

Componentização e Reúso de Software

Fundamentos de Componentes e Design Visão Interna

André Santanchè

Laboratory of Information Systems - LIS

Instituto de Computação - UNICAMP

Julho de 2018

O que é um componente?

Características Comuns

- Publica sua funcionalidade através de uma interface
 - interface guia relacionamento componente x ambiente
- Entidade concebida para ser composta
 - do latim *componens*, derivado de *componere*, que quer dizer “colocar junto”.
- Componentes podem ser aninhados em outros componentes
 - componentes e sub-componentes



O que é um componente?

Características Desejáveis

- Contém código binário que implementa a funcionalidade declarada na interface
- Serviços acessíveis exclusivamente pela interface (black-box)
- Pacote padrão para distribuição



O que é um componente?

Características de Design

- Intercambiável dentro do ambiente
- Alta Coesão e Baixo Acoplamento



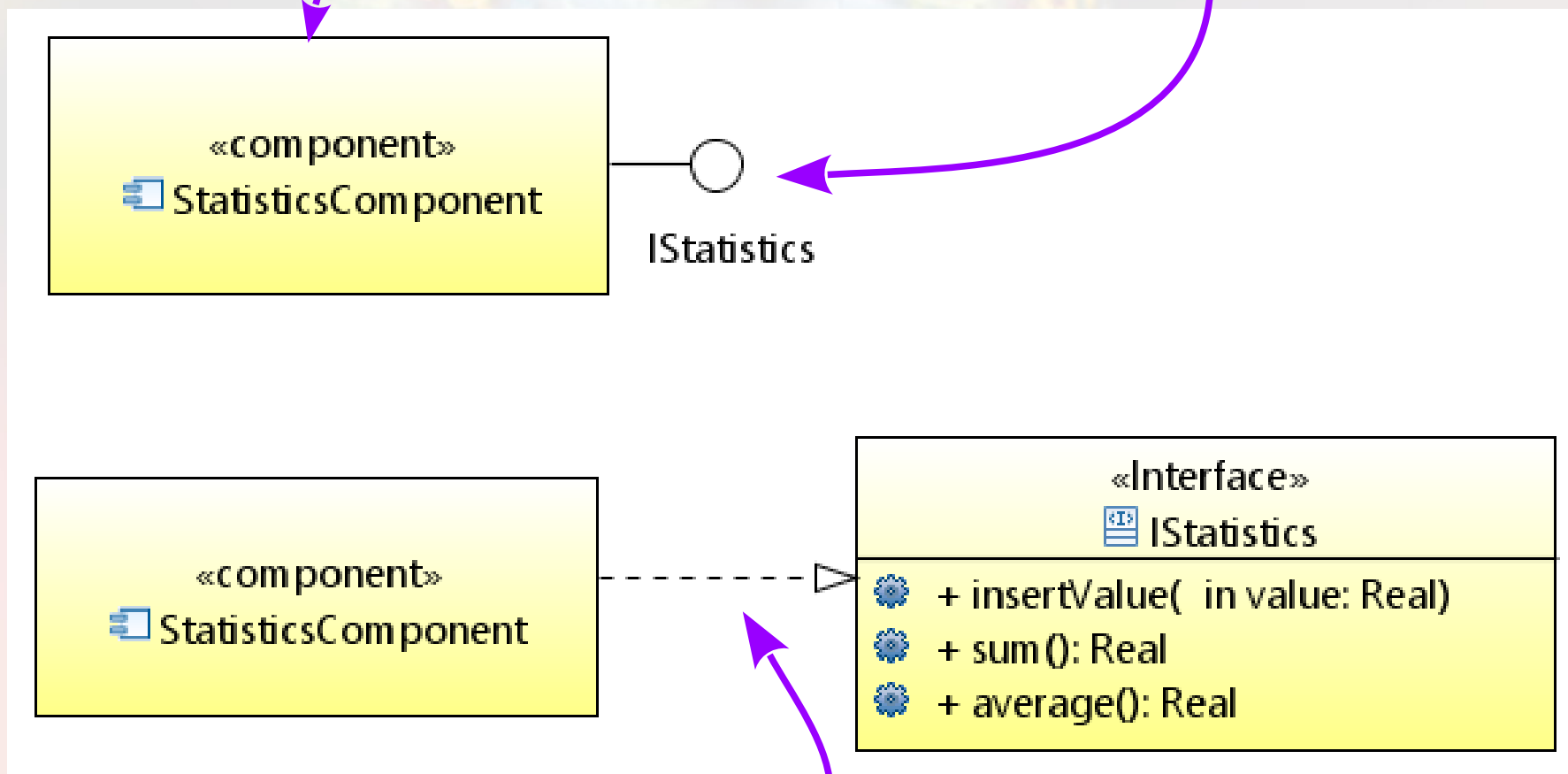
Visão Interna

- Visão Externa (aula anterior)
 - Foco: blackbox
 - Abstração das funcionalidades de um componente vendo-o externamente através de suas interfaces
 - Uso de componentes → Composição
- Visão Interna (esta aula)
 - Foco: whitebox
 - Como um componente é implementado internamente

Notação Blackbox

componente em
notação blackbox

componente realiza interface
(representação lollipop)

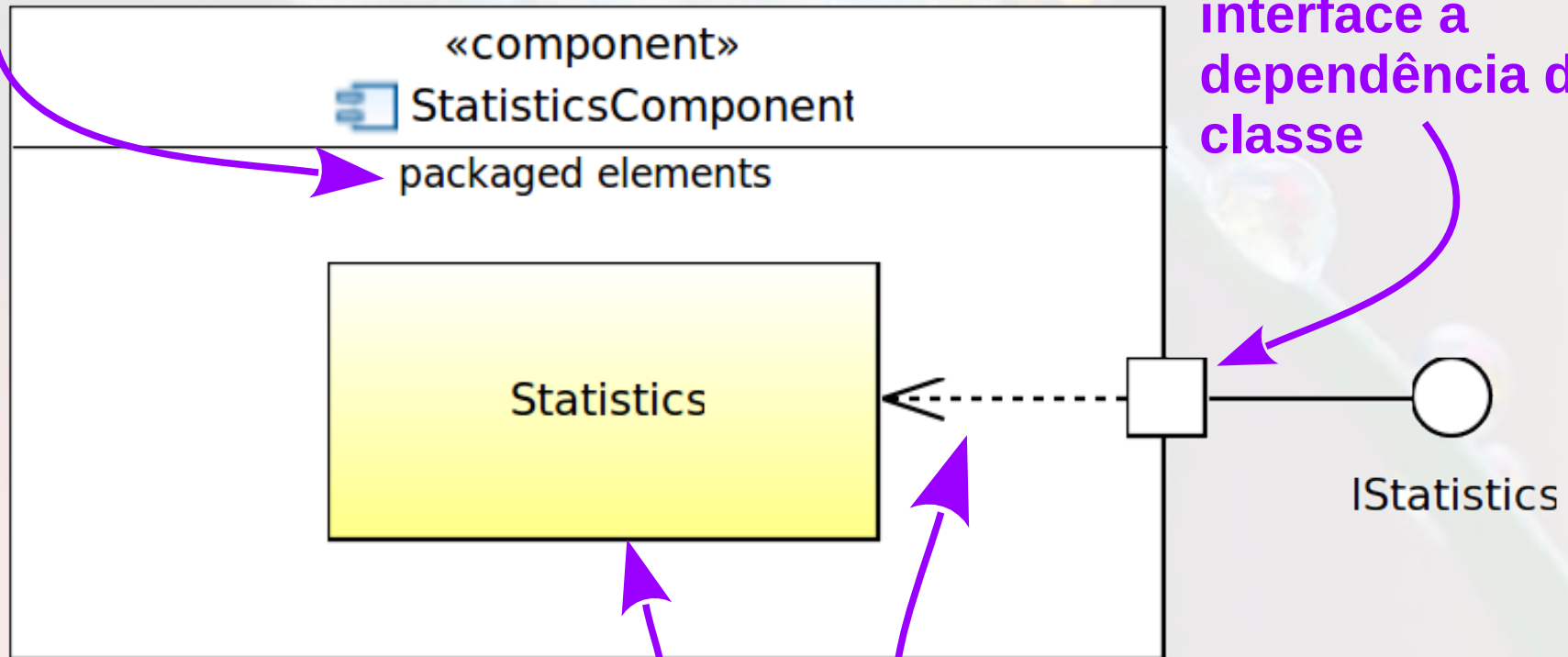


componente realiza
interface (representação
explícita)

Realizando o Componente

compartimento opcional que mostra elementos que são parte do componente

porta que associa interface a dependência de classe

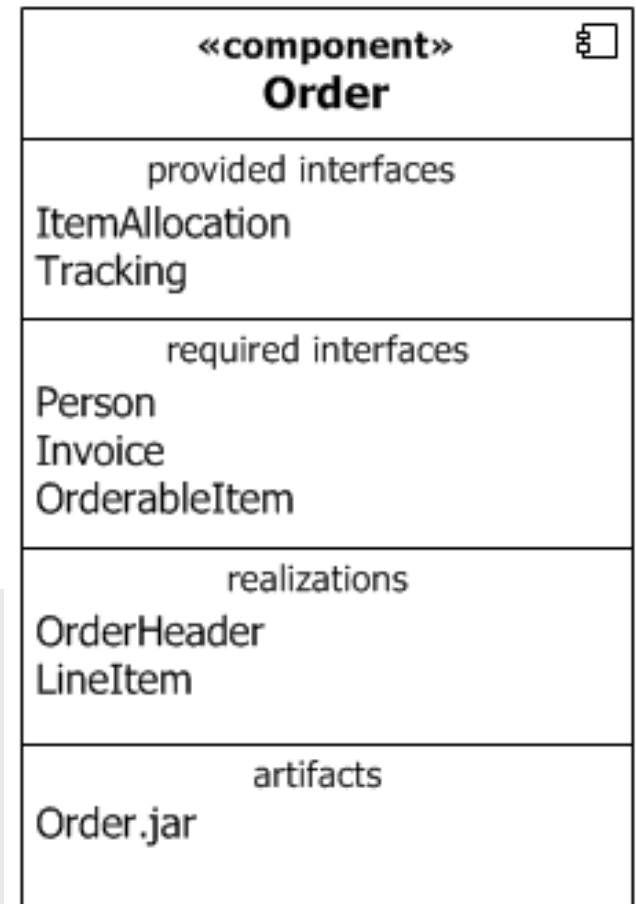
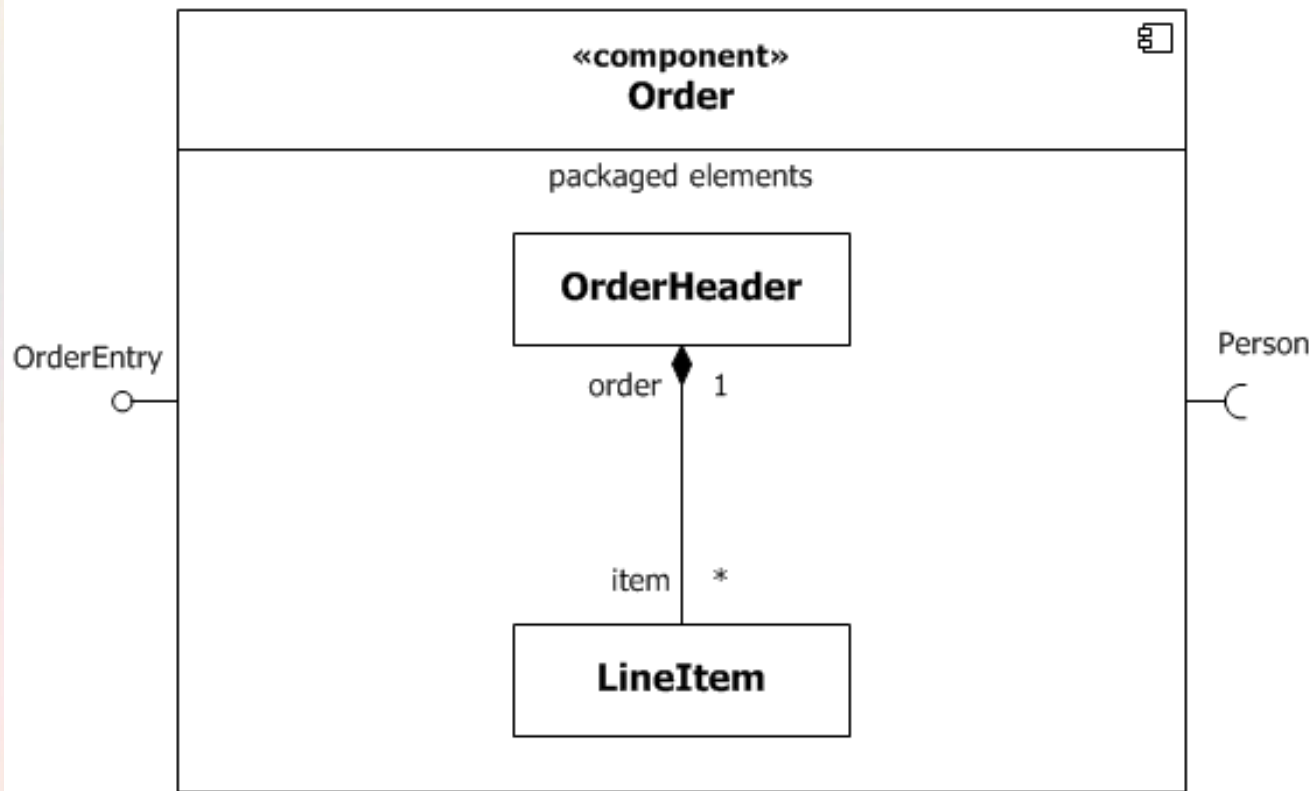


classe que realiza o componente

dependência da porta/interface com a classe que a realiza

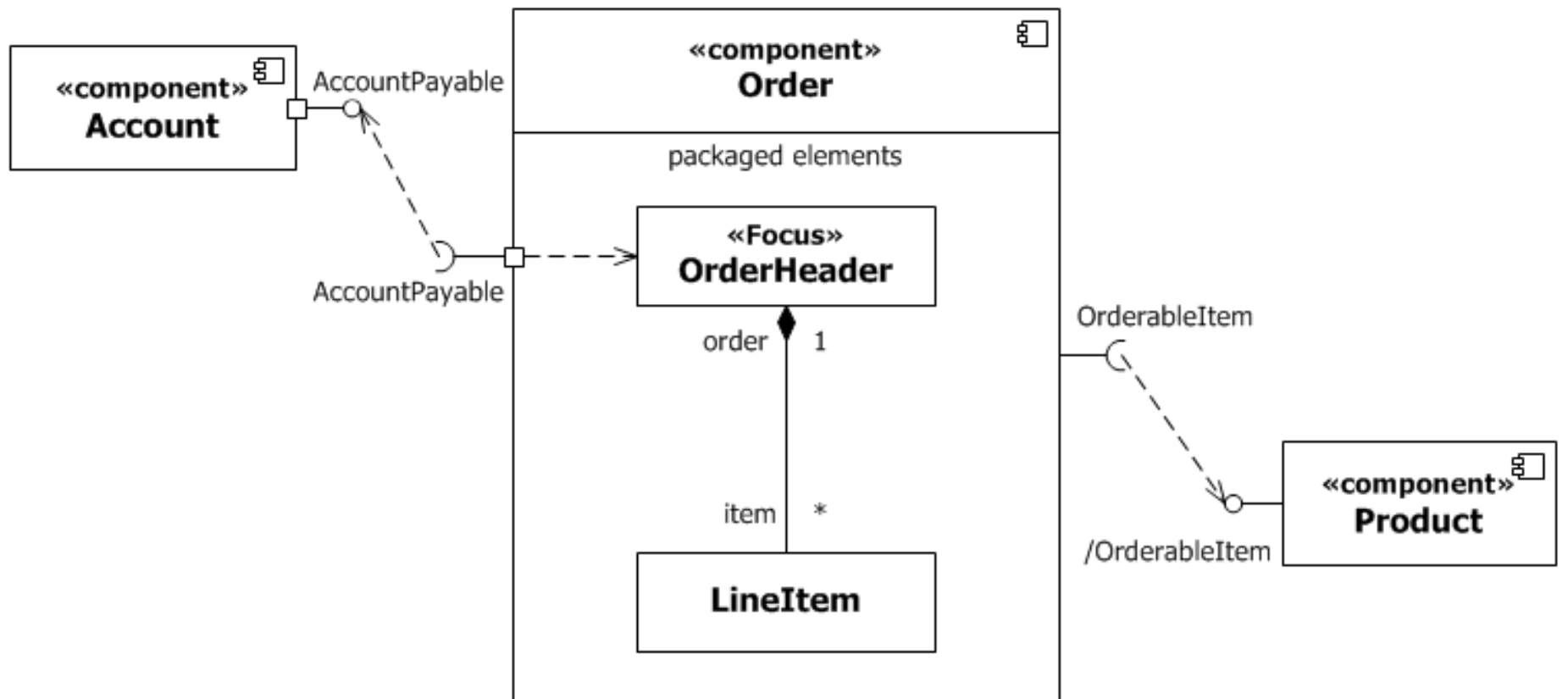
pt.c08componentes.s10statistics.s03component


Duas perspectivas de Whitebox OMG UML (Cook et al., 2015)



Whitebox com portas

OMG UML (Cook et al., 2015)





Componente Estatístico

Implementação - Whitebox

passo 1 - classe que implementa o componente

Componente Estatístico

Objetivo

- Registrar um conjunto de números e calcular a soma e média destes números.

Passo 1: Classe Estatística

Statistic

 - valueSet: Real

 - last: Integer |


 + insertValue(in value: f

 + sum(): Re

 + average(): R

Exercício 01

- Faça um programa para calcular e exibir a soma e a média dos números: 50, 70 e 30.



Componente Estatístico

Implementação - Whitebox

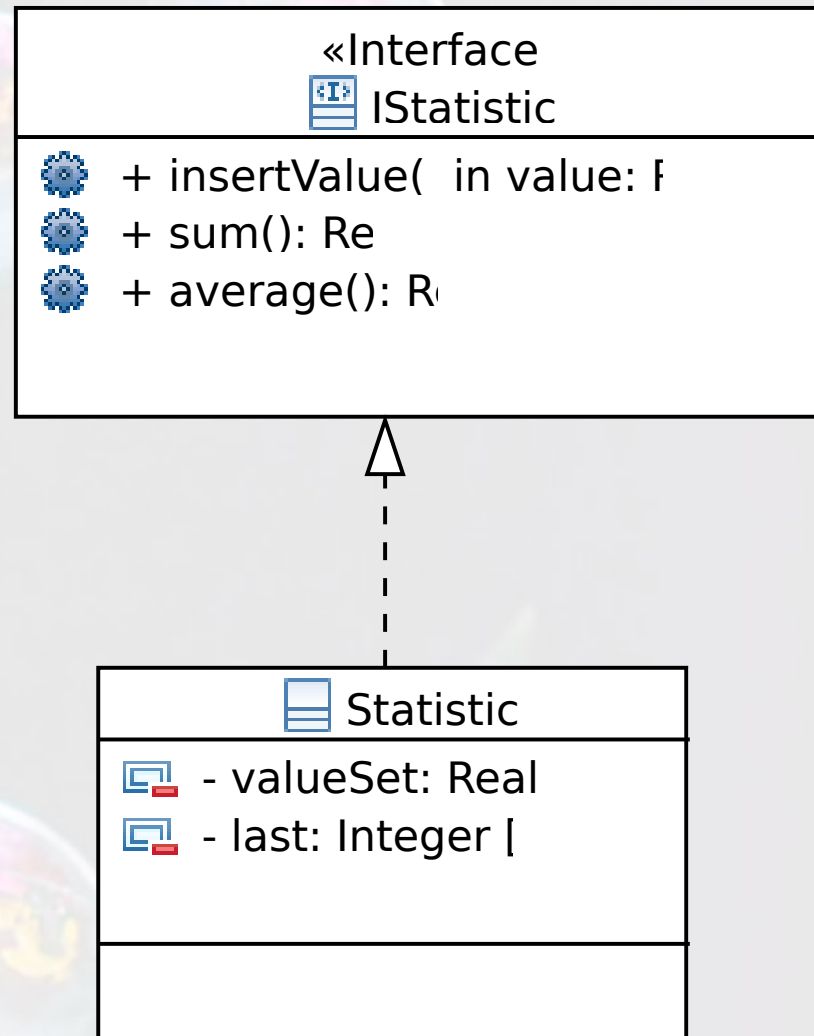
passo 2 - acesso via interface

Dependency Inversion Principle (DIP)

- “Depender das Abstrações. Não depender das Concretizações.” (Martin, 2000)




Passo 2: Interface Estatística



pt.c08componentes.s10statistics.s02interface

Exercício 02

- Adapte o programa anterior de modo que a variável seja do tipo da interface.



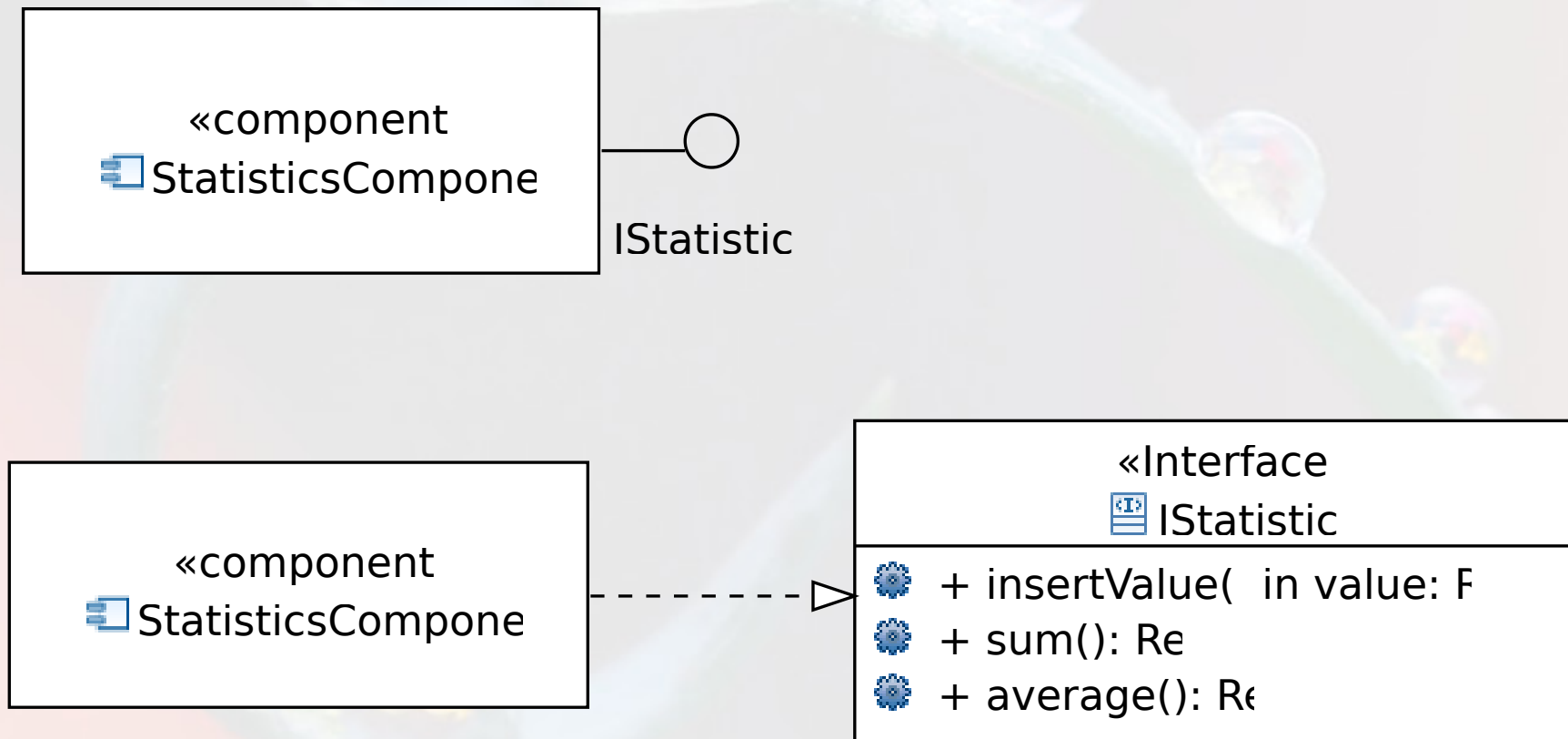
Componente Estatístico

Implementação - Whitebox

passo 3 - transformando em um componente

Notação Blackbox

- Esconde os detalhes internos da representação

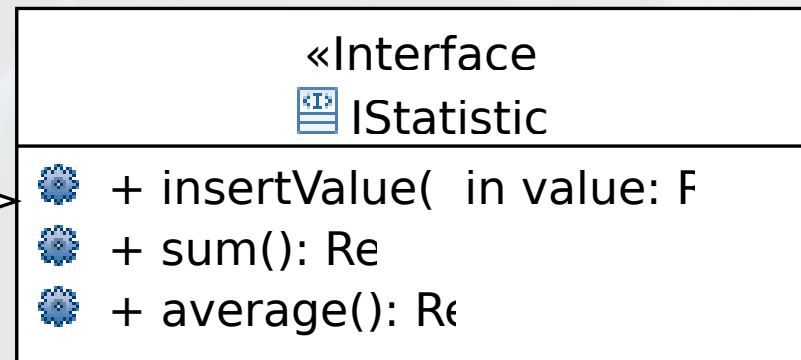
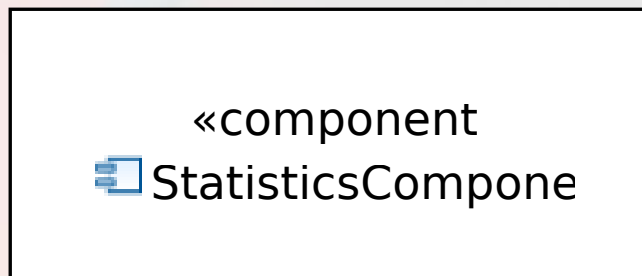
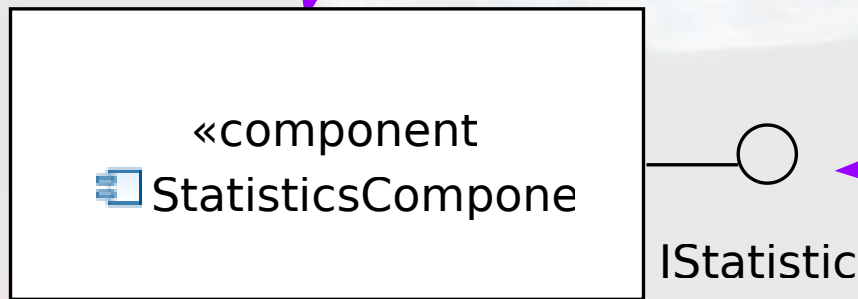


pt.c08componentes.s10statistics.s03component

Notação Blackbox

componente em
notação blackbox

componente realiza interface
(representação lollipop)

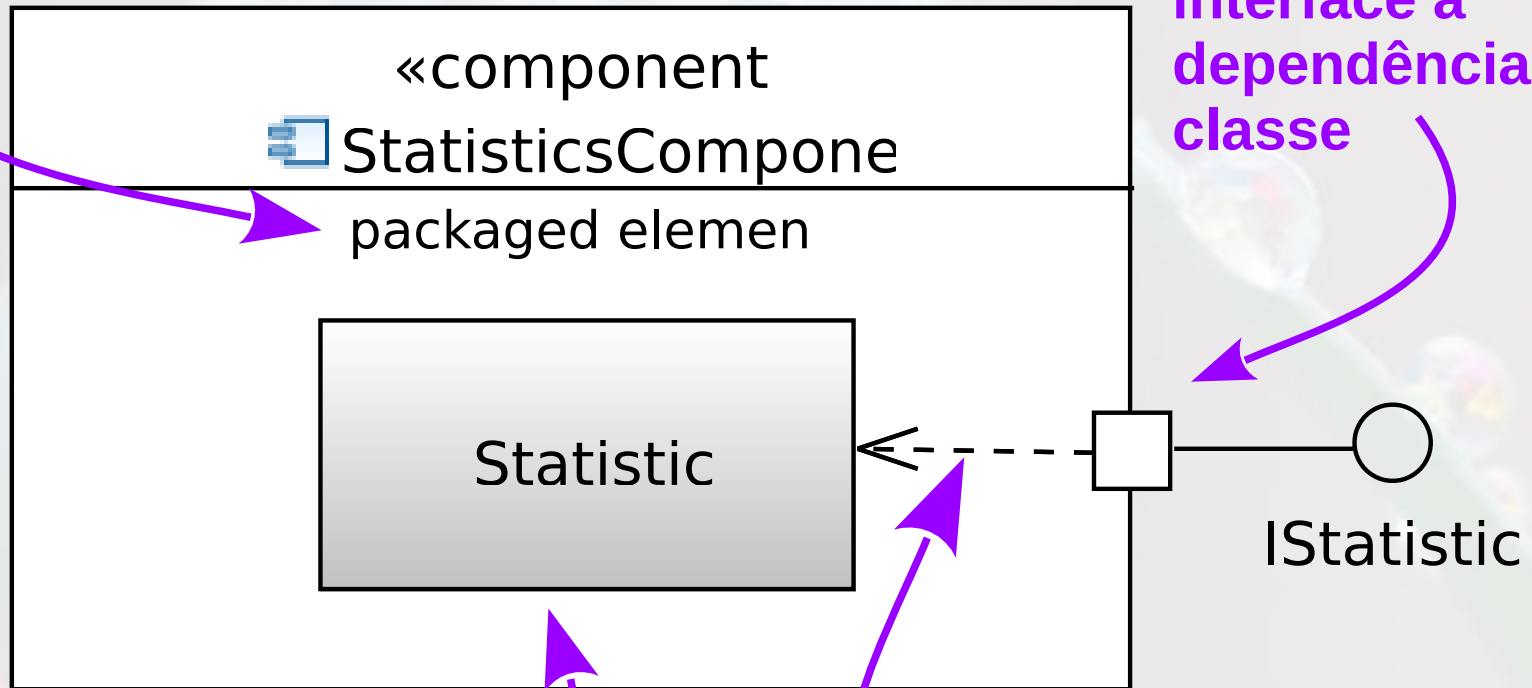


componente realiza
interface (representação
explícita)

Realizando o Componente

compartimento opcional que mostra elementos que são parte do componente

porta que associa interface a dependência de classe



classe que realiza o componente

dependência da porta/interface com a classe que a realiza

pt.c08componentes.s10statistics.s03component

Componente JavaBean

- Componentes são unidades de software auto-contidas e reusáveis que podem ser compostas visualmente em componentes compostos, applets, aplicações, e servlets usando ferramentas visuais de construção de aplicações.” (Sun, 2006)

Tradução do Inglês: “Components are self-contained, reusable software units that can be visually assembled into composite components, applets, applications, and servlets using visual application builder tools.” (Sun, 2006)

JavaBeans

- Beans - componentes em Java

Características:

- Construtor sem argumentos
- Propriedades
- Introspecção
- Customização
- Persistência
- Eventos

Perspectiva Orientada a Objetos de Componentes

- Componentes são associados a classes
 - São instanciados como objetos
 - Não é um consenso
- Propriedades externamente observáveis
 - Customizam a instância do componente
 - Não é um consenso

Construtor sem Argumentos

- Permite a criação automática do componente
- Construtor com ação padrão

```
public final static int STANDARD_SIZE = 50;  
private double valueSet[];  
private int last;  
public Statistics() {  
    this(STANDARD_SIZE);  
}  
public Statistics(int size) {  
    super();  
    valueSet = new double[size];  
    last = -1;  
}
```

Construtor sem Argumentos

- Construtor sem ação padrão

```
public class Bean01Nome {  
    public Bean01Nome() {  
        /* nada */  
    }  
}
```

- Construtor com ação padrão

```
public class Bean02Circulo {  
    private int raio;  
  
    public Bean02Circulo() {  
        raio = 50;  
    }  
}
```


Exercício 03

- Adapte a classe implementada anteriormente para que ela se comporte como um componente, no pacote:

```
pt.c08componentes.s10statistics.s03component.v01
```

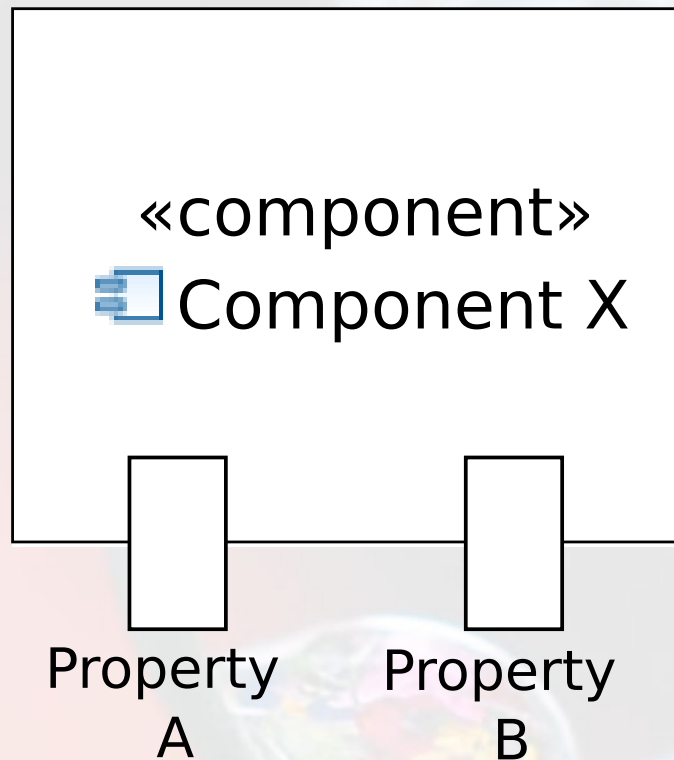


Componente Estadístico

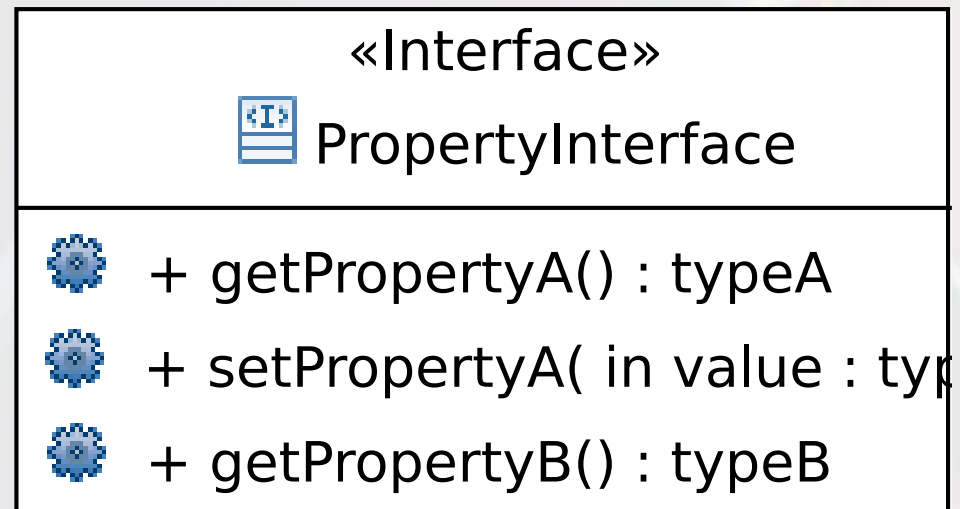
Propiedades

Propriedades

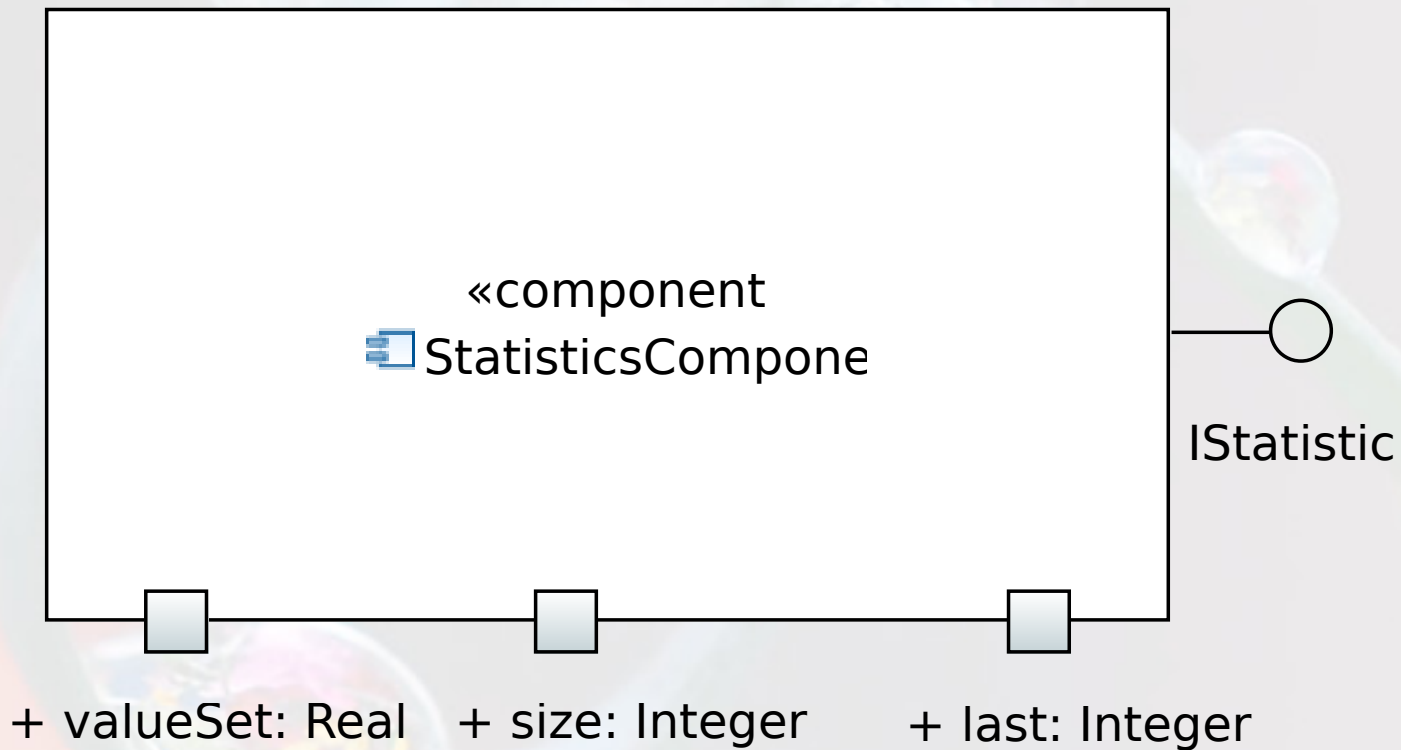
Notação CORBA Component Model



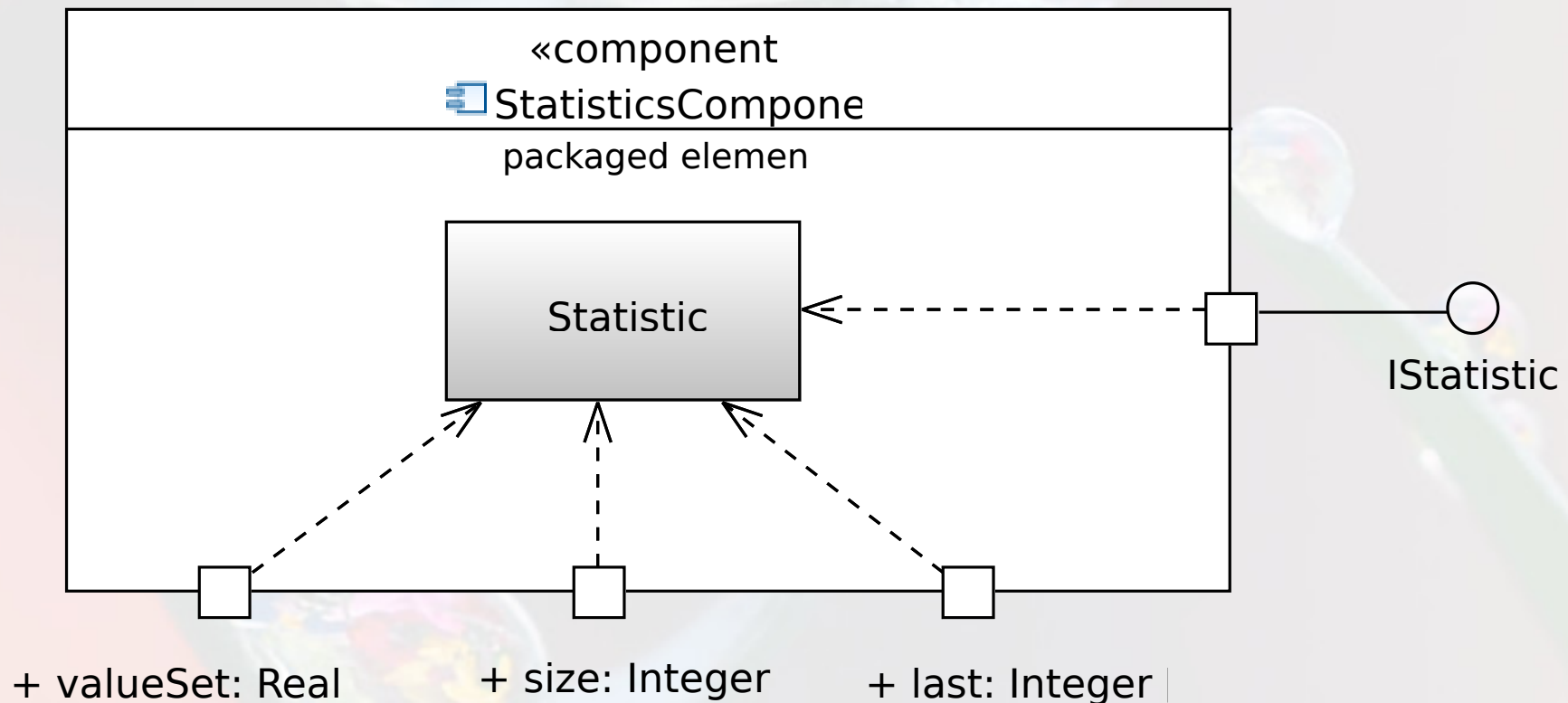
PropertyB é somente leitura



Propriedades de Statistics Blackbox



Propriedades de Statistics Whitebox





Propriedades em Javabeans

Propriedades

- Expostas através de métodos:
 - prefixo “get” → leitura
 - prefixo “set” → modificação

```
private int raio;
```

```
public int getRaio() {  
    return raio;  
}
```

```
public void setRaio(int raio) {  
    this.raio = raio;  
}
```

Propriedades

- Para diferenciar o atributo do parâmetro é usado o `this`:

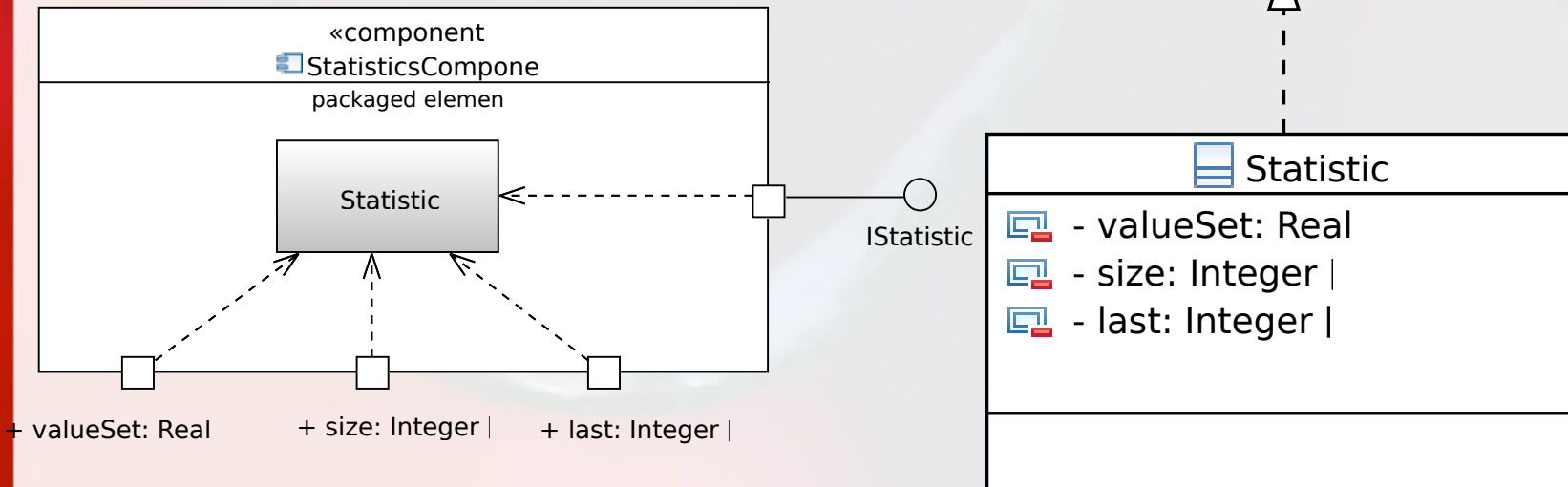
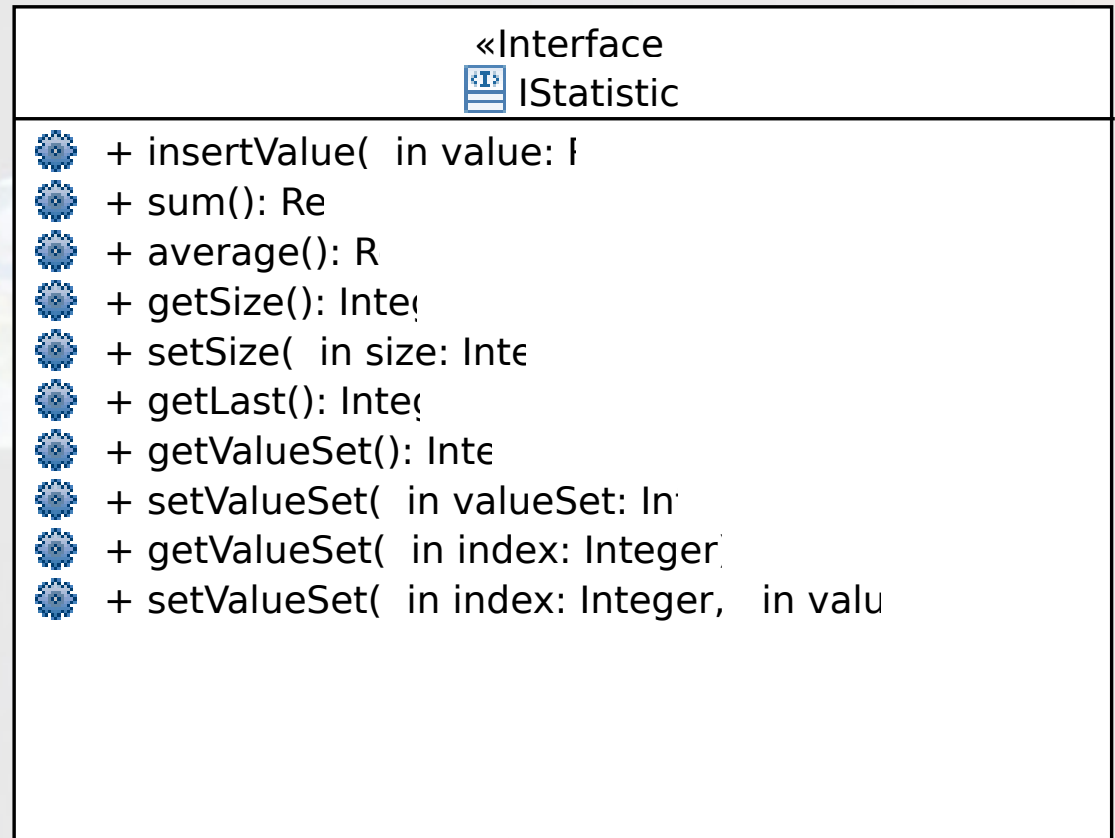
```
public void setRaio(int raio) {  
    this.raio = raio;  
}
```

Propriedades

- Somente leitura
 - não têm método “set”
- Propriedades não estão obrigadas a expor um atributo

```
public float getArea() {  
    return 3.1416f + raio * raio;  
}
```

Propriedades de Statistics Interface



Propriedades Indexadas

valueSet

- Acrescentam-se métodos:
 - prefixo “get” com index → retorna um item
 - prefixo “set” com index → altera um item

```
private double[] valueSet;
```

```
public double[] getValueSet() {...}
```

```
public void setValueSet(double[] valueSet){...}
```

```
public double getValueSet(int index) {...}
```

```
public void setValueSet(int index, double valueSet)  
{...}
```

Exercício 04

- Adapte a componente implementado anteriormente para que ele suporte as três propriedades mencionadas:

```
pt.c08componentes.s10statistics.s03component.v02
```



Componente Estadístico

Interface Requerida

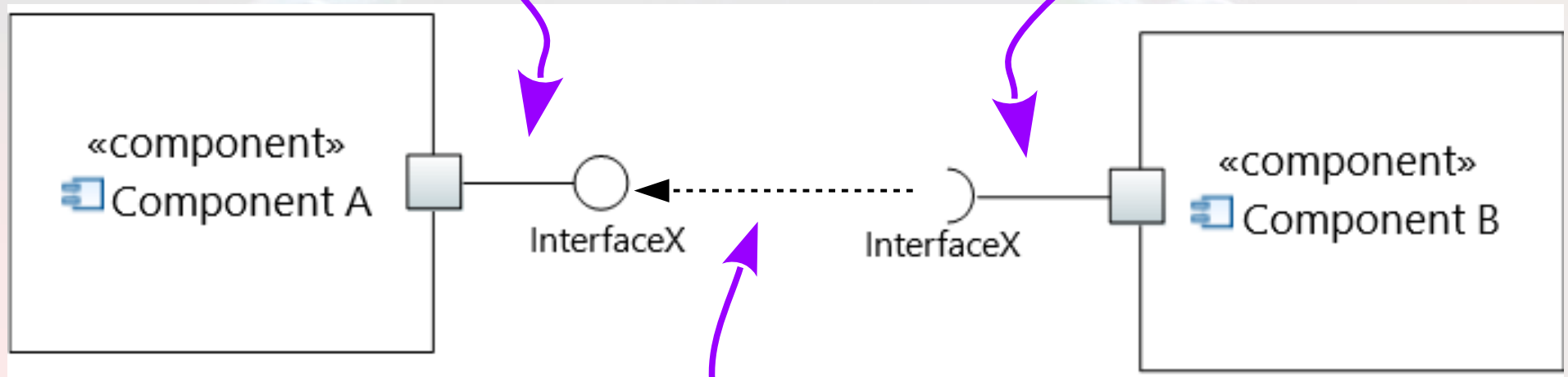
Exemplo: Mostrando Resultados

Interfaces Requeridas

Notação CORBA Component Model

interface provida

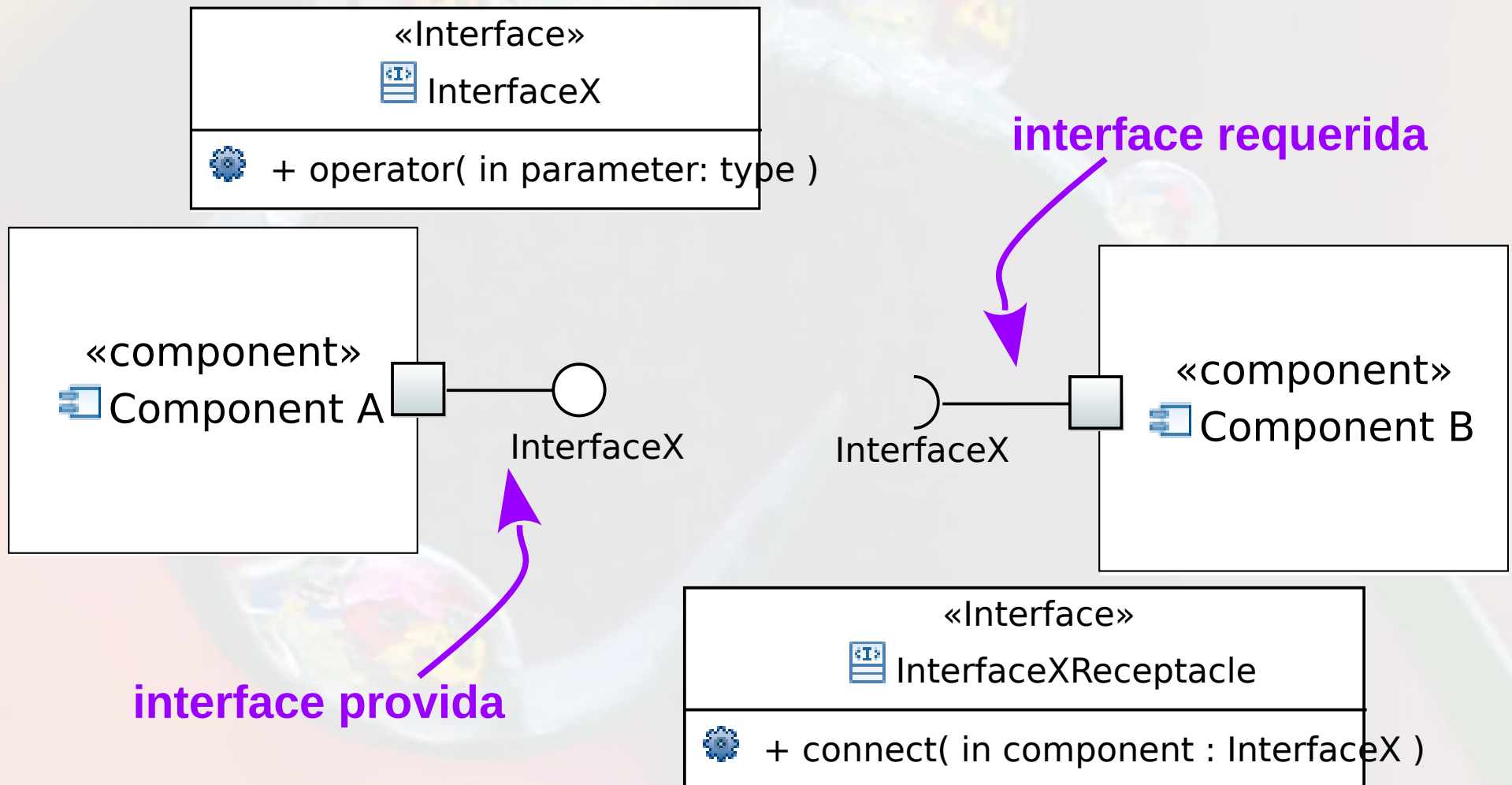
interface requerida



dependência entre a interface provida e a requerida

Interfaces Requeridas

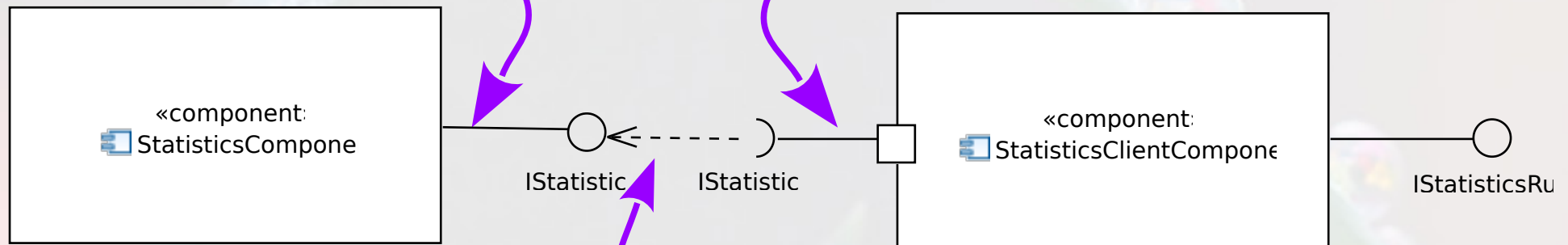
Notação CORBA Component Model



Interface Provida e Requerida Componente Client (blackbox)

interface provida

interface requerida



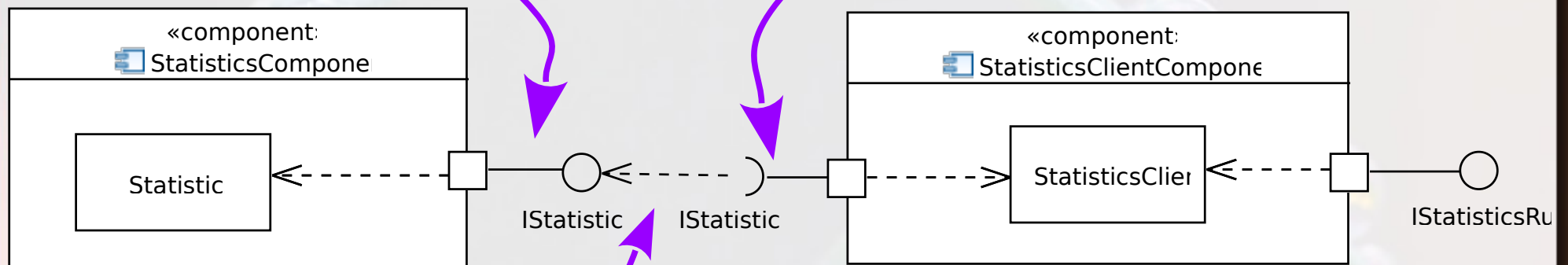
dependência entre a interface provida e a requerida

Interface Provida e Requerida

Componente Client (whitebox)

interface provida

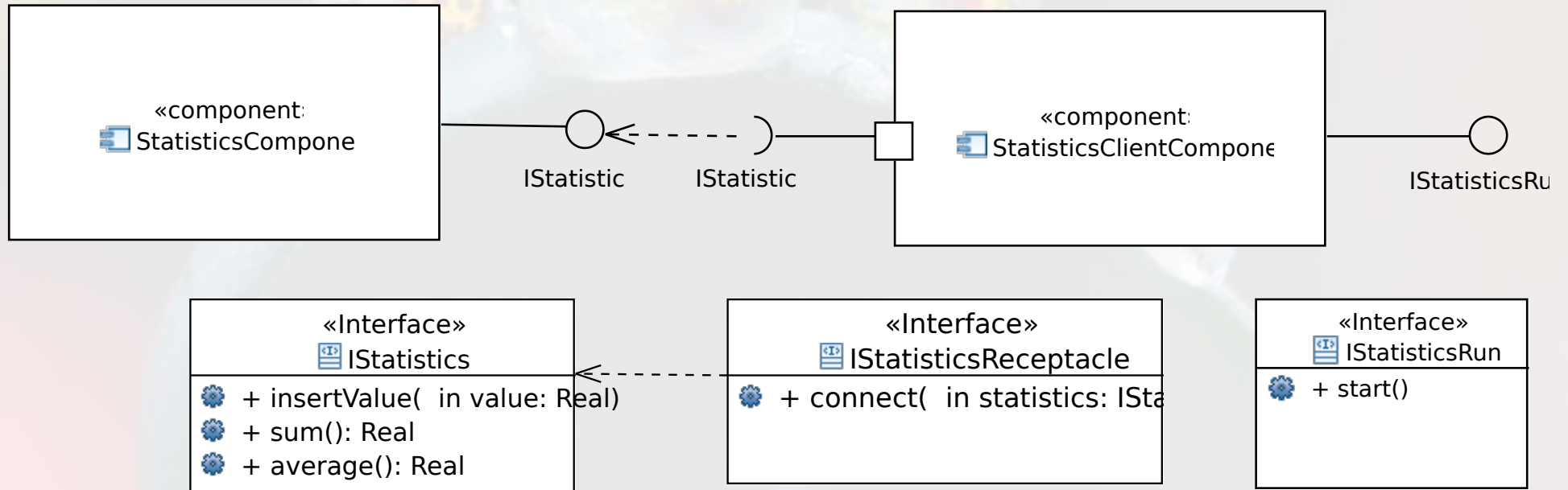
interface requerida



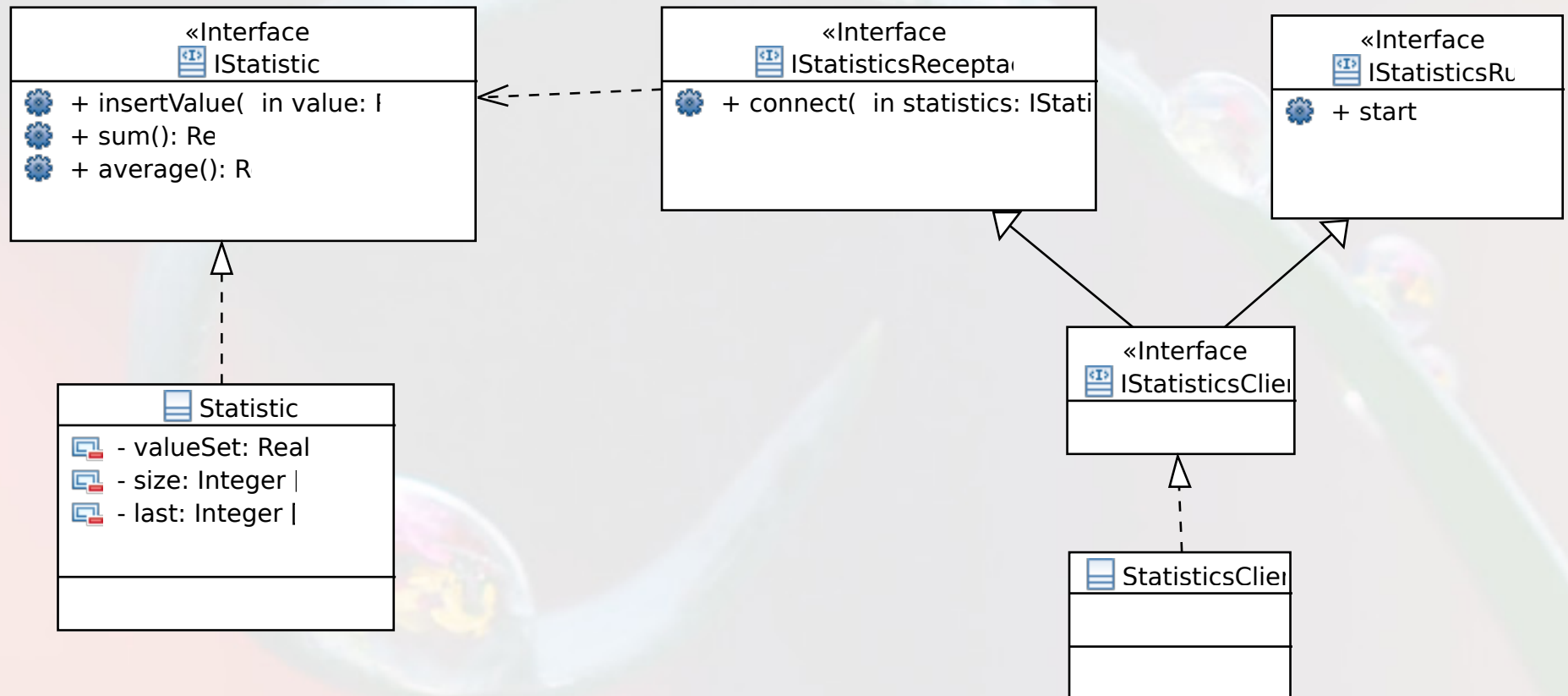
dependência entre a interface provida e a requerida

Interface Provida e Requerida

Componente Client (blackbox)



IStatisticsReceptacle

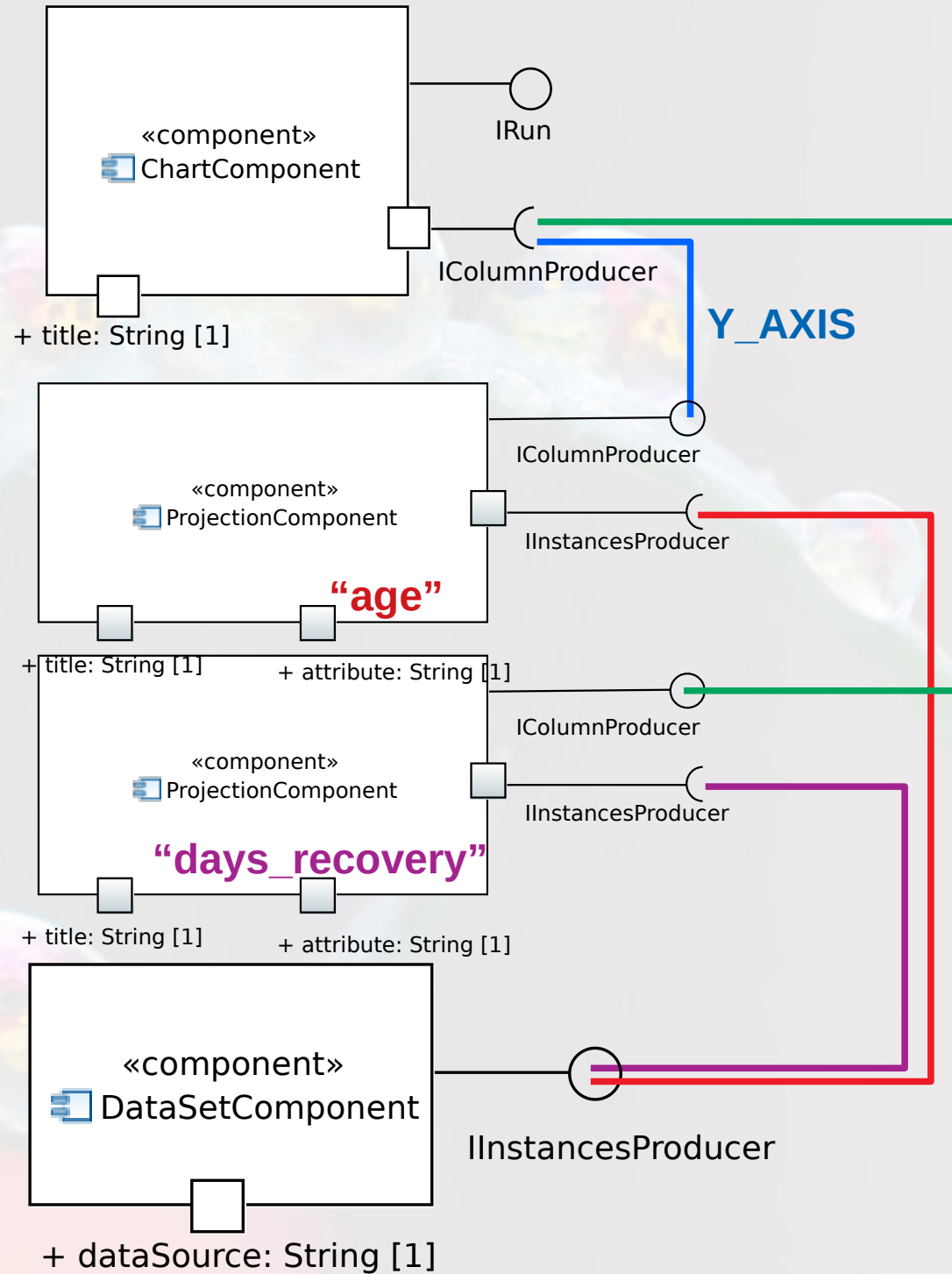
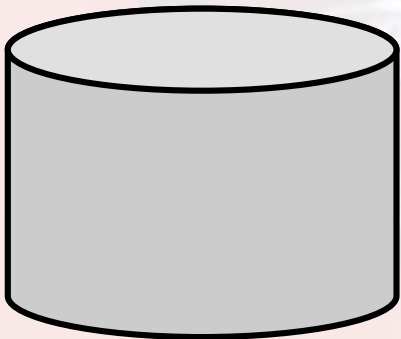
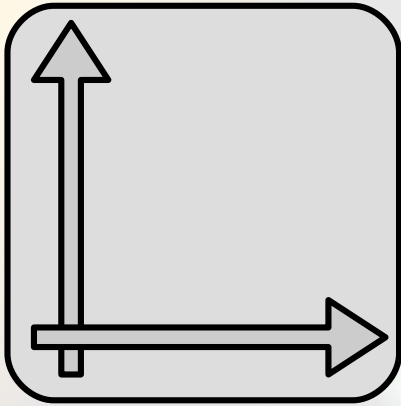


Exercício 05

- Construa um componente que se ligue ao `StatisticsComponent`, entregue a ele os 5 primeiros números de uma sequência de Fibonacci, e apresente a o somatório e a média calculados por ele.

Exercício 06

- Crie um componente `StatisticsSeries` que se incorpore ao catálogo de componentes apresentado na aula anterior.



Exercício 06

- Crie um componente `StatisticsSeries` que se incorpore ao catálogo de componentes apresentado na aula anterior.
- Este componente deve se ligar aos outros apresentados, de modo que seja possível ver a média de idade e de tempo de recuperação por diagnóstico.

Bibliografia

- Bachmann, F.; Bass, L.; Buhman, C.; Dorda, S.C.; Long, F.; Robert, J. & Wallnau, R.S.K. **Volume II: Technical Concepts of Component-Based Software Engineering**, 2nd Edition. Carnegie Mellon University, 2000.
- Broy, M.; Deimel, A.; Henn, J.; Koskimies, K.; Plásil, F.; Pomberger, G.; Pree, W.; Stal, M. & Szyperski, C. **What characterizes a (software) component?** Software -- Concepts & Tools, Springer-Verlag Heidelberg, 1998, 19, 49-56
- Hopkins, J. **Component primer**. Communications ACM, ACM Press, 2000, 43, 27-30.
- Mcilroy, M. D. Naur, P. & Randell, B. (ed.) **Mass Produced Software Components**. Software Engineering: Report of a conference sponsored by the NATO Science Committee, 1968.

Bibliografia

- Olsen, G. **From COM to Common**. Queue, ACM Press, 2006, 4, 20-26.
- Szyperski, C. **Component Software: Beyond Object-Oriented Programming**. Addison-Wesley Longman Publishing Co., Inc., 2002.

André Santanchè

<http://www.ic.unicamp.br/~santanche>

Licença

- Estes slides são concedidos sob uma Licença Creative Commons. Sob as seguintes condições: Atribuição, Uso Não-Comercial e Compartilhamento pela mesma Licença.
- Mais detalhes sobre a referida licença Creative Commons veja no link:
<http://creativecommons.org/licenses/by-nc-sa/3.0/>
- Agradecimento a Steve Wall [<http://www.flickr.com/photos/stevewall/>] por sua fotografia “Dew drops” usada na capa e nos fundos, disponível em [<http://www.flickr.com/photos/stevewall/524803118/>] vide licença específica da fotografia.