Estácio

UNIVERSIDADE ESTACIO DE SÁ DE RIBEIRÃO PRETO

POLO PARQUE ANDORINHAS

DESENVOLVIMENTO FULL STACK

2023.3 FULL STACK ALUNO

NIVEL 1: INICIANDO O CAMINHO PELO JAVA

ALESSANDRO SENDI SHIGEMATSU

1. **Título da Prática**

RPG0014 - Iniciando o caminho pelo Java

2.Objetivos 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.

3. Todos os códigos solicitados neste roteiro de aula

```
package model;


import java.io.IOException;
import java.util.ArrayList;
import java.util.NoSuchElementException;


public class main {
   public static void main ( String[] args ) throws IOException, ClassNotFoundException {
      try {
         Menu menu = new Menu();
         PessoaFisicaRepo pessoaFisicaRepo = new PessoaFisicaRepo( );
         PessoaJuridicaRepo pessoaJuridicaRepo = new PessoaJuridicaRepo( );


         while (true) {
            menu.telaInicial();
```

```
            menu.setTelaInicialResposta();


            switch( menu.getTelaInicialResposta() ) {
               case "1" -> {
                    String tipoPessoa = menu.escolhaTipoPessoa();
                    if( "F".equals(tipoPessoa) )
pessoaFisicaRepo.inserir(menu.preenchePessoaFisica() );
                    if( "J".equals(tipoPessoa) )
pessoaJuridicaRepo.inserir(menu.preenchePessoaJuridica() );
                  }
               case "2" -> {
                    int id = menu.inputApenasNumeros( "ID" );
                    String tipoPessoa = menu.escolhaTipoPessoa();

                    if( "F".equals(tipoPessoa) ) pessoaFisicaRepo.alterar(
id,menu.preenchePessoaFisica() );
                    if( "J".equals(tipoPessoa) ) pessoaJuridicaRepo.alterar(
id,menu.preenchePessoaJuridica() );
                  }
               case "3" -> {
                    int id = menu.inputApenasNumeros( "ID" );
                    String tipoPessoa = menu.escolhaTipoPessoa();

                    if( "F".equals(tipoPessoa) ) pessoaFisicaRepo.excluir( id );
                    if( "J".equals(tipoPessoa) ) pessoaJuridicaRepo.excluir( id );
                  }
               case "4" -> {
                    int id = menu.inputApenasNumeros( "ID" );
                    String tipoPessoa = menu.escolhaTipoPessoa();

                    if( "F".equals(tipoPessoa) ){
                       PessoaFisica pessoaFisica = pessoaFisicaRepo.obter( id );
```

```
                    pessoaFisica.exibir();
                }

                if( "J".equals(tipoPessoa) ){
                    PessoaJuridica pessoaJuridica = pessoaJuridicaRepo.obter( id );
                    pessoaJuridica.exibir();
                }
            }
            case "5" -> {
                String tipoPessoa = menu.escolhaTipoPessoa();

                if( "F".equals(tipoPessoa) ){
                    ArrayList<PessoaFisica> pessoasFisicas =  pessoaFisicaRepo.obterTodos();
                    pessoasFisicas.stream().forEach( pessoa -> pessoa.exibir() );
                }

                if( "J".equals(tipoPessoa) ) {
                    ArrayList<PessoaJuridica> pessoasJuridicas =
pessoaJuridicaRepo.obterTodos();
                    pessoasJuridicas.stream().forEach( pessoa -> pessoa.exibir() );
                }
            }
            case "6" -> {
                try{
                    String preFixo = menu.escolhaPrefixo();
                    pessoaFisicaRepo.persistir(preFixo + ".fisica.bin" );
                    pessoaJuridicaRepo.persistir(preFixo + ".juridica.bin" );
                } catch (Exception e) {
                    System.out.println("O sistema não pode criar o arquivo especificado!");
                }
```

```
                }
            case "7" -> {
                try{
                    String preFixo = menu.escolhaPrefixo();
                    pessoaFisicaRepo.recuperar(preFixo + ".fisica.bin" );
                    pessoaJuridicaRepo.recuperar(preFixo + ".juridica.bin" );
                } catch (Exception e) {
                    System.out.println("O sistema não pode encontrar o arquivo especificado!");
                }
            }
            case "0" -> System.exit(0);
            default -> System.out.println("Opcao invalida!");
          }

        }
    } catch(IllegalStateException | NoSuchElementException e) {
        // System.in has been closed
        System.out.println("System.in was closed; exiting");
    }


  }
}
```

```
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.stream.Collectors;

public class PessoaJuridicaRepo {

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

   public void inserir( PessoaJuridica pessoa ) {
      pessoas.add(pessoa);
      Menu.limparConsole();
      System.out.println( "Pessoa Juridica Inserida com Sucesso!" );
   }

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

   public void alterar( int id, PessoaJuridica pessoaDestino ) {
      int index = pessoas.indexOf( obter( id ) );
      pessoas.set( index, pessoaDestino);
      Menu.limparConsole();
      System.out.println( "Pessoa Juridica Alterada com Sucesso!" );
   }
```

```
    public void excluir( int id ) {
        pessoas = (ArrayList<PessoaJuridica>) pessoas.stream().filter(pessoa -> pessoa.getId() !=  id
).collect(Collectors.toList());
        Menu.limparConsole();
        System.out.println( "Pessoa Juridica Excluida com Sucesso!" );
    }

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

    public void persistir(String nomeArquivo) throws IOException {
        try (ObjectOutputStream outputStream = new ObjectOutputStream(new
FileOutputStream(nomeArquivo))) {
            outputStream.writeObject(pessoas);
            System.out.println( "Dados de Pessoas Juridicas armazenados." );
        }
    }

    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
        try (ObjectInputStream inputStream = new ObjectInputStream(new
FileInputStream(nomeArquivo))) {
            pessoas = (ArrayList<PessoaJuridica>) inputStream.readObject();
            System.out.println( "Dados de Pessoas Juridicas recuperados." );
        }
    }
}
```

```
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.stream.Collectors;

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

   public void inserir( PessoaFisica pessoa ) {
      pessoas.add( pessoa );
      Menu.limparConsole();
      System.out.println( "Pessoa Fisica Inserida com Sucesso!" );
   }

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

    public void alterar( int id , PessoaFisica pessoaDestino ) {
      int index = pessoas.indexOf( obter( id ) );
      pessoas.set( index, pessoaDestino);
      Menu.limparConsole();
      System.out.println( "Pessoa Fisica Alterada com Sucesso!" );
   }

   public void excluir( int id ) {
```

```
        pessoas = (ArrayList<PessoaFisica>) pessoas.stream().filter(pessoa -> pessoa.getId() !=  id
).collect(Collectors.toList());
        Menu.limparConsole();
        System.out.println( "Pessoa Fisica Excluida com Sucesso!" );
    }

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

    public void persistir(String nomeArquivo) throws IOException {
        try (ObjectOutputStream outputStream = new ObjectOutputStream(new
FileOutputStream(nomeArquivo))) {
            outputStream.writeObject(pessoas);
            System.out.println( "Dados de Pessoas Fisicas armazenados." );
        }
    }

    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
        try (ObjectInputStream inputStream = new ObjectInputStream(new
FileInputStream(nomeArquivo))) {
            pessoas = (ArrayList<PessoaFisica>) inputStream.readObject();
            System.out.println( "Dados de Pessoas Fisicas recuperados." );
        }
    }
}
```

```
package model;
import java.util.Scanner;

public class Menu {
   private Scanner scanner;
   private final String NEW_LINE = System.getProperty("line.separator");
   private String telaInicialResposta;

   public Menu () {
      scanner = new Scanner( System.in );
   }

   public void telaInicial() {
    System.out.println("==================="  + NEW_LINE
         + "1 - Incluir Pessoa" + NEW_LINE
         + "2 - Alterar Pessoa" + NEW_LINE
         + "3 - Excluir Pessoa" + NEW_LINE
         + "4 - Buscar por ID" + NEW_LINE
         + "5 - Exibir Todos" + NEW_LINE
         + "6 - Persistir Dados" + NEW_LINE
         + "7 - Recuperar Dados" + NEW_LINE
         + "0 - Finalizar Programa" + NEW_LINE
         + "======================");

//      ultimaResposta = scanner.nextLine();
   }

   public void setTelaInicialResposta( ) {
      telaInicialResposta = scanner.nextLine();
   }
```

```
public String getTelaInicialResposta () {
   return telaInicialResposta;
}

public String escolhaTipoPessoa() {

   while( true ) {
      System.out.println("F - Pessoa Fisica | J - Pessoa Juridica" );

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

      switch( opcao ) {
         case "J", "F" -> {
            return opcao;
         }
         default -> System.out.println("Opcao invalida!");
      }
   }

}

public String escolhaPrefixo() {
   System.out.println("Favor Digitar o Prefixo do Arquivo:" );
   return scanner.nextLine();
}

public PessoaFisica preenchePessoaFisica() {

   int id = inputApenasNumeros( "ID" );

   System.out.println("Favor Digitar o Nome:" );
```

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

    System.out.println("Favor Digitar o CPF:" );
    String cpf = scanner.nextLine();

    int idade = inputApenasNumeros( "idade" );

    return new PessoaFisica( id, nome, cpf,idade );

}

public PessoaJuridica preenchePessoaJuridica() {

    int id = inputApenasNumeros( "ID" );

    System.out.println("Favor Digitar o Nome:" );
    String nome = scanner.nextLine();

    System.out.println("Favor Digitar o CNPJ:" );
    String CNPJ = scanner.nextLine();

    return new PessoaJuridica( id, nome, CNPJ );

}

public int inputApenasNumeros( String nomeInput ){
    System.out.println("Favor Digitar o " + nomeInput +" (Somente Numeros):" );

    int reloop = 0;
    int result = 0;
    do {
```

```
      try {
        String input = scanner.nextLine(); // Scan the next line from System.in
        result = Integer.parseInt(input); // Try to parse it as an int
        reloop++;
      } catch (Exception e) {
        System.out.println("Por favor digite um NUMERO!");
      }
    } while (reloop == 0);

    return result;
}

public final static void limparConsole()
{
    try
    {
        final String os = System.getProperty("os.name");

        if (os.contains("Windows"))
        {
            Runtime.getRuntime().exec("cls");
        }
        else
        {
            Runtime.getRuntime().exec("clear");
        }
    }
    catch (final Exception e)
    {
        //  Handle any exceptions.
    }
```

```
    }


}
```

4. **Os resultados da execução dos códigos também devem ser apresentados.**

Menu

```
run:
==================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar por ID
5 - Exibir Todos
6 - Persistir Dados
7 - Recuperar Dados
0 - Finalizar Programa
======================
```

1-Inserção

```
1
F - Pessoa Fisica | J - Pessoa Juridica
f
Favor Digitar o ID (Somente Numeros):
1
Favor Digitar o Nome:
Carlos
Favor Digitar o CPF:
11111111111
Favor Digitar o idade (Somente Numeros):
20
Pessoa Fisica Inserida com Sucesso!
```

5-Exibir Todos

```
5
F - Pessoa Fisica | J - Pessoa Juridica
f
ID: 1
Nome: Carlos
CPF: 11111111111
Idade: 20
```

2-Alteração

```
2
Favor Digitar o ID (Somente Numeros):
1
F - Pessoa Fisica | J - Pessoa Juridica
f
Favor Digitar o ID (Somente Numeros):
3
Favor Digitar o Nome:
Rafael
Favor Digitar o CPF:
3333333333
Favor Digitar o idade (Somente Numeros):
30
Pessoa Fisica Alterada com Sucesso!
```

5-Exibir todos

```
======================
5
F - Pessoa Fisica | J - Pessoa Juridica
f
ID: 3
Nome: Rafael
CPF: 333333333
Idade: 30
==================
```

6-Persistir Dados

```
======================
6
Favor Digitar o Prefixo do Arquivo:
teste
Dados de Pessoas Fisicas armazenados.
Dados de Pessoas Juridicas armazenados.
==================
```

3-Exclusão

```
======================
3
Favor Digitar o ID (Somente Numeros):
3
F - Pessoa Fisica | J - Pessoa Juridica
f
Pessoa Fisica Excluida com Sucesso!
====================
```

5-Exibir todos

```
----------------------
5
F - Pessoa Fisica | J - Pessoa Juridica
f
====================
```

7 - Recuperar Dados

```
 ======================
 7
 Favor Digitar o Prefixo do Arquivo:
 teste
 Dados de Pessoas Fisicas recuperados.
 Dados de Pessoas Juridicas recuperados.
 ====================
```

5-Exibir todos

```
  ======================
  5
  F - Pessoa Fisica | J - Pessoa Juridica

  Opcao invalida!
  F - Pessoa Fisica | J - Pessoa Juridica
  5
  Opcao invalida!
  F - Pessoa Fisica | J - Pessoa Juridica
  f
  ID: 3
  Nome: Rafael
  CPF: 333333333
  Idade: 30
  ====================
```

4-Buscar por ID

```
======================
4
Favor Digitar o ID (Somente Numeros):
3
F - Pessoa Fisica | J - Pessoa Juridica
f
ID: 3
Nome: Rafael
CPF: 333333333
Idade: 30
====================
```

5. Análise e Conclusão

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

São usados como modificadores em atributos e métodos. Os elementos estáticos podem ser acessados sem precisar instanciar o objeto. O main adota esse elemento devido ao compilador (JVM) poder chamar o método main sem precisar instanciar o objeto.

b. Para que serve a classe Scanner?

É usada para coletar dados do usuário através de inputs.

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

Separou as responsabilidades de cada objeto, assim os repositórios gerenciam os dados das pessoas e a classes de pessoas somente os dados pertinentes. Facilitando a padronização e consequentemente organizando o código.