

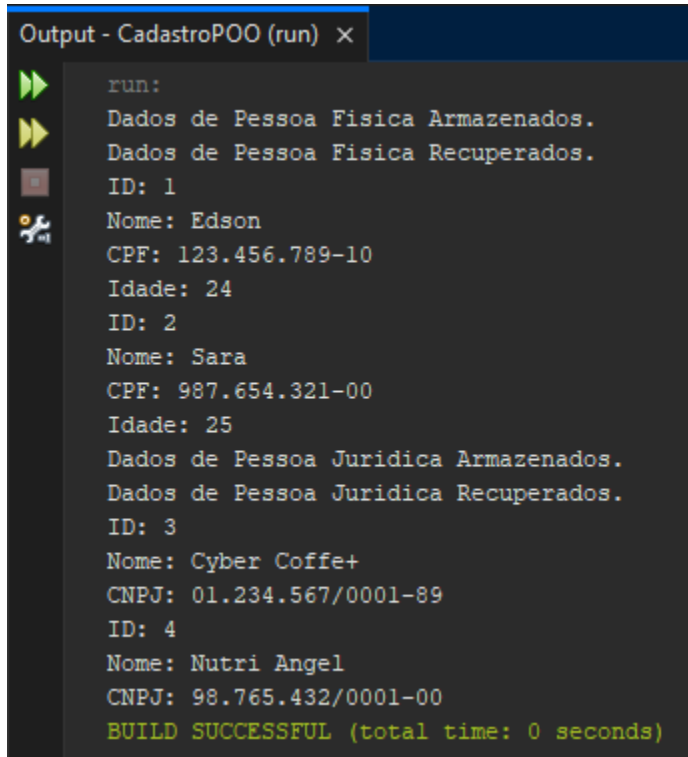


Campus:	Polo Jardim Goiás	Curso:	Desenvolvimento Full Stack
Disciplina:	RPG0014 - Iniciando o Caminho Pelo Java		
Semestre Letivo:	2024.3	Turma:	9001
Aluno:	Edson Luiz Pacheco Junior		
Repositório GIT:	EdsonJr73/cadastropoo (github.com)		

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

1. Título da Prática: “1º Procedimento | Criação das Entidades e Sistema de Persistência.
2. Objetivos da Prática:
 - a. Utilizar herança e polimorfismo na definição de entidades.
 - b. Utilizar persistência de objetos em arquivos binários.
 - c. Implementar uma interface cadastral em modo texto.
 - d. Utilizar o controle de exceções da plataforma Java.
 - e. Implementar um sistema cadastral em Java, utilizando os recursos da programação orientada a objetos e a persistência em arquivos binários.
3. Códigos solicitados nesta aula: Estarão no final do relatório.

4. Resultado da Execução do Código:



```
Output - CadastroPOO (run) x
run:
Dados de Pessoa Fisica Armazenados.
Dados de Pessoa Fisica Recuperados.
ID: 1
Nome: Edson
CPF: 123.456.789-10
Idade: 24
ID: 2
Nome: Sara
CPF: 987.654.321-00
Idade: 25
Dados de Pessoa Juridica Armazenados.
Dados de Pessoa Juridica Recuperados.
ID: 3
Nome: Cyber Coffe+
CNPJ: 01.234.567/0001-89
ID: 4
Nome: Nutri Angel
CNPJ: 98.765.432/0001-00
BUILD SUCCESSFUL (total time: 0 seconds)
```

5. Análise e Conclusão:

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

Vantagens: Reutilização de código; manutenção facilitada pois alterando código na classe pai ele será refletido nas filhas sem a necessidade de ter que ir em cada uma para alterar; compatibilidade entre tipos, o que permite tratar as classes filhas como se fossem a classe pai, fazendo com que seja possível criar códigos mais genéricos.

Desvantagens: Forte acoplamento sujeitando o código a possíveis problemas nas classes filhas caso haja alterações na classe pai; dificuldade em alterar a hierarquia; excesso de herança pode gerar um código confuso; fraco encapsulamento, as classes filhas podem alterar atributos ou métodos a depender de como foi declarado o modificador. O que pode acarretar comportamentos inesperados e bugs difíceis de rastrear.

- b. Por que a interface Serializable é necessária ao efetuar persistência em arquivos binários?
Esta interface serve para dizer ao java que o objeto pode ser transformado em bytes, o que é importante quando há a intenção de armazenar o objeto num arquivo. Sem ela, o java não saberá transformá-lo em bytes, o que fará com que não dê para salvar o estado e nem recuperar o objeto futuramente.
- c. Como o paradigma funcional é utilizado pela API stream no Java?
Streams não modificam o arquivo original e sim criam novos resultados com base nos dados processados. Poder utilizar funções como parâmetros e a possibilidade de encadear operações (filter, map, sorted).
- d. Quando trabalhamos com Java, qual padrão de desenvolvimento é adotado na persistência de dados em arquivos?
O padrão DAO (Data Access Object).

Códigos:

1. **Pessoa.java:**

```
package model;

import java.io.Serializable;

/**
 *
 * @author edson-202308892185
 */
public class Pessoa implements Serializable {

    private static final long serialVersionUID = 1L;

    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 + "\nNome: " + nome);
    }
}

```

2. **PessoaFisica.java:**

```

package model;

/**
 *
 * @author edson-202308892185
 */
public class PessoaFisica extends Pessoa {

    String cpf;

```

```

    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() {
        super.exibir();
        System.out.println("CPF: " + cpf + "\nIdade: " + idade);
    }
}

```

3. **PessoaJuridica.java:**

```
package model;
```

```
/**
```

```

*
* @author edson-202308892185
*/
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() {
        super.exibir();
        System.out.println("CNPJ: " + cnpj);
    }
}

```

4. **PessoaFisicaRepo.java:**

```

package model;

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

/**
 *

```

```

* @author edson-202308892185
*/
public class PessoaFisicaRepo {

    private ArrayList<PessoaFisica> listaPessoaFisica = new ArrayList<>();

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

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

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

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

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

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

```

```

    }
}

    public void recuperar(String nomeArquivo) throws IOException,
ClassNotFoundException {
    try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream(nomeArquivo))) {
        listaPessoaFisica = (ArrayList<PessoaFisica>) ois.readObject();
    }
}
}

```

5. **PessoaJuridicaRepo.java:**

```

package model;

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

/**
 *
 * @author edson-202308892185
 */
public class PessoaJuridicaRepo {

    private ArrayList<PessoaJuridica> listaPessoaJuridica = new ArrayList<>();

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

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

```



```

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

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

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

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

public void recuperar(String nomeArquivo) throws IOException,
    ClassNotFoundException {
    try (ObjectInputStream ois = new ObjectInputStream(new
        FileInputStream(nomeArquivo))) {
        listaPessoaJuridica = (ArrayList<PessoaJuridica>) ois.readObject();
    }
}

```

6. **CadastroPOO.java (main):**

```
package cadastropoo;
```

```
import model.*;
import java.io.*;
```

```

/**
 *
 * @author edson-202308892185
 */
public class CadastroPOO {

    public static void main(String[] args) {
        PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
        PessoaFisica pessoaFisica1 = new PessoaFisica(1, "Edson", "123.456.789-10",
24);
        PessoaFisica pessoaFisica2 = new PessoaFisica(2, "Sara", "987.654.321-00",
25);
        repo1.inserir(pessoaFisica1);
        repo1.inserir(pessoaFisica2);
        String arquivoPessoaFisica = "pessoasFisicas.bin";
        try {
            repo1.persistir(arquivoPessoaFisica);
            System.out.println("Dados de Pessoa Fisica Armazenados.");
        } catch (IOException e) {
            System.out.println("Erro ao salvar pessoas fisicas: " + e.getMessage());
        }

        PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
        try {
            repo2.recuperar(arquivoPessoaFisica);
            System.out.println("Dados de Pessoa Fisica Recuperados.");
        } catch (IOException | ClassNotFoundException e) {
            System.out.println("Erro ao recuperar pessoas fisicas: " + e.getMessage());
        }

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

        PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
        PessoaJuridica pessoaJuridica1 = new PessoaJuridica(3, "Cyber Coffe+",
"01.234.567/0001-89");
    }
}

```

```

        PessoaJuridica pessoaJuridica2 = new PessoaJuridica(4, "Nutri Angel",
"98.765.432/0001-00");
        repo3.inserir(pessoaJuridica1);
        repo3.inserir(pessoaJuridica2);

        String arquivoPessoaJuridica = "pessoasJuridicas.bin";
        try {
            repo3.persistir(arquivoPessoaJuridica);
            System.out.println("Dados de Pessoa Juridica Armazenados.");
        } catch (IOException e) {
            System.out.println("Erro ao salvar pessoas juridicas: " + e.getMessage());
        }

        PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
        try {
            repo4.recuperar(arquivoPessoaJuridica);
            System.out.println("Dados de Pessoa Juridica Recuperados.");
        } catch (IOException | ClassNotFoundException e) {
            System.out.println("Erro ao recuperar pessoas juridicas: " + e.getMessage());
        }

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

    }

}

```