Faculdade Estácio - Polo Curitiba - Centro

Curso: Desenvolvimento Full Stack
Disciplina: Desenvolvimento Full Stack
Número da Turma: RPG0014
Semestre Letivo: 3
Integrante: Mariana Lucas Fernandes Onório
Repositório: https://github.com/MariLFO/estacio-mundo3-missao-nivel-1

Sumário:

1. Título da Prática:

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 platafomra 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. Arquivos do Projeto:

Arquivo: Pessoa.java

```java
package model;

import java.io.Serializable;

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

   public Pessoa(){
   }

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

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

Arquivo: PessoaFisica.java

```java
package model;

import java.io.Serializable;

public class PessoaFisica extends Pessoa implements Serializable {
   private String cpf;
   private int idade;

   public PessoaFisica()
   {
   }

   public PessoaFisica(int id, String nome, String cpf, int idade){
       this.id = id;
       this.nome = nome;
       this.cpf = cpf;
       this.idade = idade;
   }

   @Override
   public void exibir() {
       System.out.println("id: "+ this.id + ", nome: "+ this.nome + ", cpf: "+ this.cpf + ",
idade: "+ this.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;
   }
}
```

Arquivo: PessoaJuridica.java

```java
package model;

import java.io.Serializable;

public class PessoaJuridica extends Pessoa implements Serializable {
   private String cnpj;

   public PessoaJuridica()
   {
   }

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

   @Override
   public void exibir() {
       System.out.println("id: "+ this.id + ", nome: "+ this.nome + ", cnpj: "+ this.cnpj);
   }

   public String getCnpj() {
       return cnpj;
   }

   public void setCnpj(String cnpj) {
       this.cnpj = cnpj;
   }
}
```

Arquivo: PessoaFisicaRepo.java

```java
package model;

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

public class PessoaFisicaRepo {
   private ArrayList<PessoaFisica> pessoasFisicas;

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

   public void inserir(PessoaFisica entidade) {
       this.pessoasFisicas.add(entidade);
   }

   public void alterar(PessoaFisica entidade) {
       this.excluir(entidade.id);
       this.inserir(entidade);
   }

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

   public PessoaFisica obter(int id) {
       for (PessoaFisica item : this.pessoasFisicas) {
           if (item.id == id) {
               return item;
           }
       }
       return null;
   }
   public ArrayList<PessoaFisica> obterTodos(){
       return this.pessoasFisicas;
   }

   public void persistir(String nomeArquivo) throws IOException {
       FileOutputStream fos = new FileOutputStream(nomeArquivo);
       ObjectOutputStream oos = new ObjectOutputStream(fos);
       oos.writeObject(this.pessoasFisicas);
   }

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

Arquivo: PessoaJuridicaRepo.java

```java
package model;

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

public class PessoaJuridicaRepo {
   private ArrayList<PessoaJuridica> pessoasJuridicas;

   public PessoaJuridicaRepo(){
       this.pessoasJuridicas = new ArrayList<PessoaJuridica>();
   }

   public void inserir(PessoaJuridica entidade) {
       this.pessoasJuridicas.add(entidade);
   }

   public void alterar(PessoaJuridica entidade){
       this.excluir(entidade.id);
       this.inserir(entidade);
   }

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

   public PessoaJuridica obter(int id) {
       for (PessoaJuridica item : this.pessoasJuridicas) {
           if (item.id == id) {
               return item;
           }
       }
       return null;
   }
   public ArrayList<PessoaJuridica> obterTodos(){
       return this.pessoasJuridicas;
   }

   public void persistir(String nomeArquivo) throws IOException {
       FileOutputStream fos = new FileOutputStream(nomeArquivo);
       ObjectOutputStream oos = new ObjectOutputStream(fos);
       oos.writeObject(this.pessoasJuridicas);
   }

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

Arquivo: Main.java

```
import model.PessoaFisica;
import model.PessoaFisicaRepo;
import model.PessoaJuridica;
import model.PessoaJuridicaRepo;

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

public class Main {
   private static Scanner scanner = new Scanner(System.in);
   private static PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
   private static PessoaJuridicaRepo repo2 = new PessoaJuridicaRepo();

   public static void main(String[] args) {
       int opcao = -1;
       while (opcao != 0) {
           System.out.println("Selecione uma opção:");
           System.out.println("1 - Incluir");
           System.out.println("2 - Alterar");
           System.out.println("3 - Excluir");
           System.out.println("4 - Exibir pelo id");
           System.out.println("5 - Exibir todos");
           System.out.println("6 - Salvar dados");
           System.out.println("7 - Recuperar dados");
           System.out.println("0 - Finalizar a execução");

           opcao = scanner.nextInt();
           switch (opcao) {
               case 1:
                   inserirPessoa();
                   break;
               case 2:
                   alterarPessoa();
                   break;
               case 3:
                   excluirPessoa();
                   break;
               case 4:
                   exibirPessoaPeloId();
                   break;
               case 5:
                   exibirTodasAsPessoas();
                   break;
               case 6:
                   salvarDados();
                   break;
               case 7:
                   recuperarDados();
           }

           System.out.println("---------------------------------\n");
       }
   }

   private static String lerTipoDePessoa() {
       System.out.println("Escolha o tipo:\n\tPara Pessoa Física digite F\n\tPara Pessoa
Jurídica digite J");
       String tipo = scanner.next();
```

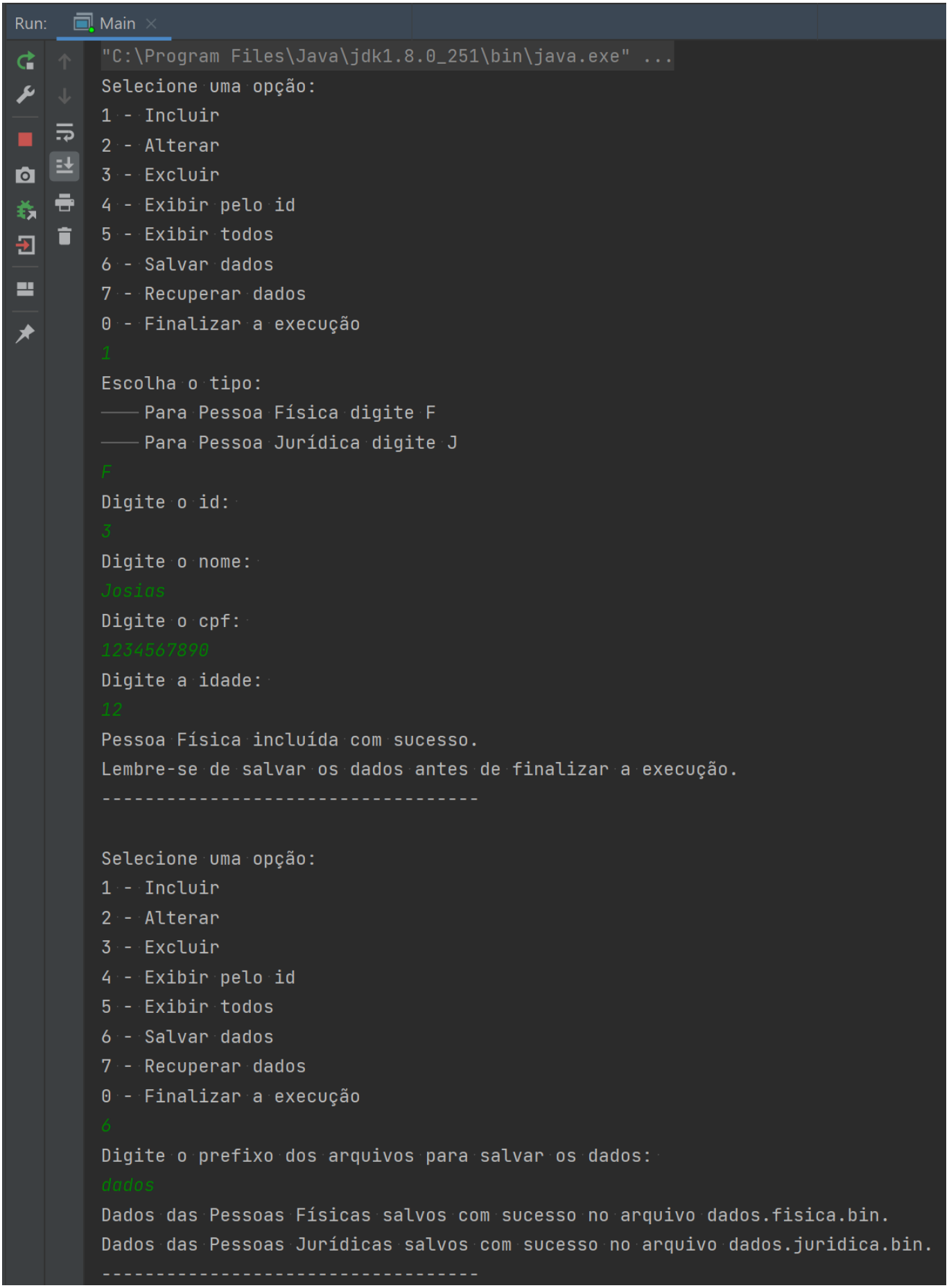
```
        if (tipo.equalsIgnoreCase("F") || tipo.equalsIgnoreCase("J")) {
            return tipo;
        } else {
            System.out.println("Opção inválida, tente novamente.");
            return lerTipoDePessoa();
        }
    }

    private static PessoaFisica definirDadosPessoaFisica(PessoaFisica pessoaFisica) {
        try {
            System.out.println("Digite o id: ");
            pessoaFisica.setId(scanner.nextInt());
            System.out.println("Digite o nome: ");
            pessoaFisica.setNome(scanner.next());
            System.out.println("Digite o cpf: ");
            pessoaFisica.setCpf(scanner.next());
            System.out.println("Digite a idade: ");
            pessoaFisica.setIdade(scanner.nextInt());
            return pessoaFisica;
        } catch (Exception e) {
            System.out.println("Erro ao inserir os dados da Pessoa física:");
            e.printStackTrace();
            System.out.println("Por favor, tente novamente.");
            return null;
        }
    }

    private static PessoaJuridica definirDadosPessoaJuridica(PessoaJuridica pessoaJuridica) {
        try {
            System.out.println("Digite o id: ");
            pessoaJuridica.setId(scanner.nextInt());
            System.out.println("Digite o nome: ");
            pessoaJuridica.setNome(scanner.next());
            System.out.println("Digite o cnpj: ");
            pessoaJuridica.setCnpj(scanner.next());
            return pessoaJuridica;
        } catch (Exception e) {
            System.out.println("Erro ao inserir os dados da Pessoa Jurídica:");
            e.printStackTrace();
            System.out.println("Por favor, tente novamente.");
            return null;
        }
    }

    private static void lembreteSalvarDados() {
        System.out.println("Lembre-se de salvar os dados antes de finalizar a execução.");
    }

    private static void inserirPessoa() {
        String tipo = lerTipoDePessoa();

        if (tipo.equalsIgnoreCase("F")) {
            PessoaFisica pessoaFisica = definirDadosPessoaFisica(new PessoaFisica());
            if (pessoaFisica != null) {
                repo1.inserir(pessoaFisica);
                System.out.println("Pessoa Física incluída com sucesso.");
            }
        } else if (tipo.equalsIgnoreCase("J")) {
            PessoaJuridica pessoaJuridica = definirDadosPessoaJuridica(new PessoaJuridica());
            if (pessoaJuridica != null) {
                repo2.inserir(pessoaJuridica);
                System.out.println("Pessoa Jurídica incluída com sucesso.");
```

```
            }
        }

        lembreteSalvarDados();
    }

    private static void alterarPessoa() {
        String tipo = lerTipoDePessoa();

        if (tipo.equalsIgnoreCase("F")) {
            System.out.println("Digite o id da Pessoa Física que deseja alterar: ");
            int idPessoaFisica = scanner.nextInt();
            PessoaFisica pessoaFisica = repo1.obter(idPessoaFisica);
            if (pessoaFisica != null) {
                pessoaFisica.exibir();
                pessoaFisica = definirDadosPessoaFisica(pessoaFisica);
                if (pessoaFisica != null) {
                    System.out.println("Pessoa Física alterada com sucesso.");
                    repo1.alterar(pessoaFisica);
                }
            } else {
                System.out.println("Pessoa Física não encontrada.");
            }
        } else if (tipo.equalsIgnoreCase("J")) {
            System.out.println("Digite o id da Pessoa Jurídica que deseja alterar: ");
            int idPessoaJuridica = scanner.nextInt();
            PessoaJuridica pessoaJuridica = repo2.obter(idPessoaJuridica);
            if (pessoaJuridica != null) {
                pessoaJuridica.exibir();
                pessoaJuridica = definirDadosPessoaJuridica(pessoaJuridica);
                if (pessoaJuridica != null) {
                    repo2.alterar(pessoaJuridica);
                    System.out.println("Pessoa Jurídica alterada com sucesso.");
                }
            } else {
                System.out.println("Pessoa Jurídica não encontrada.");
            }
        }

        lembreteSalvarDados();
    }

    private static void excluirPessoa() {
        String tipo = lerTipoDePessoa();
        if (tipo.equalsIgnoreCase("F")) {
            System.out.println("Digite o id da Pessoa Física que deseja excluir: ");
            int idPessoaFisica = scanner.nextInt();
            PessoaFisica pessoaFisica = repo1.obter(idPessoaFisica);
            if (pessoaFisica != null) {
                repo1.excluir(idPessoaFisica);
                System.out.println("Pessoa Física excluída com sucesso.");
            } else {
                System.out.println("Pessoa Física não encontrada.");
            }
        } else if (tipo.equalsIgnoreCase("J")) {
            System.out.println("Digite o id da Pessoa Jurídica que deseja excluir: ");
            int idPessoaJuridica = scanner.nextInt();
            PessoaJuridica pessoaJuridica = repo2.obter(idPessoaJuridica);
            if (pessoaJuridica != null) {
                repo2.excluir(idPessoaJuridica);
                System.out.println("Pessoa Jurídica excluída com sucesso.");
            } else {
```

```
                System.out.println("Pessoa Jurídica não encontrada.");
            }
        }

        lembreteSalvarDados();
    }

    private static void exibirPessoaPeloId() {
        String tipo = lerTipoDePessoa();

        if (tipo.equalsIgnoreCase("F")) {
            System.out.println("Digite o id da Pessoa Física que deseja exibir: ");
            int idPessoaFisica = scanner.nextInt();
            PessoaFisica pessoaFisica = repo1.obter(idPessoaFisica);
            if (pessoaFisica != null) {
                pessoaFisica.exibir();
            } else {
                System.out.println("Pessoa Física não encontrada.");
            }
        } else if (tipo.equalsIgnoreCase("J")) {
            System.out.println("Digite o id da Pessoa Jurídica que deseja exibir: ");
            int idPessoaJuridica = scanner.nextInt();
            PessoaJuridica pessoaJuridica = repo2.obter(idPessoaJuridica);
            if (pessoaJuridica != null) {
                pessoaJuridica.exibir();
            } else {
                System.out.println("Pessoa Jurídica não encontrada.");
            }
        }
    }

    private static void exibirTodasAsPessoas() {
        String tipo = lerTipoDePessoa();

        if (tipo.equalsIgnoreCase("F")) {
            ArrayList<PessoaFisica> pessoasFisicas = repo1.obterTodos();
            if (pessoasFisicas != null) {
                System.out.println("Exibindo todos os registros de Pessoas Físicas:\n");
                for (PessoaFisica pessoaFisica : pessoasFisicas) {
                    pessoaFisica.exibir();
                }
            } else {
                System.out.println("Não existem Pessoas Físicas cadastradas.");
            }
        } else if (tipo.equalsIgnoreCase("J")) {
            ArrayList<PessoaJuridica> pessoasJuridicas = repo2.obterTodos();
            if (pessoasJuridicas != null) {
                System.out.println("Exibindo todos os registros de Pessoas Jurídicas.");
                for (PessoaJuridica pessoaJuridica : pessoasJuridicas) {
                    pessoaJuridica.exibir();
                }
            } else {
                System.out.println("Não existem Pessoas Jurídicas cadastradas.");
            }
        }
    }

    private static void salvarDados() {
        System.out.println("Digite o prefixo dos arquivos para salvar os dados: ");
        String nomeArquivo = scanner.next();

        try {
```

```
            String nomeArquivoPessoaFisica = nomeArquivo + ".fisica.bin";
            repo1.persistir(nomeArquivoPessoaFisica);
            System.out.println("Dados das Pessoas Físicas salvos com sucesso no arquivo " +
nomeArquivoPessoaFisica + ".");
        } catch (IOException e) {
            System.out.println("Não foi possível salvar os dados das Pessoas Físicas.\nErro:
" + e.getMessage());
        }

        try {
            String nomeArquivoPessoaJuridica = nomeArquivo + ".juridica.bin";
            repo2.persistir(nomeArquivoPessoaJuridica);
            System.out.println("Dados das Pessoas Jurídicas salvos com sucesso no arquivo " +
nomeArquivoPessoaJuridica + ".");
        } catch (IOException e) {
            System.out.println("Não foi possível salvar os dados das Pessoas
Jurídicas.\nErro: " + e.getMessage());
        }
    }

    private static void recuperarDados() {
        System.out.println("Digite o prefixo dos arquivos para salvar os dados: ");
        String nomeArquivo = scanner.next();

        try {
            String nomeArquivoPessoaFisica = nomeArquivo + ".fisica.bin";
            repo1.recuperar(nomeArquivoPessoaFisica);
            System.out.println("Pessoas Físicas recuperadas com sucesso do arquivo " +
nomeArquivoPessoaFisica + ".");
        } catch (Exception e) {
            System.out.println("Erro ao recuperar Pessoas Físicas.\nErro: " +
e.getMessage());
        }

        try {
            String nomeArquivoPessoaJuridica = nomeArquivo + ".juridica.bin";
            repo2.recuperar(nomeArquivoPessoaJuridica);
            System.out.println("Pessoas Jurídicas recuperadas com sucesso do arquivo " +
nomeArquivoPessoaJuridica + ".");
        } catch (Exception e) {
            System.out.println("Erro ao recuperar Pessoas Jurídicas.\nErro: " +
e.getMessage());
        }
    }
}
```

4. Resultados da execução dos códigos:



5. Análise e Conclusão:

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

Resposta: Elementos estáticos são métodos ou atributos que pertencem diretamente à classe e não aos objetos dessa classe, sendo assim, para acessá-los utilizamos o nome da classe seguido do elemento em questão, ao invés de acessá-los através de uma instância dessa classe.

Um exemplo de método estático é o método main da classe Main nesse trabalho.

b. Para que serve a classe Scanner?

Resposta: Essa classe é utilizada para entrada de dados do usuário, lendo e processando instruções digitadas através do teclado. Ela pode ler diferentes tipos de dados, como inteiros, números, floats, strings (texto), entre outros.

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

Resposta: Utilizar a classes de repositório, além de melhorar a estrutura do código, proporcionou uma melhor separação de responsabilidades, pois os repositórios ficaram apenas responsáveis por manipular coleções de objetos, inserindo, alterando e excluindo objetos de uma lista, enquanto as operações específicas do objeto em si, como alterar um campo, são tratadas exclusivamente nas classes Pessoa, PessoaFisica ou PessoaJuridica.