private static void alterarPessoaJuridica(){
        int id = -1;
        while (id <0 | !verificarIdJuridicoExiste(id)){
        System.out.println( "Insira o ID da empresa que deseja alterar:" );
        String resposta = input.nextLine();
        try{
                id = Integer.parseInt(resposta);
        }
        catch(NumberFormatException exception){
                System.out.println("O ID deve ser um numero. Tente novamente.");
        }
        if (!verificarIdJuridicoExiste(id)){
                System.out.println("O ID inserido não existe ou não é de uma
Pessoa Fisica.");
        }
        }
        System.out.println("Os dados atuais para este id são:");
        PessoaJuridica pJuridica = pessoasJuridicas.obter(id);
        System.out.println("-----------------");
        pJuridica.exibir();
        System.out.println("-----------------");
        System.out.println("Entre com os dados novos:");
        pJuridica.setNome(pedirNomeEmpresa());
        pJuridica.setCnpj(pedirCnpj());

        System.out.println("As novas informações são:");
        System.out.println("-----------------");
        pJuridica.exibir();
        System.out.println("-----------------");

        String resposta = verificarInformacoes();
        if (resposta.equalsIgnoreCase("S")){
        pessoasJuridicas.alterar(pJuridica);
        }
        else if (resposta.equalsIgnoreCase("N")){
        System.out.println("Alteração cancelada. Retornando ao menu.");
        }
        }

        private static void excluirCadastro(){
        int opcao = verificarTipoPessoa();
        switch (opcao){
        case 1 -> {
                excluirCadastroFisico();
        }
        case 2 -> {
                excluirCadastroJuridico();
```

```
        }
        default ->{
                System.out.println("Retornando ao menu.");
        }

        }
        }
        private static void excluirCadastroFisico(){
        int id = pedirId();
        if (verificarIdFisicoExiste(id)){
        System.out.println("Excluindo Pessoa Física de ID: "+ id + ": " +
pessoasFisicas.obter(id).getNome());
        pessoasFisicas.excluir(id);
        }
        else {
        System.out.println("O ID fornecido não existe ou não é de uma Pessoa
Física. Retornando ao menu.");
        }
        }
        private static void excluirCadastroJuridico(){
        int id = pedirId();
        if (verificarIdJuridicoExiste(id)){
        System.out.println("Excluindo Pessoa Juridica de ID: "+ id + ": " +
pessoasJuridicas.obter(id).getNome());
        pessoasJuridicas.excluir(id);
        }
        else{
        System.out.println("O ID fornecido não existe ou não é de uma Pessoa
Jurídica. Retornando ao menu.");
        }
        }
        static void procurarID(){
        int opcao = verificarTipoPessoa();
        if (opcao == 0) {
        return;
        }
        int id = pedirId();


        switch (opcao){
        case 1 ->{
                if (verificarIdFisicoExiste(id)){
                pessoasFisicas.obter(id).exibir();
                return;
                }
        }
        case 2 ->{
                if (verificarIdJuridicoExiste(id)){
                pessoasJuridicas.obter(id).exibir();
                return;
                }
        }
        }
        System.out.println( "O ID fornecido não foi encontrado." );
```

```
        }

        static void exibirTodos(){
        int tipoPessoa = verificarTipoPessoa();
        switch(tipoPessoa){
        case 1 -> {
                System.out.println("As Pessoas Fisicas registradas são: \n");
                pessoasFisicas.obterTodos().stream().forEach(p -> p.exibir());
        }
        case 2 -> {
                System.out.println("As Pessoas Juridicas registradas são: \n");
                pessoasJuridicas.obterTodos().stream().forEach(p -> p.exibir());
        }
        default -> {
                return;
        }
        }
        }

        static void salvarArquivo(){
        System.out.println("Digite o prefixo do nome para o arquivo:");
        String resposta = input.nextLine();
        try{
        String name = resposta + ".fisica.bin";
        pessoasFisicas.persistir(name);
        System.out.println("Dados das pessoas fisicas salvos com sucesso no
arquivo " + name +".");
        }
        catch(IOException exception){
        System.out.println("Não foi possível salvar os dados das pessoas fisicas.\n
Erro: " + exception.toString());
        }
        try{
        String name = resposta + ".juridica.bin";
        pessoasJuridicas.persistir(name);
        System.out.println("Dados das pessoas juridicas salvos com sucesso no
arquivo "+ name +".");
        }
        catch(IOException exception){
        System.out.println("Não foi possível salvar os dados pessoas juridicas.\n
Erro: " + exception.toString());
        }
        }

        static void recuperarArquivo(){
        System.out.println("Digite o prefixo do arquivo:");
        String resposta = input.nextLine();
        try{
        String name = resposta + ".fisica.bin";
        pessoasFisicas.recuperar(name);
        System.out.println("Dados das pessoas fisicas recuperados com sucesso
do arquivo "+ name +".");
        }
        catch(IOException | ClassNotFoundException exception){
```

```
        System.out.println("Houve um erro tentando recuperar as Pessoas Fisicas.
Erro: "+ exception.toString());
        }
        try{
        String name = resposta + ".juridica.bin";
        pessoasJuridicas.recuperar(name);
        System.out.println("Dados das pessoas juridicas recuperados com sucesso
do arquivo " + name +".");
        }
        catch(IOException | ClassNotFoundException exception){
        System.out.println("Houve um erro tentando recuperar as Pessoas
Juridicas. Erro: "+ exception.toString());
        }
        }


        static boolean verificarIdExiste(int id){
        return  verificarIdFisicoExiste(id) | verificarIdJuridicoExiste(id);
        }

        private static boolean verificarIdFisicoExiste(int id){
        ArrayList<PessoaFisica> pFisicas = pessoasFisicas.obterTodos();
        return ( pFisicas.stream().anyMatch( p-> p.getId() == id ));
        }

        private static boolean verificarIdJuridicoExiste(int id){
        ArrayList<PessoaJuridica> pJuridicas = pessoasJuridicas.obterTodos();
        return (pJuridicas.stream().anyMatch( p-> p.getId() == id) );
        }

}
```

2. Pessoa.java

```java
package model;

import java.io.Serializable;
/**
 *
 * @author Gabriel
 */
public class Pessoa implements Serializable {

    private int id;
    private String nome;

    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;
    }

    public void exibir(){
        System.out.println( "id: " + this.getId() + ", nome: " + this.getNome());
    }
}
```

3. PessoaFisica.java

```java
package model;

import java.io.Serializable;

public class PessoaFisica extends Pessoa implements Serializable {

    private String cpf;
    private int idade;

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

    public String getCpf(){
        return this.cpf;
    }

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

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

    @Override
    public void exibir(){
        System.out.println(
                "Id: "+ this.getId() +
                "\nNome: " + this.getNome() +
                "\nCPF: " + this.getCpf() +
                ",\nIdade: " + this.getIdade());
    }

}

```

4. PessoaJuridica.java

```java
package model;

import java.io.Serializable;

public class PessoaJuridica extends Pessoa implements Serializable {

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

    public String getCnpj(){
        return this.cnpj;
    }

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

    @Override
    public void exibir(){
        System.out.println(
                    "Id: " + this.getId() +
                    "\nNome: " + this.getNome() +
                    "\nCNPJ: " + this.getCnpj());
    }

}

```

5. PessoaFisicaRepo.java

```java
package model;

import java.io.*;
import java.util.ArrayList;

public class PessoaFisicaRepo{
    private ArrayList<PessoaFisica> pessoasFisicas;

    public PessoaFisicaRepo(){
        this.pessoasFisicas = new ArrayList<PessoaFisica>();
    }

    public void inserir( PessoaFisica pessoa ){
        this.pessoasFisicas.add(pessoa);
    }

    public void alterar( PessoaFisica pessoa ){
        this.excluir(pessoa.getId());
        this.inserir(pessoa);
    }

    public void excluir( int id ){
        this.pessoasFisicas.removeIf( pessoa -> pessoa.getId() == id);
    }

    public PessoaFisica obter( int id){
        for (PessoaFisica pessoa : this.pessoasFisicas){
            if (pessoa.getId() == id){
                return pessoa;
            }
        }
        return null;
    }

    public ArrayList<PessoaFisica> obterTodos(){
        return this.pessoasFisicas;
    }

    public void persistir(String nomeArquivo)throws IOException{
        FileOutputStream fos = new FileOutputStream(nomeArquivo);
        ObjectOutputStream oos = new ObjectOutputStream(fos);
        oos.writeObject(this.pessoasFisicas);
    }

    public void recuperar ( String nomeArquivo) throws IOException, ClassNotFoundException{
        FileInputStream fis = new FileInputStream(nomeArquivo);
        ObjectInputStream ois = new ObjectInputStream(fis);
        this.pessoasFisicas = (ArrayList<PessoaFisica>) ois.readObject();
    }
}

```

6. PessoaJuridicaRepo.java

```java
package model;

import java.io.*;
import java.util.ArrayList;

public class PessoaJuridicaRepo{

    private ArrayList<PessoaJuridica> pessoasJuridicas;

    public PessoaJuridicaRepo(){
        this.pessoasJuridicas = new ArrayList<PessoaJuridica>();
    }

    public void inserir(PessoaJuridica pessoa){
        this.pessoasJuridicas.add(pessoa);
    }

    public void alterar(PessoaJuridica pessoa){
        this.excluir(pessoa.getId());
        this.inserir(pessoa);
    }

    public void excluir( int id ){
        this.pessoasJuridicas.removeIf( pessoa -> pessoa.getId() == id );
    }

    public PessoaJuridica obter( int id ){
        for (PessoaJuridica pessoa : pessoasJuridicas){
            if (pessoa.getId() == id){
                return pessoa;
            }
        }
        return null;
    }

    public ArrayList<PessoaJuridica> obterTodos(){
        return this.pessoasJuridicas;
    }

    public void persistir( String nomeArquivo ) throws IOException{
        FileOutputStream fos = new FileOutputStream(nomeArquivo);
        ObjectOutputStream oos = new ObjectOutputStream(fos);
        oos.writeObject(this.pessoasJuridicas);
    }

    public void recuperar( String nomeArquivo) throws IOException, ClassNotFoundException {
        FileInputStream fis = new FileInputStream(nomeArquivo);
        ObjectInputStream ois = new ObjectInputStream(fis);
        this.pessoasJuridicas = (ArrayList<PessoaJuridica>) ois.readObject();

    }

}

```

Resultado da execução do código

```
run:
==========================================================
Por favor, escolha um numero para continuar (Apenas n□meros):
        1 -> Incluir Pessoa.
        2 -> Alterar Pessoa.
        3 -> Excluir Pessoa.
        4 -> Buscar pelo ID.
        5 -> Exibir Todos.
        6 -> Persistir Dados.
        7 -> Recuperar Dados.
        0 -> Finalizar Programa.
==========================================================
1
Qual o tipo? (Apenas Numeros)
1 - Fisica
2 - Juridica
0 - Retornar
1
Insira o ID desejado: (Apenas Numeros)
Para retornar digite 0
1
Insira o nome da pessoa:
Gabriel
Insira o CPF da pessoa:
16491970582
Insira a idade da pessoa: (Apenas Numeros)
30
----------------------
Id: 1
Nome: Gabriel
CPF: 16491970582,
```

Pequeno trecho da Execução do Código.

Análise e Conclusão

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

    Os métodos estáticos permitem que funções sejam chamadas sem que nenhum objeto da classe seja inicializado. O método main adota esse modificar pois é automaticamente chamado pelo sistema e não depende da inicialização da classe que o contém para ser executado.

2. Para que serve a classe Scanner?

    A classe Scanner é utilizada para capturar uma informação fornecida pelo usuário do programa, seja ele uma String, ou qualquer outro tipo de entrada.

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

    A utilização de classes fez com que o código fosse compartimentalizado em pequenos pedaços e, desta forma, tornou o gerenciamento do projeto mais simples. A sua utilização também proporciona um certo nível de controle em como utilizar o código, pois ao declarar uma propriedade como privada, ele impede que elas sejam acessadas sem o devido tratamento.