UNIVERSIDADE ESTÁCIO DE SÁ

DESENVOLVIMENTO FULLSTACK

RPG0014

INICIANDO O CAMINHO PELO JAVA

202205003922

JARDS DE OLIVEIRA GUIMARÃES

MACEIÓ - AL

2023

SUMÁRIO

1. INTRODUÇÃO

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

1.1 OBJETIVO GERAL

- ✓ Utilizar herança e polimorfismo na definição de entidades.
- ✓ Utilizar persistência de objetos em arquivos binários.
- ✓ Implementar uma interface cadastral em modo texto.
- ✓ Utilizar o controle de exceções da plataforma Java.
- ✓ persistência em arquivos binários.

2. CÓDIGOS SOLICITADOS NESTE ROTEIRO DE AULA

2.1 CLASSE PESSOA

```
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;
   }

   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.printf("   ID: %d%n Nome: %s%n", this.getId(), this.getNome());
    }

}
```

2.2 CLASSE PESSOA FÍSICA

```
package model;

public class PessoaFisica extends Pessoa {

    private String cpf;
    private 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() {
        System.out.printf("   ID: %d%n Nome: %s%n  CPF: %s%nIdade: %d%n", this.getId(),
                this.getNome(), this.getCpf(), this.getIdade());
    }

}
```

2.3CLASSE PESSOA JURÍDICA

```
package model;

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() {
      System.out.printf("   ID: %d%n Nome: %s%n CNPJ: %s%n", this.getId(), this.getNome(),
            this.getCnpj());
   }

}
```

2.4CLASSE 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;

public class PessoaFisicaRepo {

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

   public void inserir(PessoaFisica pf) {
      listaDePessoasFisica.add(pf);
   }

   public void alterar(int id, PessoaFisica pf) {
      listaDePessoasFisica.set(buscarIndex(id), pf);
   }
```

```
    public void excluir(int id) {
        listaDePessoasFisica.remove(buscarIndex(id));
    }

    public PessoaFisica obter(int id) {
        return listaDePessoasFisica.get(buscarIndex(id));
    }

    private int buscarIndex(int id) {
        for (PessoaFisica pf : listaDePessoasFisica) {
            if (pf.getId() == id) {
                return listaDePessoasFisica.indexOf(pf);
            }
        }
        return -1;
    }

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

    public void persistir(String nomeArquivo) throws IOException {
        try (FileOutputStream fout = new FileOutputStream("src\\files\\" + nomeArquivo + ".fisica.bin");
                ObjectOutputStream oos = new ObjectOutputStream(fout)) {
            oos.writeObject(this.listaDePessoasFisica);
            System.out.println("Dados de Pessoa Física Armazenados.");
        }
    }

    public void recuperar(String nomeArquivo) throws ClassNotFoundException, IOException{
        try (FileInputStream fin = new FileInputStream("src\\files\\" + nomeArquivo + ".fisica.bin");
                ObjectInputStream ois = new ObjectInputStream(fin)) {
            listaDePessoasFisica = (ArrayList<PessoaFisica>) ois.readObject();
        }

    }
}
```

2.5CLASSE 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;

public class PessoaJuridicaRepo {

    private ArrayList<PessoaJuridica> listaDePessoasJuridica = new ArrayList<>();
```

```
    public void inserir(PessoaJuridica pj) {
        listaDePessoasJuridica.add(pj);
    }

    public void alterar(int id, PessoaJuridica pj) {
        listaDePessoasJuridica.set(buscarIndex(id), pj);
    }

    private int buscarIndex(int id) {
        for (PessoaJuridica pj : listaDePessoasJuridica) {
            if (pj.getId() == id) {
                return listaDePessoasJuridica.indexOf(pj);
            }
        }

        return -1;
    }

    public void excluir(int id) {
        listaDePessoasJuridica.remove(buscarIndex(id));
    }

    public PessoaJuridica obter(int id) {
        return listaDePessoasJuridica.get(buscarIndex(id));
    }

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

    public void persistir(String nomeArquivo) throws IOException {
        try (FileOutputStream fout = new FileOutputStream("src\\files\\" + nomeArquivo +
".juridica.bin");
                ObjectOutputStream oos = new ObjectOutputStream(fout)) {
            oos.writeObject(this.listaDePessoasJuridica);
            System.out.println("Dados de Pessoa Juridica Armazenados.");
        }
    }

    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
        try (FileInputStream fin = new FileInputStream("src\\files\\" + nomeArquivo + ".juridica.bin");
                ObjectInputStream ois = new ObjectInputStream(fin)) {
            listaDePessoasJuridica = (ArrayList<PessoaJuridica>) ois.readObject();
        }

    }

}
```

2.6 CLASSE PESSOAFISICASERVICE

```
package model;

import java.io.IOException;
import java.util.Scanner;

public class PessoaFisicaService {

   public static String cadastrar(Scanner escanner, PessoaFisicaRepo pessoaFisicaRepo) {
      System.out.println("Digite o Id da Pessoa:");
      int id = escanner.nextInt();

      System.out.println("Digite o Nome da Pessoa:");
      String nome = escanner.next();

      System.out.println("Digite o CPF da Pessoa:");
      String cpf = escanner.next();

      System.out.println("Digite a Idade da Pessoa:");
      int idade = escanner.nextInt();

      pessoaFisicaRepo.inserir(new PessoaFisica(id, nome, cpf, idade));

      return "Pessoa Fisica Cadastrada com Sucesso!";
   }

   public static String alterar(Scanner escanner, PessoaFisicaRepo pessoaFisicaRepo) {
      System.out.println("Digite o Id da Pessoa:");
      int id = escanner.nextInt();
      PessoaFisica pessoaFisica = pessoaFisicaRepo.obter(id);
      pessoaFisica.exibir();

      System.out.println("Digite o Nome da Pessoa:");
      String nome = escanner.next();

      System.out.println("Digite o CPF da Pessoa:");
      String cpf = escanner.next();

      System.out.println("Digite a Idade da Pessoa:");
      int idade = escanner.nextInt();

      pessoaFisica.setNome(nome);
      pessoaFisica.setCpf(cpf);
      pessoaFisica.setIdade(idade);

      pessoaFisicaRepo.alterar(id, pessoaFisica);

      return "Pessoa Fisica Alterada com Sucesso!";
   }

   public static String excluir(Scanner escanner, PessoaFisicaRepo pessoaFisicaRepo) {
```

```
        System.out.println("Digite o Id da Pessoa:");
        int id = escanner.nextInt();

        pessoaFisicaRepo.excluir(id);
        return "Pessoa Fisica Excluida com Sucesso!";
    }

    public static String obter(Scanner escanner, PessoaFisicaRepo pessoaFisicaRepo) {
        System.out.println("Digite o Id da Pessoa:");
        int id = escanner.nextInt();
        PessoaFisica pessoaFisica = pessoaFisicaRepo.obter(id);
        pessoaFisica.exibir();
        return "Pessoa Fisica Exibida com Sucesso!";
    }

    public static String exibir(Scanner escanner, PessoaFisicaRepo pessoaFisicaRepo) {
        for (PessoaFisica pf : pessoaFisicaRepo.obterTodos()) {
            pf.exibir();
        }
        return "Lista Exibida com sucesso!";
    }

    public static String salvar(Scanner escanner, PessoaFisicaRepo pessoaFisicaRepo) {
        System.out.println("Digite o Nome do arquivo:");
        String nome = escanner.next();

        try {
            pessoaFisicaRepo.persistir(nome);
            return "Dados Salvo no arquivo " + nome + ".fisica.bin";
        } catch (IOException e) {
            e.printStackTrace();
            return "Erro ao Salvar o arquivo " + nome + ".juridica.bin";
        }

    }

    public static String recuperar(Scanner escanner, PessoaFisicaRepo pessoaFisicaRepo) {
        System.out.println("Digite o Nome do arquivo:");
        String nome = escanner.next();

        try {
            pessoaFisicaRepo.recuperar(nome);
            return "Dados de Pessoas Física Recuperados!";
        } catch (ClassNotFoundException | IOException e) {
            return "Arquivo não encontrado!";
        }

    }
}
```

2.7 CLASSE PESSOAJURIDICASERVICE

```
package model;

import java.io.IOException;
import java.util.Scanner;

public class PessoaJuridicaService {

   public static String cadastrar(Scanner escanner, PessoaJuridicaRepo pessoaJuridicaRepo) {
      System.out.println("Digite o Id da Empresa:");
      int id = escanner.nextInt();

      System.out.println("Digite o Nome da Empresa:");
      String nome = escanner.next();

      System.out.println("Digite o CNPJ da Empresa:");
      String cnpj = escanner.next();

      pessoaJuridicaRepo.inserir(new PessoaJuridica(id, nome, cnpj));

      return "Pessoa Juridica Cadastrada com Sucesso!";
   }

   public static String alterar(Scanner escanner, PessoaJuridicaRepo pessoaJuridicaRepo) {
      System.out.println("Digite o Id da Empresa:");
      int id = escanner.nextInt();
      PessoaJuridica pessoaJuridica = pessoaJuridicaRepo.obter(id);
      pessoaJuridica.exibir();

      System.out.println("Digite o Nome da Empresa:");
      String nome = escanner.next();

      System.out.println("Digite o CNPJ da Empresa:");
      String cnpj = escanner.next();

      pessoaJuridica.setNome(nome);
      pessoaJuridica.setCnpj(cnpj);

      pessoaJuridicaRepo.alterar(id, pessoaJuridica);

      return "Pessoa Juridica Alterada com Sucesso!";
   }

   public static String excluir(Scanner escanner, PessoaJuridicaRepo pessoaJuridicaRepo) {
      System.out.println("Digite o Id da Empresa:");
      int id = escanner.nextInt();

      pessoaJuridicaRepo.excluir(id);
      return "Pessoa Juridica Excluida com Sucesso!";
   }
```

```
    public static String obter(Scanner escanner, PessoaJuridicaRepo pessoaJuridicaRepo) {
        System.out.println("Digite o Id da Empresa:");
        int id = escanner.nextInt();
        PessoaJuridica pessoaJuridica = pessoaJuridicaRepo.obter(id);
        pessoaJuridica.exibir();
        return "Pessoa Juridica Exibida com Sucesso!";
    }

    public static String exibir(Scanner escanner, PessoaJuridicaRepo pessoaJuridicaRepo) {
        for (PessoaJuridica pj : pessoaJuridicaRepo.obterTodos()) {
            pj.exibir();
        }
        return "Lista Exibida com sucesso!";
    }

    public static String salvar(Scanner escanner, PessoaJuridicaRepo pessoaJuridicaRepo) {
        System.out.println("Digite o Nome do arquivo:");
        String nome = escanner.next();

        try {
            pessoaJuridicaRepo.persistir(nome);
            return "Dados Salvo no arquivo " + nome + ".juridica.bin";
        } catch (IOException e) {
            e.printStackTrace();
            return "Erro ao Salvar o arquivo " + nome + ".juridica.bin";
        }

    }

    public static String recuperar(Scanner escanner, PessoaJuridicaRepo pessoaJuridicaRepo) {
        System.out.println("Digite o Nome do arquivo:");
        String nome = escanner.next();

        try {
            pessoaJuridicaRepo.recuperar(nome);
            return "Dados de Pessoas Juridica Recuperados!";
        } catch (ClassNotFoundException | IOException e) {
            return "Arquivo não encontrado!";
        }

    }
}
```

2.8CLASSE APP(MAIN)

```
import java.util.Scanner;

import model.PessoaFisicaRepo;
import model.PessoaFisicaService;
import model.PessoaJuridicaRepo;
import model.PessoaJuridicaService;
```

```
public class App {
    public static void main(String[] args) throws Exception {

        Scanner escanner = new Scanner(System.in);
        int opcao = -1;
        String resposta;
        PessoaFisicaRepo pessoaFisicaRepo = new PessoaFisicaRepo();
        PessoaJuridicaRepo pessoaJuridicaRepo = new PessoaJuridicaRepo();

        while (opcao != 0) {
            System.out.println("=================================================");
            System.out.println("1 - Incluir Pessoa");
            System.out.println("2 - Alterar Pessoa");
            System.out.println("3 - Excluir Pessoa");
            System.out.println("4 - Buscar por Id");
            System.out.println("5 - Exibir Todos");
            System.out.println("6 - Persistir Dados");
            System.out.println("7 - Recuperar Dados");
            System.out.println("0 - Finalizar Programa");
            System.out.println("=================================================");
            opcao = escanner.nextInt();

            switch (opcao) {
                case 1:
                    resposta = pessoa(escanner).equals("F")
                            ? PessoaFisicaService.cadastrar(escanner, pessoaFisicaRepo)
                            : PessoaJuridicaService.cadastrar(escanner, pessoaJuridicaRepo);
                    System.out.println(resposta);
                    break;
                case 2:
                    resposta = pessoa(escanner).equals("F")
                            ? PessoaFisicaService.alterar(escanner, pessoaFisicaRepo)
                            : PessoaJuridicaService.alterar(escanner, pessoaJuridicaRepo);
                    System.out.println(resposta);
                    break;
                case 3:
                    resposta = pessoa(escanner).equals("F")
                            ? PessoaFisicaService.excluir(escanner, pessoaFisicaRepo)
                            : PessoaJuridicaService.excluir(escanner, pessoaJuridicaRepo);
                    System.out.println(resposta);
                    break;
                case 4:
                    resposta = pessoa(escanner).equals("F")
                            ? PessoaFisicaService.obter(escanner, pessoaFisicaRepo)
                            : PessoaJuridicaService.obter(escanner, pessoaJuridicaRepo);
                    System.out.println(resposta);
                    break;
                case 5:
                    resposta = pessoa(escanner).equals("F")
                            ? PessoaFisicaService.exibir(escanner, pessoaFisicaRepo)
                            : PessoaJuridicaService.exibir(escanner, pessoaJuridicaRepo);
                    System.out.println(resposta);
```

```
                break;
            case 6:
                resposta = pessoa(escanner).equals("F")
                        ? PessoaFisicaService.salvar(escanner, pessoaFisicaRepo)
                        : PessoaJuridicaService.salvar(escanner, pessoaJuridicaRepo);
                System.out.println(resposta);
                break;
            case 7:
                resposta = pessoa(escanner).equals("F")
                        ? PessoaFisicaService.recuperar(escanner, pessoaFisicaRepo)
                        : PessoaJuridicaService.recuperar(escanner, pessoaJuridicaRepo);
                System.out.println(resposta);
                break;
            case 0:
                System.out.println("Programa Finalizado com Sucesso!");
                System.exit(0);
            default:
                System.out.println("Digite uma das opções entre 0 - 7");
                break;
        }
    }

  }

  private static String pessoa(Scanner escanner) {
    System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
    String selecao = escanner.next().toUpperCase();
    if (!(selecao.equals("F") || selecao.equals("J"))) {
      pessoa(escanner);
    }
    return selecao;
  }

}
```

3. RESULTADOS DA EXECUÇÃO DOS CÓDIGOS

```
==================================================
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
F - Pessoa Fisica | J - Pessoa Juridica
F
Digite o Id da Pessoa:
1
Digite o Nome da Pessoa:
JARDS
Digite o CPF da Pessoa:
1234567890
Digite a Idade da Pessoa:
39
Pessoa Fisica Cadastrada com Sucesso!
================================================
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
================================================
```

4. ANÁLISE E CONCLUSÃO

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

Sabemos que as classes Java são uma espécie de script de construção de objetos. Ou seja, será através das classes que instânciaremos nossos objetos, através da sintaxe "new Classe()". Porém, as classes tornam-se muito mais poderosas quando ganham a possibilidade de terem elementos estáticos. Os elementos estáticos são elementos que "existem", ou seja, estão disponíveis para uso, sem a necessidade de serem instânciados. Isto quer dizer que podem ser utilizados em código sem a necessidade de existirem objetos produzidos (sem a necessidade de um comando "new Classe()").

Como estamos a falar dos métodos, é importante destacar algumas questões: os métodos estáticos não podem utilizar variáveis globais da classe que não sejam estáticas;

os métodos estáticos não devem guardar estado; estes métodos devem, basicamente, possuir lógicas e funcionalidades "auto contidas".

Toda solução Java possui uma classe principal, que é utilizada para correr a aplicação. Isto porque na arquitetura Java, a máquina virtual está preparada para buscar o "ponto de arranque" das aplicações. Este ponto de arranque é o método "main". Entretanto você já observou a assinatura deste método?

```
public static void main(String args){ ... }
```

Isto mesmo, este método possui a palavra "static" em sua assinatura. Esta palavra foi convencionada pela arquitetura, para definir os elementos estáticos.

4.2 Para que serve a classe Scanner?

A classe Scanner tem como objetivo separar a entrada dos textos em blocos, gerando os conhecidos tokens, que são sequências de caracteres separados por delimitadores que por padrão correspondem aos espaços em branco, tabulações e mudança de linha.

Com essa classe podem ser convertidos textos para tipos primitivos, sendo que esses textos podem ser considerados como objetos do tipo String, InputStream e arquivos. Essa classe ajuda na leitura dos dados informados. Para fazer essa ação na prática, é necessário criar um objeto do tipo Scanner que passa como argumento o objeto System.in dentro construtor, do seguinte modo.

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

Através das classes Repo do projeto foi possível se ter uma maior organização do código além de se poder usar um dos pilares da programação orientada a objetos que é o reaproveitamento de código.