

CADASTRO POO EM JAVA – DOCUMENTAÇÃO DO 1º PROCEDIMENTO

Objetivo da Prática: Realizar um CRUD em Java com Persistência de Dados

CÓDIGOS SOLICITADOS

Classe Main

```
package cadastrapoo;
import model.PessoaFisica;
import model.PessoaFisicaRepo;
import model.PessoaJuridica;
import model.PessoaJuridicaRepo;

public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        PessoaFisicaRepo repol = new PessoaFisicaRepo();

        PessoaFisica pfl = new PessoaFisica("Henrique", 1, 20, "156.895.666-59");
        PessoaFisica pf2 = new PessoaFisica("Maria", 2, 76, "756.666.666-59");

        repol.inserir(pfl);
        repol.inserir(pf2);
        repol.persistir("pessoasfisicas");

        PessoaFisicaRepo repo2 = new PessoaFisicaRepo();

        repo2.recuperar("pessoasfisicas");
        repo2.obterTodos();

        PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();

        PessoaJuridica pj1 = new PessoaJuridica("Jaime", 3, "7361723");
        PessoaJuridica pj2 = new PessoaJuridica("Vanessa", 4, "545645");

        repo3.inserir(pj1);
        repo3.inserir(pj2);
        repo3.persistir("pessoasjuridicas");

        PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();

        repo4.recuperar("pessoasjuridicas");
        repo4.obterTodos();
```

Classe Pessoa

```
package model;
import java.io.Serializable;

public class Pessoa implements Serializable{

    private int id;
    private String nome;

    public Pessoa(String nome, int id)
    {
        this.nome = nome;
        this.id = id;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public void exibir()
    {
        System.out.println("NOME: " + this.nome + "\nID: " + this.id);
    }
}
```

Classe PessoaFisica

```
package model;
import java.io.Serializable;

public class PessoaFisica extends Pessoa implements Serializable{

    private int idade;
    private String cpf;

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

    public int getIdade() {
        return idade;
    }

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

    public String getCpf() {
        return cpf;
    }

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

    @Override
    public void exibir()
    {
        System.out.println("NOME: " + super.getNome() + "\nID: " + super.getId());
        System.out.println("CPF: " + this.cpf + "\nIDADE: " + this.idade);
    }
}
```

Classe PessoaFisicaRepo

```
package model;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

public class PessoaFisicaRepo {

    private List<PessoaFisica> pessoas;

    public PessoaFisicaRepo() {
        pessoas = new ArrayList<>();
    }

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

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

    public void excluir(int id) {
        Iterator<PessoaFisica> iterator = pessoas.iterator();
        while (iterator.hasNext()) {
            PessoaFisica pessoa = iterator.next();
            if (pessoa.getId() == id) {
                iterator.remove();
                break;
            }
        }
    }
}
```

```

public PessoaFisica obter(int id) {
    for (PessoaFisica pessoa : pessoas) {
        if (pessoa.getId() == id) {
            System.out.println("Nome: " + pessoa.getNome());
            System.out.println("CPF: " + pessoa.getCpf());
            System.out.println("Idade: " + pessoa.getIdade());
            System.out.println("ID: " + pessoa.getId());
            System.out.println();
        }
    }
    return null; // Retorna null se não encontrar
}

public void obterTodos() {
    for (PessoaFisica pessoa : pessoas) {
        System.out.println("Nome: " + pessoa.getNome());
        System.out.println("CPF: " + pessoa.getCpf());
        System.out.println("Idade: " + pessoa.getIdade());
        System.out.println("ID: " + pessoa.getId());
        System.out.println();
    }
}

public void persistir(String nomeArquivo) {
    try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
        oos.writeObject(pessoas);
        System.out.println("Dados persistidos no arquivo " + nomeArquivo);
    } catch (IOException e) {
        System.err.println("Erro ao persistir os dados: " + e.getMessage());
    }
}

public void recuperar(String nomeArquivo) {
    try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
        pessoas = (List<PessoaFisica>) ois.readObject();
        System.out.println("Dados recuperados do arquivo " + nomeArquivo);
    } catch (IOException | ClassNotFoundException e) {
        System.err.println("Erro ao recuperar os dados: " + e.getMessage());
    }
}

```

Classe PessoaJuridica

```
package model;
import java.io.Serializable;

public class PessoaJuridica extends Pessoa implements Serializable{

    private String cnpj;

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

    public String getCnpj() {
        return cnpj;
    }

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

    @Override
    public void exibir()
    {
        System.out.println("NOME: " + super.getNome() + "\nID: " + super.getId());
        System.out.println("CNPJ: " + this.cnpj);
    }

}
```

Classe PessoaJuridicaRepo

```
package model;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

public class PessoaJuridicaRepo {

    private List<PessoaJuridica> pessoas;

    public PessoaJuridicaRepo() {
        pessoas = new ArrayList<>();
    }

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

    public void alterar(int id, PessoaJuridica pessoa) {
        for (int i = 0; i < pessoas.size(); i++) {
            if (pessoas.get(i).getId() == id) {
                pessoas.set(i, pessoa);
                break;
            }
        }
    }

    public void excluir(int id) {
        Iterator<PessoaJuridica> iterator = pessoas.iterator();
        while (iterator.hasNext()) {
            PessoaJuridica pessoa = iterator.next();
            if (pessoa.getId() == id) {
                iterator.remove();
                break;
            }
        }
    }
}
```

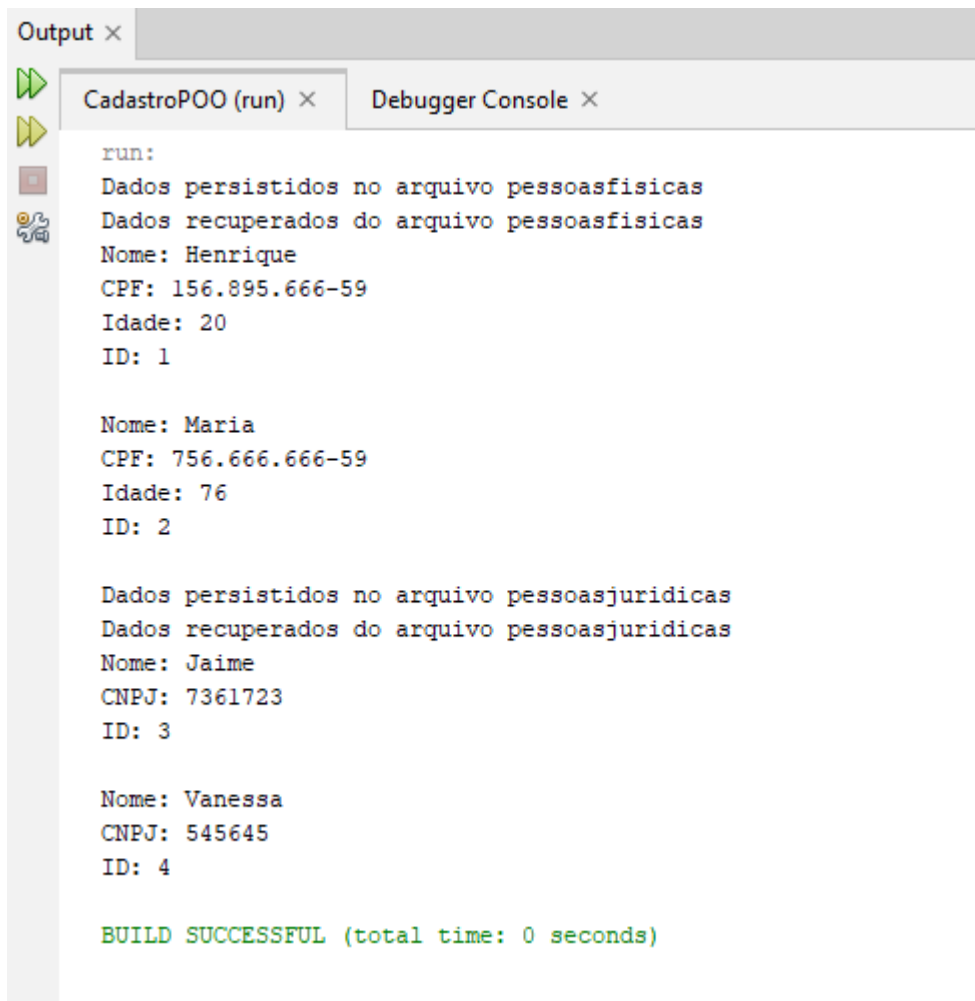
```
public PessoaJuridica obter(int id) {
    for (PessoaJuridica pessoa : pessoas) {
        if (pessoa.getId() == id) {
            System.out.println("Nome: " + pessoa.getNome());
            System.out.println("CNPJ: " + pessoa.getCnpj());
            System.out.println("ID: " + pessoa.getId());
            System.out.println();
        }
    }
    return null; // Retorna null se não encontrar
}

public void obterTodos() {
    for (PessoaJuridica pessoa : pessoas) {
        System.out.println("Nome: " + pessoa.getNome());
        System.out.println("CNPJ: " + pessoa.getCnpj());
        System.out.println("ID: " + pessoa.getId());
        System.out.println();
    }
}

public void persistir(String nomeArquivo) {
    try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
        oos.writeObject(pessoas);
        System.out.println("Dados persistidos no arquivo " + nomeArquivo);
    } catch (IOException e) {
        System.err.println("Erro ao persistir os dados: " + e.getMessage());
    }
}

public void recuperar(String nomeArquivo) {
    try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
        pessoas = (List<PessoaJuridica>) ois.readObject();
        System.out.println("Dados recuperados do arquivo " + nomeArquivo);
    } catch (IOException | ClassNotFoundException e) {
        System.err.println("Erro ao recuperar os dados: " + e.getMessage());
    }
}
}
```


RESULTADOS DE EXECUÇÃO



```
run:
Dados persistidos no arquivo pessoasfisicas
Dados recuperados do arquivo pessoasfisicas
Nome: Henrique
CPF: 156.895.666-59
Idade: 20
ID: 1

Nome: Maria
CPF: 756.666.666-59
Idade: 76
ID: 2

Dados persistidos no arquivo pessoasjuridicas
Dados recuperados do arquivo pessoasjuridicas
Nome: Jaime
CNPJ: 7361723
ID: 3

Nome: Vanessa
CNPJ: 545645
ID: 4

BUILD SUCCESSFUL (total time: 0 seconds)
```

ANÁLISE E CONCLUSÃO

1. Quais as vantagens e desvantagens do uso da herança?

Vantagens: Reutilização de código, Organização de código e Extensibilidade.

Desvantagens: Acoplamento, Complexidade, Rigidez.

2. Por que a interface Serializable é necessária ao efetuar persistência em arquivos binários?
Serializable é essencial para permitir que objetos Java sejam persistidos em arquivos binários e recuperados posteriormente, garantindo a integridade dos dados.

3. Como o paradigma funcional é utilizado pela API stream no Java?
Em resumo, a API Stream no Java simplifica a manipulação de coleções, reduz o tamanho do código e segue os princípios da programação funcional.

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

A JPA é uma ferramenta poderosa para a persistência de dados em Java, tornando o desenvolvimento mais eficiente e abstraindo a complexidade das operações de banco de dados.