Título da Prática;

Objetivo da Prática;

Todos os códigos solicitados neste roteiro de aula;

Os resultados da execução dos códigos também devem ser apresentados;

Análise e Conclusão:

Quais as vantagens e desvantagens do uso de herança?

Vantagens da Herança:

Reutilização de código: A herança permite que você reutilize código existente de classes pai (ou superclasses) em classes filhas (ou subclasses). Isso pode economizar tempo e esforço de desenvolvimento, pois você não precisa reescrever funcionalidades comuns.

Hierarquia de classes: A herança permite criar hierarquias de classes, onde classes mais específicas (subclasses) herdam características gerais de classes mais abstratas (superclasses). Isso ajuda na organização e estruturação do código.

Polimorfismo: A herança suporta polimorfismo, o que significa que objetos de subclasses podem ser tratados como objetos de superclasses. Isso permite escrever código mais genérico e flexível, onde diferentes objetos podem ser tratados de maneira uniforme.

Desvantagens da Herança:

Acoplamento forte: A herança cria um acoplamento forte entre a classe pai e a classe filha. Qualquer alteração na classe pai pode afetar todas as subclasses, o que torna o código mais frágil e difícil de manter.

Herança múltipla complexa: Em algumas linguagens de programação, como C++, a herança múltipla pode levar a problemas de ambiguidade e complexidade. Isso ocorre quando uma classe herda de várias classes ao mesmo tempo, o que pode tornar o código confuso.

Inflexibilidade: A herança define uma relação permanente entre a classe pai e a classe filha. Isso pode ser problemático quando as necessidades de design mudam, pois pode ser difícil modificar a hierarquia de classes sem causar efeitos colaterais.

Violação do princípio da substituição de Liskov: A herança mal projetada pode violar o princípio da substituição de Liskov, que estabelece que as subclasses devem ser substituíveis por suas superclasses sem afetar o comportamento do programa. Uma má hierarquia de herança pode quebrar essa regra.

Complexidade desnecessária: Às vezes, a herança é usada quando a composição (outra técnica de reutilização de código) seria mais apropriada. Isso pode adicionar complexidade desnecessária ao código.

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

Conversão de objetos em bytes: A interface Serializable fornece um mecanismo padrão para serializar objetos, ou seja, converter objetos em uma sequência de bytes. Essa serialização é importante para armazenar objetos em arquivos binários, pois os arquivos binários lidam com dados brutos, enquanto os objetos em memória contêm informações complexas e estruturas de dados.

Transferência de objetos pela rede: Além de persistência em arquivos, a serialização também é útil quando você precisa transmitir objetos pela rede. Ao serializar um objeto, você pode enviá-lo através da rede como uma sequência de bytes e, em seguida, desserializá-lo do lado do receptor.

Persistência de estado: Quando você deseja salvar o estado de um objeto para que ele possa ser recuperado posteriormente, a serialização é uma técnica eficaz. Isso é comumente usado em aplicativos que precisam salvar o estado do usuário, como jogos, aplicativos de produtividade, entre outros.

Facilita a interoperabilidade: A interface Serializable é uma convenção amplamente adotada em linguagens como Java, o que significa que objetos serializados em uma plataforma podem ser desserializados em outra plataforma que também suporta a serialização. Isso ajuda na interoperabilidade entre diferentes sistemas e linguagens.

Suporte a bibliotecas e estruturas de persistência: Muitas bibliotecas e frameworks de persistência, como Hibernate em Java, usam a serialização para armazenar objetos em bancos de dados ou em sistemas de armazenamento de arquivos binários. Ao implementar a interface Serializable, você permite que essas bibliotecas persistam seus objetos de maneira eficaz.

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

Operações de alto nível: A API Stream oferece uma série de operações de alto nível que permitem que você processe e transforme coleções de dados de maneira declarativa. Essas operações incluem map, filter, reduce, flatMap, distinct, sorted, entre outras. Essas operações permitem que você defina o que deseja fazer com os dados, em vez de como fazê-lo.

Expressões lambda: Você pode passar funções lambda como argumentos para as operações de Stream. Isso permite que você especifique o comportamento que deseja aplicar aos elementos da coleção de uma maneira concisa e flexível. Por exemplo, ao usar map ou filter, você pode passar funções lambda que descrevem como transformar ou filtrar os elementos.

Exemplo usando map:

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

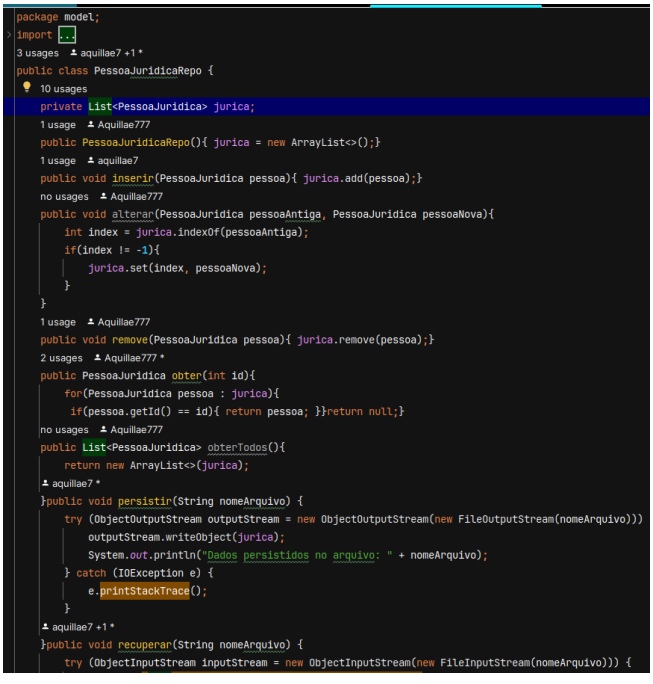
Padrão DAO (Data Access Object): O padrão DAO é usado para encapsular todas as operações de acesso a dados em uma interface e suas implementações concretas. Isso permite que você separe a lógica de negócios da lógica de acesso a dados e torna o código mais modular e testável. O DAO define métodos para criar, ler, atualizar e excluir registros de dados, por exemplo.

Exemplo de interface DAO:

```
public interface UserDao {

    User getById(int id);

    void save(User user);

    void update(User user);

    void delete(int id);

}
```

```java
import java.io.*;

5 usages  Aquillae777 +1
public class PessoaFisicaRepo {
    /* Lista privada para armazenar pessoas */
    10 usages
    private List<PessoaFIsica> pessoas;

    2 usages  Aquillae777
    public PessoaFisicaRepo(){
        pessoas = new ArrayList<>();

    }

    1 usage  Aquillae777
    public void inserir(PessoaFIsica pessoa) { pessoas.add(pessoa); }

    1 usage  Aquillae777
    public void alterar(PessoaFIsica pessoaAntiga, PessoaFIsica pessoaNova){
        int index = pessoas.indexOf(pessoaAntiga);
        if (index != -1){
            pessoas.set(index,pessoaNova);
        }
    }
    1 usage  Aquillae777
    public void excluir(PessoaFIsica pessoa) { pessoas.remove(pessoa); }

    1 usage  Aquillae777
    public PessoaFIsica obter(int id){
            for(PessoaFIsica pessoa : pessoas){
            if(pessoa.getId() == id){
                return pessoa;
            }
            }
        }
        }
        return null;
    }
    1 usage  Aquillae777 +1
    public List<PessoaFIsica> obterTodos() { return Collections.unmodifiableList(pessoas); }

    1 usage  aquillae7
    public void persistir(String nomeArquivo) {
        try (ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
            outputStream.writeObject(pessoas);
            System.out.println("Dados persistidos no arquivo: " + nomeArquivo);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }


    1 usage  aquillae7
    public void recuperar(String nomeArquivo) {
        try (ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
            pessoas = (List<PessoaFIsica>) inputStream.readObject();
            System.out.println("Dados de Pessoa Física recuperados do arquivo: " + nomeArquivo);
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
```

```
    6 usages   aquillae7
    public void imprimirTodos() {
        for (PessoaFIsica pessoa : pessoas) {
            System.out.println(pessoa); // Supondo que sua classe PessoaFisica tenha um método toString adequado
        }
    }

}
```



```
package model;
import ...
3 usages   aquillae7 +1 *
public class PessoaJuridicaRepo {
    10 usages
    private List<PessoaJuridica> jurica;
    1 usage   Aquillae777
    public PessoaJuridicaRepo(){ jurica = new ArrayList<>();}
    1 usage   aquillae7
    public void inserir(PessoaJuridica pessoa){ jurica.add(pessoa);}
    no usages   Aquillae777
    public void alterar(PessoaJuridica pessoaAntiga, PessoaJuridica pessoaNova){
        int index = jurica.indexOf(pessoaAntiga);
        if(index != -1){
            jurica.set(index, pessoaNova);
        }
    }
    1 usage   Aquillae777
    public void remove(PessoaJuridica pessoa){ jurica.remove(pessoa);}
    2 usages   Aquillae777 *
    public PessoaJuridica obter(int id){
        for(PessoaJuridica pessoa : jurica){
         if(pessoa.getId() == id){ return pessoa; }}return null;}
    no usages   Aquillae777
    public List<PessoaJuridica> obterTodos(){
        return new ArrayList<>(jurica);
    aquillae7 *
    }public void persistir(String nomeArquivo) {
        try (ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
            outputStream.writeObject(jurica);
            System.out.println("Dados persistidos no arquivo: " + nomeArquivo);
        } catch (IOException e) {
            e.printStackTrace();
        }
    aquillae7 +1 *
    }public void recuperar(String nomeArquivo) {
        try (ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
```

```
    aquillae7 +1 *
    }public void recuperar(String nomeArquivo) {
        try (ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
            jurica = (List<PessoaJuridica>) inputStream.readObject();
            System.out.println("Dados de Pessoa Jurídica recuperados do arquivo: " + nomeArquivo);
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    aquillae7 *
    }public void imprimirTodos(){
        for(PessoaJuridica pessoa : jurica){
            System.out.println(pessoa);
        }
    }
}
```

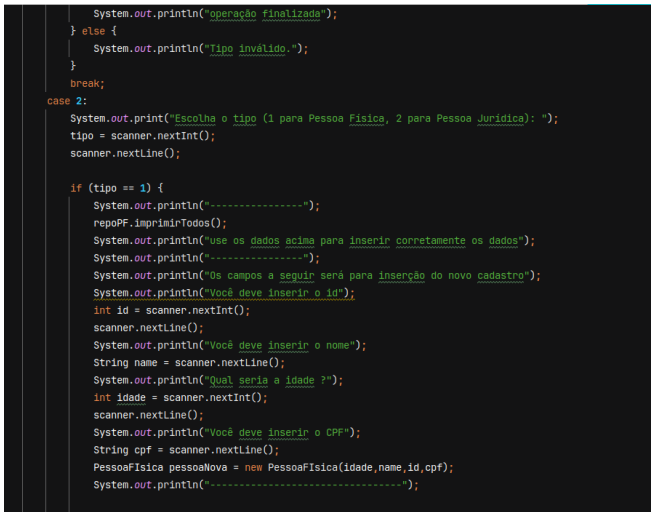
```
package model;

import java.io.Serializable;

15 usages   Aquillae777 +1
public class PessoaJuridica extends Pessoa implements Serializable {
    5 usages
    String cnpj;

    2 usages   Aquillae777
```

```java
    public PessoaJuridica(int idade, String name, int id, String cnpj ) {
        super(idade, name, id);
        this.cnpj = cnpj;

    }

    no usages  Aquillae777
    public String getCnpj() { return cnpj; }

    1 usage  Aquillae777 +1
    public void setCnpj(String novoCNPJ) { this.cnpj = cnpj ; }


    4 usages  aquillae7
    @Override
    public String toString() {
        return "[\n" +
                "  id = " + id + "\n" +
                "  Nome = " + nome + "\n" +
                "  Cnpj = " + cnpj + "\n" +
                "  Idade = " + idade + "\n" +
                "]";
    }
```

```java
public class Main {
    aquillae7 +1
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        PessoaFisicaRepo repoPF = new PessoaFisicaRepo();
         PessoaJuridicaRepo repoPJ = new PessoaJuridicaRepo();
        String nomeArquivoPF = "pessoas_fisicas.bin";
        String nomeArquivoPJ = "pessoas_juridicas.bin";

        int opcao;

        do {
            System.out.println("=== Menu de Opções ===");
            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");

            System.out.print("Digite a opção desejada: ");
            opcao = scanner.nextInt();
            scanner.nextLine(); // Consumir a quebra de linha

            switch (opcao) {
                case 1:
                    System.out.print("Escolha o tipo (1 para Pessoa Física, 2 para Pessoa Jurídica): ");
                    int tipo = scanner.nextInt();
                    scanner.nextLine(); // Consumir a quebra de linha
```

```
            if (tipo == 1) {
                System.out.println("Você deve inserir o id");
                int id = scanner.nextInt();
```

```
                int id = scanner.nextInt();
                scanner.nextLine();
                System.out.println("Você deve inserir o nome");
                String nome = scanner.nextLine();
                System.out.println("Qual seria a idade ?");
                int idade = scanner.nextInt();
                scanner.nextLine();
                System.out.println("Você deve inserir o CPF");
                String cpf = scanner.nextLine();
                PessoaFIsica pessoa = new PessoaFIsica(idade, nome, id, cpf);
                repoPF.inserir(pessoa);
                repoPF.imprimirTodos();


                repoPF.obterTodos();
                System.out.println("operação finalizada");

            } else if (tipo == 2) {
                System.out.println("Você deve inserir id");
                int id = scanner.nextInt();
                System.out.println("Você deve inserir o nome");
                String nome = scanner.next();
                System.out.println("Qual seria a idade ?");
                int idade = scanner.nextInt();;
                System.out.println("você deve inserir o cpf");
                String cnpj = scanner.next();
                PessoaJuridica pesssoa = new PessoaJuridica(idade,nome,id,cnpj);
                repoPJ.inserir(pesssoa);


                System.out.println("operação finalizada");
            } else {
                System.out.println("Tipo inválido.");
            }
            break;
        case 2:
            System.out.print("Escolha o tipo (1 para Pessoa Física, 2 para Pessoa Jurídica): ");
```



```
            PessoaFIsica pessoaAntiga = new PessoaFIsica(idade,name,id,cpf);
            repoPF.alterar(pessoaAntiga, pessoaNova);

        } else if (tipo == 2) {
            // Pedir ao usuário o ID da pessoa jurídica que deseja alterar
            System.out.println("Informe o ID da pessoa jurídica que deseja alterar:");
```

```
     PessoaFIsica pessoaAntiga = new PessoaFIsica(idade,name,id,cpf);
     repoPF.alterar(pessoaAntiga, pessoaNova);

 } else if (tipo == 2) {
     // Pedir ao usuário o ID da pessoa jurídica que deseja alterar
     System.out.println("Informe o ID da pessoa jurídica que deseja alterar:");
     int idParaAlterar = scanner.nextInt();
     scanner.nextLine(); // Consumir a quebra de linha

     PessoaJuridica pessoaExistente = repoPJ.obter(idParaAlterar);

     if (pessoaExistente != null) {
         System.out.println("Informe os novos dados:");

         System.out.println("Você deve inserir o nome:");
         String novoNome = scanner.nextLine();

         System.out.println("Qual seria a idade?");
         int novaIdade = scanner.nextInt();
         scanner.nextLine(); // Consumir a quebra de linha

         System.out.println("Você deve inserir o novo CPF:");
         String novoCNPJ = scanner.nextLine();

         // Atualizar apenas os campos relevantes da pessoa jurídica existente
         pessoaExistente.setNome(novoNome);
         pessoaExistente.setIdade(novaIdade);
         pessoaExistente.setCnpj(novoCNPJ);

         System.out.println("Pessoa jurídica atualizada com sucesso!");
     } else {
         System.out.println("Pessoa jurídica com o ID informado não encontrada.");
     }
```

```
    } else {
        System.out.println("Tipo inválido.");
    }
    break;
case 3:
    System.out.print("Escolha o tipo (1 para Pessoa Fisica, 2 para Pessoa Juridica): ");
    tipo = scanner.nextInt();
    scanner.nextLine(); // Consumir a quebra de linha

    if (tipo == 1) {
        System.out.println("----------------");
        repoPF.imprimirTodos();
        System.out.println("use os dados acima para inserir corretamente os dados");
        System.out.println("----------------");
        System.out.println("Você deve inserir o id");
        int id = scanner.nextInt();
        scanner.nextLine();
        System.out.println("Você deve inserir o nome");
        String nome = scanner.nextLine();
        System.out.println("Qual seria a idade ?");
        int idade = scanner.nextInt();
        scanner.nextLine();
        System.out.println("Você deve inserir o CPF");
        String cpf = scanner.nextLine();

        repoPF.excluir(new PessoaFIsica(idade,nome,id,cpf));
    } else if (tipo == 2) {
        System.out.println("----------------");
        repoPF.imprimirTodos();
        System.out.println("use os dados acima para inserir corretamente os dados");
        System.out.println("----------------");
        System.out.println("Você deve inserir o id");
        int id = scanner.nextInt();
        scanner.nextLine();
        System.out.println("Você deve inserir o nome");
        String nome = scanner.nextLine();
```

```
        String nome = scanner.nextLine();
        System.out.println("Qual seria a idade ?");
        int idade = scanner.nextInt();
        scanner.nextLine();
        System.out.println("Você deve inserir o CPF");
        String cnpj = scanner.nextLine();

        repoPJ.remove(new PessoaJuridica(idade,nome,id,cnpj));
    } else {
        System.out.println("Tipo inválido.");
    }
    break;
case 4:
    System.out.print("Escolha o tipo (1 para Pessoa Fisica, 2 para Pessoa Juridica): ");
```

```
            tipo = scanner.nextInt();
            scanner.nextLine(); // Consumir a quebra de linha

            if (tipo == 1) {
                System.out.println("Informe o ID da pessoa jurídica que deseja alterar:");
                int idParaAlterar = scanner.nextInt();
                scanner.nextLine();

                repoPF.obter(idParaAlterar);
                System.out.println();

            } else if (tipo == 2) {
                System.out.println("Informe o ID da pessoa jurídica que deseja alterar:");
                int idParaAlterar = scanner.nextInt();
                repoPJ.obter(idParaAlterar);
            } else {
                System.out.println("Tipo inválido.");
            }
            break;
        case 5:
            System.out.print("Escolha o tipo (1 para Pessoa Física, 2 para Pessoa Jurídica): ");
            tipo = scanner.nextInt();
            scanner.nextLine(); // Consumir a quebra de linha
```

```
                        System.out.println("----------------------");
                        repoPF.imprimirTodos();
                        System.out.println("----------------------");


                    } else if (tipo == 2) {
                        System.out.println("----------------------");
                        repoPJ.imprimirTodos();
                        System.out.println("----------------------");

                    } else {
                        System.out.println("Tipo inválido.");
                    }
                    break;
                case 6:
                    repoPF.persistir(nomeArquivoPF);
                    repoPJ.persistir(nomeArquivoPJ);
                    System.out.println("Dados persistidos nos arquivos.");
                    break;
                case 7:
                    repoPF.recuperar(nomeArquivoPF);
                    repoPJ.recuperar(nomeArquivoPJ);
                    System.out.println("Dados recuperados dos arquivos.");
                    break;
                case 0:
                    System.out.println("Finalizando a execução.");
                    break;
                default:
                    System.out.println("Opção inválida. Digite novamente.");
            }
        } while (opcao != 0);

        scanner.close();
    }
}
```

```
public abstract class Pessoa implements Serializable {
    6 usages
    int idade;
    6 usages
    String nome;
    4 usages
    int id;

    2 usages  Aquillae777
    public Pessoa(int idade, String nome, int id){
        this.idade = idade;
        this.nome = nome;
        this.id = id;

    }

    4 usages  2 overrides  Aquillae777
    public String toString (){
        return "Objeto:" + "\n\t- Classe: " + getClass().getName() + "\n\t- Hash: " + Integer.toHexString(hashCode()) + "\n\t- Nome: " + nome ;
    }

    no usages  Aquillae777
```

```java
public int getIdade() { return idade; }

1 usage  Aquillae777
public void setIdade(int Idade) { this.idade = idade; }

no usages  Aquillae777
public String getNome() { return nome; }

1 usage  Aquillae777
public void setNome(String nome) { this.nome = nome; }

2 usages  Aquillae777
public int getId() { return id; }
```