Campus: 202301037751 - POLO CENTRO - PORTO REAL - RJ
Curso: Desenvolvimento Full Stack
Disciplina: Nível 1: Iniciando o Caminho Pelo Java
Turma: 9001
Semestre letivo: 3º Semestre
Nome do estudante: Leonardo Naves de Lima Araujo

Missão Prática | Nível 1 | Mundo 3
RPG0014 - Iniciando o caminho pelo Java

2º Procedimento | Criação das Entidades e Sistema de Persistência

Implementação de Cadastro de Clientes em Java com Persistência em Arquivos

Objetivos 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,
6. utilizando os recursos da programação orientada a objetos e a persistência em
7. arquivos binários.

Códigos da Prática

Main.java ——

```
package model;

import java.io.*;
import java.util.Scanner;
import java.util.List;

public class Main {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        PessoaFisicaRepo repoPessoaFisica = new PessoaFisicaRepo();
        PessoaJuridicaRepo repoPessoaJuridica = new PessoaJuridicaRepo();
        String prefixoArquivo;

        boolean continuar = true;
        while (continuar) {
            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 Dados");
            System.out.println("7 - Recuperar Dados");
            System.out.println("0 - Finalizar Programa");
            System.out.println("============================");
            System.out.print("Escolha uma opção: ");
            int opcao = scanner.nextInt();
            scanner.nextLine();
            try {
                switch (opcao) {
                    case 1:
                        incluirPessoa(scanner, repoPessoaFisica, repoPessoaJuridica);
                        break;
                    case 2:
                        alterarPessoa(scanner, repoPessoaFisica, repoPessoaJuridica);
                        break;
                    case 3:
                        excluirPessoa(scanner, repoPessoaFisica, repoPessoaJuridica);
                        break;
                    case 4:
                        buscarPessoaPorId(scanner, repoPessoaFisica, repoPessoaJuridica);
                        break;
```

```
                case 5:
                    exibirTodos(scanner, repoPessoaFisica, repoPessoaJuridica);
                    break;
                case 6:
                    System.out.print("Digite o prefixo para o arquivo de dados: ");
                    prefixoArquivo = scanner.nextLine();
                    persistirDados(repoPessoaFisica, repoPessoaJuridica, prefixoArquivo);
                    break;
                case 7:
                    System.out.print("Digite o prefixo para o arquivo de dados a ser recuperado:
");
                    prefixoArquivo = scanner.nextLine();
                    recuperarDados(repoPessoaFisica, repoPessoaJuridica, prefixoArquivo);
                    break;
                case 0:
                    continuar = false;
                    break;
                default:
                    System.out.println("Opção inválida.");
                    break;
            }
        } catch (IOException e) {
            System.err.println("Ocorreu um erro de entrada/saída: " + e.getMessage());
        } catch (ClassNotFoundException e) {
            System.err.println("Classe não encontrada durante a deserialização: " +
e.getMessage());
        }
    }
    scanner.close();
    System.out.println("Programa finalizado.");
  }

  private static void incluirPessoa(Scanner scanner, PessoaFisicaRepo repoPF,
PessoaJuridicaRepo repoPJ) throws IOException {
    System.out.println("Escolha o tipo de pessoa para incluir (F - Física, J - Jurídica): ");
    String tipo = scanner.nextLine();
    if (tipo.equalsIgnoreCase("F")) {
        System.out.print("Nome: ");
        String nome = scanner.nextLine();
        System.out.print("CPF: ");
        String cpf = scanner.nextLine();
        System.out.print("Idade: ");
        int idade = Integer.parseInt(scanner.nextLine());
        int id = repoPF.obterTodos().size() + 1;
        PessoaFisica pf = new PessoaFisica(id, nome, cpf, idade);
        repoPF.inserir(pf);
    } else if (tipo.equalsIgnoreCase("J")) {
        System.out.print("Nome: ");
```

```
            String nome = scanner.nextLine();
            System.out.print("CNPJ: ");
            String cnpj = scanner.nextLine();
            int id = repoPJ.obterTodos().size() + 1;
            PessoaJuridica pj = new PessoaJuridica(id, nome, cnpj);
            repoPJ.inserir(pj);
        } else {
            System.out.println("Tipo inválido. Tente novamente.");
        }
    }

    private static void alterarPessoa(Scanner scanner, PessoaFisicaRepo repoPF,
PessoaJuridicaRepo repoPJ) throws IOException {
        System.out.println("Escolha o tipo de pessoa para alterar (F - Física, J - Jurídica): ");
        String tipo = scanner.nextLine();
        System.out.print("Digite o id da pessoa: ");
        int id = Integer.parseInt(scanner.nextLine());

        if (tipo.equalsIgnoreCase("F")) {
            PessoaFisica pf = repoPF.obter(id);
            if (pf != null) {
                System.out.print("Nome (" + pf.getNome() + "): ");
                pf.setNome(scanner.nextLine());
                System.out.print("CPF (" + pf.getCpf() + "): ");
                pf.setCpf(scanner.nextLine());
                System.out.print("Idade (" + pf.getIdade() + "): ");
                pf.setIdade(Integer.parseInt(scanner.nextLine()));
                repoPF.alterar(pf);
            } else {
                System.out.println("Pessoa Física não encontrada.");
            }
        } else if (tipo.equalsIgnoreCase("J")) {
            PessoaJuridica pj = repoPJ.obter(id);
            if (pj != null) {
                System.out.print("Nome (" + pj.getNome() + "): ");
                pj.setNome(scanner.nextLine());
                System.out.print("CNPJ (" + pj.getCnpj() + "): ");
                pj.setCnpj(scanner.nextLine());
                repoPJ.alterar(pj);
            } else {
                System.out.println("Pessoa Jurídica não encontrada.");
            }
        } else {
            System.out.println("Tipo inválido. Tente novamente.");
        }
    }
```

```
    private static void excluirPessoa(Scanner scanner, PessoaFisicaRepo repoPF,
PessoaJuridicaRepo repoPJ) throws IOException {
        System.out.println("Escolha o tipo de pessoa para excluir (F - Física, J - Jurídica): ");
        String tipo = scanner.nextLine();
        System.out.print("Digite o id da pessoa: ");
        int id = Integer.parseInt(scanner.nextLine());

        if (tipo.equalsIgnoreCase("F")) {
            repoPF.excluir(id);
        } else if (tipo.equalsIgnoreCase("J")) {
            repoPJ.excluir(id);
        } else {
            System.out.println("Tipo inválido. Tente novamente.");
        }
    }

    private static void buscarPessoaPorId(Scanner scanner, PessoaFisicaRepo repoPF,
PessoaJuridicaRepo repoPJ) throws IOException {
        System.out.println("Escolha o tipo de pessoa para buscar (F - Física, J - Jurídica): ");
        String tipo = scanner.nextLine();
        System.out.print("Digite o id da pessoa: ");
        int id = Integer.parseInt(scanner.nextLine());

        if (tipo.equalsIgnoreCase("F")) {
            PessoaFisica pf = repoPF.obter(id);
            if (pf != null) {
                pf.exibir();
            } else {
                System.out.println("Pessoa Física não encontrada.");
            }
        } else if (tipo.equalsIgnoreCase("J")) {
            PessoaJuridica pj = repoPJ.obter(id);
            if (pj != null) {
                pj.exibir();
            } else {
                System.out.println("Pessoa Jurídica não encontrada.");
            }
        } else {
            System.out.println("Tipo inválido. Tente novamente.");
        }
    }

    private static void exibirTodos(Scanner scanner, PessoaFisicaRepo repoPF,
PessoaJuridicaRepo repoPJ) throws IOException {
        System.out.println("Escolha o tipo de pessoa para exibir todos (F - Física, J - Jurídica):
");
        String tipo = scanner.nextLine();
```

```
        if (tipo.equalsIgnoreCase("F")) {
            List<PessoaFisica> todasPessoasFisicas = repoPF.obterTodos();
            todasPessoasFisicas.forEach(PessoaFisica::exibir);
        } else if (tipo.equalsIgnoreCase("J")) {
            List<PessoaJuridica> todasPessoasJuridicas = repoPJ.obterTodos();
            todasPessoasJuridicas.forEach(PessoaJuridica::exibir);
        } else {
            System.out.println("Tipo inválido. Tente novamente.");
        }
    }

    private static void persistirDados(PessoaFisicaRepo repoPF, PessoaJuridicaRepo
repoPJ, String prefixoArquivo) throws IOException {
        repoPF.persistir(prefixoArquivo + ".fisica.bin");
        repoPJ.persistir(prefixoArquivo + ".juridica.bin");
        System.out.println("Dados persistidos com sucesso.");
    }

    private static void recuperarDados(PessoaFisicaRepo repoPF, PessoaJuridicaRepo
repoPJ, String prefixoArquivo) throws IOException, ClassNotFoundException {
        repoPF.recuperar(prefixoArquivo + ".fisica.bin");
        repoPJ.recuperar(prefixoArquivo + ".juridica.bin");
        System.out.println("Dados recuperados com sucesso.");
    }

}
```

Pessoa.java ----------

```
package model;

import java.io.Serializable;

public class Pessoa implements Serializable {
    private static final long serialVersionUID = 1L;
    private int id;
    private String nome;

    // Construtor padrão
    public Pessoa() {
    }

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

```
    // Getters e Setters
    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.println("Id: " + id);
        System.out.println("Nome: " + nome);
    }

    // Sobrescrita do método toString
    @Override
    public String toString() {
        return "Pessoa{" + "id=" + id + ", nome='" + nome + '\'' + '}';
    }
}
```

PessoaFisica.java ----------

```
package model;

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

    // Construtor padrão
    public PessoaFisica() {
        super();
    }

    // Construtor completo
    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() {
        super.exibir();
        System.out.println("CPF: " + cpf);
        System.out.println("Idade: " + idade);
    }

    @Override
    public String toString() {
        return super.toString() + ", CPF: " + cpf + ", Idade: " + idade;
    }
}
```

PessoaFisicaRepo.java ----------

```
package model;

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

public class PessoaFisicaRepo {
    private ArrayList<PessoaFisica> pessoasFisicas;

    public PessoaFisicaRepo() {
        pessoasFisicas = new ArrayList<>();
```

```
    }

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

    public void alterar(PessoaFisica pessoaFisica) {
        int index = pessoasFisicas.indexOf(pessoaFisica);
        if (index != -1) {
            pessoasFisicas.set(index, pessoaFisica);
        }
    }

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

    public PessoaFisica obter(int id) {
        return pessoasFisicas.stream()
                .filter(p -> p.getId() == id)
                .findFirst()
                .orElse(null);
    }

    public List<PessoaFisica> obterTodos() {
        return new ArrayList<>(pessoasFisicas);
    }

    public void persistir(String nomeArquivo) throws IOException {
        try (ObjectOutputStream out = new ObjectOutputStream(new
FileOutputStream(nomeArquivo))) {
            out.writeObject(pessoasFisicas);
        }
    }

    @SuppressWarnings("unchecked")
    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException
{
        try (ObjectInputStream in = new ObjectInputStream(new
FileInputStream(nomeArquivo))) {
            pessoasFisicas = (ArrayList<PessoaFisica>) in.readObject();
        }
    }
```

PessoaJuridica.java ----------

```
package model;
```

```
public class PessoaJuridica extends Pessoa {
    private String cnpj;


    public PessoaJuridica() {
        super();
    }

    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() {
        super.exibir();
        System.out.println("CNPJ: " + cnpj);
    }

    @Override
    public String toString() {
        return super.toString() + ", CNPJ: " + cnpj;
    }
}
```

PessoaJuridicaRepo.java ----------

```
package model;

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

public class PessoaJuridicaRepo {
    private ArrayList<PessoaJuridica> pessoasJuridicas;

    public PessoaJuridicaRepo() {
        pessoasJuridicas = new ArrayList<>();
```

```
    }

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

    public void alterar(PessoaJuridica pessoaJuridica) {
        int index = pessoasJuridicas.indexOf(pessoaJuridica);
        if (index != -1) {
            pessoasJuridicas.set(index, pessoaJuridica);
        }
    }

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

    public PessoaJuridica obter(int id) {
        return pessoasJuridicas.stream()
                .filter(p -> p.getId() == id)
                .findFirst()
                .orElse(null);
    }

    public List<PessoaJuridica> obterTodos() {
        return new ArrayList<>(pessoasJuridicas);
    }

    public void persistir(String nomeArquivo) throws IOException {
        try (ObjectOutputStream out = new ObjectOutputStream(new
FileOutputStream(nomeArquivo))) {
            out.writeObject(pessoasJuridicas);
        }
    }

    @SuppressWarnings("unchecked")
    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException
{
        try (ObjectInputStream in = new ObjectInputStream(new
FileInputStream(nomeArquivo))) {
            pessoasJuridicas = (ArrayList<PessoaJuridica>) in.readObject();
        }
    }
}
```

Resultados da Execução

```
run:
==========================
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
==========================
Escolha uma op��o: 1
Escolha o tipo de pessoa para incluir (F - F�sica, J - Jur�dica):
f
Nome: Leonardo
CPF: 12345
Idade: 19
==========================
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
==========================
Escolha uma op��o: 1
Escolha o tipo de pessoa para incluir (F - F�sica, J - Jur�dica):
j
Nome: Estacio
CNPJ: 12345
==========================
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
==========================
Escolha uma op��o: 2
Escolha o tipo de pessoa para alterar (F - F�sica, J - Jur�dica):
j
```

Digite o id da pessoa: 1
Nome (Estacio): Google
CNPJ (12345): 12345
==========
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
==========
Escolha uma op��o: 3
Escolha o tipo de pessoa para excluir (F - F�sica, J - Jur�dica):
f
Digite o id da pessoa: 1
==========
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
==========
Escolha uma op��o: 5
Escolha o tipo de pessoa para exibir todos (F - F�sica, J - Jur�dica):
f
==========
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
==========
Escolha uma op��o: 5
Escolha o tipo de pessoa para exibir todos (F - F�sica, J - Jur�dica):
j
Id: 1
Nome: Google
CNPJ: 12345
==========
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
==========
Escolha uma op��o: 6
Digite o prefixo para o arquivo de dados: teste
Dados persistidos com sucesso.
==========
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
==========
Escolha uma op��o: 0
Programa finalizado.
BUILD SUCCESSFUL (total time: 3 minutes 18 seconds)

run:
==========
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
==========
Escolha uma op��o: 6
Digite o prefixo para o arquivo de dados: teste
Dados persistidos com sucesso.
==========
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

==========
Escolha uma op��o: 5
Escolha o tipo de pessoa para exibir todos (F - F�sica, J - Jur�dica):
j
Id: 1
Nome: Google
CNPJ: 12345
==========
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
==========
Escolha uma op��o:

Análise e Conclusão

1. Elementos Estáticos e o Método Main:
Elementos estáticos, como campos estáticos (variáveis) e métodos estáticos, são associados à classe em que são declarados, ao invés de a uma instância específica de uma classe. Isso significa que eles podem ser acessados diretamente através do nome da classe. O método `main` é declarado como estático porque é o ponto de entrada do programa e deve ser acessível pelo ambiente de execução do Java sem a necessidade de instanciar a classe, o que facilita o início da execução do programa.

2. Para que Serve a Classe Scanner:
A classe `Scanner` é uma ferramenta do Java utilizada para ler a entrada de dados primitivos como strings e números. Em um sistema de cadastro em modo texto, `Scanner` permite ler a entrada do usuário a partir do console. Isso é essencial para coletar informações como nomes, identificações e outros dados relevantes durante a interação com o usuário.

3. Como o Uso de Classes de Repositório Impactou na Organização do Código:
As classes de repositório centralizam a lógica para manipular coleções de objetos, como criar, buscar, atualizar e deletar entidades. Isso ajuda a desacoplar a lógica de negócios da lógica de apresentação e armazenamento de dados. No nosso caso, `PessoaFisicaRepo` e `PessoaJuridicaRepo` fornecem uma abstração sobre como os objetos `PessoaFisica` e `PessoaJuridica` são mantidos, facilitando mudanças futuras na forma de armazenamento (por exemplo, migrar de arquivos para banco de dados) sem impactar o restante do código.

Github do projeto:
https://github.com/Navesz/Iniciando-o-caminho-pelo-Java-2