Nome: **Ariel Freitas dos Santos**
Matrícula: **202208291694**
Curso: **Desenvolvimento Full Stack**
Turma : **2023.3**
Campus: **POLO ST MORADA DO SOL - GOIÂNIA - GO**
Github: **https://github.com/ArielFSantos/Java_Estacio_Nivel_1**

Missão Pratica | Mundo 3 | Nível 1

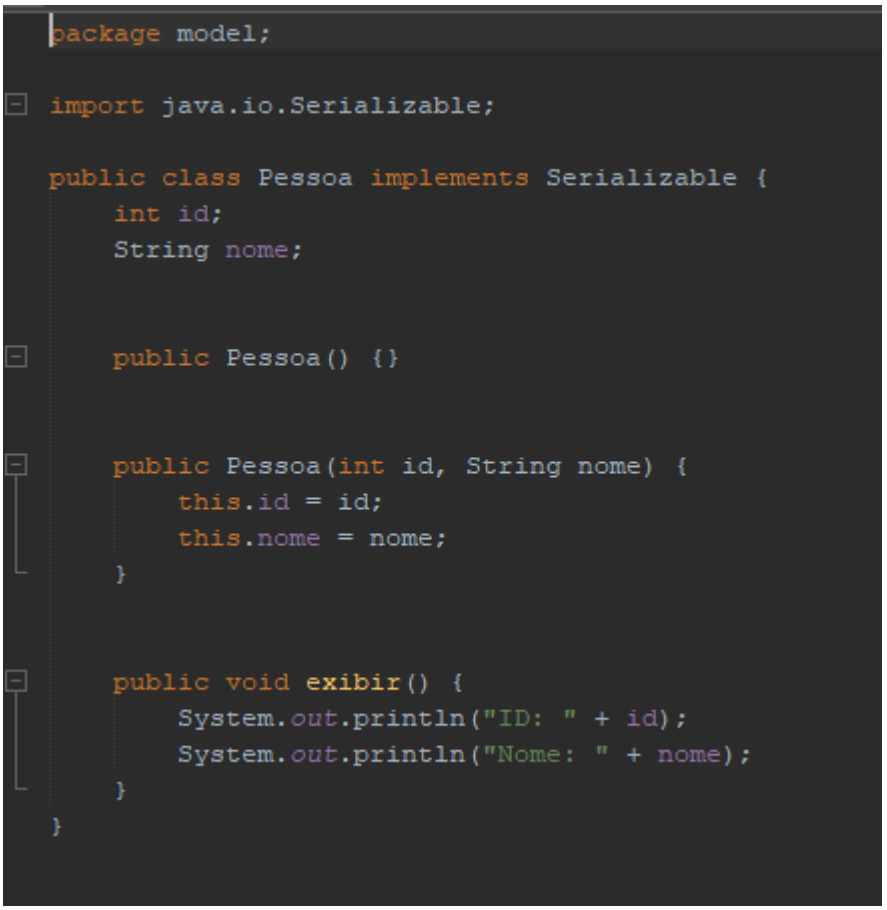
RPG0014 - Iniciando o caminho pelo Java

Objetivo:

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.

Códigos Solicitados:

Classe Pessoa :



```java
package model;

import java.io.Serializable;

public class Pessoa implements Serializable {
    int id;
    String nome;


    public Pessoa() {}


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


    public void exibir() {
        System.out.println("ID: " + id);
        System.out.println("Nome: " + nome);
    }
}
```

Classe PessoaFisica :

```java
import java.io.Serializable;

public class PessoaFisica extends Pessoa implements Serializable {
    private static int proximoId = 1;
    private String cpf;
    private int idade;

    public PessoaFisica( String nome, String cpf, int idade) {
        this.id = proximoId++;
        this.nome = 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);
        System.out.println("Idade: " + idade);
    }

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

    public int getId() {
        return id;
}

}
```

Classe PessoaJuridica:

```java
package model;

import java.io.Serializable;

public class PessoaJuridica extends Pessoa implements Serializable {
    private static int proximoId = 1;
    private String cnpj;


    public PessoaJuridica(String nome, String cnpj) {
        this.id = proximoId++;
        this.nome = nome;
        this.cnpj = cnpj;
    }

    @Override
    public void exibir() {
        super.exibir();
        System.out.println("CNPJ: " + cnpj);
    }

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

    int getId() {
        return id;
    }
}
```

Classe PessoaFisicaRepo:

```java
package model;

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

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

    public void inserir(PessoaFisica pessoaFisica) {
        pessoasFisicas.add(e: pessoaFisica);
    }

    public void alterar(PessoaFisica pessoaFisica) {
        for (int i = 0; i < pessoasFisicas.size(); i++) {
            PessoaFisica p = pessoasFisicas.get(index: i);
            if (p.getId() == pessoaFisica.getId()) {
                pessoasFisicas.set(index: i, element:pessoaFisica);
                return;
            }
        }
    }

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

    public PessoaFisica obterPorId(int id) {
        for (PessoaFisica pessoaFisica : pessoasFisicas) {
            if (pessoaFisica.getId() == id) {
                return pessoaFisica;
            }
        }
        return null;
    }

    public List<PessoaFisica> obterTodos() {
        return new ArrayList<>(c: pessoasFisicas);
    }

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

    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(name: nomeArquivo))) {
            pessoasFisicas = (List<PessoaFisica>) ois.readObject();
        }
    }
}
```

Classe PessoaJuridicaRepo:

```java
package model;

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

public class PessoaJuridicaRepo {
    private List<PessoaJuridica> pessoasJuridicas = new ArrayList<>();

    public void inserir(PessoaJuridica pessoaJuridica) {
        pessoasJuridicas.add(e: pessoaJuridica);
    }

    public void alterar(PessoaJuridica pessoaJuridica) {
        for (int i = 0; i < pessoasJuridicas.size(); i++) {
            PessoaJuridica p = pessoasJuridicas.get(index: i);
            if (p.getId() == pessoaJuridica.getId()) {
                pessoasJuridicas.set(index: i, element:pessoaJuridica);
                return;
            }
        }
    }

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

    public PessoaJuridica obterPorId(int id) {
        for (PessoaJuridica pessoaJuridica : pessoasJuridicas) {
            if (pessoaJuridica.getId() == id) {
                return pessoaJuridica;
            }
        }
        return null;
    }


    public List<PessoaJuridica> obterTodos() {
        return new ArrayList<>(c: pessoasJuridicas);
    }

    public void persistir(String nomeArquivo) throws IOException {
        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(name: nomeArquivo))) {
            oos.writeObject(obj:pessoasJuridicas);
        }
    }
```

```java
    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(name: nomeArquivo))) {
            pessoasJuridicas = (List<PessoaJuridica>) ois.readObject();
        }
    }
}
```

Classe CadastroPOO:

```
import java.io.IOException;
import java.util.List;
import java.util.Scanner;
import model.PessoaFisica;
import model.PessoaFisicaRepo;
import model.PessoaJuridica;
import model.PessoaJuridicaRepo;

public class CadastroPOO {
    public static void main(String[] args) {
        try (Scanner scanner = new Scanner(source: System.in)) {
            PessoaFisicaRepo pessoaFisicaRepo = new PessoaFisicaRepo();
            PessoaJuridicaRepo pessoaJuridicaRepo = new PessoaJuridicaRepo();

            String prefixoArquivos = "cadastro";

            int opcao;
            do {
                System.out.println(x: "======================");
                System.out.println(x: "Escolha uma Opcao:");
                System.out.println(x: "1 - Incluir");
                System.out.println(x: "2 - Alterar");
                System.out.println(x: "3 - Excluir");
                System.out.println(x: "4 - Exibir pelo ID");
                System.out.println(x: "5 - Exibir todos");
                System.out.println(x: "6 - Salvar dados");
                System.out.println(x: "7 - Recuperar dados");
                System.out.println(x: "0 - Sair");
                System.out.println(x: "======================");
                opcao = scanner.nextInt();
                scanner.nextLine();

                switch (opcao) {
                    case 1 -> {
                        System.out.println(x: "Escolha o tipo (F - Pessoa Fisica, J - Pessoa Juridica):");
                        String tipo = scanner.nextLine();
                        switch (tipo) {
                            case "F" -> {
                                PessoaFisica pessoaFisica = lerDadosPessoaFisica(scanner);
                                pessoaFisicaRepo.inserir(pessoaFisica);
                            }
                            case "J" -> {
                                PessoaJuridica pessoaJuridica = lerDadosPessoaJuridica(scanner);
                                pessoaJuridicaRepo.inserir(pessoaJuridica);
                            }
                            default -> System.out.println(x: "Opção inválida.");
                        }
                    }
```

```
case 2 -> {
    System.out.println(x: "Escolha o tipo (F - Pessoa Fisica, J - Pessoa Juridica):");
    String tipoAlterar = scanner.nextLine();
    switch (tipoAlterar) {
        case "F" -> {
            System.out.println(x: "Informe o ID da Pessoa Fisica que deseja alterar:");
            int idAlterar = scanner.nextInt();
            scanner.nextLine();
            PessoaFisica pessoaFisicaAlterar = lerDadosPessoaFisica(scanner);
            pessoaFisicaAlterar.setId(idAlterar);
            pessoaFisicaRepo.alterar(pessoaFisica: pessoaFisicaAlterar);
        }
        case "J" -> {
            System.out.println(x: "Informe o ID da Pessoa Juridica que deseja alterar:");
            int idAlterar = scanner.nextInt();
            scanner.nextLine();
            PessoaJuridica pessoaJuridicaAlterar = lerDadosPessoaJuridica(scanner);
            pessoaJuridicaAlterar.setId(idAlterar);
            pessoaJuridicaRepo.alterar(pessoaJuridica: pessoaJuridicaAlterar);
        }
        default -> System.out.println(x: "Opção inválida.");
    }
}

case 3 -> {
    System.out.println(x: "Escolha o tipo (F - Pessoa Fisica, J - Pessoa Juridica):");
    String tipoExcluir = scanner.nextLine();
    switch (tipoExcluir) {
        case "F" -> {
            System.out.println(x: "Informe o ID da Pessoa Fisica que deseja excluir:");
            int idExcluir = scanner.nextInt();
            scanner.nextLine();
            pessoaFisicaRepo.excluir(id: idExcluir);
        }
        case "J" -> {
            System.out.println(x: "Informe o ID da Pessoa Juridica que deseja excluir:");
            int idExcluir = scanner.nextInt();
            scanner.nextLine();
            pessoaJuridicaRepo.excluir(id: idExcluir);
        }
        default -> System.out.println(x: "Opção inválida.");
    }
}
```

```java
case 4 -> {
    System.out.println(x: "Escolha o tipo (F - Pessoa Fisica, J - Pessoa Juridica):");
    String tipoExibir = scanner.nextLine();
    switch (tipoExibir) {
        case "F" -> {
            System.out.println(x: "Informe o ID da Pessoa Fisica que deseja exibir:");
            int idExibir = scanner.nextInt();
            scanner.nextLine();
            PessoaFisica pessoaFisicaExibir = pessoaFisicaRepo.obterPorId(id: idExibir);
            if (pessoaFisicaExibir != null) {
                pessoaFisicaExibir.exibir();
            } else {
                System.out.println(x: "Pessoa Fisica não encontrada.");
            }
        }
        case "J" -> {
            System.out.println(x: "Informe o ID da Pessoa Juridica que deseja exibir:");
            int idExibir = scanner.nextInt();
            scanner.nextLine();
            PessoaJuridica pessoaJuridicaExibir = pessoaJuridicaRepo.obterPorId(id: idExibir);
            if (pessoaJuridicaExibir != null) {
                pessoaJuridicaExibir.exibir();
            } else {
                System.out.println(x: "Pessoa Juridica não encontrada.");
            }
        }
        default -> System.out.println(x: "Opção inválida.");
    }
}

case 5 -> {
    System.out.println(x: "Escolha o tipo (F - Pessoa Fisica, J - Pessoa Juridica):");
    String tipoExibirTodos = scanner.nextLine();
    switch (tipoExibirTodos) {
        case "F" -> {
            List<PessoaFisica> todasPessoasFisicas = pessoaFisicaRepo.obterTodos();
            for (PessoaFisica pf : todasPessoasFisicas) {
                pf.exibir();
                System.out.println();
            }
        }
        case "J" -> {
            List<PessoaJuridica> todasPessoasJuridicas = pessoaJuridicaRepo.obterTodos();
            for (PessoaJuridica pj : todasPessoasJuridicas) {
                pj.exibir();
                System.out.println();
            }
        }
```

```
                    default -> System.out.println(x: "Opção inválida.");
                }
            }

            case 6 -> {
                System.out.println(x: "Digite o prefixo para os arquivos (por exemplo, 'cadastro'):");
                String prefixoSalvar = scanner.nextLine();

                try {
                    pessoaFisicaRepo.persistir(prefixoSalvar + ".fisica.bin");
                    pessoaJuridicaRepo.persistir(prefixoSalvar + ".juridica.bin");
                    System.out.println(x: "Dados salvos com sucesso.");
                } catch (IOException e) {
                    System.out.println("Erro ao salvar os dados: " + e.getMessage());
                }
            }
            case 7 -> {
                System.out.println(x: "Digite o prefixo para os arquivos (por exemplo, 'cadastro'):");
                String prefixoRecuperar = scanner.nextLine();

                try {
                    pessoaFisicaRepo.recuperar(prefixoRecuperar + ".fisica.bin");
                    pessoaJuridicaRepo.recuperar(prefixoRecuperar + ".juridica.bin");
                    System.out.println(x: "Dados recuperados com sucesso.");
                } catch (IOException | ClassNotFoundException e) {
                    System.out.println("Erro ao recuperar os dados: " + e.getMessage());
                }
            }
            case 0 -> System.out.println(x: "Saindo...");
            default -> System.out.println(x: "Opção inválida.");
        }
    } while (opcao != 0);
  }
}

private static PessoaFisica lerDadosPessoaFisica(Scanner scanner) {
    System.out.println(x: "Digite o nome da Pessoa Fisica:");
    String nome = scanner.nextLine();

    System.out.println(x: "Digite o CPF da Pessoa Fisica:");
    String cpf = scanner.nextLine();

    System.out.println(x: "Digite a idade da Pessoa Fisica:");
    int idade = scanner.nextInt();

    scanner.nextLine();

    return new PessoaFisica(nome, cpf, idade);
}
```

```
    private static PessoaJuridica lerDadosPessoaJuridica(Scanner scanner) {
        System.out.println(x: "Digite o nome da Pessoa Juridica:");
        String nome = scanner.nextLine();

        System.out.println(x: "Digite o CNPJ da Pessoa Juridica:");
        String cnpj = scanner.nextLine();

        return new PessoaJuridica(nome, cnpj);
    }
}
```

Resultados da Execução:

```
======================
Escolha uma Opcao:
1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir todos
6 - Salvar dados
7 - Recuperar dados
0 - Sair
======================
1
Escolha o tipo (F - Pessoa Fisica, J - Pessoa Juridica):
F

Digite o nome da Pessoa Fisica:
Ari
Digite o CPF da Pessoa Fisica:
70068894139
Digite a idade da Pessoa Fisica:
27
```

```
======================
Escolha uma Opcao:
1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir todos
6 - Salvar dados
7 - Recuperar dados
0 - Sair
======================
2
Escolha o tipo (F - Pessoa Fisica, J - Pessoa Juridica):
F
Informe o ID da Pessoa Fisica que deseja alterar:
2
Digite o nome da Pessoa Fisica:
Ari3
Digite o CPF da Pessoa Fisica:
7454795471
Digite a idade da Pessoa Fisica:
29
```

```
======================
Escolha uma Opcao:
1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir todos
6 - Salvar dados
7 - Recuperar dados
0 - Sair
======================
3
Escolha o tipo (F - Pessoa Fisica, J - Pessoa Juridica):
F
Informe o ID da Pessoa Fisica que deseja excluir:
1
```

```
======================
Escolha uma Opcao:
1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir todos
6 - Salvar dados
7 - Recuperar dados
0 - Sair
======================
5
Escolha o tipo (F - Pessoa Fisica, J - Pessoa Juridica):
F
ID: 1
Nome: Ari
CPF: 70068894139
Idade: 27

ID: 2
Nome: Ari2
CPF: 70064714798
Idade: 28
```

Análise e Conclusão:

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

Elementos estáticos são partes de uma classe que são compartilhadas por todas as instâncias dessa classe, em vez de pertencerem a uma instância específica.
O método main é marcado como estático para indicar que ele pertence à classe em si, em vez de a objetos individuais dessa classe. Isso é feito para que o sistema possa executar o programa sem precisar criar uma instância da classe, usando o método main diretamente como ponto de partida.

Para que serve a classe Scanner?

A classe Scanner em Java é utilizada para obter entrada do usuário a partir do teclado ou para processar informações em strings ou arquivos.

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

O uso de classes de repositório impactou positivamente na organização do código, pois permitiu a separação das operações de acesso a dados do restante do programa. Isso torna o código mais modular e fácil de manter, além de facilitar a implementação de diferentes estratégias de persistência de dados.