



Estácio

Campus:

Norte Shopping - [Av. Dom Hélder Câmara 5080, Rio de Janeiro, RJ, 20771-004](#)

Curso: Desenvolvimento Full Stack

Disciplina: RPG0014 - Iniciando o caminho pelo Java

Turma: 9001

Semestre Letivo: 2024.3

Aluno: Gabriel Bernardo Carneiro

Matrícula: 202308953541

Link do repositório GitHub: [GbDev1907/mundo-3-missao-1 \(github.com\)](https://github.com/GbDev1907/mundo-3-missao-1)

Título da Prática:

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.

1º Procedimento | Criação das Entidades e Sistema de Persistência

Todos os códigos solicitados:

```
package model;

import java.io.Serializable;

public class Pessoa implements Serializable {

    private int id;

    private String nome;

    public Pessoa() {}

    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: " + id + ", Nome: " + nome);

    }

}
```

2 – PessoaFisica.java

```
Package model;

public class PessoaFisica extends Pessoa {

    private String cpf;

    private int idade;


    public PessoaFisica() {
    }

    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;

    }


    @Override

    public void exibir() {

        System.out.println("Id: " + getId());

        System.out.println("Nome: " + getNome());

        System.out.println("CPF: " + this.cpf);

        System.out.println("Idade: " + this.idade);

    }

}
```

3 – PessoaJuridica.java

```
package model;

public class PessoaJuridica extends Pessoa {

    private String cnpj;

    public PessoaJuridica() {

    }

    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;

    }

    @Override
    public void exibir() {

        System.out.println("Id: " + getId());

        System.out.println("Nome: " + getNome());

        System.out.println("CNPJ: " + this.cnpj);

    }

}
```

4 – PessoaFisicaRepo.java

```
package model;
import java.io.*;
import java.util.ArrayList;
public class PessoaFisicaRepo {
    private ArrayList<PessoaFisica> pessoasFisicas = new ArrayList<>();
    public void inserir(PessoaFisica pessoa) {
        pessoasFisicas.add(pessoa);
    }
    public void alterar(PessoaFisica pessoa) {
        for (int i = 0; i < pessoasFisicas.size(); i++) {
            if (pessoasFisicas.get(i).getId() == pessoa.getId()) {
                pessoasFisicas.set(i, pessoa);
                return;
            }
        }
    }
    public void excluir(int id) {
        pessoasFisicas.removeIf(p -> p.getId() == id);
    }
    public PessoaFisica obter(int id) {
        for (PessoaFisica pessoa : pessoasFisicas) {
            if (pessoa.getId() == id) {
                return pessoa;
            }
        }
        return null;
    }
    public ArrayList<PessoaFisica> obterTodos() {
        return pessoasFisicas;
    }
    public void persistir(String nomeArquivo) throws IOException {
        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
            oos.writeObject(pessoasFisicas);
        }
    }
    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
            pessoasFisicas = (ArrayList<PessoaFisica>)ois.readObject();
        }
    }
}
```





5 – PessoaJuridicaRepo.java

```
package model;
import java.io.*;
import java.util.ArrayList;
public class PessoaJuridicaRepo {
    private ArrayList<PessoaJuridica> pessoasJuridicas = new ArrayList<>();
    public void inserir(PessoaJuridica pessoa) {
        pessoasJuridicas.add(pessoa);
    }
    public void alterar(PessoaJuridica pessoa) {
        for (int i = 0; i < pessoasJuridicas.size(); i++) {
            if (pessoasJuridicas.get(i).getId() == pessoa.getId()) {
                pessoasJuridicas.set(i, pessoa);
                return;
            }
        }
    }
    public void excluir(int id) {
        pessoasJuridicas.removeIf(p -> p.getId() == id);
    }
    public PessoaJuridica obter(int id) {
        for (PessoaJuridica pessoa : pessoasJuridicas) {
            if (pessoa.getId() == id) {
                return pessoa;
            }
        }
        return null;
    }
    public ArrayList<PessoaJuridica> obterTodos() {
        return pessoasJuridicas;
    }
    public void persistir(String nomeArquivo) throws IOException {
        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
            oos.writeObject(pessoasJuridicas);
        }
    }
    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
            pessoasJuridicas = (ArrayList<PessoaJuridica>)ois.readObject();
        }
    }
}
```

6 – CadastroPOO.java (main)

```
package cadastropoo;
import model.PessoaFisicaRepo;
import model.PessoaFisica;
import model.PessoaJuridicaRepo;
import model.PessoaJuridica;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.io.IOException;
public class CadastroPOO {
    private static final Logger LOGGER = Logger.getLogger(CadastroPOO.class.getName());
    public static void main(String[] args) {
        PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
        PessoaFisica pf1 = new PessoaFisica(1, "Ana", "11111111111", 25);
        PessoaFisica pf2 = new PessoaFisica(2, "Carlos", "22222222222", 52);
        repo1.inserir(pf1);
        repo1.inserir(pf2);
        try {
            repo1.persistir("pessoasFisicas.dat");
        } catch (IOException e) {
            LOGGER.log(Level.SEVERE, "Erro ao persistir dados", e);
        }
        PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
        try {
            repo2.recuperar("pessoasFisicas.dat");
        } catch (IOException | ClassNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Erro ao recuperar dados", e);
        }
        System.out.println("Dados de Pessoa Fisica Armazenados.");
        System.out.println("Dados de Pessoa Fisica Recuperados.");
        for (PessoaFisica pf : repo2.obterTodos()) {
            pf.exibir();
        }
        PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
        PessoaJuridica pj1 = new PessoaJuridica(3, "XPTO Sales", "33333333333333");
        PessoaJuridica pj2 = new PessoaJuridica(4, "XPTO Solutions", "44444444444444");
        repo3.inserir(pj1);
        repo3.inserir(pj2);
        try {
            repo3.persistir("pessoasJuridicas.dat");
        } catch (IOException e) {
            LOGGER.log(Level.SEVERE, "Erro ao persistir dados", e);
        }
        PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
        try {
            repo4.recuperar("pessoasJuridicas.dat");
        } catch (IOException | ClassNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Erro ao persistir dados", e);
        }
        System.out.println("Dados de Pessoa Jurídica Armazenados.");
        System.out.println("Dados de Pessoa Jurídica Recuperados.");
        for (PessoaJuridica pj : repo4.obterTodos()) {
            pj.exibir();
        }
    }
}
```


Resultado da execução dos códigos:

```
run:
Dados de Pessoa Fisica Armazenados.
Dados de Pessoa Fisica Recuperados.
Id: 1
Nome: Ana
CPF: 11111111111
Idade: 25
Id: 2
Nome: Carlos
CPF: 22222222222
Idade: 52
Dados de Pessoa Juridica Armazenados.
Dados de Pessoa Juridica Recuperados.
```

Análise e Conclusão:

1) Quais as vantagens e desvantagens do uso de herança ?

As vantagens é que podemos reutilizar o código, evitando a duplicação de código, outra vantagem é a facilitação de manutenção, alterações no código da superclasse são automaticamente refletidas nas subclasses. Uma das desvantagens é que a complexidade do sistema pode aumentar, especialmente quando há múltiplos níveis de herança.

2) Por que a interface Serializable é necessário ao efetuar persistência em arquivos binários?

A interface Serializable é necessária para a persistência em arquivos binários porque ela permite que objetos Java sejam convertidos em um formato de bytes.

3) Como o paradigma funcional é utilizado pela API stream em Java ?

A API Stream do Java adota o paradigma funcional, possibilitando o processamento de coleções de dados de maneira declarativa e sem modificar a coleção original. Com ela, utilizamos expressões lambda para realizar operações como filtragem, mapeamento e redução de dados, mantendo a imutabilidade e evitando efeitos colaterais.

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

O padrão de serialização/deserialização envolve a utilização da interface Serializable para marcar classes cujos objetos podem ser convertidos em uma sequência de bytes e depois reconstruídos. Esse processo facilita o armazenamento em arquivos e a transmissão de dados pela rede, permitindo que objetos sejam gravados e lidos de forma eficiente.

2º Procedimento | Criação do Cadastro em Modo Texto

Todos os códigos solicitados:

```
package cadastropoo;

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

public class CadastroPOO {
    public static void main(String[] args) {
        try (Scanner scanner = new Scanner(System.in)) {
            PessoaFisicaRepo repoFisica = new PessoaFisicaRepo();
            PessoaJuridicaRepo repoJuridica = new PessoaJuridicaRepo();

            int opcao;
            do {

System.out.println("=====");
                System.out.println("Escolha uma opcao:");
                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 Dados");
                System.out.println("7 - Recuperar Dados");
                System.out.println("0 - Finalizar programa");

System.out.println("=====");
                opcao = scanner.nextInt();
                scanner.nextLine();

                switch (opcao) {
                    case 1 ->
                        incluir(scanner, repoFisica, repoJuridica);
                    case 2 ->
                        alterar(scanner, repoFisica, repoJuridica);
                    case 3 ->
                        excluir(scanner, repoFisica, repoJuridica);
                    case 4 ->
                        exibirPeloid(scanner, repoFisica, repoJuridica);
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

```

        case 5 ->
            exibirTodos(scanner, repoFisica, repoJuridica);
        case 6 ->
            salvarDados(scanner, repoFisica, repoJuridica);
        case 7 ->
            recuperarDados(scanner, repoFisica, repoJuridica);
        case 0 ->
            System.out.println("Finalizando...");
        default ->
            System.out.println("Opcao invalida!");
    }
} while (opcao != 0);
}
}

```

```

private static void incluir(Scanner scanner, PessoaFisicaRepo repoFisica,
PessoaJuridicaRepo repoJuridica) {
    System.out.println("Escolha o tipo (F - Fisica, J - Juridica):");
    char tipo = scanner.next().toUpperCase().charAt(0);
    scanner.nextLine();

    switch (tipo) {
        case 'F' -> {
            System.out.println("Digite o id da pessoa:");
            int id = scanner.nextInt();
            scanner.nextLine();
            System.out.println("Insira os dados...");
            System.out.println("Nome:");
            String nome = scanner.nextLine();
            System.out.println("CPF:");
            String cpf = scanner.nextLine();
            System.out.println("Idade:");
            int idade = scanner.nextInt();
            scanner.nextLine();

            PessoaFisica pf = new PessoaFisica(id, nome, cpf, idade);
            repoFisica.inserir(pf);
        }
        case 'J' -> {
            System.out.println("Digite o id da pessoa:");
            int id = scanner.nextInt();
            scanner.nextLine();
            System.out.println("Insira os dados...");
            System.out.println("Digite o nome:");
            String nome = scanner.nextLine();

```

```

        System.out.println("CNPJ:");
        String cnpj = scanner.nextLine();

        PessoaJuridica pj = new PessoaJuridica(id, nome, cnpj);
        repoJuridica.inserir(pj);
    }
    default ->
        System.out.println("Tipo invalido!");
    }
}

```

```

private static void alterar(Scanner scanner, PessoaFisicaRepo repoFisica,
PessoaJuridicaRepo repoJuridica) {
    System.out.println("Escolha o tipo (F - Fisica, J - Juridica):");
    char tipo = scanner.next().toUpperCase().charAt(0);
    scanner.nextLine();

    switch (tipo) {
        case 'F' -> {
            System.out.println("Digite o ID da pessoa fisica a ser alterada:");
            int id = scanner.nextInt();
            scanner.nextLine();
            PessoaFisica pf = repoFisica.obter(id);
            if (pf != null) {
                System.out.println("Digite o novo nome:");
                String nome = scanner.nextLine();
                System.out.println("Digite o novo CPF:");
                String cpf = scanner.nextLine();
                System.out.println("Digite a nova idade:");
                int idade = scanner.nextInt();
                scanner.nextLine();
                pf.setNome(nome);
                pf.setCpf(cpf);
                pf.setIdade(idade);
                repoFisica.alterar(pf);
                System.out.println("Pessoa fisica alterada com sucesso!");
            } else {
                System.out.println("Pessoa fisica nao encontrada!");
            }
        }
        case 'J' -> {
            System.out.println("Digite o ID da pessoa juridica a ser alterada:");
            int id = scanner.nextInt();
            scanner.nextLine();
            PessoaJuridica pj = repoJuridica.obter(id);

```

```

        if (pj != null) {
            System.out.println("Digite o novo nome:");
            String nome = scanner.nextLine();
            System.out.println("Digite o novo CNPJ:");
            String cnpj = scanner.nextLine();
            pj.setNome(nome);
            pj.setCnpj(cnpj);
            repoJuridica.alterar(pj);
            System.out.println("Pessoa juridica alterada com sucesso!");
        } else {
            System.out.println("Pessoa juridica nao encontrada!");
        }
    }
    default -> System.out.println("Tipo invalido!");
}
}

```

```

private static void excluir(Scanner scanner, PessoaFisicaRepo repoFisica,
PessoaJuridicaRepo repoJuridica) {
    System.out.println("Escolha o tipo (F - Fisica, J - Juridica):");
    String tipo = scanner.nextLine();

    switch (tipo.toUpperCase()) {
        case "F" -> {
            System.out.println("Digite o ID da pessoa fisica a ser excluida:");
            int id = scanner.nextInt();
            scanner.nextLine();
            if (repoFisica.excluir(id)) {
                System.out.println("Pessoa fisica removida com sucesso!");
            } else {
                System.out.println("pessoa fisica nao encontrada!");
            }
        }
        case "J" -> {
            System.out.println("Digite o ID da pessoa juridica a ser excluida:");
            int id = scanner.nextInt();
            scanner.nextLine();
            if (repoJuridica.excluir(id)) {
                System.out.println("Pessoa juridica removida com sucesso!");
            } else {
                System.out.println("Pessoa juridica nao encontrada!");
            }
        }
        default ->
            System.out.println("Tipo invalido");
    }
}

```

```
}  
}
```

```
private static void exibirPeloid(Scanner scanner, PessoaFisicaRepo repoFisica,  
PessoaJuridicaRepo repoJuridica) {  
    System.out.println("Escolha o tipo (F - Fisica, J - Juridica):");  
    char tipo = scanner.next().toUpperCase().charAt(0);  
    scanner.nextLine();  
  
    switch (tipo) {  
        case 'F' -> {  
            System.out.println("Digite o ID da pessoa fisica:");  
            int id = scanner.nextInt();  
            scanner.nextLine();  
            PessoaFisica pf = repoFisica.obter(id);  
            if (pf != null) {  
                System.out.println(pf);  
            } else {  
                System.out.println("Pessoa fisica nao encontrada!");  
            }  
        }  
        case 'J' -> {  
            System.out.println("Digite o ID da pessoa juridica:");  
            int id = scanner.nextInt();  
            scanner.nextLine();  
            PessoaJuridica pj = repoJuridica.obter(id);  
            if (pj != null) {  
                System.out.println(pj);  
            } else {  
                System.out.println("Pessoa juridica nao encontrada!");  
            }  
        }  
        default ->  
            System.out.println("Tipo invalido!");  
    }  
}
```

```
private static void exibirTodos(Scanner scanner, PessoaFisicaRepo repoFisica,  
PessoaJuridicaRepo repoJuridica) {  
    System.out.println("Escolha o tipo (F - Fisica, J - Juridica):");  
    char tipo = scanner.next().toUpperCase().charAt(0);  
    scanner.nextLine();  
  
    switch (tipo) {  
        case 'F' ->
```

```

        repoFisica.obterTodos().forEach(System.out::println);
    case 'J' ->
        repoJuridica.obterTodos().forEach(System.out::println);
    default ->
        System.out.println("Tipo invalido!");
    }
}

```

```

private static void salvarDados(Scanner scanner, PessoaFisicaRepo repoFisica,
PessoaJuridicaRepo repoJuridica) {
    System.out.println("Digite o prefixo dos arquivos:");
    String prefixo = scanner.nextLine();

    try {
        repoFisica.persistir(prefixo + ".fisica.bin");
        repoJuridica.persistir(prefixo + ".juridica.bin");
        System.out.println("Dados salvos com sucesso!");
    } catch (IOException e) {
        System.out.println("Erro ao salvar dados: " + e.getMessage());
    }
}

```

```

private static void recuperarDados(Scanner scanner, PessoaFisicaRepo
repoFisica, PessoaJuridicaRepo repoJuridica) {
    System.out.println("Digite o prefixo dos arquivos:");
    String prefixo = scanner.nextLine();

    try {
        repoFisica.recuperar(prefixo + ".fisica.bin");
        repoJuridica.recuperar(prefixo + ".juridica.bin");
    } catch (IOException | ClassNotFoundException e) {
        System.out.println("Erro ao recuperar dados: " + e.getMessage());
    }
}
}

```


Resultado da execução:

```
=====
Escolha uma opcao:
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
Escolha o tipo (F - Fisica, J - Juridica):
f
Digite o id da pessoa:
1
Insira os dados...
Nome:
Gabriel
CPF:
123456789
Idade:
20
```

Análise e Conclusão:

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

O método main é o ponto inicial de execução de qualquer programa Java. Quando o programa começa, a JVM procura a classe indicada e chama o método main dessa classe para iniciar a execução. Se o método main não fosse declarado como estático, a JVM teria que criar uma instância da classe antes de poder executar o programa, o que não seria eficiente e nem apropriado para o propósito de iniciar a aplicação.

2) Para que serve a classe Scanner?

Serve para obter dados de várias fontes, como entrada do usuário no console, strings, arquivos, entre outros. É utilizada para ler e processar tipos primitivos de maneira simples e eficiente.

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

Nas arquiteturas MVC, é comum utilizar camadas para separar responsabilidades. Dessa forma, é uma boa prática trabalhar com classes de repositório, que permitem, por exemplo, isolar certas lógicas das classes mais gerais. Isso torna a manutenção do código mais fácil e organizado.

