



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

**Thomas Antonio Santana de Oliveira**

**Matricula 202208999417**

**Polo W3 Sul – Brasília – DF.**

**RPG0014 - Iniciando o caminho pelo Java – Turma 2024.3 – 3 Semestre.**

### **Objetivo da Prática**

Utilizar herança e polimorfismo na definição de entidades; persistência de objetos em arquivos binários. Implementar uma interface cadastral em modo texto. Utilizar o controle de exceções da plataforma Java.

No final do projeto, terá sido implementado um sistema cadastral em Java, utilizando os recursos da programação orientada a objetos e a persistência em arquivos binários.

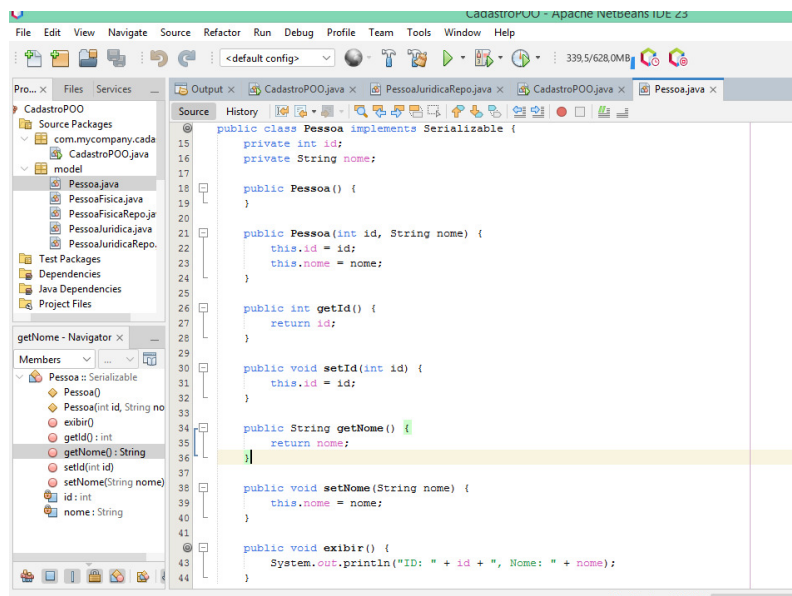
Descreva nessa seção qual o objetivo da sua prática. Todos os Relatórios de Práticas deverão ser confeccionados em arquivo no formato PDF, com a **Logo da Universidade, nome do Campus, nome do Curso, nome da Disciplina, número da Turma, semestre letivo**. Além disso, o projeto deve ser armazenado em um repositório no GIT e o respectivo endereço deve constar na documentação e essa documentação deve estar no no GIT. O código deve estar versionado no GIT de forma organizada.

Lembre-se que a organização contará pontos.

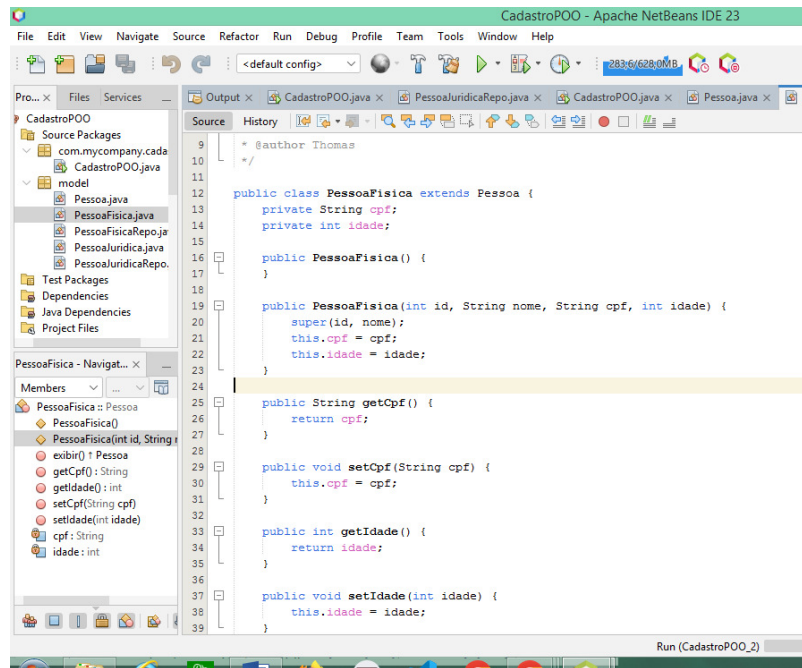
Esse template é um modelo a ser seguido. O aluno pode optar por seguir outro modelo, **desde que atenda a todas as etapas disponíveis na Missão Prática**. O documento final deve estar em pdf.

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

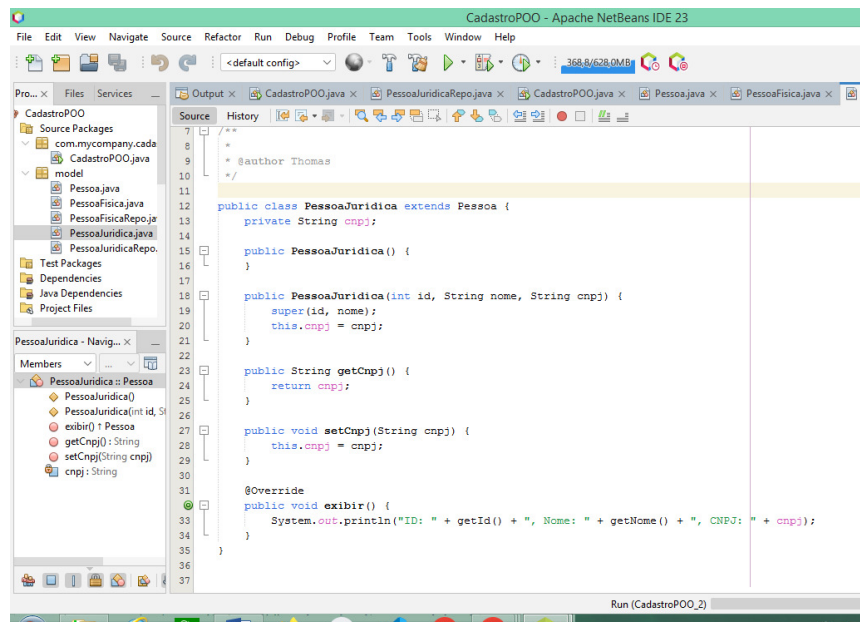
1. Inicialmente foi necessária a criação do projeto, e pacotes utilizando a ferramenta NetBeans, conforme a descrição: Criar um projeto do tipo Ant..Java Application no NetBeans, utilizando o nome CadastroPOO para o projeto.
2. Criar um pacote com o nome "model", para as entidades e gerenciadores.
  - a. Na ferramenta NetBeans é possível criar o pacote e alterar o nome para “model” como requerido.
3. No pacote model criar as entidades, com as seguintes características:
  - a. Classe Pessoa, com os campos id (inteiro) e nome (texto), método exibir, para impressão dos dados, construtor padrão e completo, além de getters e setters para todos os campos.



- b. Classe PessoaFisica, herdando de Pessoa, com o acréscimo dos campos cpf (texto) e idade (inteiro), método exibir polimórfico, construtores, getters e setters.



c. Classe PessoaJuridica, herdando de Pessoa, com o acréscimo do campo cnpj (texto), método exibir polimórfico, construtores, getters e setters.



d. Adicionar interface Serializable em todas as classes.

Para tanto foi realizada a importação e a implementação, em todas as classes

```
import java.io.Serializable;
```

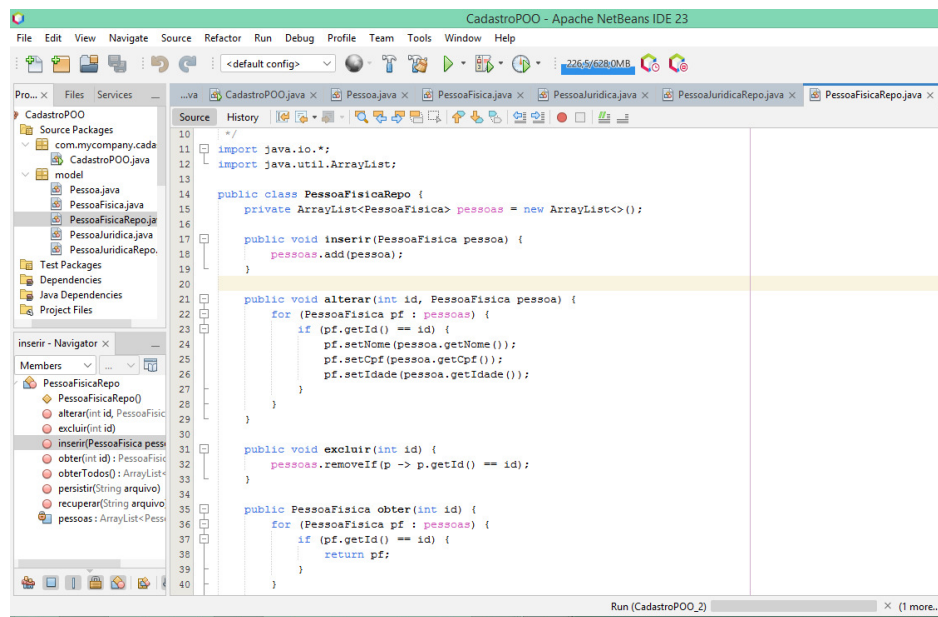
```
public class xxxx implements Serializable {
```

```
XXXX
```

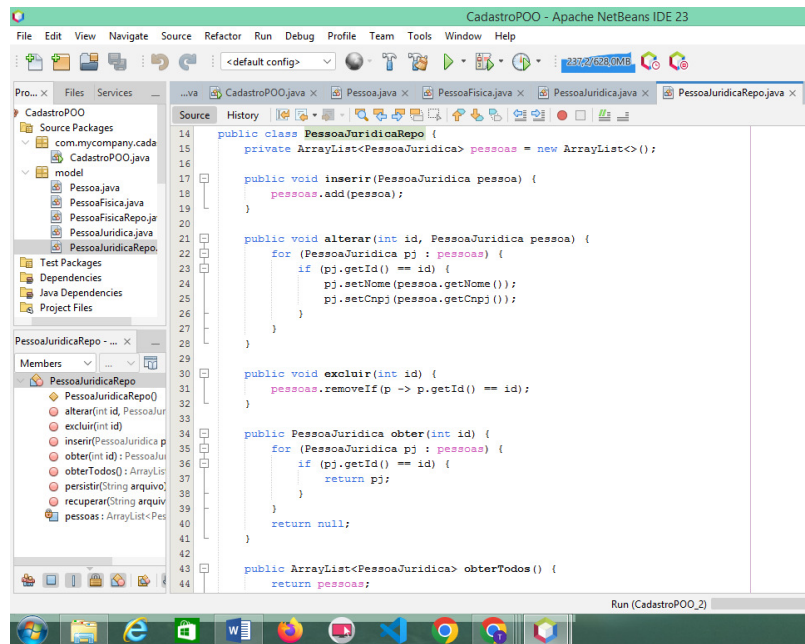
XXXX

}

4. No pacote model criar os gerenciadores, com as seguintes características:
  - a. Classe PessoaFisicaRepo, contendo um ArrayList de PessoaFisica, nível de acesso privado, e métodos públicos inserir, alterar, excluir, obter e obter Todos, para gerenciamento das entidades contidas no ArrayList.



- b. Classe PessoaJuridicaRepo, com um ArrayList de PessoaJuridica, nível de acesso privado, e métodos públicos inserir, alterar, excluir, obter e obter Todos, para gerenciamento das entidades contidas no ArrayList .



- c. Em ambos os gerenciadores adicionar o método público persistir, com a recepção do nome do arquivo, para armazenagem dos dados no disco.

```
public void persistir(String arquivo) throws IOException {

    try (ObjectOutputStream oos = new ObjectOutputStream(new
        FileOutputStream(arquivo))) {

        oos.writeObject(pessoas);

    }

}
```

- d. Em ambos os gerenciadores adicionar o método público recuperar, com a recepção do nome do arquivo, para recuperação dos dados do disco.

- e. Os métodos persistir e recuperar devem ecoar (throws) exceções

- f. O método obter deve retornar uma entidade a partir do id

```
public void recuperar(String arquivo) throws IOException,
    ClassNotFoundException {

    try (ObjectInputStream ois = new ObjectInputStream(new
        FileInputStream(arquivo))) {
```

```

        pessoas = (ArrayList<PessoaJuridica>) ois.readObject();
    }
}
}

```

g.Os métodos inserir e alterar devem ter entidades como parâmetros

h.O método excluir deve receber o id da entidade para exclusão

```

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

```

```

public void alterar(int id, PessoaJuridica pessoa) {
    for (PessoaJuridica pj : pessoas) {
        if (pj.getId() == id) {
            pj.setNome(pessoa.getNome());
            pj.setCnpj(pessoa.getCnpj());
        }
    }
}

```

```

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

```

5. Alterar o método main da classe principal para testar os repositórios:

a. Instanciar um repositório de pessoas físicas (repo1).

```
PessoaFisicaRepo repo1 = new PessoaFisicaRepo();

PessoaFisica pessoa1 = new PessoaFisica(1, "Thomas Santana",
"123.456.789-12", 37);

PessoaFisica pessoa2 = new PessoaFisica(2, "Joana Oliveira",
"321.654.987-21", 53);
```

b. Adicionar duas pessoas físicas, utilizando o construtor completo.

```
repo1.inserir(pessoa1);

repo1.inserir(pessoa2);
```

c. Invocar o método de persistência em repo1, fornecendo um nome de arquivo fixo, através do código.

```
repo1.persistir("pessoas_fisicas.dat");
```

d. Instanciar outro repositório de pessoas físicas (repo2).

```
PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
```

e. Invocar o método de recuperação em repo2, fornecendo o mesmo nome de arquivo utilizado anteriormente.

```
System.out.println("Dados de Pessoa Física Recuperados");
```

f. Exibir os dados de todas as pessoas físicas recuperadas.

```
for (PessoaFisica pf : repo2.obterTodos()) {

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

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

    System.out.println("CPF: " + pf.getCpf());

    System.out.println("Idade: " + pf.getIdade());

    System.out.println();    // Quebra de linha para melhor
formatação

}
```

g. Instanciar um repositório de pessoas jurídicas (repo3).

```
PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
```

```
    PessoaJuridica empresa1 = new PessoaJuridica(1, "Empresa  
Alpha", "33.333.333/0001-11");
```

```
    PessoaJuridica empresa2 = new PessoaJuridica(2, "Empresa  
Beta", "44.444.444/0001-22");
```

h. Adicionar duas pessoas jurídicas, utilizando o construtor completo.

```
repo3.inserir(empresa1);
```

```
repo3.inserir(empresa2);
```

i. Invocar o método de persistência em repo3, fornecendo um nome de arquivo fixo, através do código.

```
repo3.persistir("pessoas_juridicas.dat");
```

j. Instanciar outro repositório de pessoas jurídicas (repo4).

```
PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
```

k. Invocar o método de recuperação em repo4, fornecendo o mesmo nome de arquivo utilizado anteriormente.

```
repo4.recuperar("pessoas_juridicas.dat");
```

l. Exibir os dados de todas as pessoas jurídicas recuperadas.

```
for (PessoaJuridica pj : repo4.obterTodos()) {
```

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

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

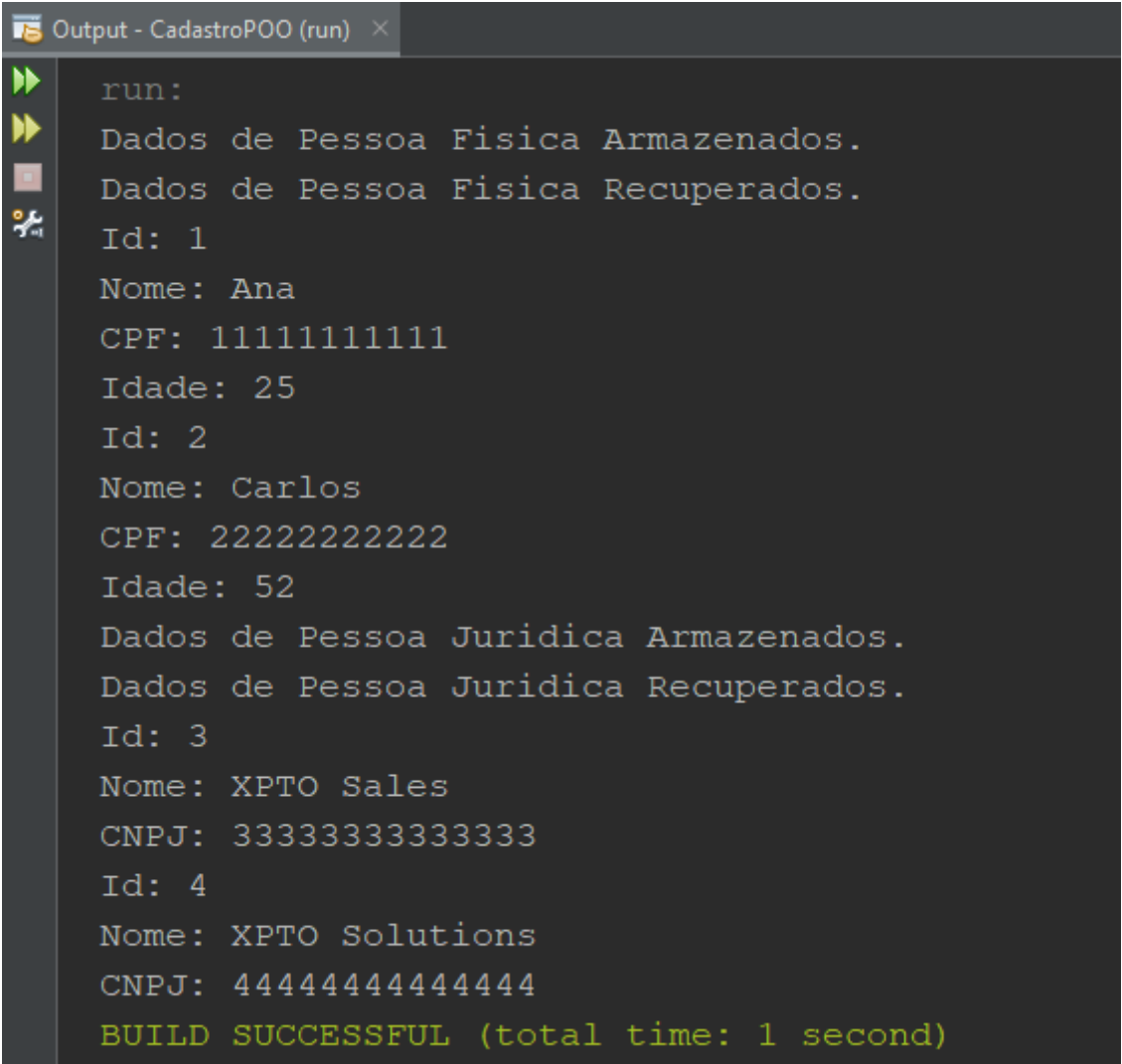
```
    System.out.println("CNPJ: " + pj.getCnpj());
```

```
    System.out.println();    // Quebra de linha para melhor  
    formatação
```

```
}
```



6. Ajustar as características para obter uma execução como a seguinte:



```
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.
Id: 3
Nome: XPTO Sales
CNPJ: 33333333333333
Id: 4
Nome: XPTO Solutions
CNPJ: 44444444444444
BUILD SUCCESSFUL (total time: 1 second)
```

Para tanto, foi necessário implementar/importar o Scanner:

```
import java.util.Scanner;
```

```
public class CadastroPOO {
```

```
    public static void main(String[] args) {
```

```
        PessoaFisicaRepo repoFisica = new PessoaFisicaRepo();
```

```
        PessoaJuridicaRepo repoJuridica = new PessoaJuridicaRepo();
```

```
        Scanner scanner = new Scanner(System.in);
```

```
int opcao = -1;

while (opcao != 0) {

    // Exibe o menu de opções

    System.out.println("=== MENU ===");

    System.out.println("1 - Incluir");

    System.out.println("2 - Alterar");

    System.out.println("3 - Excluir");

    System.out.println("4 - Exibir por ID");

    System.out.println("5 - Exibir todos");

    System.out.println("6 - Salvar dados");

    System.out.println("7 - Recuperar dados");

    System.out.println("0 - Sair");

    System.out.print("Escolha uma opção: ");

    opcao = scanner.nextInt();

    scanner.nextLine(); // Limpa o buffer

    switch (opcao) {

        case 1:

            incluir(scanner, repoFisica, repoJuridica);

            break;

        case 2:

            alterar(scanner, repoFisica, repoJuridica);

            break;

        case 3:

            excluir(scanner, repoFisica, repoJuridica);
```

```

        break;

    case 4:

        exibirPorId(scanner, repoFisica, repoJuridica);

        break;

    case 5:

        exibirTodos(scanner, repoFisica, repoJuridica);

        break;

    case 6:

        salvarDados(scanner, repoFisica, repoJuridica);

        break;

    case 7:

        recuperarDados(scanner, repoFisica, repoJuridica);

        break;

    case 0:

        System.out.println("Encerrando o programa...");

        break;

    default:

        System.out.println("Opção inválida. Tente novamente.");

    }

}

scanner.close();

}

```

```

private static void incluir(Scanner scanner, PessoaFisicaRepo repoFisica,
PessoaJuridicaRepo repoJuridica) {

```

```
System.out.print("Incluir Pessoa Física ou Jurídica? (F/J): ");

String tipo = scanner.nextLine().toUpperCase();

if (tipo.equals("F")) {

    System.out.print("ID: ");

    int id = scanner.nextInt();

    scanner.nextLine(); // Limpa o buffer

    System.out.print("Nome: ");

    String nome = scanner.nextLine();

    System.out.print("CPF: ");

    String cpf = scanner.nextLine();

    System.out.print("Idade: ");

    int idade = scanner.nextInt();

    PessoaFisica pessoaFisica = new PessoaFisica(id, nome, cpf, idade);

    repoFisica.inserir(pessoaFisica);

} else if (tipo.equals("J")) {

    System.out.print("ID: ");

    int id = scanner.nextInt();

    scanner.nextLine(); // Limpa o buffer

    System.out.print("Nome: ");

    String nome = scanner.nextLine();

    System.out.print("CNPJ: ");

    String cnpj = scanner.nextLine();

    PessoaJuridica pessoaJuridica = new PessoaJuridica(id, nome, cnpj);

    repoJuridica.inserir(pessoaJuridica);

} else {
```

```
        System.out.println("Tipo inválido. Tente novamente.");  
    }  
}
```

```
private static void alterar(Scanner scanner, PessoaFisicaRepo repoFisica,  
PessoaJuridicaRepo repoJuridica) {
```

```
    System.out.print("Alterar Pessoa Física ou Jurídica? (F/J): ");
```

```
    String tipo = scanner.nextLine().toUpperCase();
```

```
    if (tipo.equals("F")) {
```

```
        System.out.print("ID: ");
```

```
        int id = scanner.nextInt();
```

```
        scanner.nextLine(); // Limpa o buffer
```

```
        PessoaFisica pessoaFisica = repoFisica.obter(id);
```

```
        if (pessoaFisica != null) {
```

```
            System.out.println("Dados atuais: ");
```

```
            pessoaFisica.exibir();
```

```
            System.out.print("Novo nome: ");
```

```
            String nome = scanner.nextLine();
```

```
            System.out.print("Novo CPF: ");
```

```
            String cpf = scanner.nextLine();
```

```
            System.out.print("Nova idade: ");
```

```
            int idade = scanner.nextInt();
```

```
            scanner.nextLine(); // Limpa o buffer
```

```
            pessoaFisica.setNome(nome);
```

```
            pessoaFisica.setCpf(cpf);
```

```
        pessoaFisica.setIdade(idade);

        repoFisica.alterar(pessoaFisica);

    } else {

        System.out.println("Pessoa Física não encontrada.");

    }

} else if (tipo.equals("J")) {

    System.out.print("ID: ");

    int id = scanner.nextInt();

    scanner.nextLine(); // Limpa o buffer

    PessoaJuridica pessoaJuridica = repoJuridica.obter(id);

    if (pessoaJuridica != null) {

        System.out.println("Dados atuais: ");

        pessoaJuridica.exibir();

        System.out.print("Novo nome: ");

        String nome = scanner.nextLine();

        System.out.print("Novo CNPJ: ");

        String cnpj = scanner.nextLine();

        pessoaJuridica.setNome(nome);

        pessoaJuridica.setCnpj(cnpj);

        repoJuridica.alterar(pessoaJuridica);

    } else {

        System.out.println("Pessoa Jurídica não encontrada.");

    }

} else {

    System.out.println("Tipo inválido. Tente novamente.");

}
```

```
}  
}
```

```
private static void excluir(Scanner scanner, PessoaFisicaRepo repoFisica,  
PessoaJuridicaRepo repoJuridica) {
```

```
    System.out.print("Excluir Pessoa Física ou Jurídica? (F/J): ");
```

```
    String tipo = scanner.nextLine().toUpperCase();
```

```
    System.out.print("ID: ");
```

```
    int id = scanner.nextInt();
```

```
    scanner.nextLine(); // Limpa o buffer
```

```
    if (tipo.equals("F")) {
```

```
        repoFisica.excluir(id);
```

```
    } else if (tipo.equals("J")) {
```

```
        repoJuridica.excluir(id);
```

```
    } else {
```

```
        System.out.println("Tipo inválido. Tente novamente.");
```

```
    }
```

```
}
```

```
private static void exibirPorId(Scanner scanner, PessoaFisicaRepo repoFisica,  
PessoaJuridicaRepo repoJuridica) {
```

```
    System.out.print("Exibir Pessoa Física ou Jurídica? (F/J): ");
```

```
    String tipo = scanner.nextLine().toUpperCase();
```

```
    System.out.print("ID: ");
```

```
    int id = scanner.nextInt();
```

```
    scanner.nextLine(); // Limpa o buffer
```

```

if (tipo.equals("F")) {

    PessoaFisica pessoaFisica = repoFisica.obter(id);

    if (pessoaFisica != null) {

        pessoaFisica.exibir();

    } else {

        System.out.println("Pessoa Física não encontrada.");

    }

} else if (tipo.equals("J")) {

    PessoaJuridica pessoaJuridica = repoJuridica.obter(id);

    if (pessoaJuridica != null) {

        pessoaJuridica.exibir();

    } else {

        System.out.println("Pessoa Jurídica não encontrada.");

    }

} else {

    System.out.println("Tipo inválido. Tente novamente.");

}

}

```

```

private static void exibirTodos(Scanner scanner, PessoaFisicaRepo repoFisica,
PessoaJuridicaRepo repoJuridica) {

    System.out.print("Exibir todas as Pessoas Físicas ou Jurídicas? (F/J): ");

    String tipo = scanner.nextLine().toUpperCase();

    if (tipo.equals("F")) {

        for (PessoaFisica pessoaFisica : repoFisica.obterTodos()) {

```



```

        pessoaFisica.exibir();
    }
} else if (tipo.equals("J")) {
    for (PessoaJuridica pessoaJuridica : repoJuridica.obterTodos()) {
        pessoaJuridica.exibir();
    }
} else {
    System.out.println("Tipo inválido. Tente novamente.");
}
}

```

```

private static void salvarDados(Scanner scanner, PessoaFisicaRepo repoFisica,
PessoaJuridicaRepo repoJuridica) {
    System.out.print("Informe 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.print("Informe o prefixo dos arquivos: ");

String prefixo = scanner.nextLine();

try {

    repoFisica.recuperar(prefixo + ".fisica.bin");

    repoJuridica.recuperar(prefixo + ".juridica.bin");

    System.out.println("Dados recuperados com sucesso.");

} catch (IOException | ClassNotFoundException e) {

    System.out.println("Erro ao recuperar dados: " + e.getMessage());

}

}

}

```

Conclusão:

- a. Quais as vantagens e desvantagens do uso de herança?

Uma das grandes vantagens na utilização da herança é a possibilidade de reutilização do código, sendo uma desvantagem deixar o código complexo e engessado, o Java também não aceita herança múltipla da classe superior, apenas da classe “filha”.

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

É necessária para gravação de dados binários em formato que possa ser gravada em disco e posterior retorno para leitura do objeto.

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

É utilizado de forma a garantir a imutabilidade dos dados e para gerenciamento de coleções, garantindo uma melhora no controle de fluxo de dados.

- d. Quando trabalhamos com Java, qual padrão de desenvolvimento é adotado

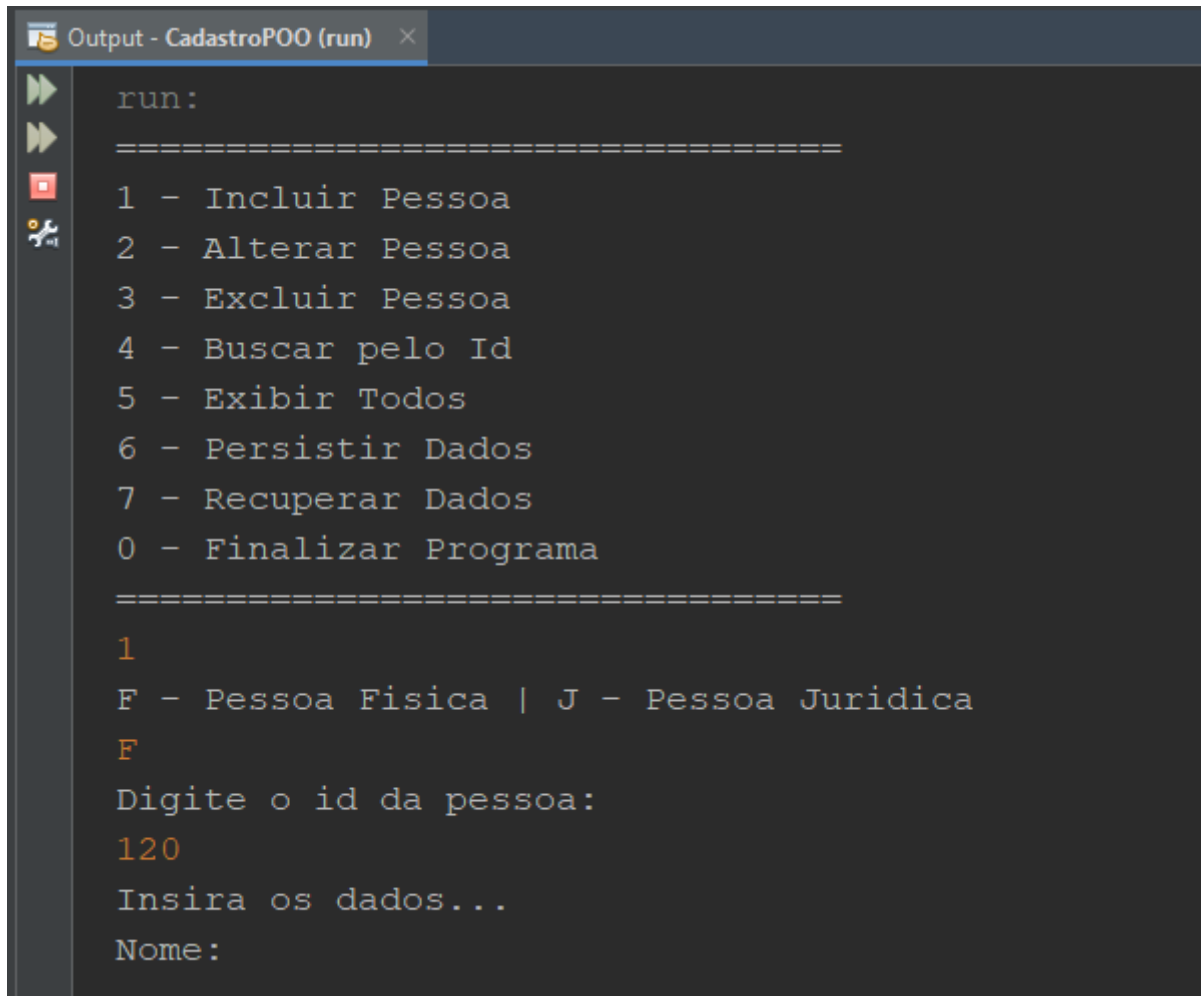
na persistência de dados em arquivos?

O Java Persistence API (JPA), permite a persistência, atualização, recuperação ou remoção de objetos, o padrão query é utilizado.

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

1. Alterar o método main da classe principal do projeto, para implementação do cadastro em modo texto:
  - a. Apresentar as opções do programa para o usuário, sendo 1 para incluir, 2 para alterar, 3 para excluir, 4 para exibir pelo id, 5 para exibir todos, 6 para salvar dados, 7 para recuperar dados e 0 para finalizar a execução.
  - b. Selecionada a opção incluir, escolher o tipo (Física ou Jurídica), receber os dados a partir do teclado e adicionar no repositório correto.
  - c. Selecionada a opção alterar, escolher o tipo (Física ou Jurídica), receber o id a partir do teclado, apresentar os dados atuais, solicitar os novos dados e alterar no repositório correto.
  - d. Selecionada a opção excluir, escolher o tipo (Física ou Jurídica), receber o id a partir do teclado e remover do repositório correto.
  - e. Selecionada a opção obter, escolher o tipo (Física ou Jurídica), receber o id a partir do teclado e apresentar os dados atuais para a entidade.
  - f. Selecionada a opção obterTodos, escolher o tipo (Física ou Jurídica) e apresentar os dados de todas as entidades do repositório correto.
  - g. Selecionada a opção salvar, solicitar o prefixo dos arquivos e persistir os dados nos arquivos [prefixo].fisica.bin e [prefixo].juridica.bin.
  - h. Selecionada a opção recuperar, solicitar o prefixo dos arquivos e obter os dados a partir dos arquivos [prefixo].fisica.bin e [prefixo].juridica.bin.
  - i. Nas opções salvar e recuperar devem ser tratadas as exceções.
  - j. Selecionada a opção sair, finalizar a execução do sistema.

2. Ajustar as características para obter uma execução como a seguinte.



```
Output - CadastroPOO (run) x
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
=====
1
F - Pessoa Fisica | J - Pessoa Juridica
F
Digite o id da pessoa:
120
Insira os dados...
Nome:
```

```
package com.mycompany.cadastropoo;
```

```
/**
```

```
*
```

```
* @author Thomas
```

```
*/
```

```
import model.PessoaFisica;
```

```
import model.PessoaFisicaRepo;

import model.PessoaJuridica;

import model.PessoaJuridicaRepo;


import java.io.IOException;

import java.util.Scanner;


public class CadastroPOO {

    public static void main(String[] args) {

        PessoaFisicaRepo repoFisica = new PessoaFisicaRepo();

        PessoaJuridicaRepo repoJuridica = new PessoaJuridicaRepo();

        Scanner scanner = new Scanner(System.in);


        int opcao = -1;

        while (opcao != 0) {

            // Exibe o menu de opções

            System.out.println("=== MENU ===");

            System.out.println("1 - Incluir");

            System.out.println("2 - Alterar");

            System.out.println("3 - Excluir");

            System.out.println("4 - Exibir por ID");

            System.out.println("5 - Exibir todos");

            System.out.println("6 - Salvar dados");

            System.out.println("7 - Recuperar dados");
```

```
System.out.println("0 - Sair");

System.out.print("Escolha uma opção: ");

opcao = scanner.nextInt();

scanner.nextLine(); // Limpa o buffer


switch (opcao) {

    case 1:

        incluir(scanner, repoFisica, repoJuridica);

        break;

    case 2:

        alterar(scanner, repoFisica, repoJuridica);

        break;

    case 3:

        excluir(scanner, repoFisica, repoJuridica);

        break;

    case 4:

        exibirPorId(scanner, repoFisica, repoJuridica);

        break;

    case 5:

        exibirTodos(scanner, repoFisica, repoJuridica);

        break;

    case 6:

        salvarDados(scanner, repoFisica, repoJuridica);

        break;

    case 7:
```

```

        recuperarDados(scanner, repoFisica, repoJuridica);

        break;

    case 0:

        System.out.println("Encerrando o programa...");

        break;

    default:

        System.out.println("Opção inválida. Tente novamente.");

    }

}

scanner.close();

}

```

```

private static void incluir(Scanner scanner, PessoaFisicaRepo repoFisica,
PessoaJuridicaRepo repoJuridica) {

```

```

    System.out.print("Incluir Pessoa Física ou Jurídica? (F/J): ");

```

```

    String tipo = scanner.nextLine().toUpperCase();

```

```

    if (tipo.equals("F")) {

```

```

        System.out.print("ID: ");

```

```

        int id = scanner.nextInt();

```

```

        scanner.nextLine(); // Limpa o buffer

```

```

        System.out.print("Nome: ");

```

```

        String nome = scanner.nextLine();

```

```

        System.out.print("CPF: ");

```

```

        String cpf = scanner.nextLine();

```

```

        System.out.print("Idade: ");

```

```

        int idade = scanner.nextInt();

        PessoaFisica pessoaFisica = new PessoaFisica(id, nome, cpf, idade);

        repoFisica.inserir(pessoaFisica);
    } else if (tipo.equals("J")) {

        System.out.print("ID: ");

        int id = scanner.nextInt();

        scanner.nextLine(); // Limpa o buffer

        System.out.print("Nome: ");

        String nome = scanner.nextLine();

        System.out.print("CNPJ: ");

        String cnpj = scanner.nextLine();

        PessoaJuridica pessoaJuridica = new PessoaJuridica(id, nome, cnpj);

        repoJuridica.inserir(pessoaJuridica);
    } else {

        System.out.println("Tipo inválido. Tente novamente.");

    }
}

```

```

private static void alterar(Scanner scanner, PessoaFisicaRepo repoFisica,
PessoaJuridicaRepo repoJuridica) {

    System.out.print("Alterar Pessoa Física ou Jurídica? (F/J): ");

    String tipo = scanner.nextLine().toUpperCase();

    if (tipo.equals("F")) {

        System.out.print("ID: ");

        int id = scanner.nextInt();
    }
}

```



```

scanner.nextLine(); // Limpa o buffer

PessoaFisica pessoaFisica = repoFisica.obter(id);

if (pessoaFisica != null) {

    System.out.println("Dados atuais: ");

    pessoaFisica.exibir();

    System.out.print("Novo nome: ");

    String nome = scanner.nextLine();

    System.out.print("Novo CPF: ");

    String cpf = scanner.nextLine();

    System.out.print("Nova idade: ");

    int idade = scanner.nextInt();

    scanner.nextLine(); // Limpa o buffer

    pessoaFisica.setNome(nome);

    pessoaFisica.setCpf(cpf);

    pessoaFisica.setIdade(idade);

    repoFisica.alterar(pessoaFisica);

} else {

    System.out.println("Pessoa Física não encontrada.");

}

} else if (tipo.equals("J")) {

    System.out.print("ID: ");

    int id = scanner.nextInt();

    scanner.nextLine(); // Limpa o buffer

    PessoaJuridica pessoaJuridica = repoJuridica.obter(id);

    if (pessoaJuridica != null) {

```

```

        System.out.println("Dados atuais: ");

        pessoaJuridica.exibir();

        System.out.print("Novo nome: ");

        String nome = scanner.nextLine();

        System.out.print("Novo CNPJ: ");

        String cnpj = scanner.nextLine();

        pessoaJuridica.setNome(nome);

        pessoaJuridica.setCnpj(cnpj);

        repoJuridica.alterar(pessoaJuridica);

    } else {

        System.out.println("Pessoa Jurídica não encontrada.");

    }

} else {

    System.out.println("Tipo inválido. Tente novamente.");

}

}

```

```

private static void excluir(Scanner scanner, PessoaFisicaRepo repoFisica,
PessoaJuridicaRepo repoJuridica) {

    System.out.print("Excluir Pessoa Física ou Jurídica? (F/J): ");

    String tipo = scanner.nextLine().toUpperCase();

    System.out.print("ID: ");

    int id = scanner.nextInt();

    scanner.nextLine(); // Limpa o buffer

    if (tipo.equals("F")) {

```

```

        repoFisica.excluir(id);
    } else if (tipo.equals("J")) {
        repoJuridica.excluir(id);
    } else {
        System.out.println("Tipo inválido. Tente novamente.");
    }
}

```

```

private static void exibirPorId(Scanner scanner, PessoaFisicaRepo repoFisica,
PessoaJuridicaRepo repoJuridica) {

```

```

    System.out.print("Exibir Pessoa Física ou Jurídica? (F/J): ");

```

```

    String tipo = scanner.nextLine().toUpperCase();

```

```

    System.out.print("ID: ");

```

```

    int id = scanner.nextInt();

```

```

    scanner.nextLine(); // Limpa o buffer

```

```

    if (tipo.equals("F")) {

```

```

        PessoaFisica pessoaFisica = repoFisica.obter(id);

```

```

        if (pessoaFisica != null) {

```

```

            pessoaFisica.exibir();

```

```

        } else {

```

```

            System.out.println("Pessoa Física não encontrada.");

```

```

        }

```

```

    } else if (tipo.equals("J")) {

```

```

        PessoaJuridica pessoaJuridica = repoJuridica.obter(id);

```

```

        if (pessoaJuridica != null) {

```

```

        pessoaJuridica.exibir();

    } else {

        System.out.println("Pessoa Jurídica não encontrada.");

    }

    } else {

        System.out.println("Tipo inválido. Tente novamente.");

    }

}

```

```

private static void exibirTodos(Scanner scanner, PessoaFisicaRepo repoFisica,
PessoaJuridicaRepo repoJuridica) {

    System.out.print("Exibir todas as Pessoas Físicas ou Jurídicas? (F/J): ");

    String tipo = scanner.nextLine().toUpperCase();

    if (tipo.equals("F")) {

        for (PessoaFisica pessoaFisica : repoFisica.obterTodos()) {

            pessoaFisica.exibir();

        }

    } else if (tipo.equals("J")) {

        for (PessoaJuridica pessoaJuridica : repoJuridica.obterTodos()) {

            pessoaJuridica.exibir();

        }

    } else {

        System.out.println("Tipo inválido. Tente novamente.");

    }

}

```

```

private static void salvarDados(Scanner scanner, PessoaFisicaRepo repoFisica,
PessoaJuridicaRepo repoJuridica) {

    System.out.print("Informe 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.print("Informe o prefixo dos arquivos: ");

    String prefixo = scanner.nextLine();

    try {

        repoFisica.recuperar(prefixo + ".fisica.bin");

        repoJuridica.recuperar(prefixo + ".juridica.bin");

        System.out.println("Dados recuperados com sucesso.");

    } catch (IOException | ClassNotFoundException e) {

        System.out.println("Erro ao recuperar dados: " + e.getMessage());

    }

}

```

}

Resultado como esperado:

