



Arquitetura em camadas

DC – UFRPE
Programação II

Prof. Gustavo Callou
gustavo.callou@ufrpe.br

Roteiro

- ▶ Arquitetura em Camadas



Arquitetura de software

- ▶ Modelo de alto nível de um sistema de software
 - ▶ Componentes (de software)
 - ▶ Interfaces (serviços visíveis)
 - ▶ Papéis, relacionamentos e interações entre componentes



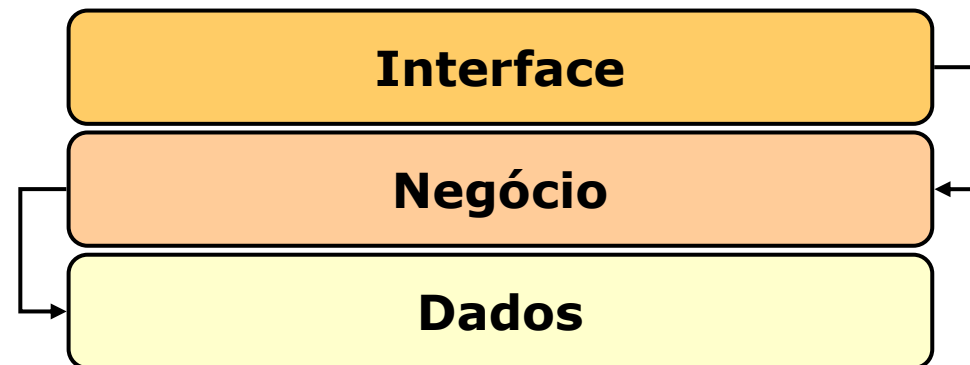
Arquitetura de software



- ▶ Um sistema pode aderir a um ou mais estilos de arquitetura
 - ▶ Cliente-Servidor, Camadas, Pipelining, MVC, etc
- ▶ **Interfaces** são fundamentais para definir papéis dentro da arquitetura

Arquitetura em camadas

- ▶ Organiza o sistema em um conjunto de camadas lógicas com interfaces definidas
- ▶ Camada superior é cliente da camada inferior
- ▶ Principal camada é a de negócio
- ▶ A seguir uma arquitetura com 3 camadas



Arquitetura em três camadas

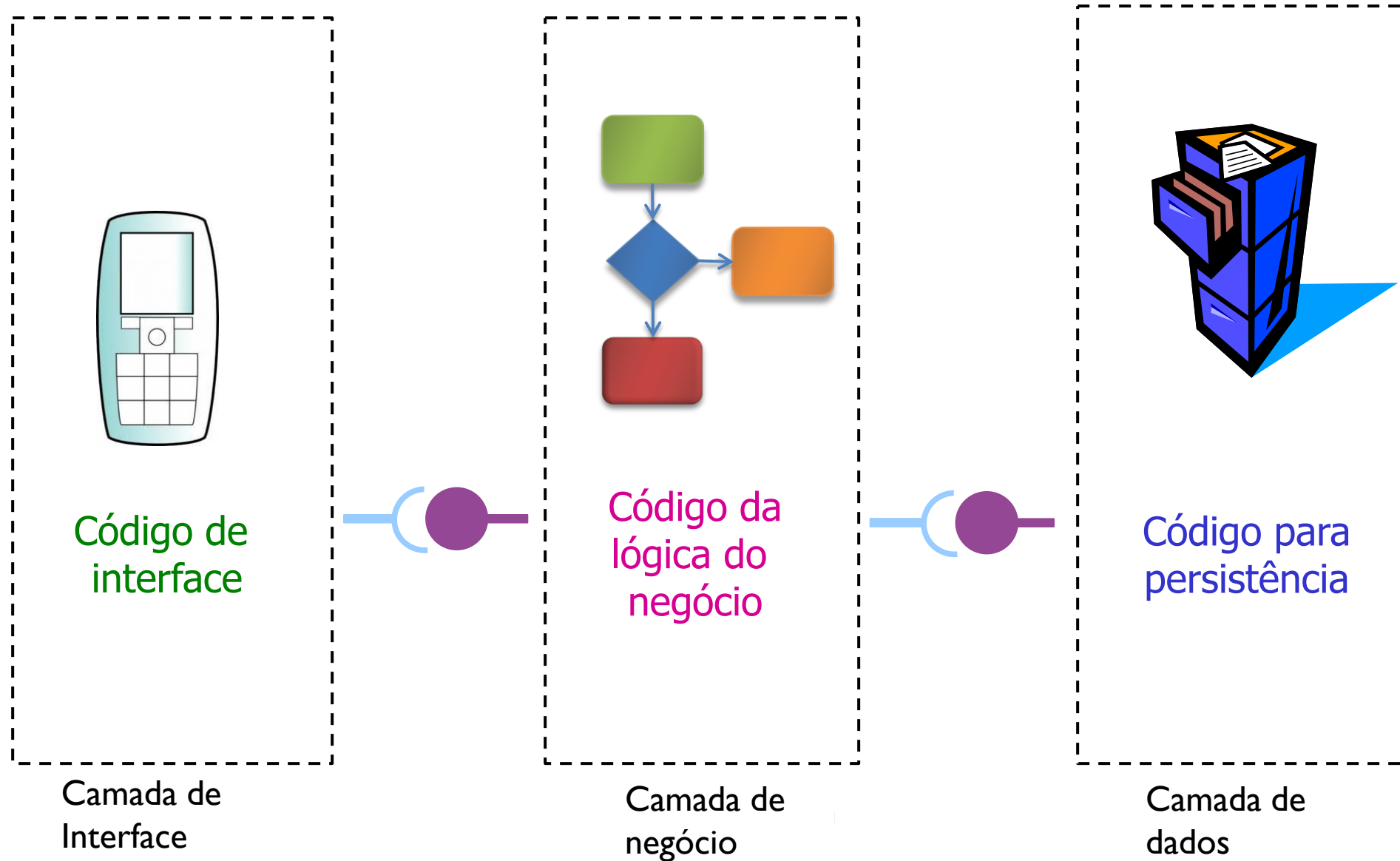
- ▶ Uma forma de organizar o código das camadas é colocar o código em pacotes ou projetos diferentes
- ▶ Exemplo: no sistema do banco cada camada está em um pacote separado

`br.ufrpe.poo.banco.gui`

`br.ufrpe.poo.banco.negocio`

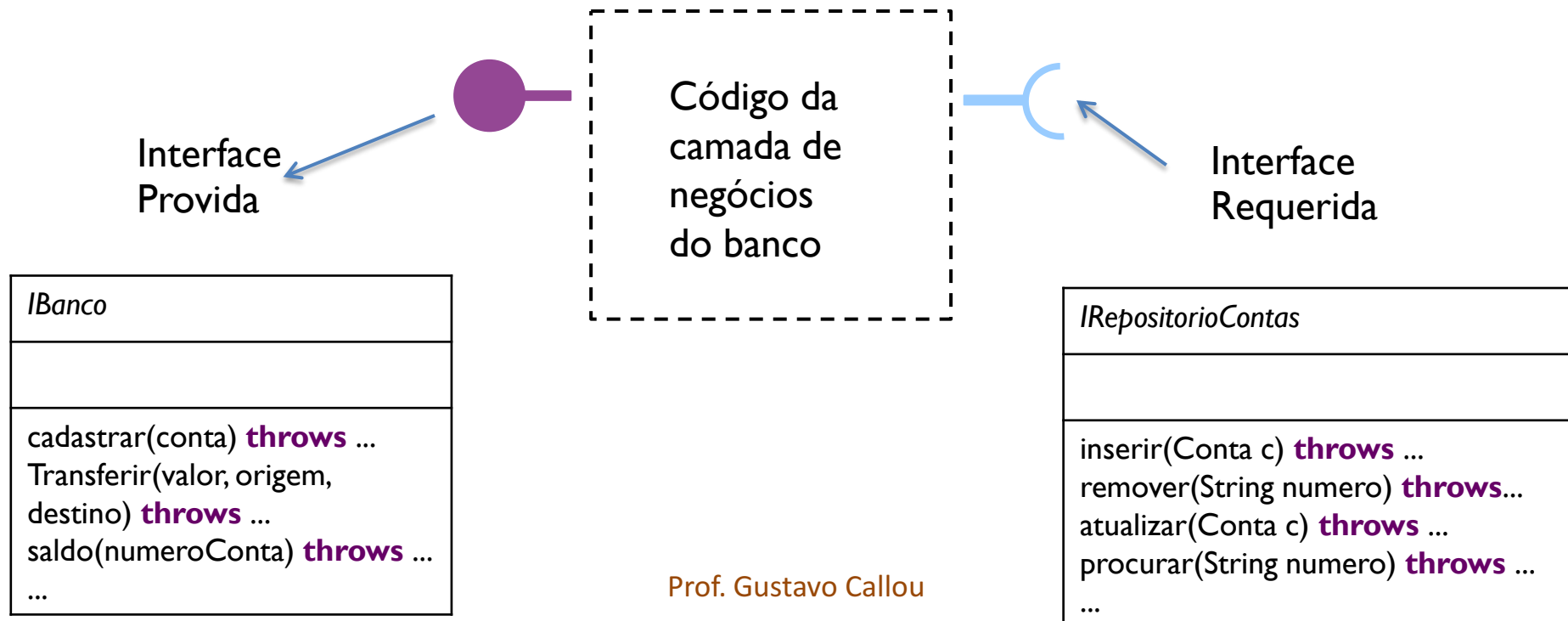
`br.ufrpe.poo.banco.dados`

Arquitetura em três camadas



Camada de negócio

- ▶ Implementa uma interface (**interface provida**) que é utilizada pelas classes da camada de dados (**interfaces requeridas**)
- ▶ Exemplo: camada de negócio do banco



Camada de negócio



- ▶ Classes **básicas** : representam as entidades
 - ▶ Exemplo: Conta, Poupanca, etc
- ▶ **Exceções** : representam os possíveis erros durante o uso do sistema
 - ▶ Exemplo: SaldoInsuficienteException
- ▶ **Classe** do sistema : classe que implementa a interface da camada de negócios
 - ▶ Exemplo: Classe Banco implementa IBanco

Classe do sistema



- ▶ **Implementa a interface provida** pela camada de negócios
- ▶ Corresponde a **fachada** (ponto de acesso) do sistema
 - ▶ API com os métodos chamados por uma ou mais interfaces de usuário
- ▶ **Guarda repositórios** das entidades do sistema como atributo
 - ▶ Tipo dos repositórios são as interfaces providas pela camada de dados

Classe do sistema

- ▶ Exemplo: modelagem parcial da classe que corresponde a fachada do sistema do banco
 - ▶ Implementa a interface IBanco

Banco
contas : IRepositorioContas <i>instance</i> : IBanco ...
IBanco getInstance() throws InicializacaoSistemaException cadastrar(conta) throws ContaJaExisteException transferir(valor, origem, destino) throws SaldoInsuficienteException, ContaNaoEncontradaException saldo(numeroConta) throws ContaNaoEncontradaException ...

Classe do sistema

- ▶ Exemplo: implementação do método getSaldo da classe Banco

```
@Override
```

```
public double getSaldo(String numero) throws  
    RepositorioException, ContaNaoEncontradaException {
```

```
    ContaAbstrata c = contas.procurar(numero);  
    if (c == null) {  
        throw new ContaNaoEncontradaException();  
    }  
    return c.getSaldo();
```

```
}
```

Classe do sistema

- ▶ Implementa o padrão de projeto chamado **singleton** que garante que será criado apenas uma instancia do sistema
- ▶ Classe que implementa singleton tem:
 - ▶ Atributo estático aponta para a única instância
 - ▶ Construtor privativo
 - ▶ Método público chamado `getInstance ()`
 - ▶ Retorna uma instância da classe

Classe do sistema

► Atributo estático, construtor e getInstance da classe Banco

```
private static IBanco instance;

private Banco(IRepositorioContas rep) {
    this.contas = rep;
}

public static IBanco getInstance() throws InicializacaoSistemaException {
    if (Banco.instance == null) {
        try {
            Banco.instance = new Banco(new RepositorioContasArquivoTxt(new java.io.File("a.txt")));
        } catch (RepositorioException e) {
            throw new InicializacaoSistemaException();
        }
    }
    return Banco.instance;
}
```

Classe do sistema

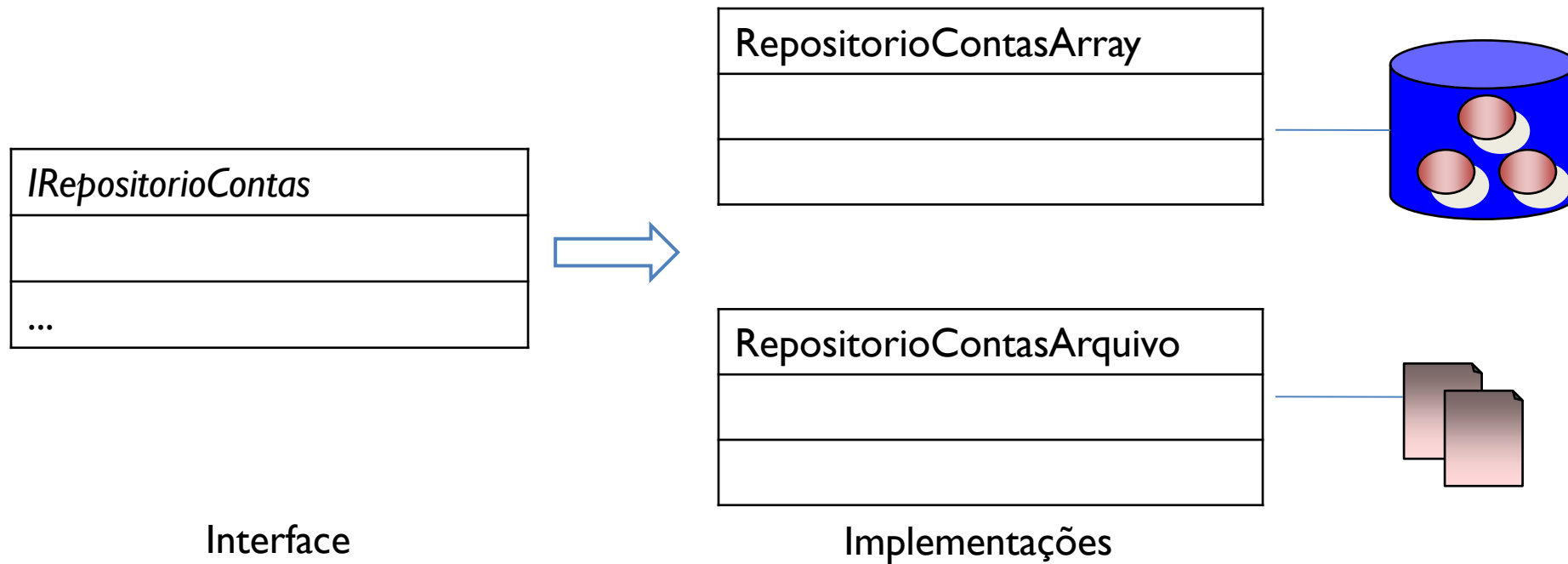
- ▶ Exemplo: obtendo uma instância do sistema e consultando saldo de conta

```
IBanco banco = Banco.getInstance();

try{
    double saldo = banco.getSaldo("1");
    System.out.println("Saldo da conta 1 = " + saldo);
}catch(ContaNaoEncontradaException e){
    ...
}catch(RepositorioException e){
    ...
}
```

Camada de dados

- ▶ Uma primeira implementação para a interface `IRepositorioContas` é `RepositorioContasArray`. Em outro passo podemos implementar `RepositorioContasArquivoTxt` que persiste em um arquivo, ou, implementar `RepositorioContasArquivoBin` que persiste em arquivo binário



Camada de dados

- ▶ Implementa interfaces para armazenar, recuperar e atualizar coleções de objetos
- ▶ Exemplo: A Interface `IRepositorioContas` e suas implementações fazem parte da camada de dados do sistema do banco

<i>IRepositorioContas</i>
<code>inserir(Conta c) throws RepositorioException</code>
<code>remover(String numero) throws RepositorioException</code>
<code>atualizar(Conta c) throws RepositorioException</code>
<code>Iterator getIterator() throws RepositorioException</code>
...

Camada de dados

- ▶ Exemplo: modelagem parcial da Classe **RepositorioContasArray** que implementa a **Interface IRepositorioContas**

RepositorioContasArray
- contas : ContaAbstrata[] - indice : int
+ inserir(Conta c) throws RepositorioException + remover(String numero) throws RepositorioException + atualizar(Conta c) throws RepositorioException - getIndice(String numero) ...

Camada de dados

- ▶ Exemplo: implementação do **método procurar** da Classe **RepositorioContasArray**

@Override

```
public ContaAbstrata procurar(String numero) throws RepositorioException {  
    ContaAbstrata resposta = null;  
    int i = this.getIndice(numero);  
    if (i < this.indice) {  
        resposta = this.contas[i];  
    }  
    return resposta;  
}
```

Camada de apresentação



- ▶ Interface apresentada ao usuário do sistema
 - ▶ GUI
 - ▶ Textual
 - ▶ Etc
- ▶ Programação facilitada pois métodos do sistema estão bem definidos na interface provida pelo negócio

Camada de apresentação

▶ Exemplo: parte do código da Classe InterfaceTextualBanco

```
public class InterfaceTextualBanco {  
  
    private IBanco fachadaSistema;  
  
    public InterfaceTextualBanco() throws InicializacaoSistemaException {  
        this.fachadaSistema = Banco.getInstance();  
    }  
  
    public void menuPrincipal(){  
        System.out.println("Escolha uma das opção a seguir");  
        System.out.println("1 - Saldo");  
        System.out.println("2 - Transferência");  
        ...  
    }  
}
```

Camada de apresentação

- ▶ Exemplo: método da classe InterfaceTextualBanco que apresenta o saldo da conta escolhida

```
private void saldo() {  
    String numero = numeroDigitadoUsuario;  
    try {  
        double saldo = fachadaBanco.getSaldo(numero);  
        sucesso("O saldo da conta "+ numero+" eh "+saldo);  
    } catch (ContaNaoEncontradaException e) {  
        tratarErroContaNaoExiste(e);  
    } catch (RepositorioException e) {  
        tratarErroRepositorio(e);  
    }  
}
```

Arquitetura em camadas

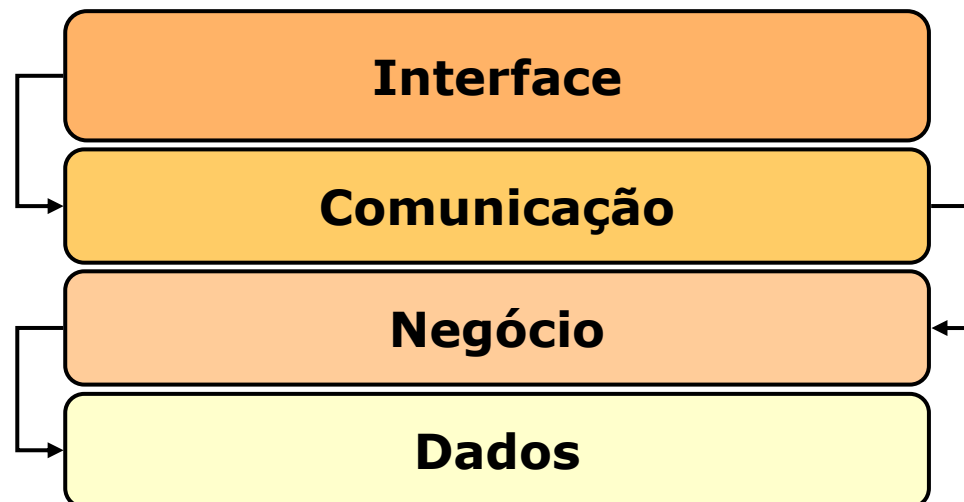


▶ Alguns benefícios

- ▶ **Separação de conceitos:** não mistura códigos que possuem finalidades diferentes
 - ▶ Facilita a manutenção
- ▶ **Modularidade:** mudanças em uma camada não afetam as outras, desde que as interfaces sejam preservadas
 - ▶ Uma mesma versão de uma camada trabalha com diferentes versões de outra camada

Arquitetura em camadas

- ▶ Quando interface e negócios estão distribuídos é comum considerar uma extra (camada de comunicação) para lidar com a transmissão/recepção das informações pela rede



- ▶ A camada de comunicação utiliza tecnologias como RMI, CORBA, etc