

# Relatório discente de acompanhamento



Missão Prática | Nível 1 | Mundo 3

Amanda Duque Kawauchi

- Campus: Morumbi
- Curso: Desenvolvimento Full-Stack
- Número da Turma: 2023.3
- Repositório GitHub: <https://github.com/madukisp/java-missao1-CadastroP00>

## Conteúdo da Documentação:

**Título da Prática:**  
**CadastroP00**

### Objetivo da Prática

Desenvolver um sistema de cadastro de clientes utilizando conceitos de programação orientada a objetos, como herança e polimorfismo, e persistência de dados em arquivos binários.

### Códigos

## 1º Procedimento | Criação das Entidades e Sistema de Persistência

```
package model;

import java.io.Serializable;

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

```

```

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

```

```

    }

    @Override
    public String toString() {
        return "ID: " + getId() + ", Nome: " + getNome() + ", CPF: " + cpf + ", Idade: " + idade;
    }
}

```

```

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 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: " + cnpj);
    }
}

```

```

package model;

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

public class PessoaFisicaRepo {
    private ArrayList<PessoaFisica> pessoasFisicas = new ArrayList<>();

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

```

```

    public void alterar(PessoaFisica pessoaFisica) {
        for (int i = 0; i < pessoasFisicas.size(); i++) {
            if (pessoasFisicas.get(i).getId() == pessoaFisica.getId()) {
                pessoasFisicas.set(i, pessoaFisica);
                break;
            }
        }
    }

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

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

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

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

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

```

```

package model;

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

public class PessoaJuridicaRepo {
    private ArrayList<PessoaJuridica> pessoasJuridicas = new ArrayList<>();
}

```

```

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

    public void alterar(PessoaJuridica pessoaJuridica) {
        for (int i = 0; i < pessoasJuridicas.size(); i++) {
            if (pessoasJuridicas.get(i).getId() == pessoaJuridica.getId()) {
                pessoasJuridicas.set(i, pessoaJuridica);
            }
        }
    }

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

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

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

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

    public void recuperar(String nomeArquivo) throws IOException,
ClassNotFoundException {
        FileInputStream fileIn = new FileInputStream(nomeArquivo);
        ObjectInputStream in = new ObjectInputStream(fileIn);
        pessoasJuridicas = (ArrayList<PessoaJuridica>) in.readObject();
        in.close();
        fileIn.close();
    }
}

```

```

package model;

import java.io.IOException;

public class Main {

```

```

public static void main(String[] args) {
    PessoaFisicaRepo repo1 = new PessoaFisicaRepo();

    repo1.inserir(new PessoaFisica(1, "Ana", "1111111111", 25));
    repo1.inserir(new PessoaFisica(2, "Carlos", "2222222222", 52));

    try {
        repo1.persistir("pessoasFisicas.dat");
        System.out.println("Dados de Pessoa Fisica Armazenados.");
    } catch (IOException e) {
        e.printStackTrace();
    }

    PessoaFisicaRepo repo2 = new PessoaFisicaRepo();

    try {
        repo2.recuperar("pessoasFisicas.dat");
        System.out.println("Dados de Pessoa Fisica Recuperados.");
    } catch (IOException | ClassNotFoundException e) {
        e.printStackTrace();
    }

    for (PessoaFisica pf : repo2.obterTodos()) {
        pf.exibir();
    }

    PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();

    repo3.inserir(new PessoaJuridica(3, "XPTO Sales", "33333333333333"));
    repo3.inserir(new PessoaJuridica(4, "XPTO Solutions", "44444444444444"));

    try {
        repo3.persistir("pessoasJuridicas.dat");
        System.out.println("Dados de Pessoa Juridica Armazenados.");
    } catch (IOException e) {
        e.printStackTrace();
    }

    PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();

    try {
        repo4.recuperar("pessoasJuridicas.dat");
        System.out.println("Dados de Pessoa Juridica Recuperados.");
    } catch (IOException | ClassNotFoundException e) {
        e.printStackTrace();
    }

    for (PessoaJuridica pj : repo4.obterTodos()) {
        pj.exibir();
    }
}

```

## Resultados do 1º Procedimento

```
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.  
Id: 3  
Nome: XPTO Sales  
CNPJ: 33333333333333  
Id: 4  
Nome: XPTO Solutions  
CNPJ: 44444444444444
```

## 2º Procedimento | Criação do Cadastro em Modo Texto

```
import model.PessoaFisica;  
import model.PessoaFisicaRepo;  
import model.PessoaJuridica;  
import model.PessoaJuridicaRepo;  
import java.io.IOException;  
import java.util.Scanner;  
  
public class Main {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        PessoaFisicaRepo repoFisica = new PessoaFisicaRepo();  
        PessoaJuridicaRepo repoJuridica = new PessoaJuridicaRepo();  
  
        while (true) {  
            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("Digite uma opção: ");  
  
            int opcao = sc.nextInt();  
            sc.nextLine();  
        }  
    }  
}
```

```

switch (opcao) {
    case 1:
        System.out.println("Digite F para Pessoa Física ou J para Pessoa
Jurídica");

        char tipo = sc.nextLine().charAt(0);
        System.out.println("Digite o id da pessoa: ");
        int id = sc.nextInt();
        sc.nextLine();
        System.out.println("Insira os dados...");
        System.out.print("Nome: ");
        String nome = sc.nextLine();

        if (tipo == 'F' || tipo == 'f') {
            System.out.print("CPF: ");
            String cpf = sc.nextLine();
            System.out.print("Idade: ");
            int idade = sc.nextInt();
            repoFisica.inserir(new PessoaFisica(id, nome, cpf, idade));
        } else if (tipo == 'J' || tipo == 'j') {
            System.out.print("CNPJ: ");
            String cnpj = sc.nextLine();
            repoJuridica.inserir(new PessoaJuridica(id, nome, cnpj));
        }
        break;

    case 2:
        System.out.println("Digite F para Pessoa Física ou J para Pessoa
Jurídica");

        char tipoAlterar = sc.nextLine().charAt(0);
        System.out.println("Digite o id da pessoa que deseja alterar:");
        int idAlterar = sc.nextInt();
        sc.nextLine();

        if (tipoAlterar == 'F' || tipoAlterar == 'f') {
            PessoaFisica pfAtual = repoFisica.obter(idAlterar);
            if (pfAtual != null) {
                System.out.println("Dados atuais:");
                pfAtual.exibir();
                System.out.println("Insira os novos dados...");
                System.out.print("Nome: ");
                String novoNome = sc.nextLine();
                System.out.print("CPF: ");
                String novoCpf = sc.nextLine();
                System.out.print("Idade: ");
                int novaIdade = sc.nextInt();
                PessoaFisica pfAlterada = new PessoaFisica(idAlterar,
novoNome, novoCpf, novaIdade);
                repoFisica.alterar(pfAlterada);
            } else {
                System.out.println("Pessoa Física com ID " + idAlterar + "
não encontrada.");
            }
        }
    }
}

```



```

    }
} else if (tipoAlterar == 'J' || tipoAlterar == 'j') {
    PessoaJuridica pjAtual = repoJuridica.obter(idAlterar);
    if (pjAtual != null) {
        System.out.println("Dados atuais:");
        pjAtual.exibir();
        System.out.println("Insira os novos dados...");
        System.out.print("Nome: ");
        String novoNomeJ = sc.nextLine();
        System.out.print("CNPJ: ");
        String novoCnpj = sc.nextLine();
        PessoaJuridica pjAlterada = new PessoaJuridica(idAlterar,
novoNomeJ, novoCnpj);

        repoJuridica.alterar(pjAlterada);
    } else {
        System.out.println("Pessoa Jurídica com ID " + idAlterar +
" não encontrada.");
    }
}
}
break;

case 3:
    System.out.println("Digite F para Pessoa Física ou J para Pessoa
Jurídica");

    char tipoExcluir = sc.nextLine().charAt(0);
    System.out.println("Digite o id da pessoa que deseja excluir:");
    int idExcluir = sc.nextInt();
    sc.nextLine();

    if (tipoExcluir == 'F' || tipoExcluir == 'f') {
        PessoaFisica pfExcluir = repoFisica.obter(idExcluir);
        if (pfExcluir != null) {
            repoFisica.excluir(idExcluir);
            System.out.println("Pessoa Física com ID " + idExcluir + "
excluída com sucesso.");
        } else {
            System.out.println("Pessoa Física com ID " + idExcluir + "
não encontrada.");
        }
    } else if (tipoExcluir == 'J' || tipoExcluir == 'j') {
        PessoaJuridica pjExcluir = repoJuridica.obter(idExcluir);
        if (pjExcluir != null) {
            repoJuridica.excluir(idExcluir);
            System.out.println("Pessoa Jurídica com ID " + idExcluir +
" excluída com sucesso!");
        } else {
            System.out.println("Pessoa Jurídica com ID " + idExcluir +
" não encontrada.");
        }
    }
}
break;

```

```

        case 4:
            System.out.println("Digite F para Pessoa Física ou J para Pessoa
Jurídica");

            char tipoBuscar = sc.nextLine().charAt(0);
            System.out.println("Digite o id da pessoa que deseja buscar:");
            int idBuscar = sc.nextInt();
            sc.nextLine();

            if (tipoBuscar == 'F' || tipoBuscar == 'f') {
                PessoaFisica pfBuscar = repoFisica.obter(idBuscar);
                if (pfBuscar != null) {
                    pfBuscar.exibir();
                } else {
                    System.out.println("Pessoa Física com ID " + idBuscar + "
não encontrada.");
                }
            } else if (tipoBuscar == 'J' || tipoBuscar == 'j') {
                PessoaJuridica pjBuscar = repoJuridica.obter(idBuscar);
                if (pjBuscar != null) {
                    pjBuscar.exibir();
                } else {
                    System.out.println("Pessoa Jurídica com ID " + idBuscar +
" não encontrada.");
                }
            }
            break;

        case 5:
            System.out.println("Digite F para Pessoa Física ou J para Pessoa
Jurídica");

            char tipoExibir = sc.nextLine().charAt(0);
            if (tipoExibir == 'F' || tipoExibir == 'f') {
                System.out.println("Pessoas Físicas: ");
                for (PessoaFisica pf : repoFisica.obterTodos()) {
                    System.out.println(pf); // Isso chama o método toString()
da instância pf
                }
            } else if (tipoExibir == 'J' || tipoExibir == 'j') {
                System.out.println("Pessoas Jurídicas: ");
                for (PessoaJuridica pj : repoJuridica.obterTodos()) {
                    System.out.println(pj); // Isso chama o método toString()
da instância pj
                }
            }
            break;

        case 6:
            System.out.println("Digite F para Pessoa Física ou J para Pessoa
Jurídica");

            char tipoPersistir = sc.nextLine().charAt(0);

            System.out.println("Digite o prefixo para o nome do arquivo: ");

```

```

String prefixo = sc.nextLine();

if (tipoPersistir == 'F' || tipoPersistir == 'f') {
    try {
        repoFisica.persistir(prefixo + ".fisica.bin");
        System.out.println("Dados de Pessoa Física persistidos com
sucesso!" + prefixo + ".fisica.bin");
    } catch (IOException e) {
        System.out.println("Erro ao persistir dados de Pessoa
Física: " + e.getMessage());
    }
} else if (tipoPersistir == 'J' || tipoPersistir == 'j') {
    try {
        repoJuridica.persistir(prefixo + ".juridica.bin");
        System.out.println("Dados de Pessoas Jurídica persistidos
com sucesso" + prefixo + ".juridica.bin");
    } catch (IOException e) {
        System.out.println("Erro ao persistir dados de Pessoa
Jurídica: ");
    }
}
break;

case 7:
    System.out.println("Digite F para Pessoa Física ou J para Pessoa
Jurídica");

    char tipoRecuperar = sc.nextLine().charAt(0);

    System.out.println("Digite o prefixo do nome do arquivo para
recuperar os dados:");
    String prefixoRecuperar = sc.nextLine();

    if (tipoRecuperar == 'F' || tipoRecuperar == 'f') {
        try {
            repoFisica.recuperar(prefixoRecuperar + ".fisica.bin");
            System.out.println("Dados de Pessoa Física recuperados com
sucesso de " + prefixoRecuperar + ".fisica.bin");
        } catch (IOException | ClassNotFoundException e) {
            System.out.println("Erro ao recuperar dados de Pessoa
Física: " + e.getMessage());
        }
    } else if (tipoRecuperar == 'J' || tipoRecuperar == 'j') {
        try {
            repoJuridica.recuperar(prefixoRecuperar +
".juridica.bin");

            System.out.println("Dados de Pessoa Jurídica recuperados
com sucesso de " + prefixoRecuperar + ".juridica.bin");
        } catch (IOException | ClassNotFoundException e) {
            System.out.println("Erro ao recuperar dados de Pessoa
Jurídica: " + e.getMessage());
        }
    }
}

```

```

        break;

        case 0:
            System.out.println("Finalizando o programa");
            return;
        default:
            System.out.println("Opção inválida!");
    }
}
}
}
}

```

## Resultados do 2º Procedimento

```

=====
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
=====
Digite uma opção: 1
Digite F para Pessoa Física ou J para Pessoa Jurídica
f
Digite o id da pessoa:
120
Isira os dados...
Nome: XPT0
CPF: 333333333
Idade: 20
=====
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
=====
Digite uma opção:

```

## Análise e Conclusão:

- Quais as vantagens e desvantagens do uso de herança?

*Vantagens: A reutilização de código evita a duplicação de código ao herdar características comuns de uma classe pai. Organização, permite uma estrutura hierárquica de classes. Extensibilidade, facilita a adição de novas características a classes existentes sem alterar as classes existentes.*

*Desvantagens: Rigidez, quando se define a hierarquia de classes, pode ser difícil modificá-la. Pode levar a complexidade se não for usado corretamente.*

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

*A interface `Serializable` é necessária porque ela indica ao Java que a classe pode ser convertida em uma sequência de bytes, o que permite que os objetos da classe sejam gravados em arquivos binários ou transmitidos através de redes. Sem implementar esta interface, o Java não permitirá que os objetos da classe sejam serializados.*

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

*A API `Stream` em Java utiliza o paradigma funcional ao permitir operações de transformação e agregação de dados em coleções, como listas, de uma maneira declarativa. Ela introduz conceitos como `map`, `filter` e `reduce`, que são comuns em linguagens funcionais. A API `Stream` permite processar dados de forma mais concisa e legível em comparação com abordagens imperativas.*

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

*Ao trabalhar com Java, o padrão de desenvolvimento comumente adotado para persistência de dados em arquivos é o padrão `DAO` (`Data Access Object`). O `DAO` abstrai e encapsula todos os acessos aos dados, permitindo uma separação clara entre a lógica de negócios e as operações de acesso aos dados. Isso facilita a manutenção e a escalabilidade do código.*

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

*Elementos estáticos pertencem à classe, não às instâncias. O método `main` é estático porque é o ponto de entrada do programa e é chamado sem criar uma instância da classe.*

- Para que serve a classe `Scanner`?

*A classe `Scanner` é usada para ler entradas do usuário, como dados do teclado. É usada para receber entrada de dados primitivos como `int`, `double`, `string`, etc.,*

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

*Classes de repositório centralizam o acesso aos dados, promovendo organização, evitando duplicação e facilitando manutenção, seja de um banco de dados, arquivo ou qualquer outra fonte de dados.*