UNIVERSIDADE ESTÁCIO DE SÁ

DESENVOLVIMENTO FULLSTACK

Mundo 03 - Nível 01

RPG0014 - Iniciando o caminho pelo Java

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

Ruan Hernandes Finamor Correia

Matrícula 202208175252

Gravataí – RS

2023

Objetivo da Prática

1. Utilizar herança e polimorfismo na definição de entidades.

2. Utilizar persistência de objetos em arquivos binários.

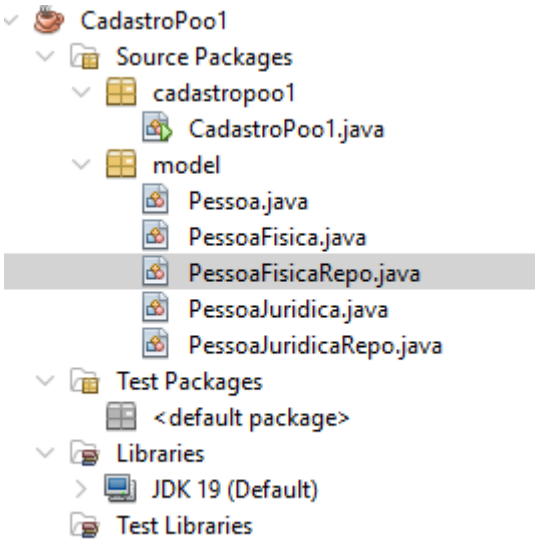
3. Implementar uma interface cadastral em modo texto.

4. Utilizar o controle de exceções da plataforma Java.

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

1º Procedimento

Códigos solicitados no roteiro desta aula



Pessoa.java | PessoaFisica.java | PessoaJuridica.java | CadastroPoo1.java | PessoaFisicaRepo.java | PessoaJuridicaRepo.java

Source | History

```java
/**
 *
 * @author ruanf
 */
import java.util.Scanner;
import model.PessoaFisica;
import model.PessoaFisicaRepo;
import model.PessoaJuridica;
import model.PessoaJuridicaRepo;

public class CadastroPoo1 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(source: System.in);
        PessoaFisicaRepo repoFisica = new PessoaFisicaRepo();
        PessoaJuridicaRepo repoJuridica = new PessoaJuridicaRepo();

        while (true) {
            System.out.println(x: "\nEscolha uma opção:");
            System.out.println(x: "######################");
            System.out.println(x: "1 - Incluir Pessoa");
            System.out.println(x: "2 - Alterar Pessoa");
            System.out.println(x: "3 - Excluir Pessoa");
            System.out.println(x: "4 - Buscar pelo ID");
            System.out.println(x: "5 - Exibir todos");
            System.out.println(x: "6 - Persistir dados");
            System.out.println(x: "7 - Recuperar dados");
            System.out.println(x: "0 - Finalizar Programa");
            System.out.println(x: "######################");


            int opcao = scanner.nextInt();
            scanner.nextLine();

            switch (opcao) {
                case 1:
                    /* Incluir Pessoa no Sistema*/
```

Pessoa.java | PessoaFisica.java | PessoaJuridica.java | CadastroPoo1.java | PessoaFisicaRe

Source | History

```java

/**
 *
 * @author ruanf
 */
package model;

import java.io.Serializable;

public class Pessoa implements Serializable {
    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;
    }
```

Pessoa.java | PessoaFisica.java | PessoaJuridica.java | CadastroPoo1.java | PessoaFisicaRepo.java

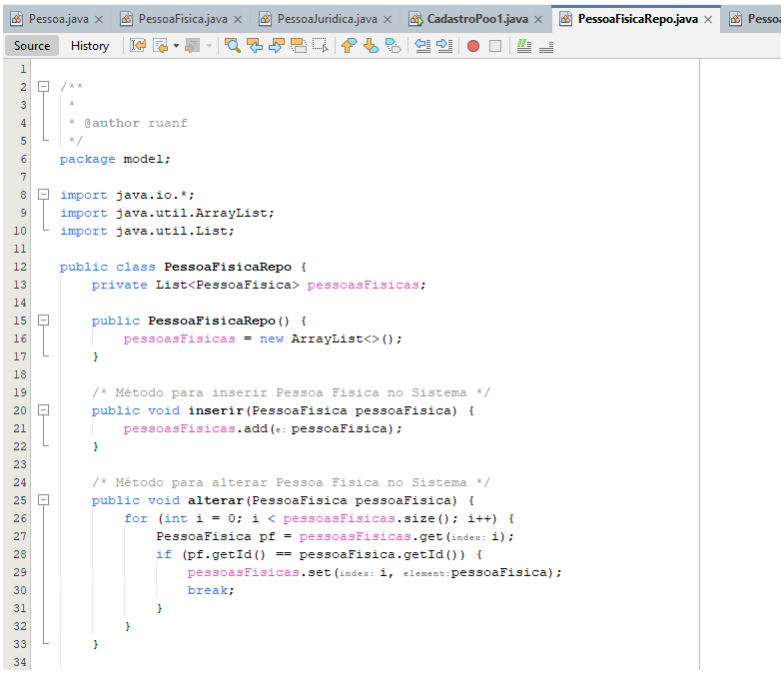
Source | History

```java
/**
 *
 * @author ruanf
 */
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;
    }
```

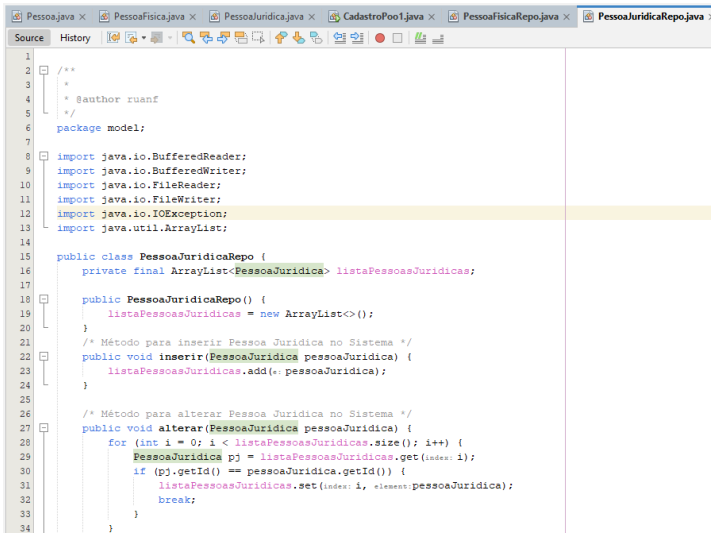
Pessoa.java | PessoaFisica.java | PessoaJuridica.java | CadastroPoo1.java | PessoaFisicaRe

Source | History

```java
 *
 * @author ruanf
 */
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() {
        super.exibir();
        System.out.println("CNPJ: " + cnpj);
```



```java
/**
 *
 * @author ruanf
 */
package model;

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

public class PessoaFisicaRepo {
    private List<PessoaFisica> pessoasFisicas;

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

    /* Método para inserir Pessoa Fisica no Sistema */
    public void inserir(PessoaFisica pessoaFisica) {
        pessoasFisicas.add(e: pessoaFisica);
    }

    /* Método para alterar Pessoa Fisica no Sistema */
    public void alterar(PessoaFisica pessoaFisica) {
        for (int i = 0; i < pessoasFisicas.size(); i++) {
            PessoaFisica pf = pessoasFisicas.get(index: i);
            if (pf.getId() == pessoaFisica.getId()) {
                pessoasFisicas.set(index: i, element:pessoaFisica);
                break;
            }
        }
    }
```



```java
/**
 *
 * @author ruanf
 */
package model;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;

public class PessoaJuridicaRepo {
    private final ArrayList<PessoaJuridica> listaPessoasJuridicas;

    public PessoaJuridicaRepo() {
        listaPessoasJuridicas = new ArrayList<>();
    }
    /* Método para inserir Pessoa Juridica no Sistema */
    public void inserir(PessoaJuridica pessoaJuridica) {
        listaPessoasJuridicas.add(e: pessoaJuridica);
    }

    /* Método para alterar Pessoa Juridica no Sistema */
    public void alterar(PessoaJuridica pessoaJuridica) {
        for (int i = 0; i < listaPessoasJuridicas.size(); i++) {
            PessoaJuridica pj = listaPessoasJuridicas.get(index: i);
            if (pj.getId() == pessoaJuridica.getId()) {
                listaPessoasJuridicas.set(index: i, element:pessoaJuridica);
                break;
            }
        }
```

Análise e conclusão:

P - Quais as vantagens e desvantagens do uso de herança?

R – Algumas das vantagens são a reutilização de código, extensibilidade e polimorfismo, já algumas desvantagens podem ser notadas como complexidade e acoplamento forte entre classes.

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

R – Ela faz com que o Java possa gravar e ler objetos em formato binário através das classes, o que é essencial para a persistência de dados.

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

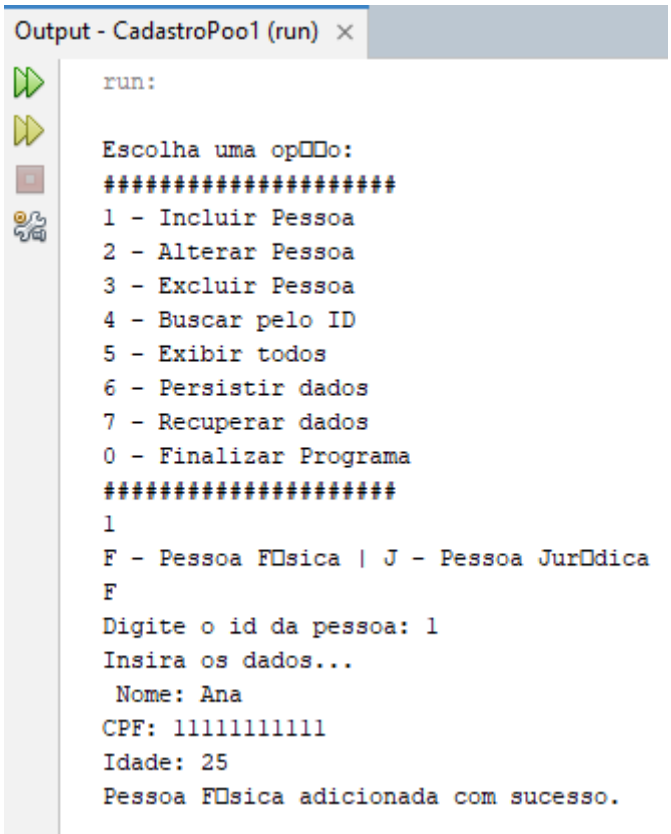
R – O paradigma funcional é utilizado no processamento de coleções de dados, permitindo operações declarativas e funcionais.

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

R - A serialização.

2º Procedimento

Resultado da execução dos códigos



```
Escolha uma op□□o:
######################
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 F□sica | J - Pessoa Jur□dica
J
Digite o id da pessoa: 3
Insira os dados...
Nome da Empresa: XPTO Sales
CNPJ: 33333333333333
Pessoa Jur□dica adicionada com sucesso.
```

```
Escolha uma op□□o:
######################
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
######################
2
F - Pessoa F□sica | J - Pessoa Jur□dica
F

Digite o ID da Pessoa para alterar: 1
Novo Nome: Ana2
Novo CPF: 22222222222
Nova Idade: 21
Pessoa F□sica alterada com sucesso.
```

```
Escolha uma op□□o:
######################
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
######################
3
F - Pessoa F□sica | J - Pessoa Jur□dica
J

Digite o ID da Pessoa que para excluir: 3
Pessoa Jur□dica exclu□da com sucesso.


Escolha uma op□□o:
######################
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
######################
4
F - Pessoa F□sica | J - Pessoa Jur□dica
F

Digite o ID da Pessoa para exibir: 1
ID: 1
Nome: Ana2
CPF: 22222222222
Idade: 21
```

```
Escolha uma op□□o:
######################
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
######################
5
F - Pessoa F□sica | J - Pessoa Jur□dica
J
ID: 4
Nome: XPTO Solutions
CNPJ: 44444444444444


Escolha uma op□□o:
######################
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
######################
6
Digite um id para Salvar o arquivo: 12
Dados salvos com sucesso.

Escolha uma op□□o:
######################
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
######################
7
Digite o id do arquivo para recupera□□o: 12
Dados das pessoas jur□dicas foram recuperados do arquivo
Dados recuperados com sucesso.
```

```
Escolha uma op□□o:
####################
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
####################
0
Saindo do programa.
BUILD SUCCESSFUL (total time: 3 minutes 22 seconds)
```

Análise e conclusão:

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

R – Elementos estáticos são variáveis ou métodos que pertencem à classe ao invés de instâncias especificas. O método main adota esse modificador para que ele possa ser chamado sem a necessidade de criar um objeto da classe que contém o método.

P – Para que serve a classe Scanner?

R – A classe Scanner serve para ler a entrada de dados.

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

R – Essas classes centralizam a lógica de acesso e manipulação de dados, separando-a das demais camadas da aplicação, como a camada de negócios e a camada de apresentação. Isso promove um design mais limpo e modular, facilitando a manutenção, teste e escalabilidade do sistema.