INSTITUTO FEDERAL
Pernambuco

Campus Garanhuns

S.O.L.I.D

PRINCÍPIO DE SEGREGAÇÃO DE INTERFACE

ISP

- **O Princípio da Segregação de Interface (Interface Segregation Principle - ISP) é um dos cinco princípios do SOLID, um conjunto de princípios de design de software que visam promover a manutenção, extensibilidade e reutilização de código.**

- **O ISP afirma que os clientes não devem ser forçados a depender de interfaces que não utilizam. Em outras palavras, as interfaces devem ser específicas para cada cliente, atendendo apenas às suas necessidades.**

ISP

- **A ideia de que é melhor ter interfaces menores e mais específicas em vez de uma única interface grande e genérica. Isso evita que as classes implementem métodos que não são relevantes para elas, reduzindo a complexidade e o acoplamento entre as classes.**

- **Ao seguir o ISP, as interfaces devem ser projetadas de forma granular, de modo que cada classe cliente dependa apenas das interfaces que são necessárias para o seu funcionamento. Isso permite que as classes sejam mais coesas, ou seja, que tenham uma única responsabilidade bem definida.**

ISP

- **Além disso, o ISP promove a flexibilidade e a extensibilidade do código, pois as classes podem implementar apenas as interfaces relevantes para elas. Isso torna mais fácil adicionar novas funcionalidades ao sistema sem afetar as classes existentes.**

- **Em resumo, o Princípio da Segregação de Interface sugere que as interfaces devem ser específicas e coesas, atendendo apenas às necessidades dos clientes individuais. Isso promove um design de software mais flexível, extensível e de fácil manutenção.**

EXEMPLO

- **Exemplo de um sistema de gerenciamento de documentos. Sem ISP:**

```java
package Com_ISP.Interfaces;

interface UsuarioFuncoes {
    void adicionarDocumento(String documento);
    void removerDocumento(String documento);
    void atribuirPermissao(String documento, String usuario, String permissao);
    void editarDocumento(String documento);
    void visualizarDocumento(String documento);
}
```

EXEMPLO

- **Exemplo de um sistema de gerenciamento de documentos. Com ISP:**

```java
package Com_ISP.Interfaces;

import Com_ISP.Entidades.Documento;
import Com_ISP.Entidades.Permissao;
import Com_ISP.Entidades.Usuario;

public interface AdminFuncoes {
    void adicionarDocumento(Documento documento);
    void removerDocumento(Documento documento);
    void atribuirPermissao(Documento documento, Usuario usuario, Permissao permissao);
}
```

```java
package Com_ISP.Interfaces;

import Com_ISP.Entidades.Documento;

public interface EditorFuncoes {
    void editarDocumento(Documento documento);
}
```

```java
package Com_ISP.Interfaces;

import Com_ISP.Entidades.Documento;

public interface VisualizadorFuncoes {
    void visualizarDocumento(int documento);
}

```

EXEMPLO

```java
public class Administrador implements AdminFuncoes, EditorFuncoes, VisualizadorFuncoes {

    ArrayList<Documento> documentos = new ArrayList<>();
    @Override
    public void adicionarDocumento(Documento documento) {
        documentos.add(documento);
        System.out.println(x:"Documento adicionado por um administrador: ");
    }

    @Override
    public void removerDocumento(Documento documento) {
        documentos.remove(documento);
        System.out.println(x:"Documento removido por um administrador: ");
    }

    @Override
    public void atribuirPermissao(Documento documento, Usuario usuario, Permissao permissao) {
        System.out.println("Permissão atribuída por um administrador - Documento: " + documento
                + ", Usuário: " + usuario + ", Permissão: " + permissao);
    }

    @Override
    public void editarDocumento(Documento documento) {
        documentos.add(documento.id,documento);
        System.out.println(x:"Documento editado por um administrador: ");
    }

    @Override
    public void visualizarDocumento(int documento) {

        System.out.println("ID: "+ documentos.get(documento).getId() +" Titulo: "+ documentos.ge
    }
```

```java
// Classe para representar um editor
public class Editor implements EditorFuncoes, VisualizadorFuncoes {
    ArrayList<Documento> documentos = null;
    @Override
    public void editarDocumento(Documento documento) {
        System.out.println("Documento editado por um editor: "
         + documento);
    }

    @Override
    public void visualizarDocumento(int documento) {
        System.out.println("Documento visualizado por um editor: "
         + documentos.get(documento).getTitulo());
    }
}
```

```java
public class Visualizador implements VisualizadorFuncoes {
    ArrayList<Documento> documentos = null;

    @Override
        public void visualizarDocumento(int documento) {
            System.out.println("Documento visualizado por um visualizador: "
             + documentos.get(documento).getTitulo());
        }

    }
```