

Componentização e Reúso de Software

Fundamentos de Componentes e Design Visão Externa

André Santanchè

Laboratory of Information Systems - LIS

Instituto de Computação - UNICAMP

Junho de 2019



Problema

Complexidade do Hospital

Hospital Tycoon

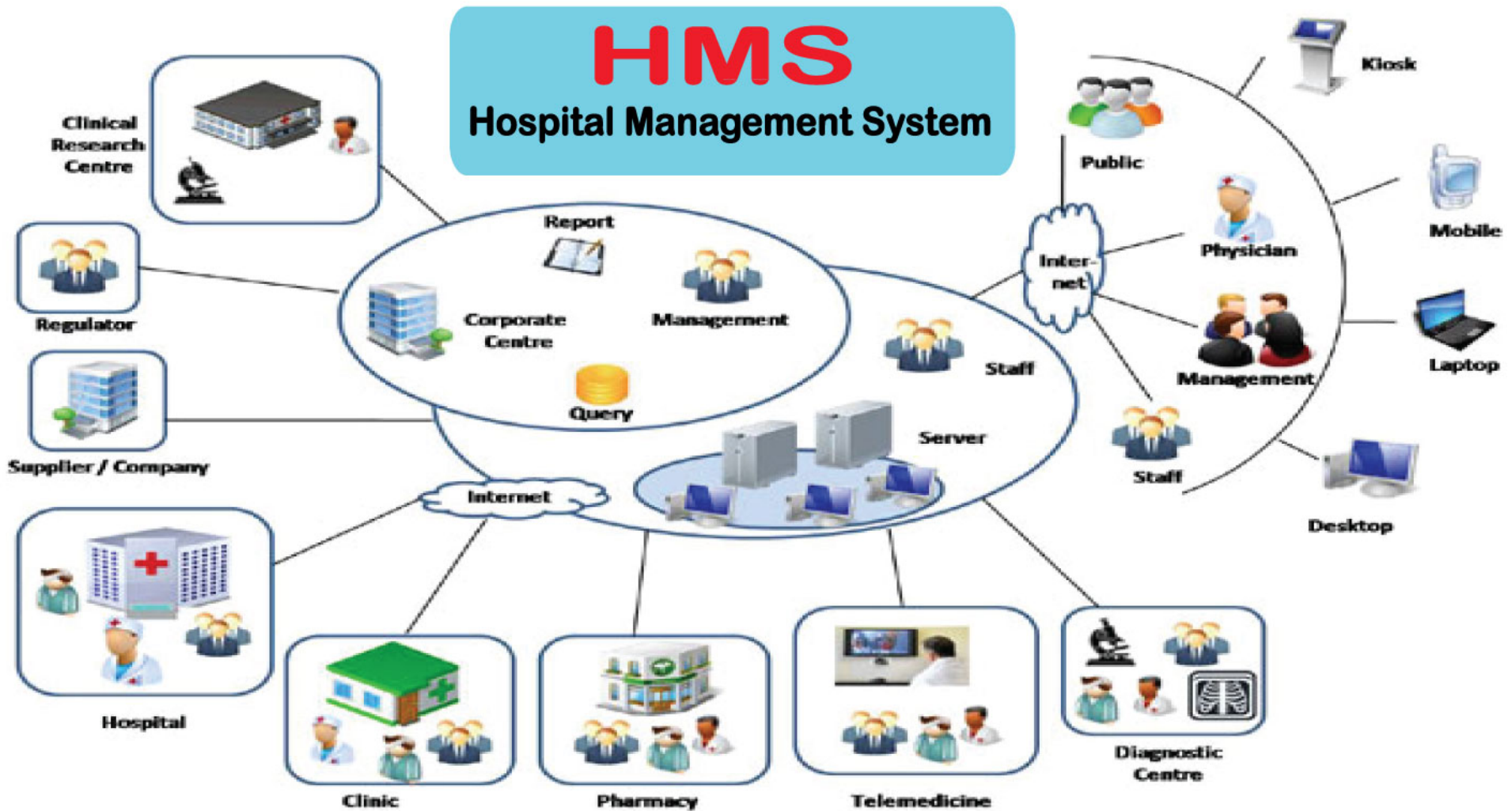
http://store.steampowered.com/app/11590/Hospital_Tycoon/



Hospital Havoc 2

<https://itunes.apple.com/us/app/hospital-havoc-2/id437134954?mt=8>

HMS



Innovanza Solutions

<http://innovanza.co/hospital-management-system/>

Quanta Modules

PATIENT REGISTRATION	WARD / ROOM MANAGEMENT	EMERGENCY/ CASUALTY	DOCTORS WORKBENCH	IPD PATIENT MODULE
LABORATORY INFORMATION SYSTEM	RADIOLOGY INFORMATION SYSTEM	OPERATION THEATER MANAGEMENT	CERTIFICATES ISSUE	ROSTER MODULE
COMMUNICATION MODULE	INVENTORY & FIXED ASSET	MEDICAL STORE MANAGEMENT	WARD PHARMACY/ INVENTORY MANAGEMENT	PHYSIOTHERAPY MODULE
DIET MODULE	HOUSE KEEPING MODULE	PAY ROLL MODULE	FINANCIAL ACCOUNTING MODULE	DENTAL MODULE
BLOOD BANK	MEDICAL RESEARCH	ICD - 10 DIAGNOSIS MODULE	DONATION MODULE	BILLING MODULE
AMBULANCE MODULE	USER MANAGEMENT	ADDRESS DIRECTORY	BACKUP/ RESTORE MODULE	MIS REPORTS
EQUIPMENT MAINTENANCE	CENTRAL STERILIZED SUPPLY DEPARTMENT	ENERGY AUDIT	COSTING	LIBRARY MANAGEMENT
BUDGETS	HELP DESK	FACILITY MANAGEMENT	WASTE MANAGEMENT	PACS & TELE RADIOLOGY
KITCHEN	EMR MODULE			

www.birlamedisoft.com





Dilema

Fazer Tudo x Comprar Tudo

- Extremos no desenvolvimento de software tradicional:
 - Desenvolver projeto da estaca zero
 - Comprar sistema pronto (configurado)

(Szyperski, 2002)

Exercício 1

- Liste vantagens e desvantagens dos extremos no desenvolvimento de software tradicional:
 - Desenvolver projeto da estaca zero
 - Comprar sistema pronto

Implementar Tudo

- Vantagens:
 - se adapta as necessidades do usuário
 - explora conhecimentos e práticas domésticas
 - diferencial → vantagem competitiva

(Szyperski, 2002)

Implementar Tudo

- Desvantagens:

- caro
- soluções sob medida geralmente são localizadas
- difícil de acompanhar o estado da arte (ex.: acesso Web)
- barreiras de interoperabilidade
- pode chegar “muito tarde”

(Szyperski, 2002)

Comprar Tudo

■ Vantagens

- custo pode ser pré-contratado
- software pré-fabricado diminui tempo de implantação
- estado da arte e interoperabilidade → tarefa de quem vende

(Szyperski, 2002)

Comprar Tudo

- Desvantagens

- adaptação dos negócios ao software
- sem diferencial → sem vantagem competitiva
- não se adapta rapidamente a novas necessidades

(Szyperski, 2002)

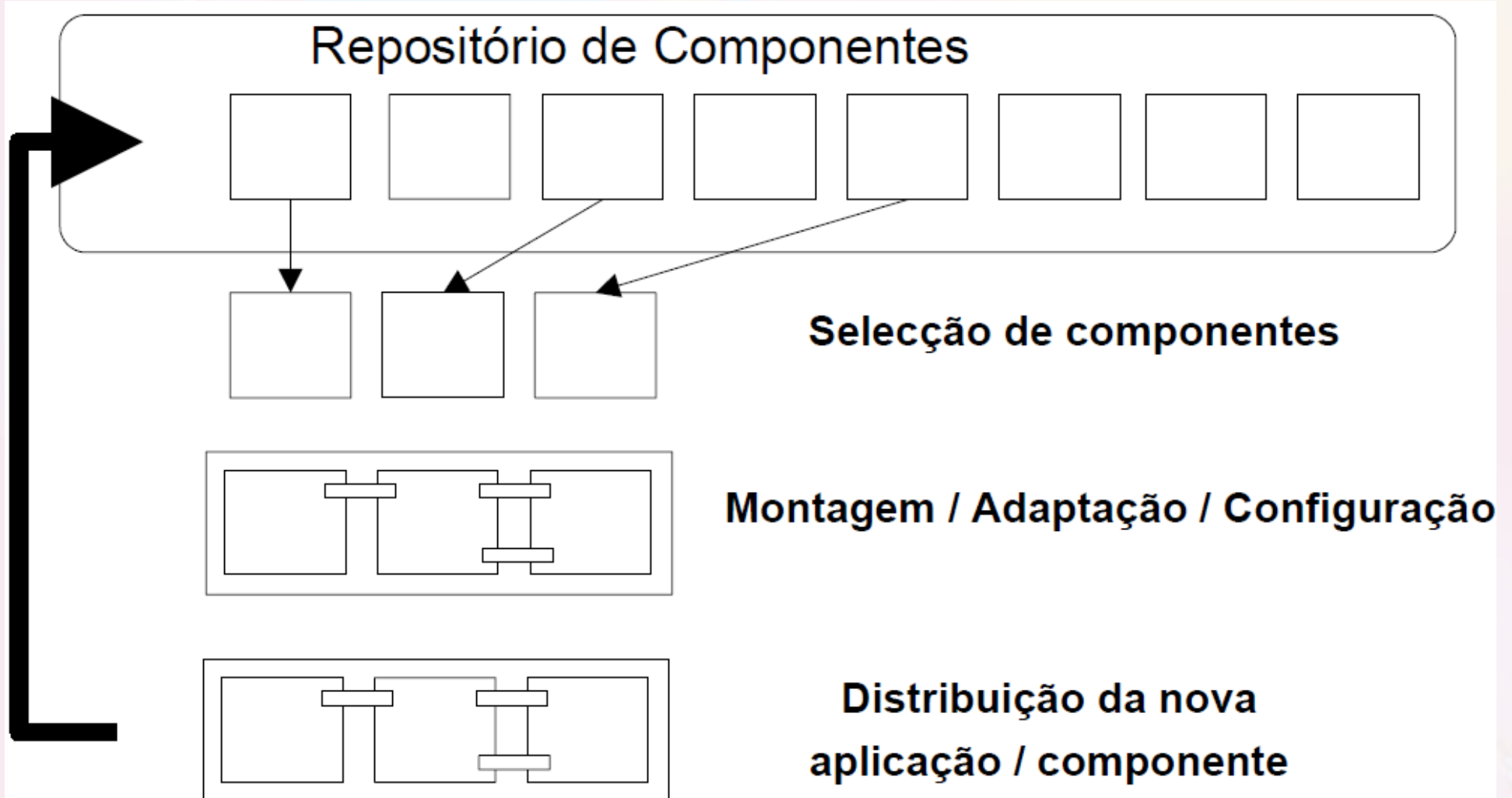
Component Software

Via Intermediária

- Component Software (Software de Componentes)
 - sistema feito de componentes de software
- “O conceito de *component software* representa uma via intermediária que pode resolver este problema.” (Szyperski, 2002)

Tradução do original feita pelo autor: “The concept of component software represents a middle path that could solve this problem.” (Szyperski, 2002)

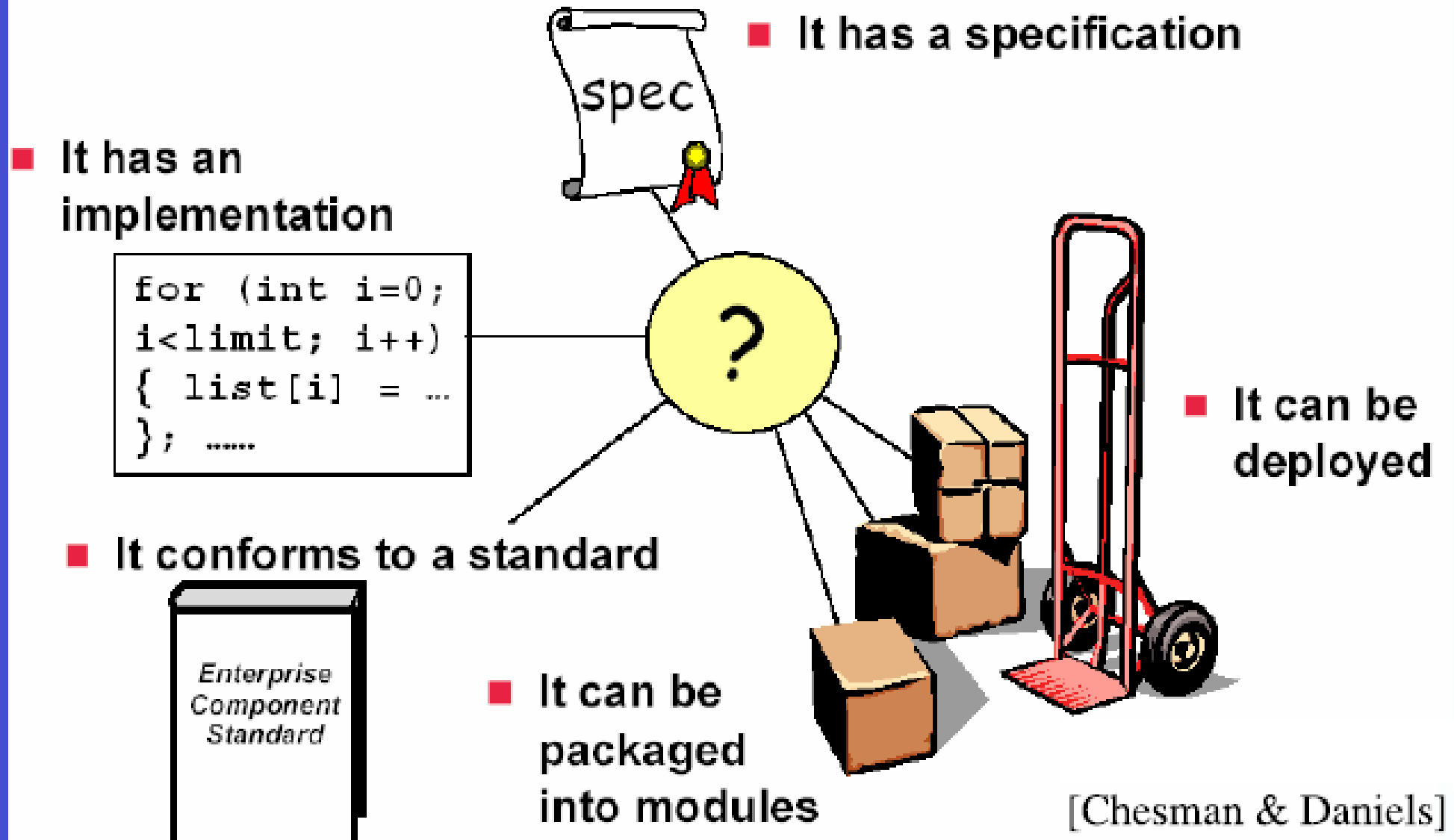
Programação por Componentes (Composição)



(Caires, 2002)

O que é um componente?

(Cheesman & Daniels, 2000)



Desafio

- Montar componentes para suporte à decisão em um Prontuário Eletrônico.

The screenshot displays the 'Handy patients enterprise edition' software interface. The top window title is 'Handy patients enterprise edition' with standard Windows window controls. The interface is divided into several sections:

- Patient Information:** Shows a photo of a baby, name 'David (8 month and 10 days)', age '10m (2 years and 3 months)', and family details: 'Mother: Teacher', 'Father: Financial advisor', 'Parents: Married'. Fields for 'Last: Anderson', 'First: David', 'Boy', and 'Birth: 5 January 2009' are visible. 'Age: 8 month and 10 days' and 'Patient nb: 3' are also shown.
- Navigation and Tools:** Includes 'File Edit View Help' menus, 'Appointments', 'Forms' (Meeting (Doctor), Full status (Doctor), Assistant, Billing, Reports, Statistics), 'Sheets' (Neurologic, Vascular, Cardiac, Respiratory, Abdomen, Exams, Radiology, Summary, Patient documents, Letter), and a 'Meetings' table.
- Meetings Table:**

Meeting	Date	Time
2 month checkup	5 Mar 09	2m,0d
1 month checkup	5 Feb 09	1m,0d
Respiration problem	22 Jan 09	17d
10 days checkup	13 Jan 09	8d
Control for return at home	9 Jan 09	4d
Birth	5 Jan 09	0d
- Diagnosis and Notes:** 'Diagnosis' section includes 'General', 'My Diagnosis', and 'Social'. 'Notes' section contains the text: 'Father ask many questions, add 10 minutes to consultation'. 'Current doctor' is 'Dr Herman'.
- Digestive Examination Report:** Titled 'Digestive' and dated 'Thursday, 22 Jan 2009'. It includes sections for 'Digestive inspection' (Normal), 'Digestive auscultation' (Normal abdomen noises), 'Digestive palpation' (Little pain on the right lower area), 'Liver' (No hepatomegaly), and 'Rectal'. Below the text are two anatomical diagrams: a photograph of a human torso with red arrows pointing to the lower right abdomen, and a diagram of the digestive system with a red arrow pointing to the lower right quadrant.
- Additional Elements:** A 'Documents manager' icon, a 'Page 1/1' indicator, and 'Previous page' and 'Next page' buttons are also visible.

By Oguntoye patients electronic medical record (free open source version), GPL,
<https://commons.wikimedia.org/w/index.php?curid=8894074>

Componente em UML

- “[...] sistemas de software de tamanho e complexidade arbitrários.”¹
- Componente²:
 - unidade modular
 - com interfaces bem definidas
 - substituível dentro do ambiente

1. “[...] software systems of arbitrary size and complexity” (Cook, 2015)

2. “[...] Component as a modular unit with well-defined Interfaces that is replaceable within its environment.” (Cook, 2015)

Especificação UML

- **OMG Unified Modeling Language (OMG UML) - version 2.5.1**
OMG
(2017)
<https://www.omg.org/spec/UML/2.5.1/>
- **Seção 11.6 - Components**





Componente DataSet

Componente DataSet

Objetivo

- Dar acesso a um conjunto de dados na forma de tabela, recuperados a partir de um arquivo CSV.

Passo 1: Classe DataSetComponent

DataSetComponent

- dataSource: String
- attributes: String[]
- instances: String[][]

- + «constructor» DataSetComponent()
- + getDataSource(): String
- + setDataSource(dataSource: String)
- + requestAttributes(): String[]
- + requestInstances(): String[][]
- readDS()

Passo 1: Classe DataSetComponent

- Atributos e métodos de interesse:

DataSetComponent

- dataSource: String

+ «constructor» DataSetComponent()
+ getSource(): String
+ setDataSource(dataSource: String)
+ requestAttributes(): String[]
+ requestInstances(): String[][]

Dependency Inversion Principle (DIP)

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

Passo 2: Interfaces

«interface»

IDataSetProperties

+ getSource(): String
+ setSource(dataSource: String)

«interface»

ITableProducer

+ requestAttributes(): String[]
+ requestInstances(): String[][]



DataSetComponent

- dataSource: String

+ «constructor» DataSetComponent()



Passo 3: Transformando em um Componente

Passo 2: Interfaces

«interface»

IDataSetProperties

+ getSource(): String
+ setSource(dataSource: String)

«interface»

ITableProducer

+ requestAttributes(): String[]
+ requestInstances(): String[][]



DataSetComponent

- dataSource: String

+ «constructor» DataSetComponent()

Passo 3: Componente

«interface»

IDataSetProperties

+ getSource(): String
+ setSource(dataSource: String)

«interface»

ITableProducer

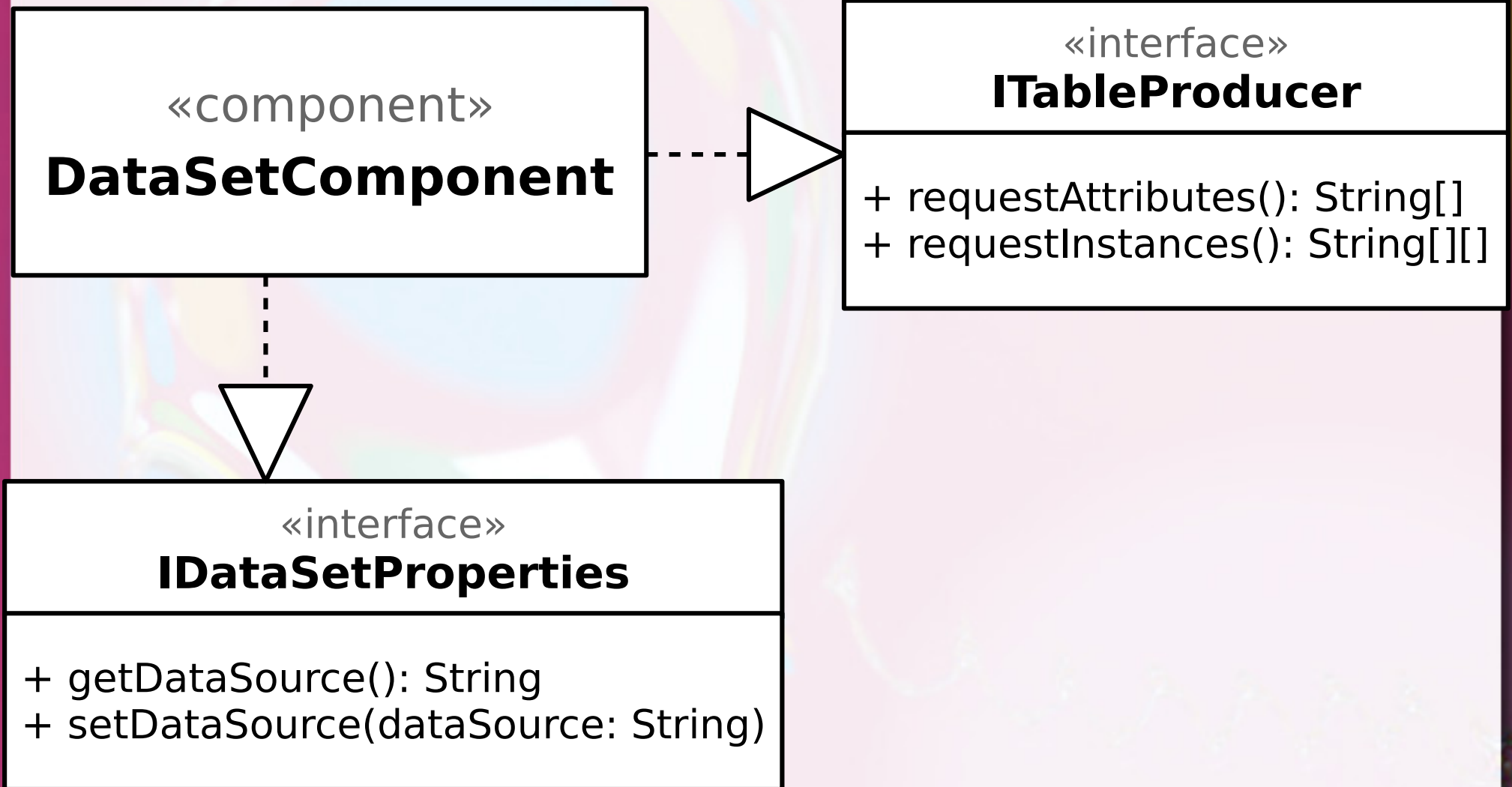
+ requestAttributes(): String[]
+ requestInstances(): String[][]



«component»

DataSetComponent

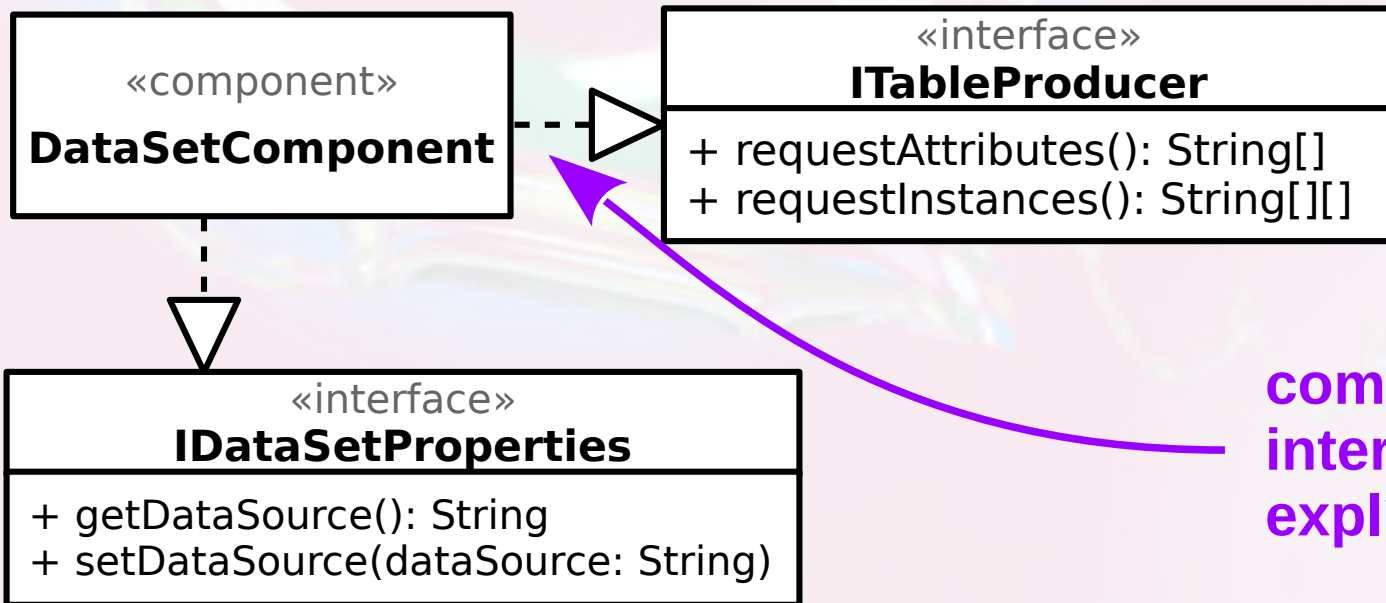
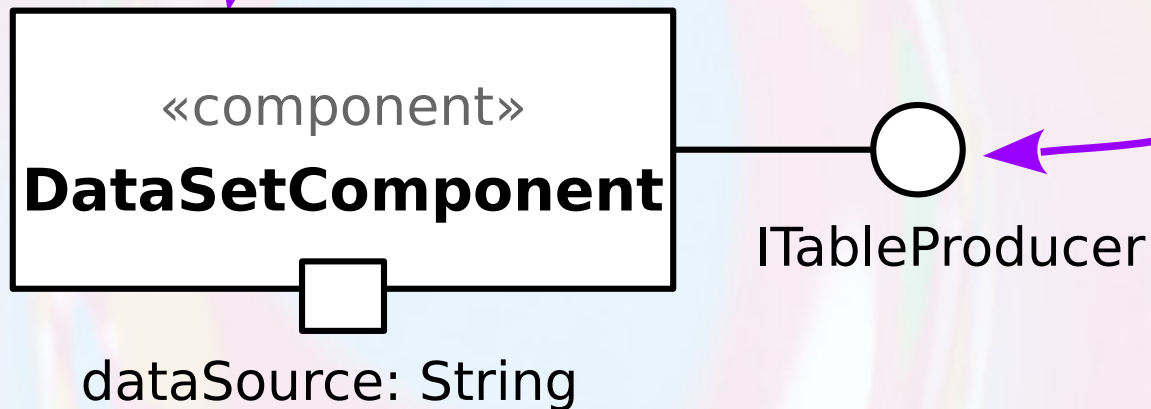
Passo 3: Componente



Notação Blackbox

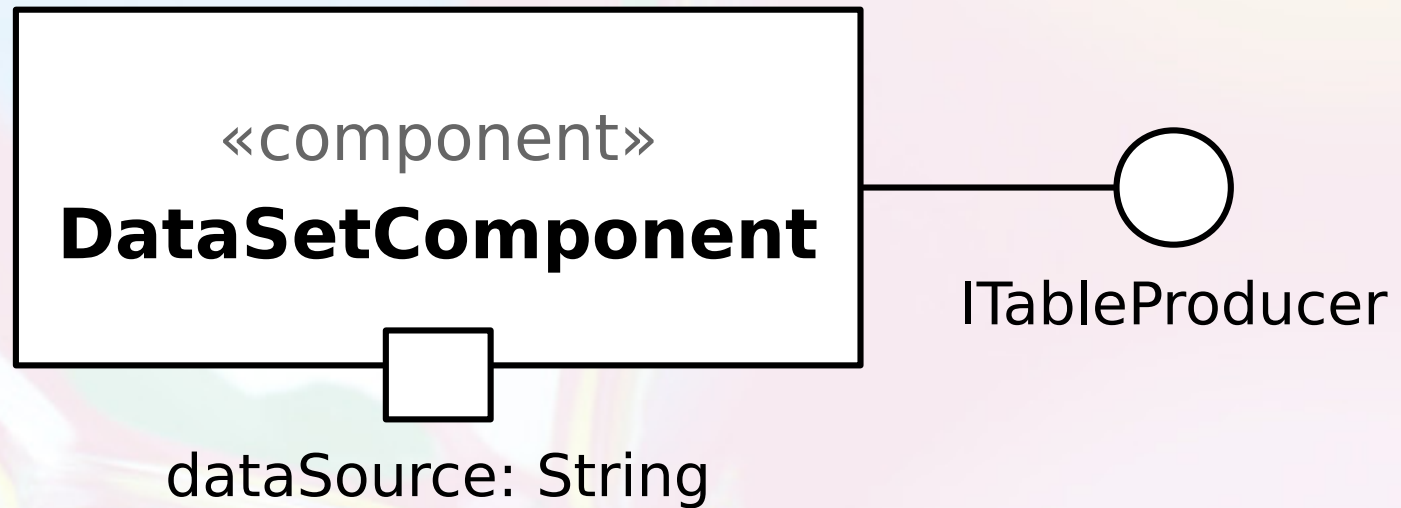
componente em notação blackbox

componente realiza interface (representação lollipop)



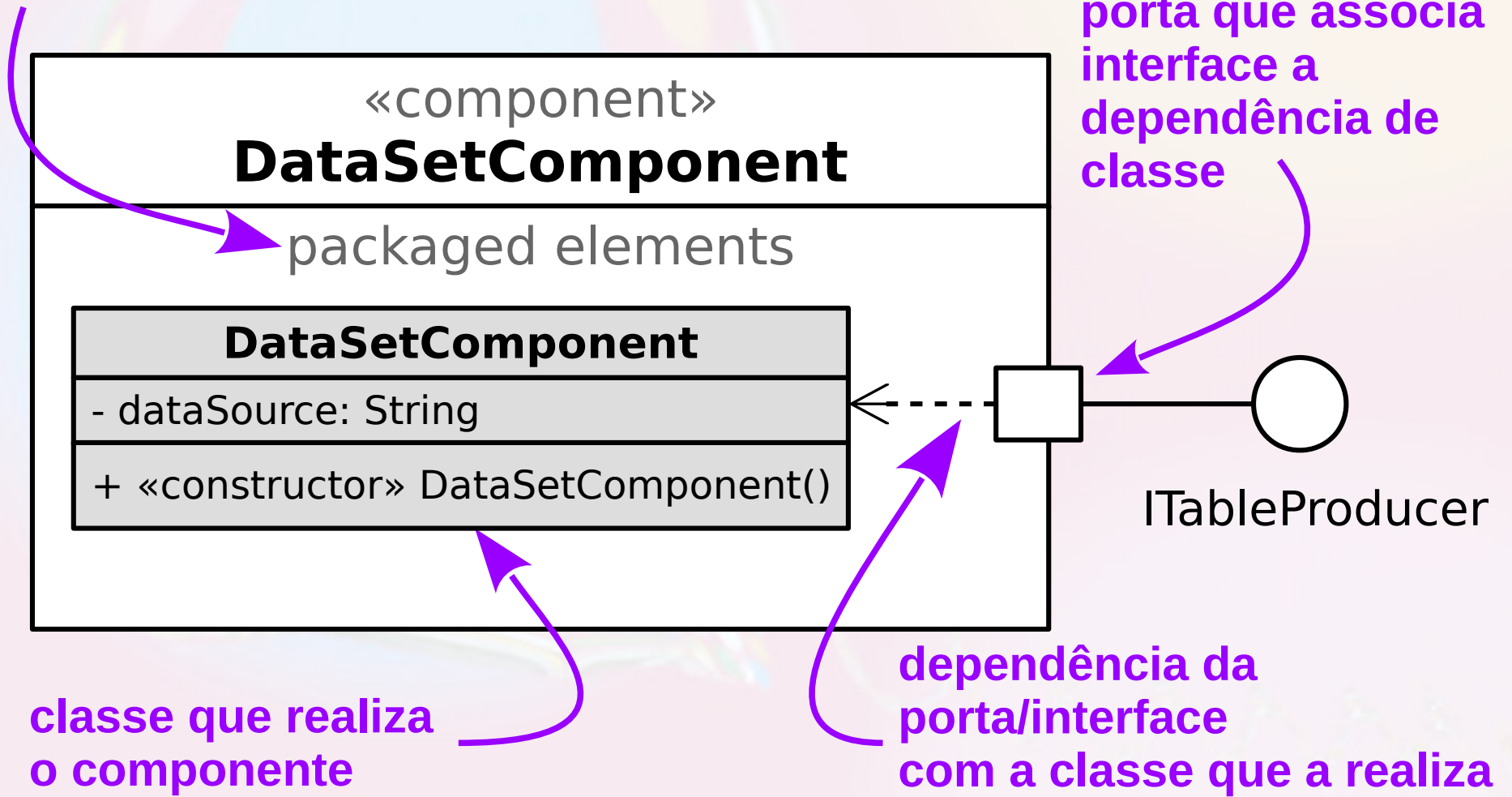
componente realiza interface (representação explícita)

Componente DataSet

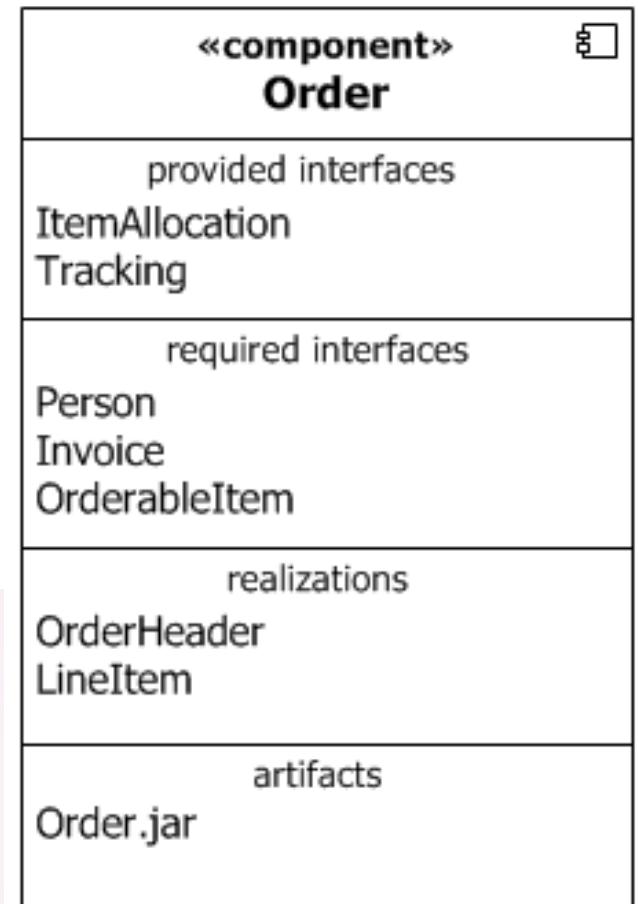
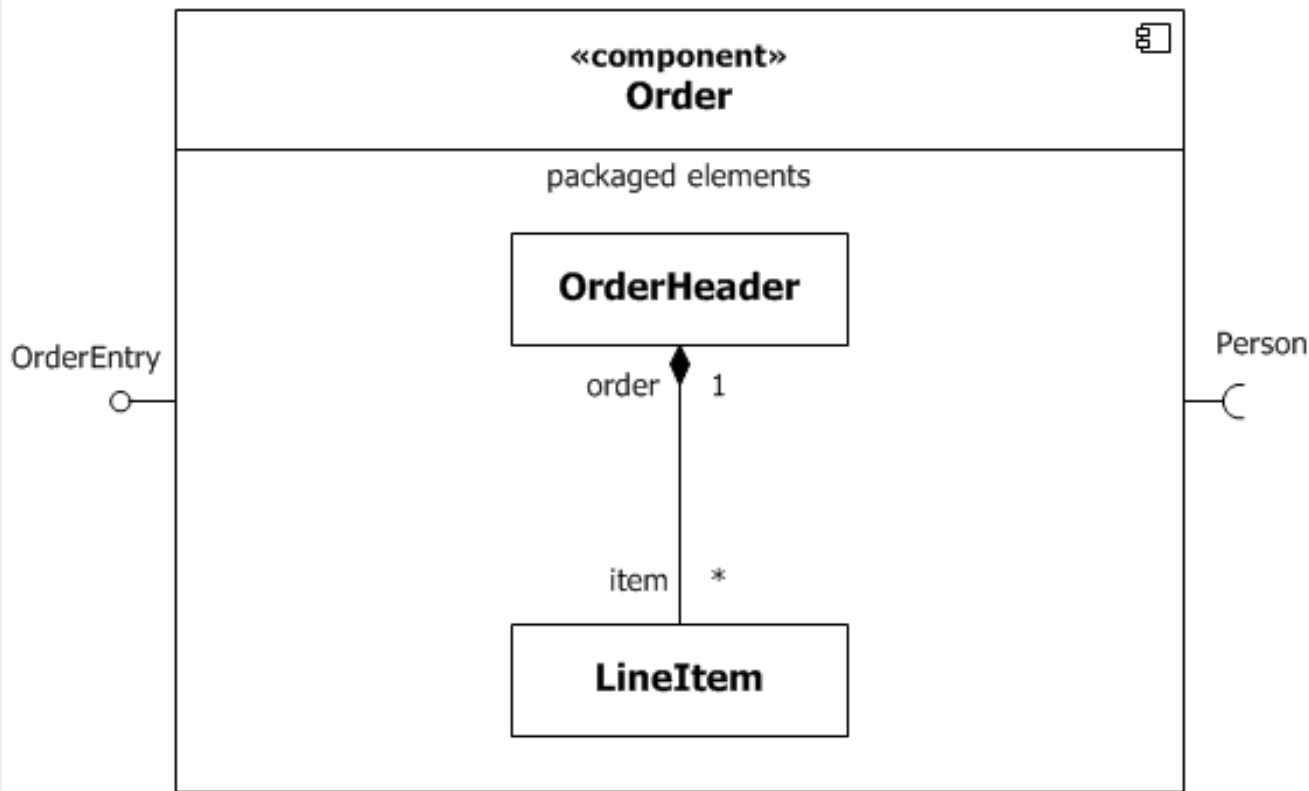


Realizando o Componente

compartimento opcional que mostra elementos que são parte do componente

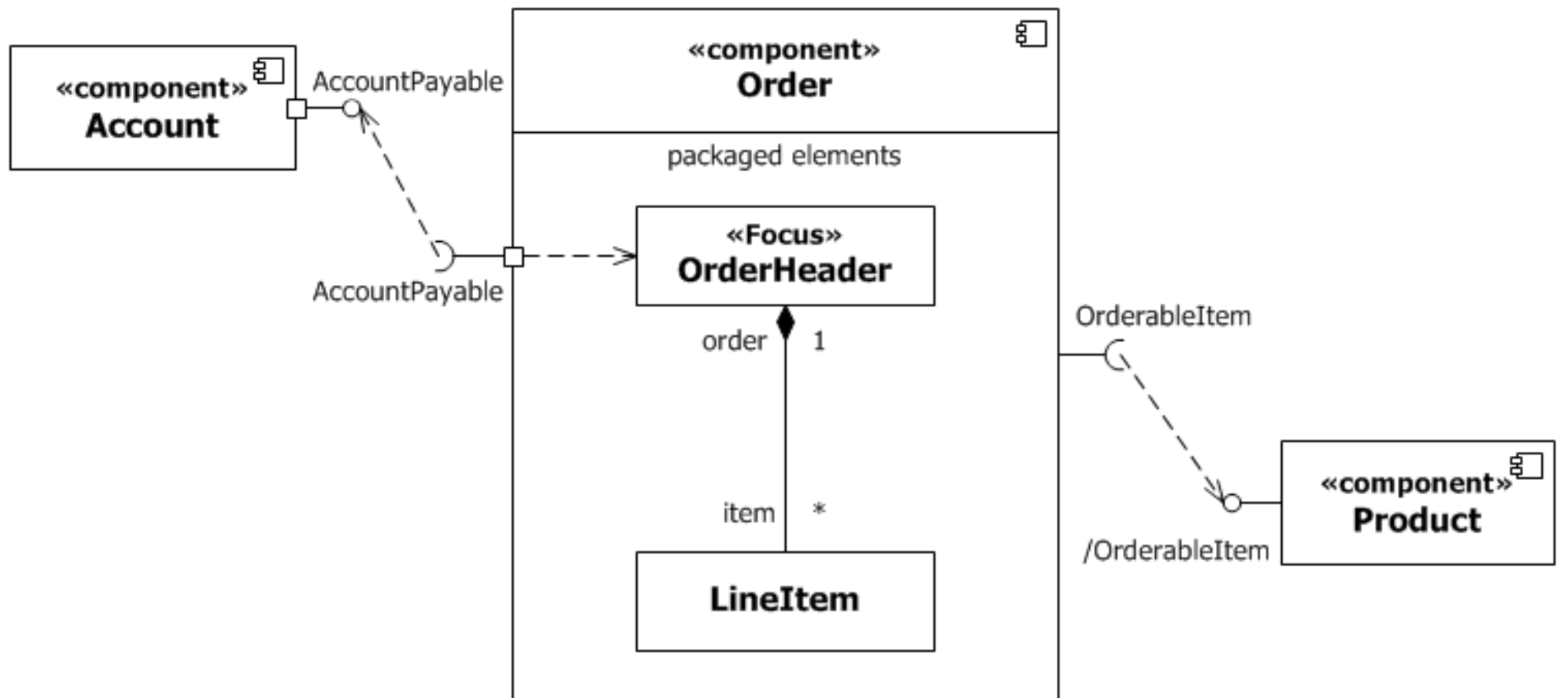


Duas perspectivas de Whitebox OMG UML (Cook et al., 2015)



Whitebox com portas

OMG UML (Cook et al., 2015)



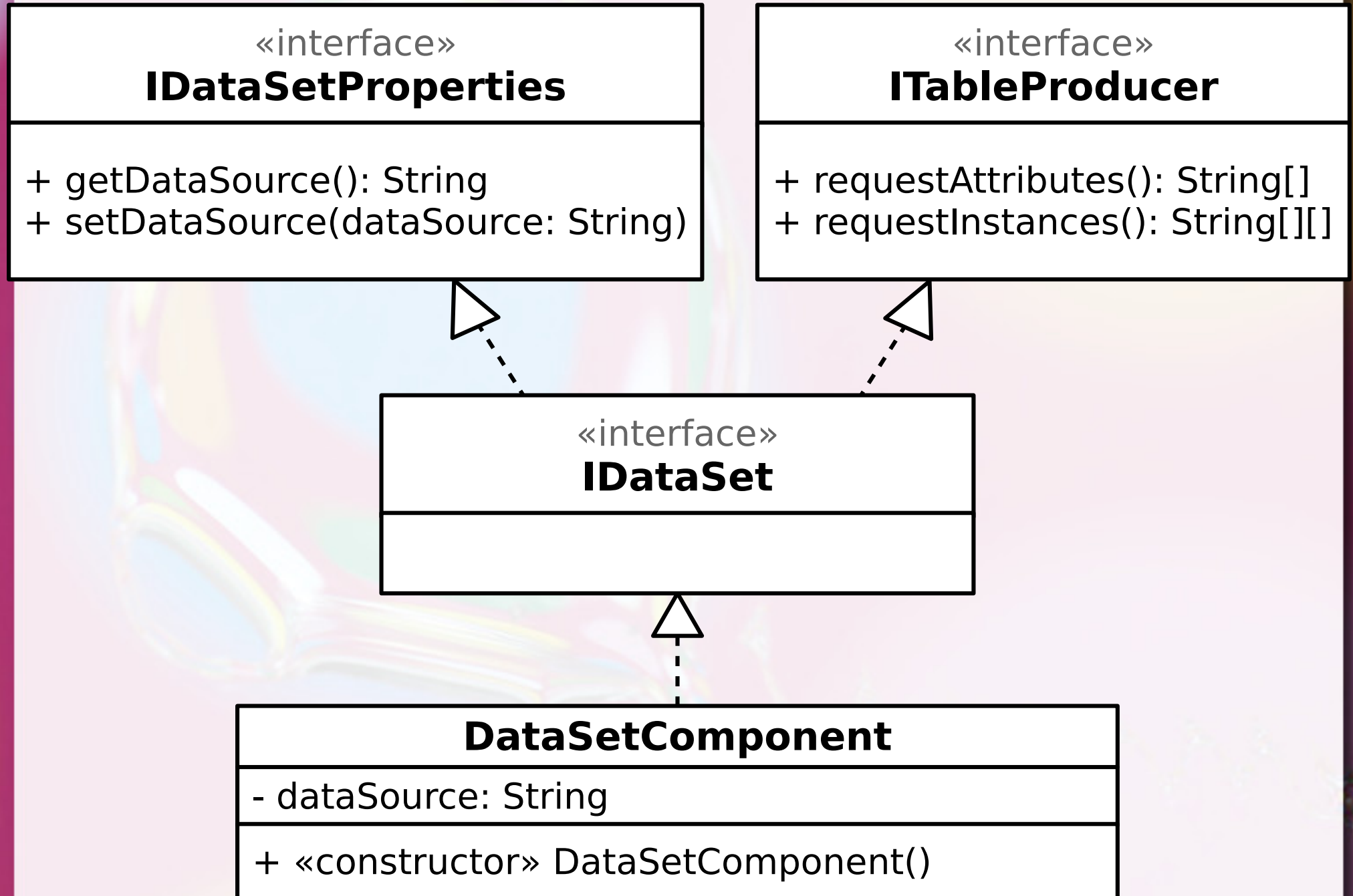
Visão Externa

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



Unificando Interfaces

Unificando Interfaces



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

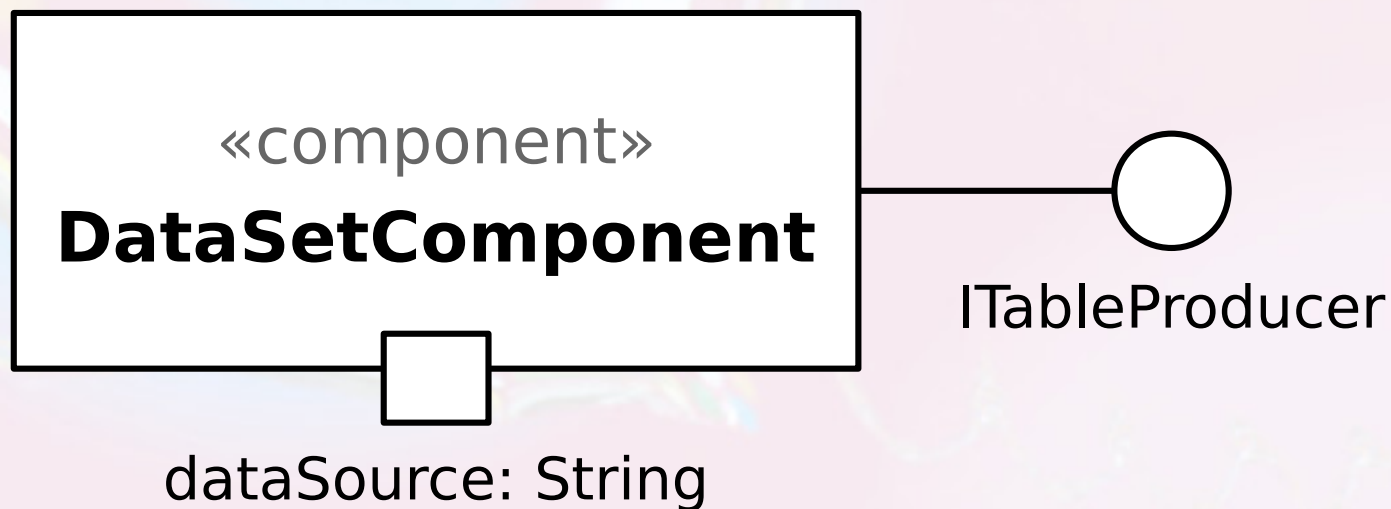
Perspectiva Orientada a Objetos de Componentes

- 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

```
IDataSet ds = new DataSetComponent ();
```





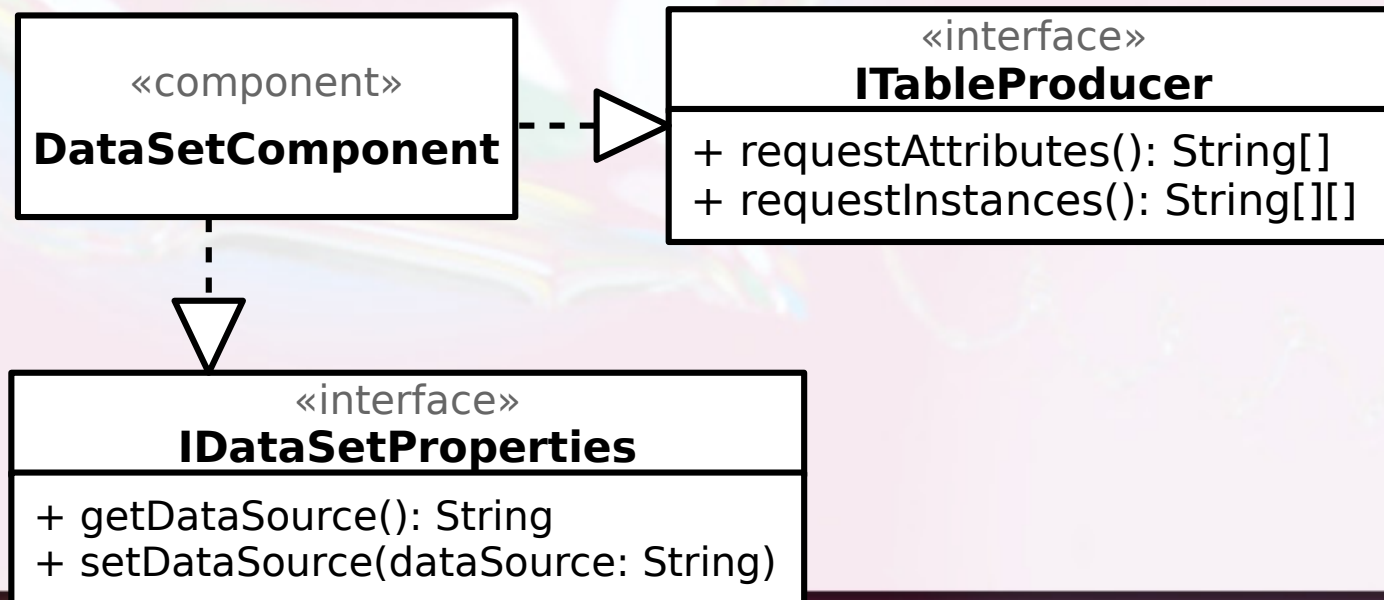
Usando Serviços do Componente

Acessando os Serviços

```
IDataSet ds = new DataSetComponent();
ds.setDataSource("...");

System.out.println("=== Attributes ===");
String attributes[] = ds.requestAttributes();
for (int a = 0; a < attributes.length-1; a++)
    System.out.print(attributes[a] + ", ");
System.out.println(attributes[attributes.length-1]);

System.out.println();
System.out.println("=== Instances ===");
String instances[][] = ds.requestInstances();
for (int i = 0; i < instances.length; i++) {
    for (int a = 0; a < attributes.length-1; a++)
        System.out.print(instances[i][a] + ", ");
    System.out.println(instances[i][attributes.length-1]);
}
```





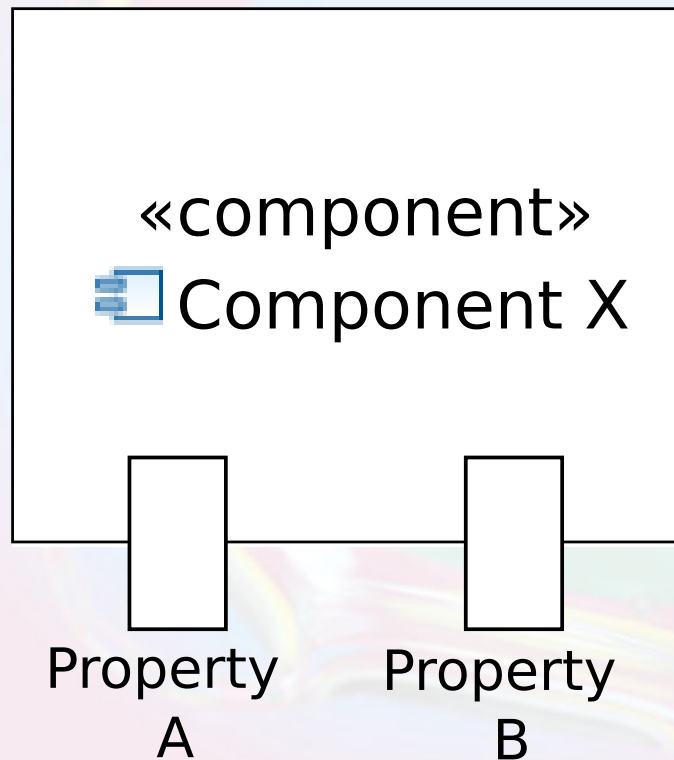
Propriedades

Propriedades

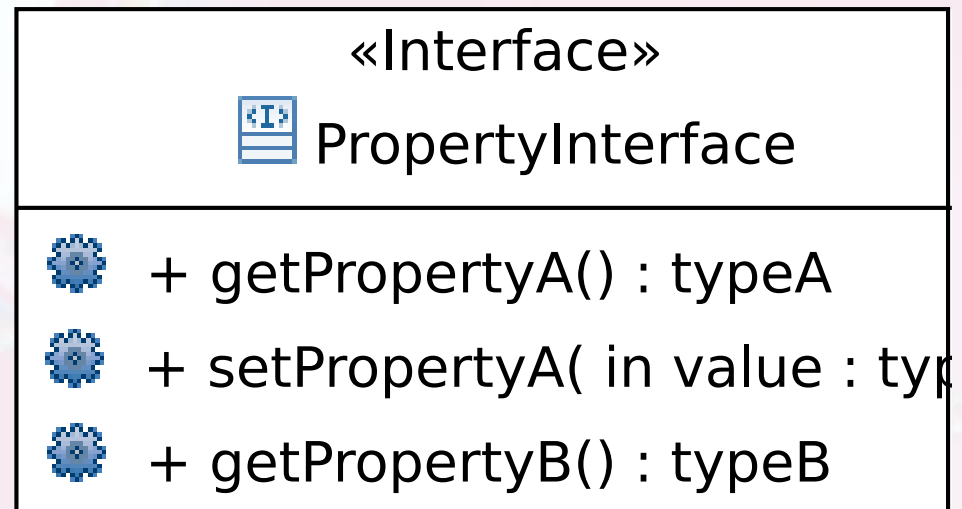
- Campos com valores que podem ser consultados externamente e eventualmente modificados.
- Permitem a customização externa do componente.
- Ligadas ao princípio de instância de componente e componente *stateful*.

Propriedades

Notação CORBA Component Model



PropertyB é somente leitura

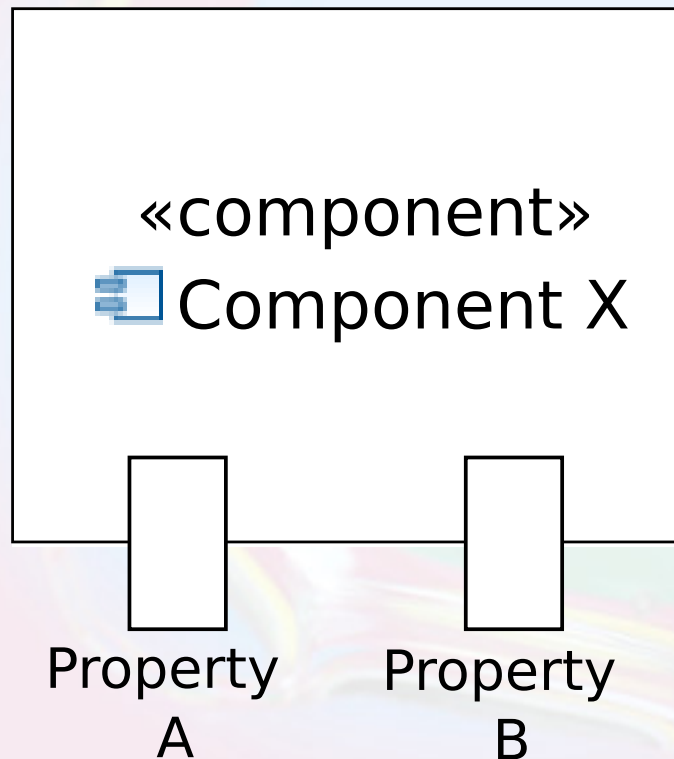


Propriedades em Java

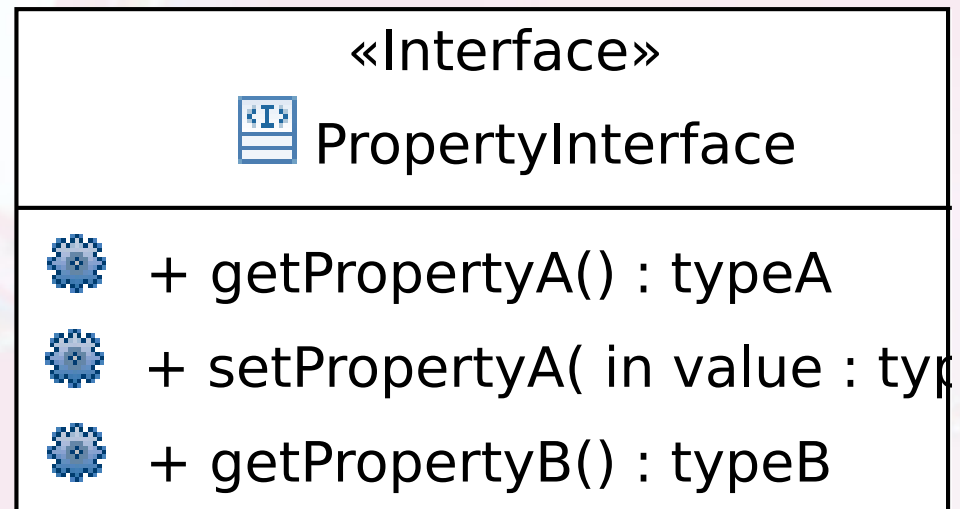
- Expostas através de métodos:
 - prefixo “get” → leitura
 - prefixo “set” → modificação
- Somente leitura
 - não têm método “set”

Propriedades

Notação CORBA Component Model



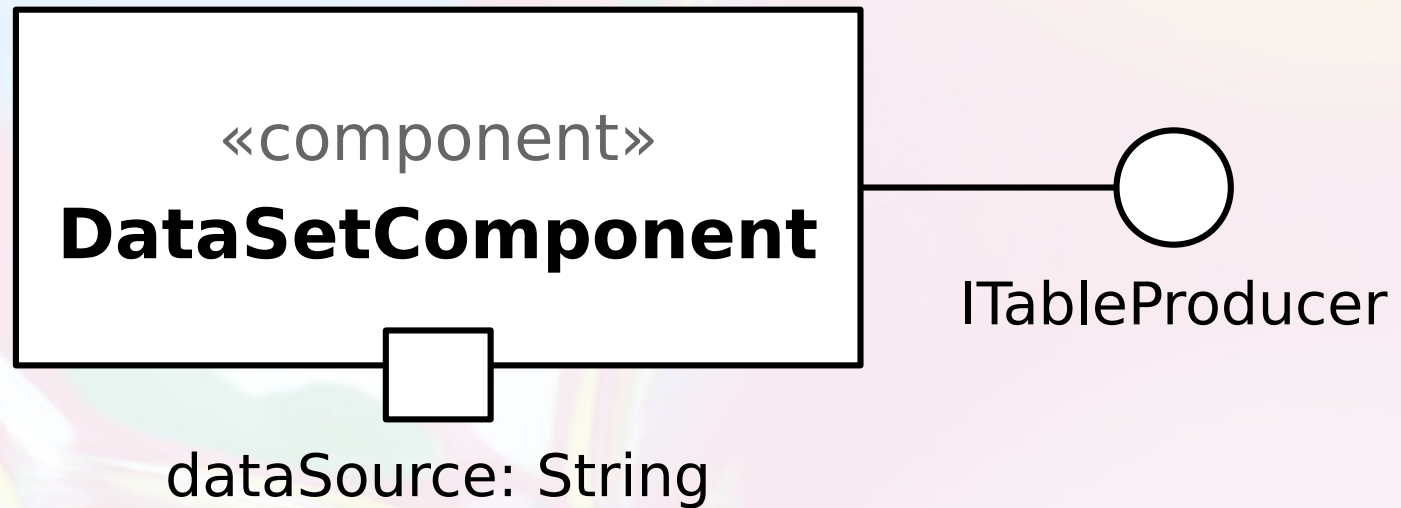
PropertyB é somente leitura





Estudo Caso

Componente DataSet





Interface Requerida

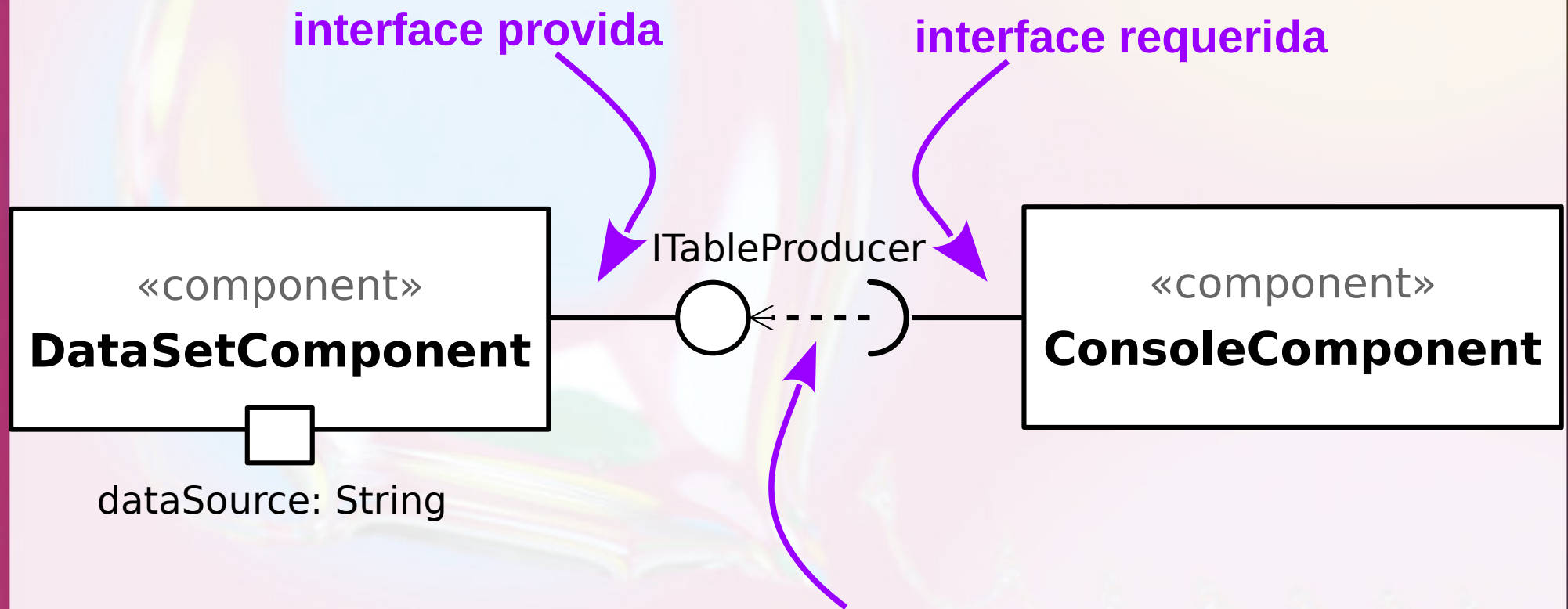
Interface Requerida

- Explícita a dependência de um componente por uma interface de outro componente.
- Interface X requerida por A = A requer um componente que oferece interface X



**Interfaces Providas e Requeridas
tornam explícitas todas as
dependências**

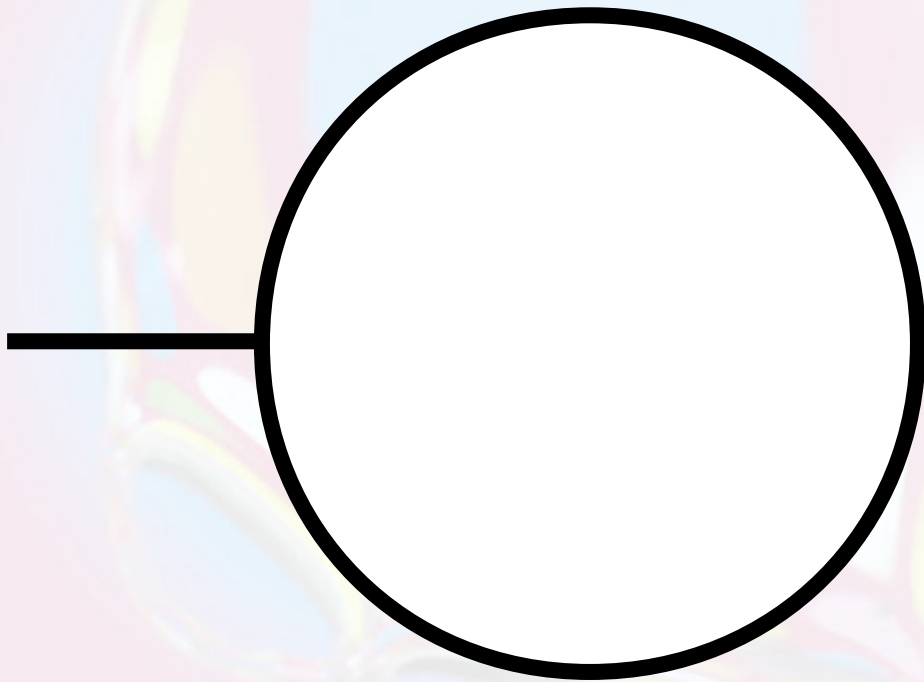
Interface Provida e Requerida Componente Console (blackbox)



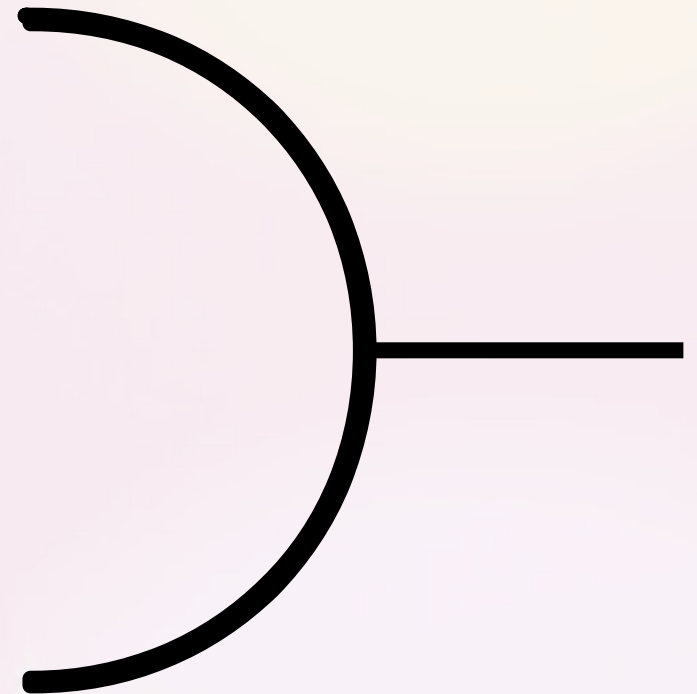
dependência entre a interface provida e a requerida

Interface Provida e Requerida

Provida

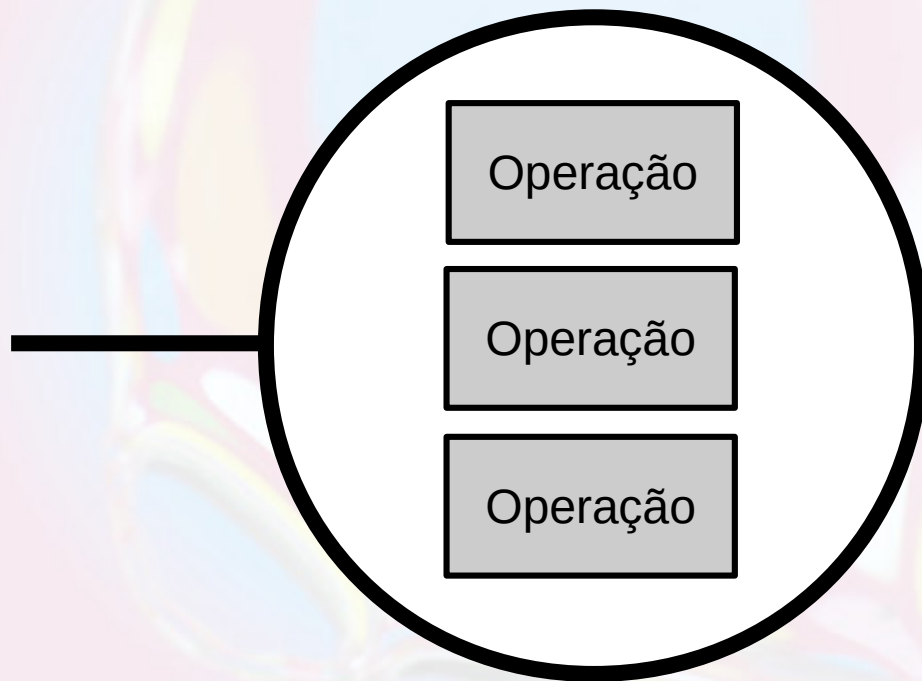


Requerida

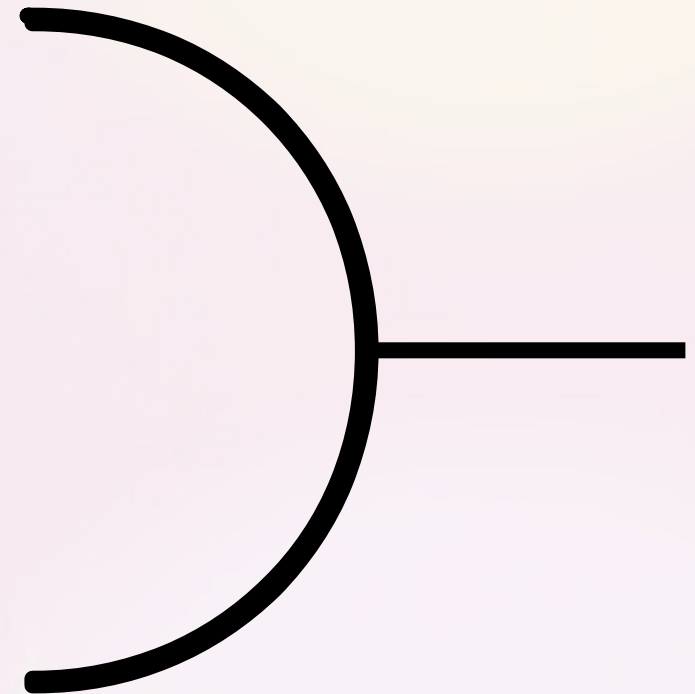


Interface Provida e Requerida

Provida



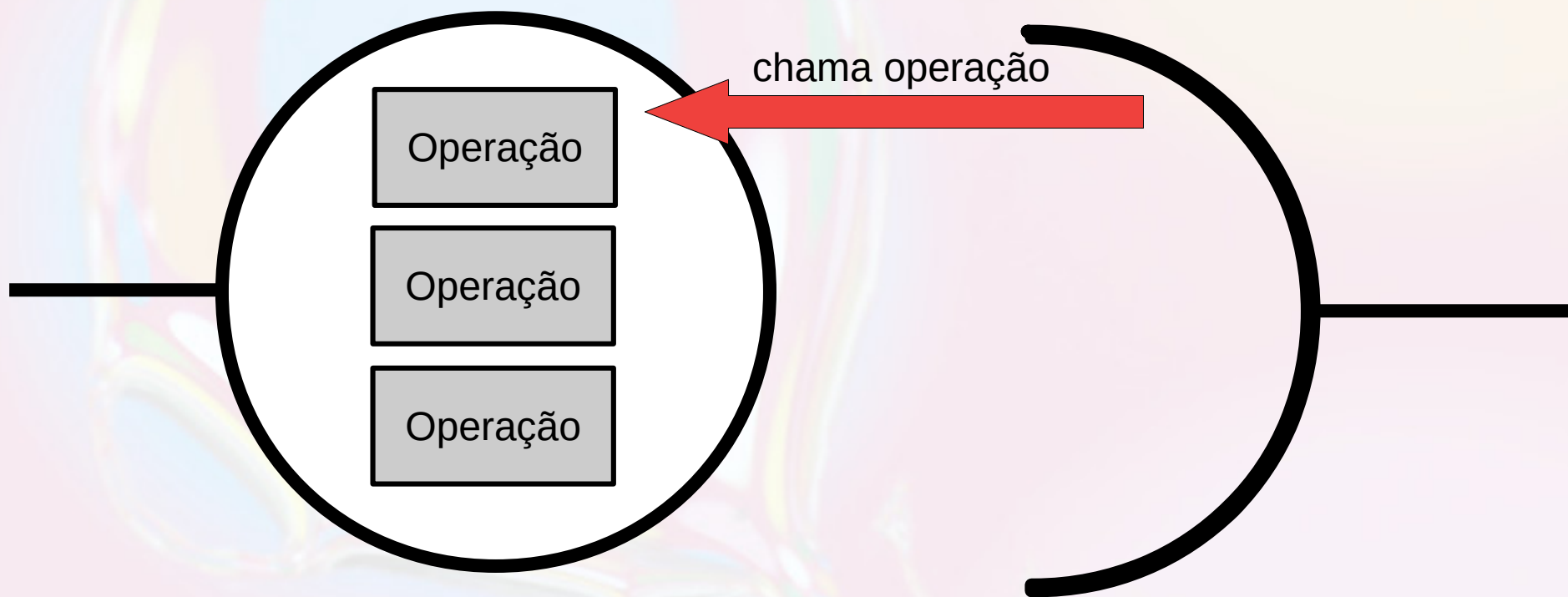
Requerida



Interface Provida e Requerida

Provida

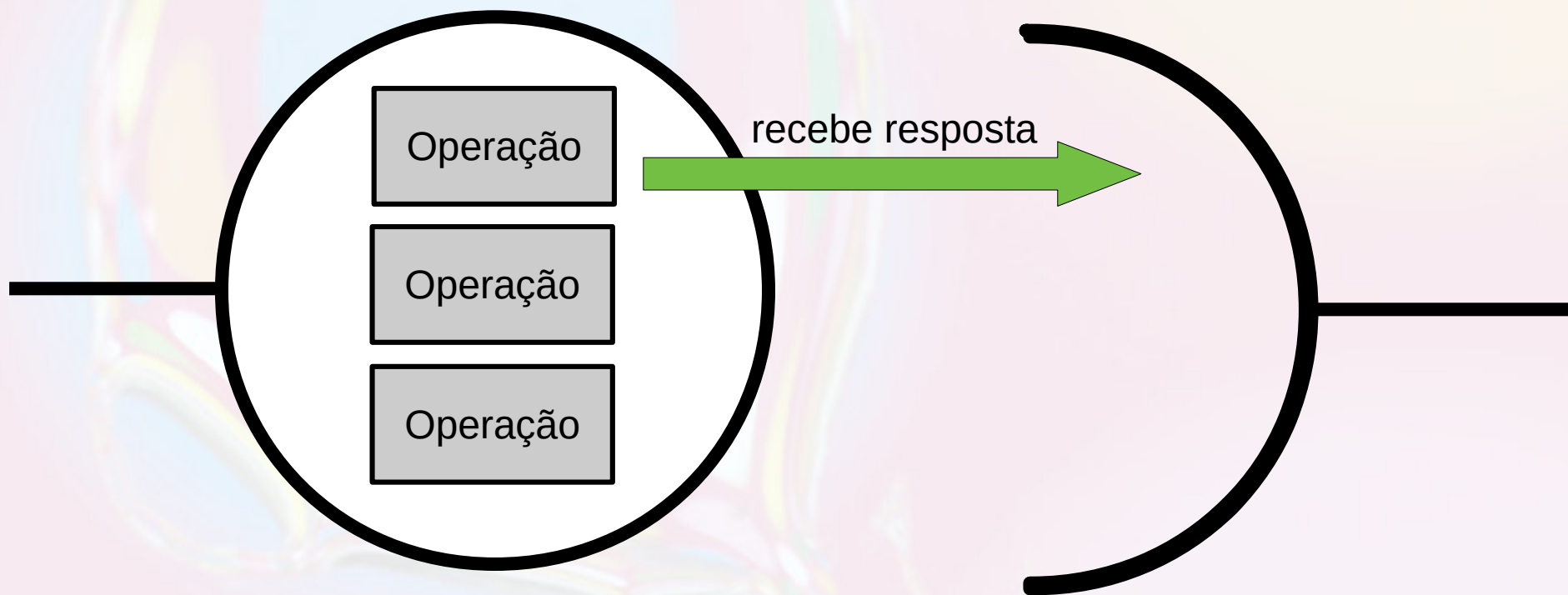
Requerida



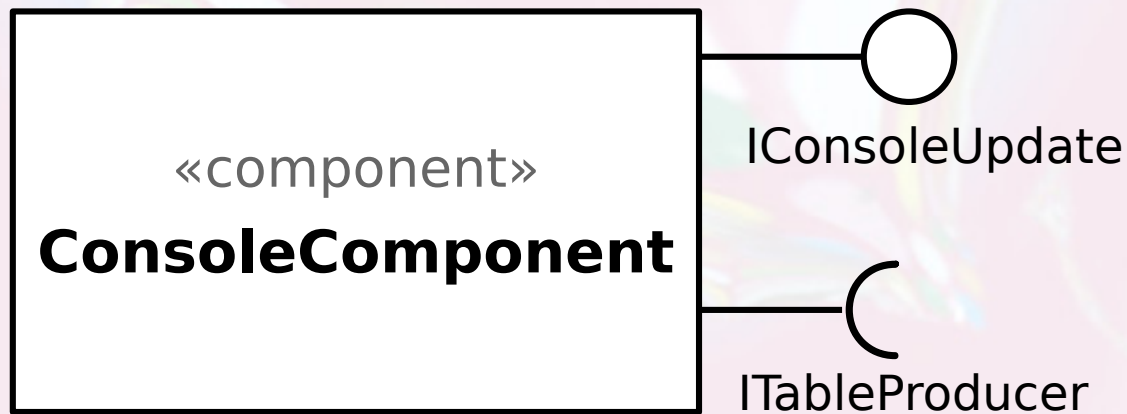
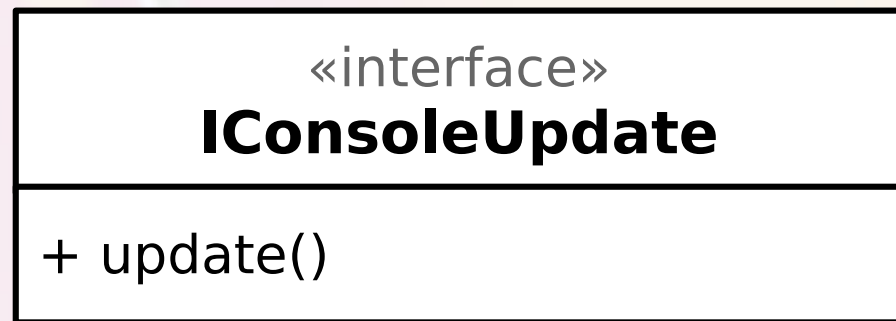
Interface Provida e Requerida

Provida

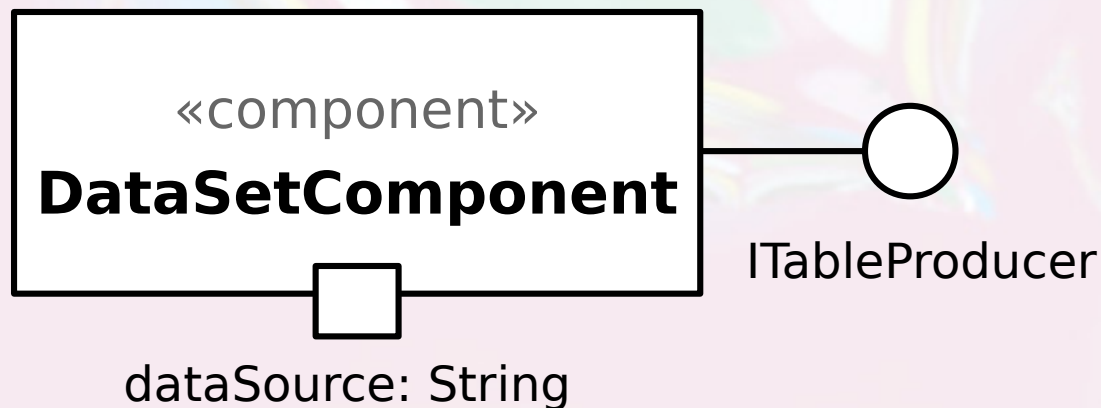
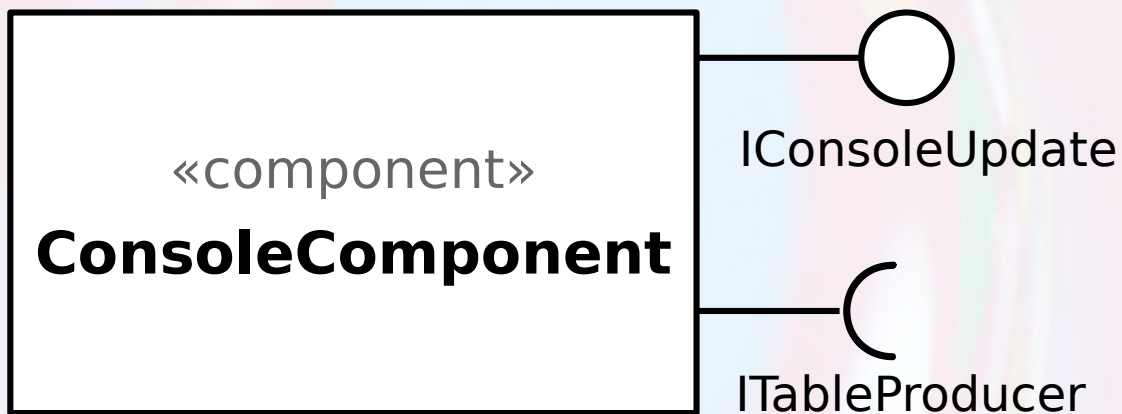
Requerida



Componente Console

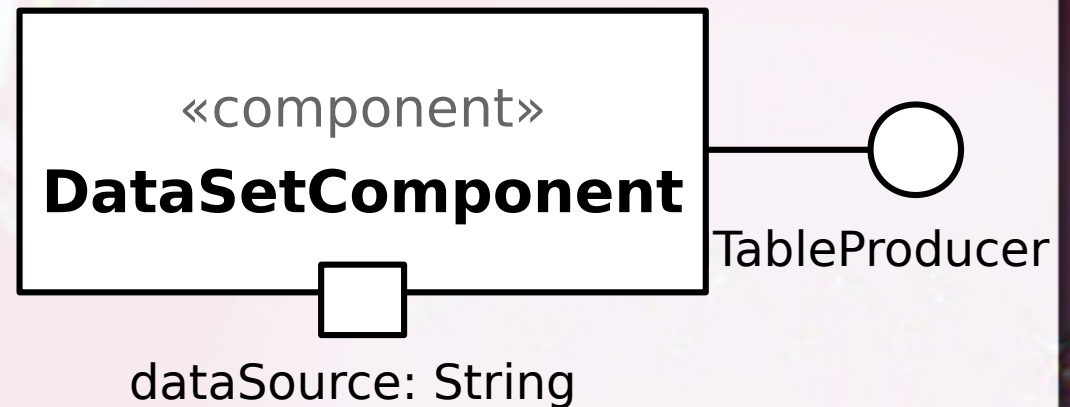


Conectando Componentes



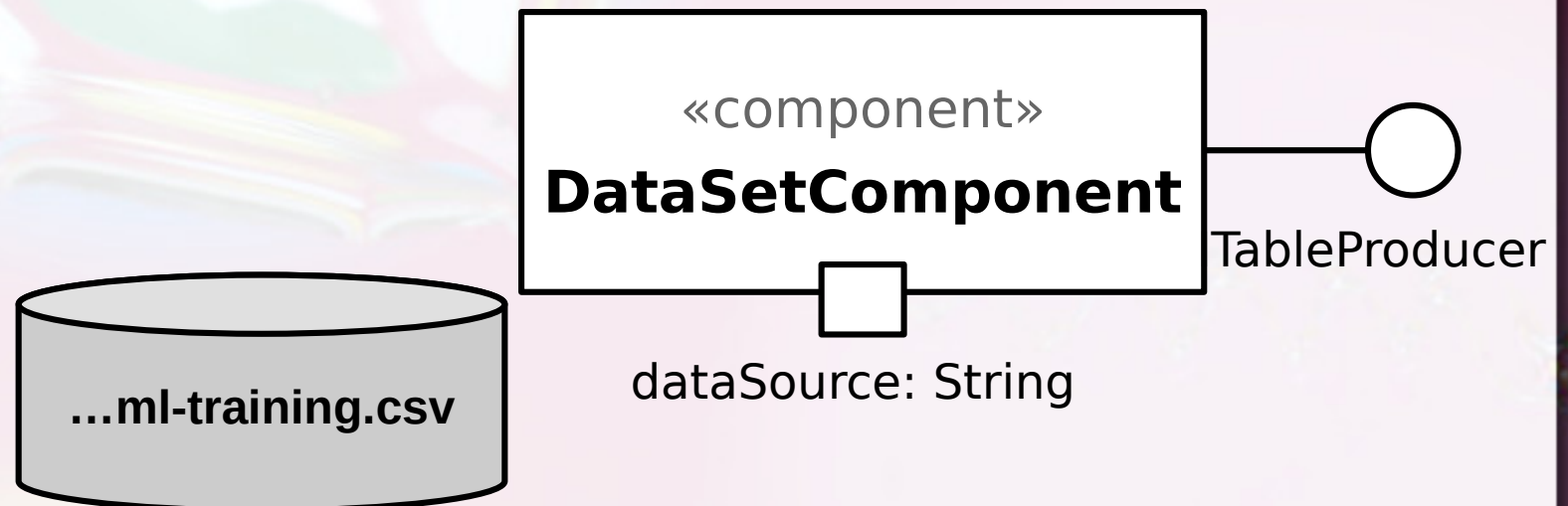
Conectando Componentes

```
IDataSet dataset = new DataSetComponent();
```



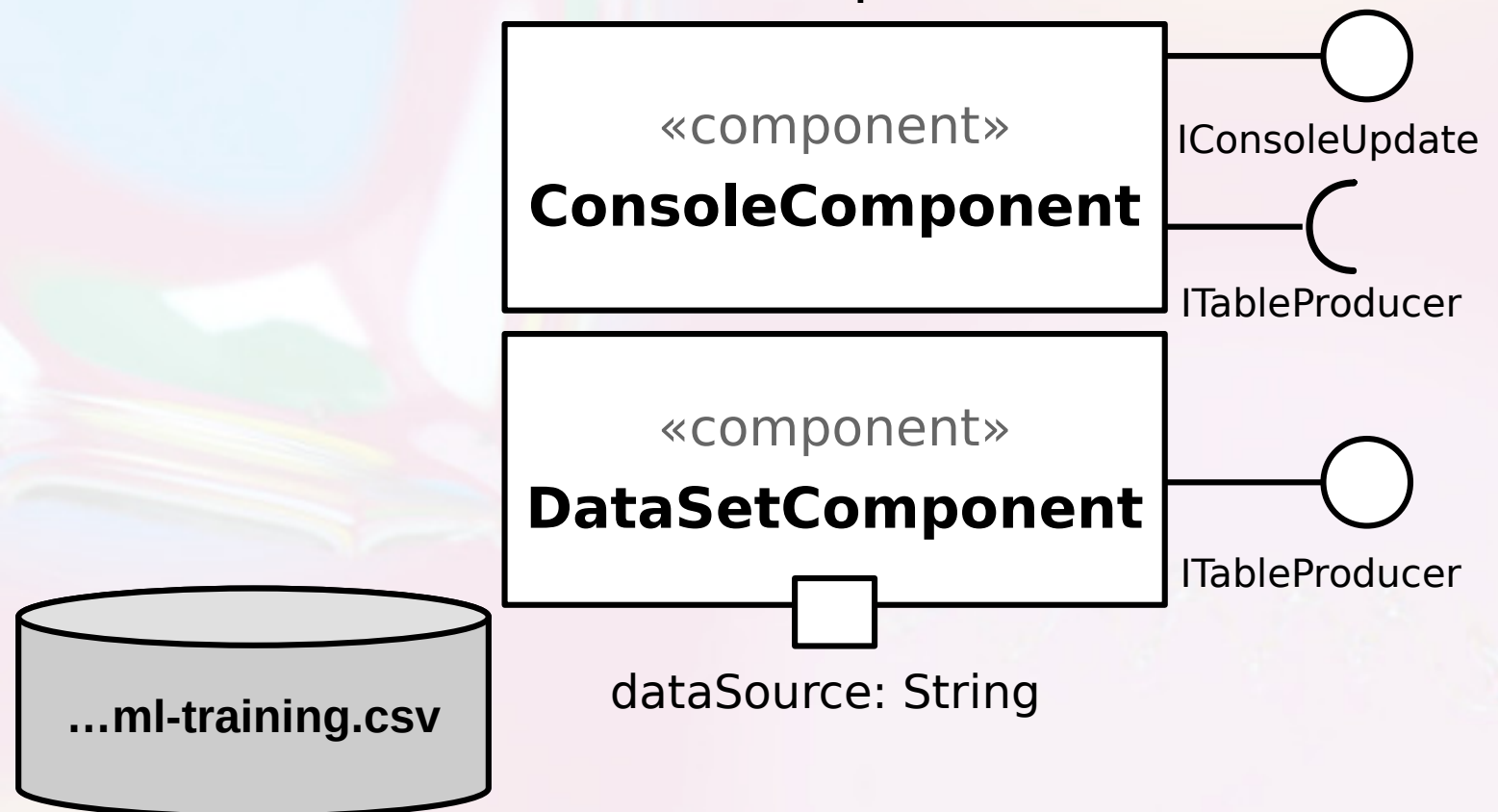
Conectando Componentes

```
IDataSet dataset = new DataSetComponent();  
dataset.setDataSource("...ml-training.csv");
```



Conectando Componentes

```
IDataSet dataset = new DataSetComponent();  
dataset.setDataSource("...csv");  
IConsole console = new ConsoleComponent();
```



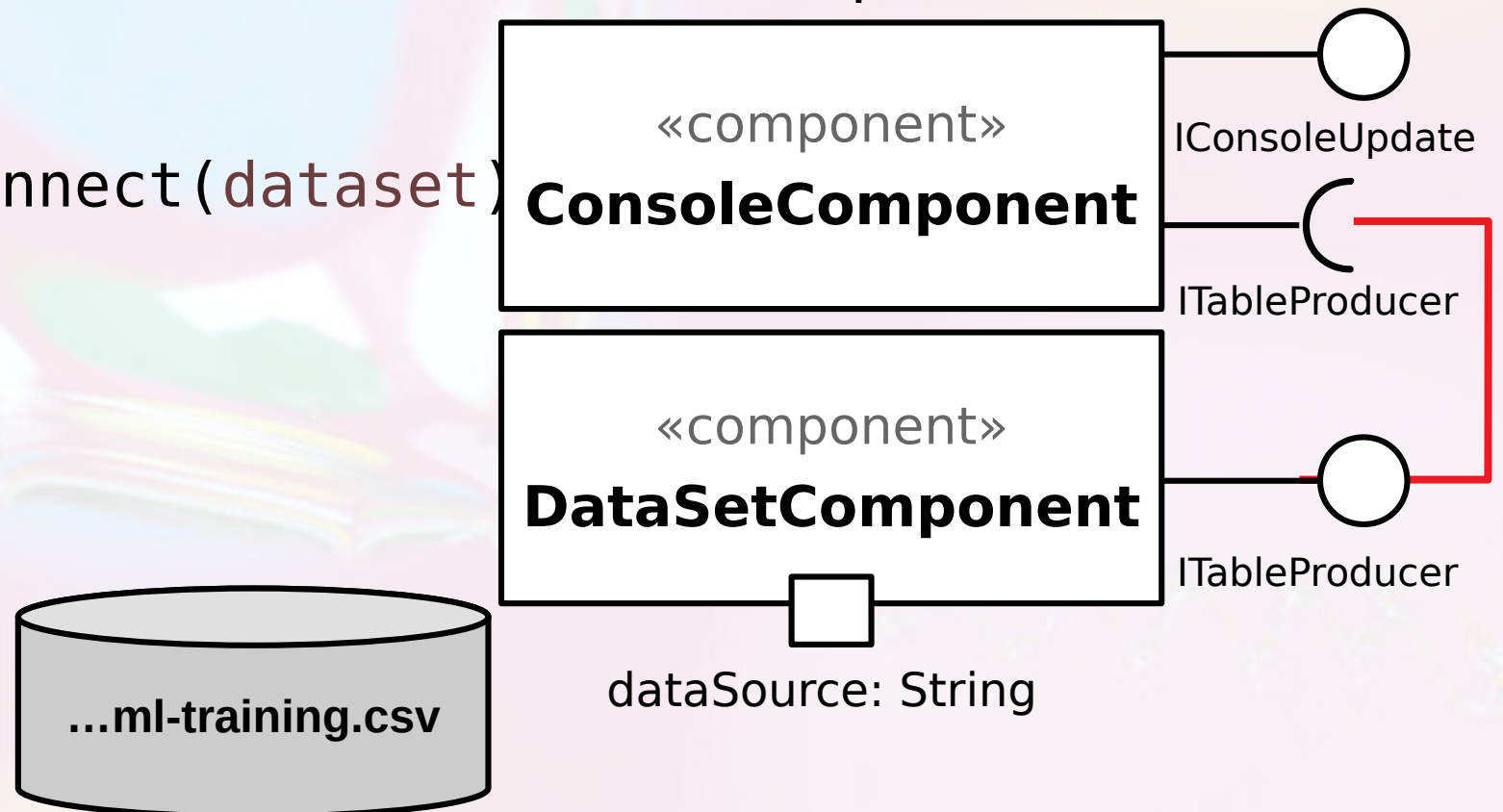
Conectando Componentes

```
IDataSet dataset = new DataSetComponent();
```

```
dataset.setDataSource("...csv");
```

```
IConsole console = new ConsoleComponent();
```

```
console.connect(dataset)
```



Conectando Componentes

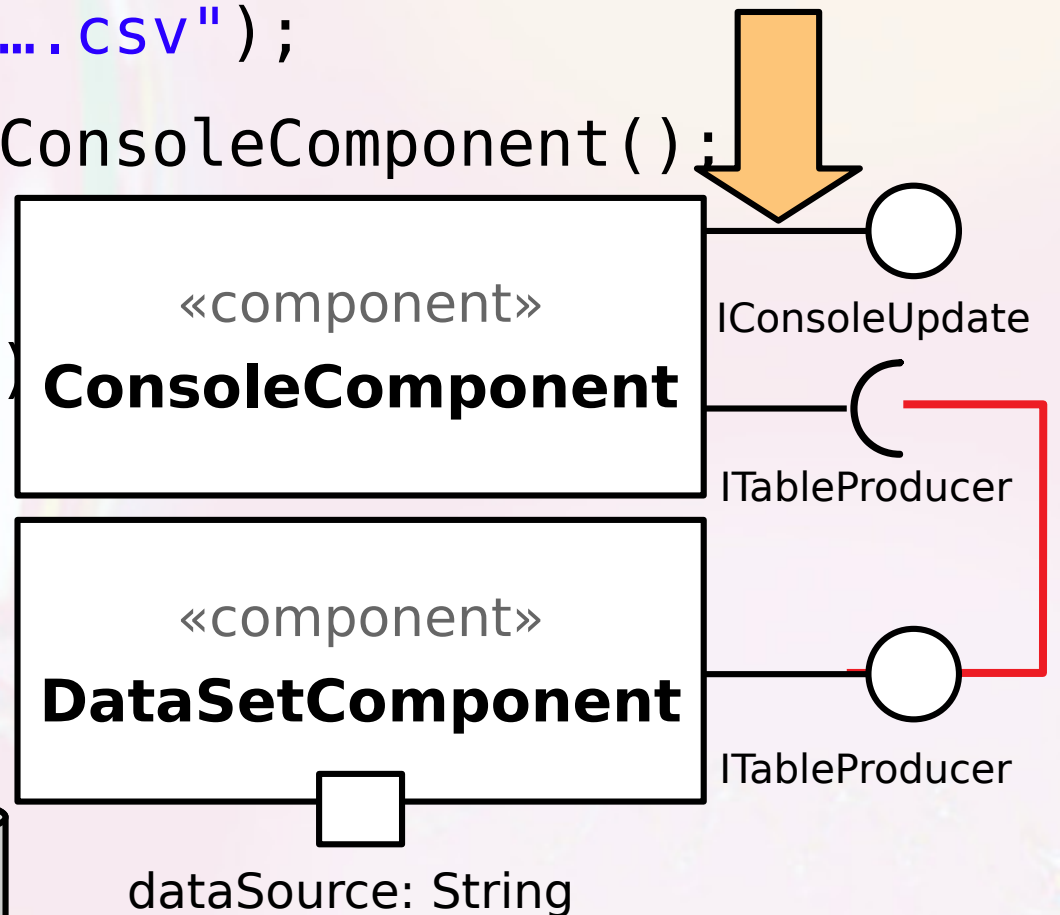
```
IDataSet dataset = new DataSetComponent();
```

```
dataset.setDataSource("...ml-training.csv");
```

```
IConsole console = new ConsoleComponent();
```

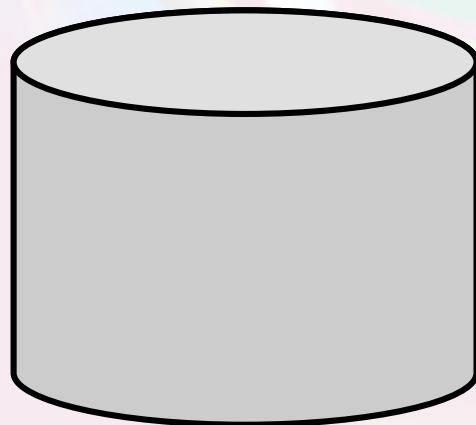
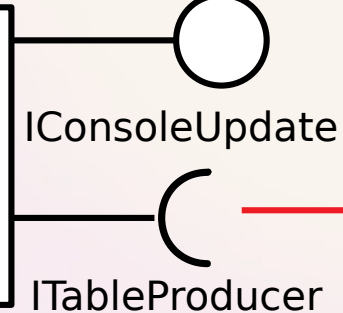
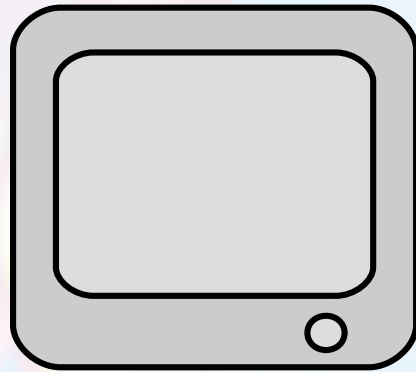
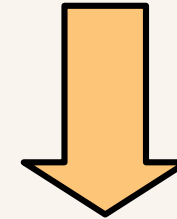
```
console.connect(dataset);
```

```
console.update();
```

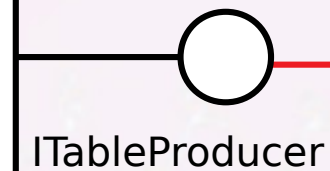


Conectando Componentes

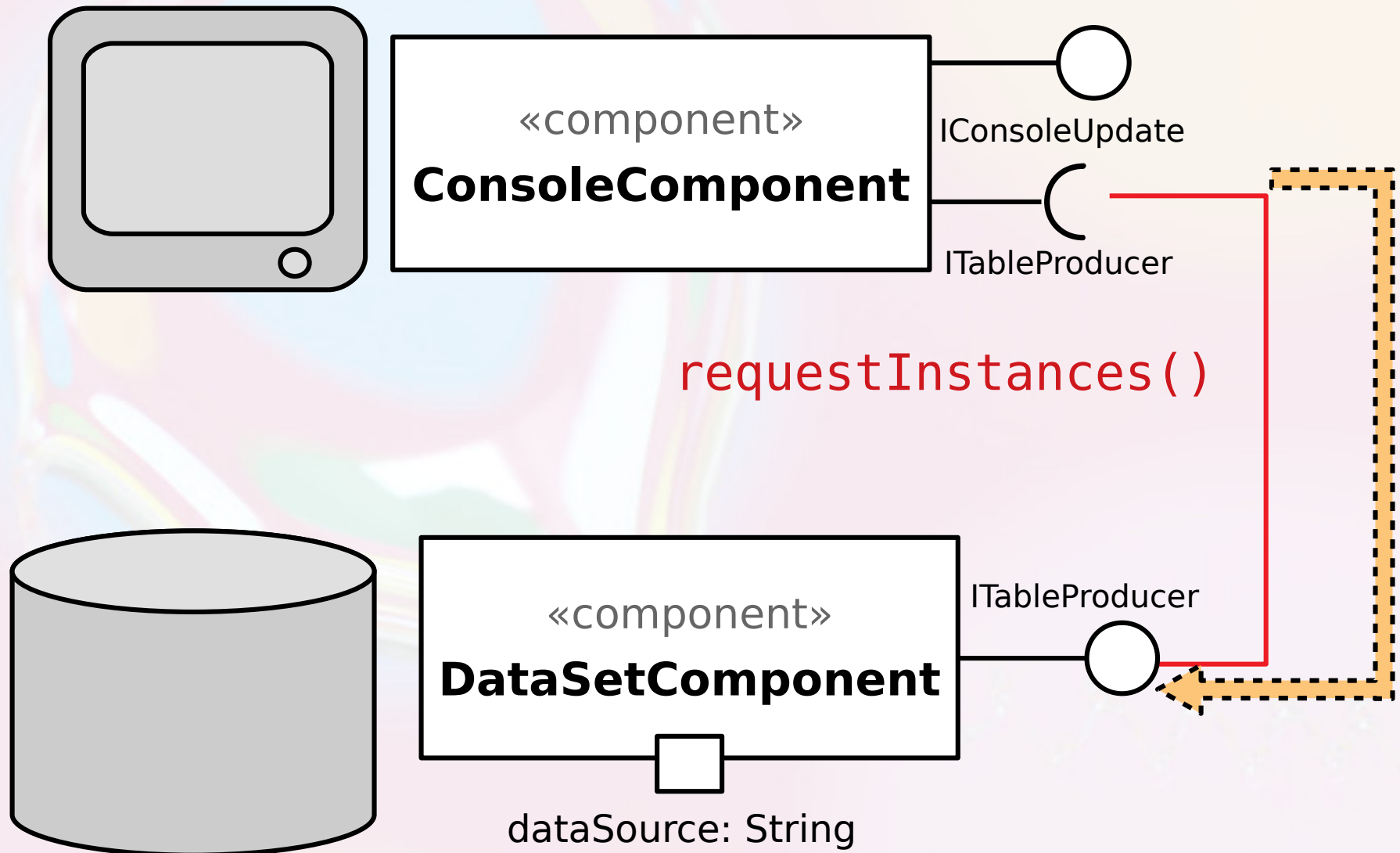
```
console.update();
```



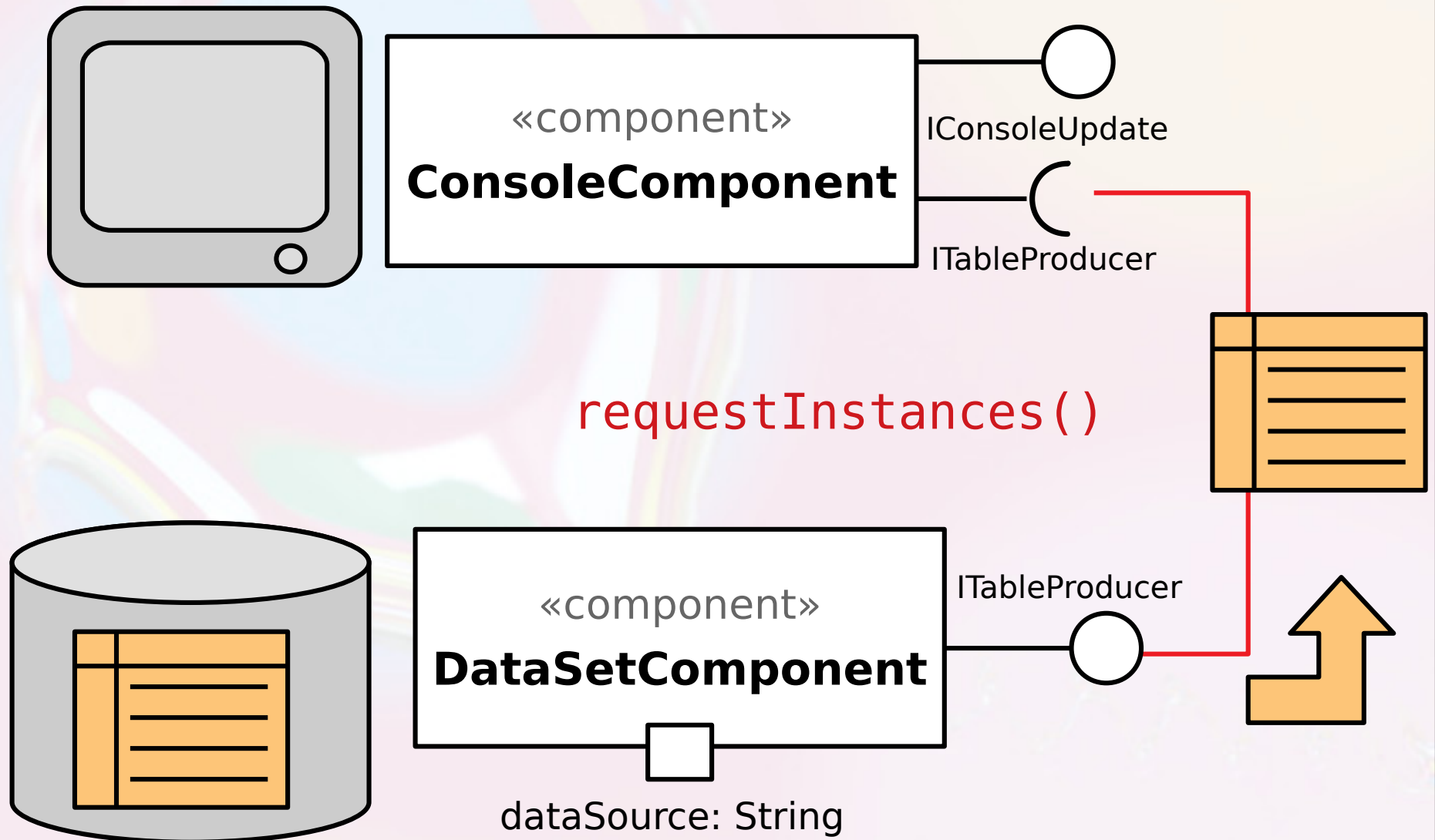
dataSource: String



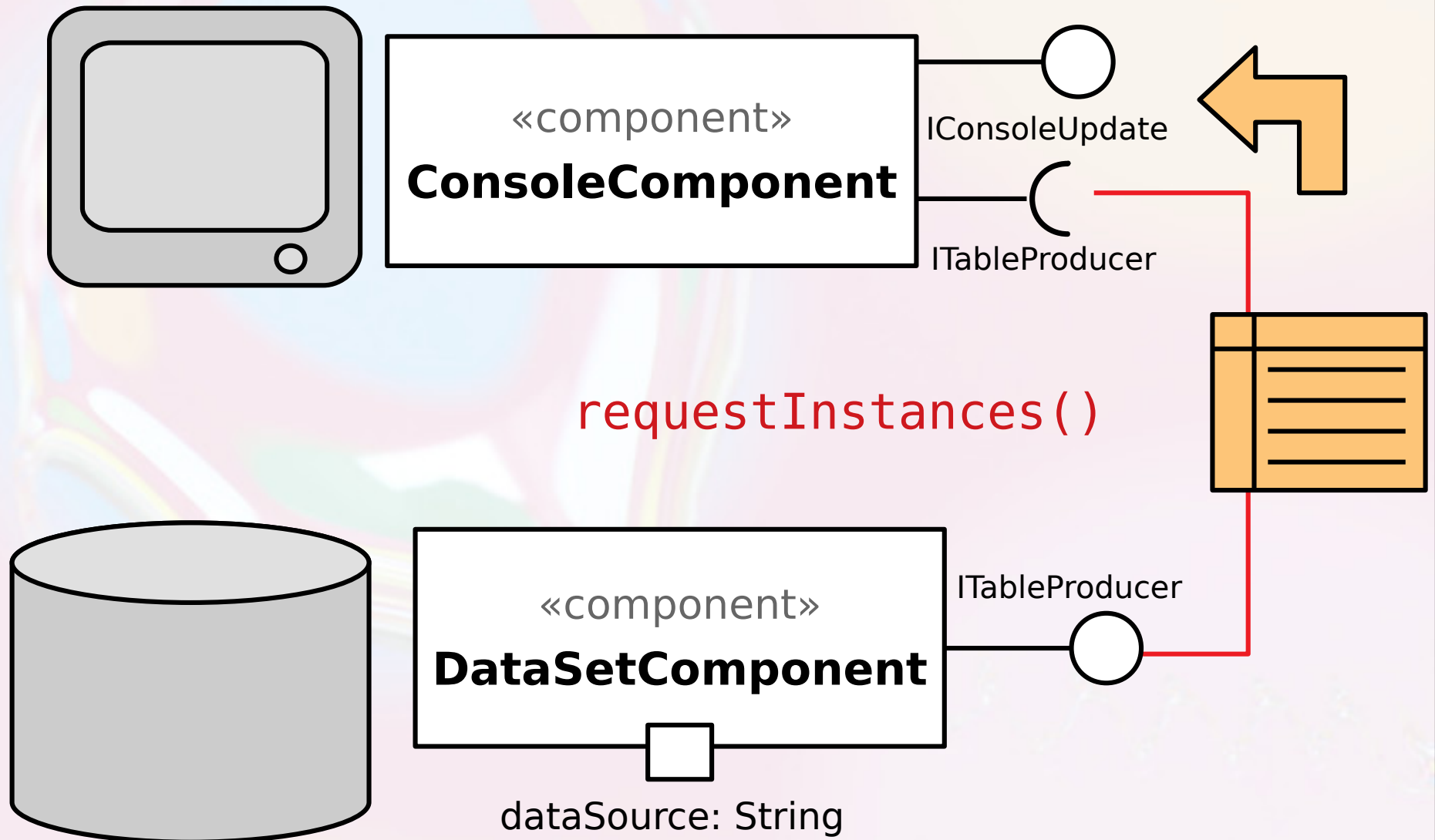
Conectando Componentes



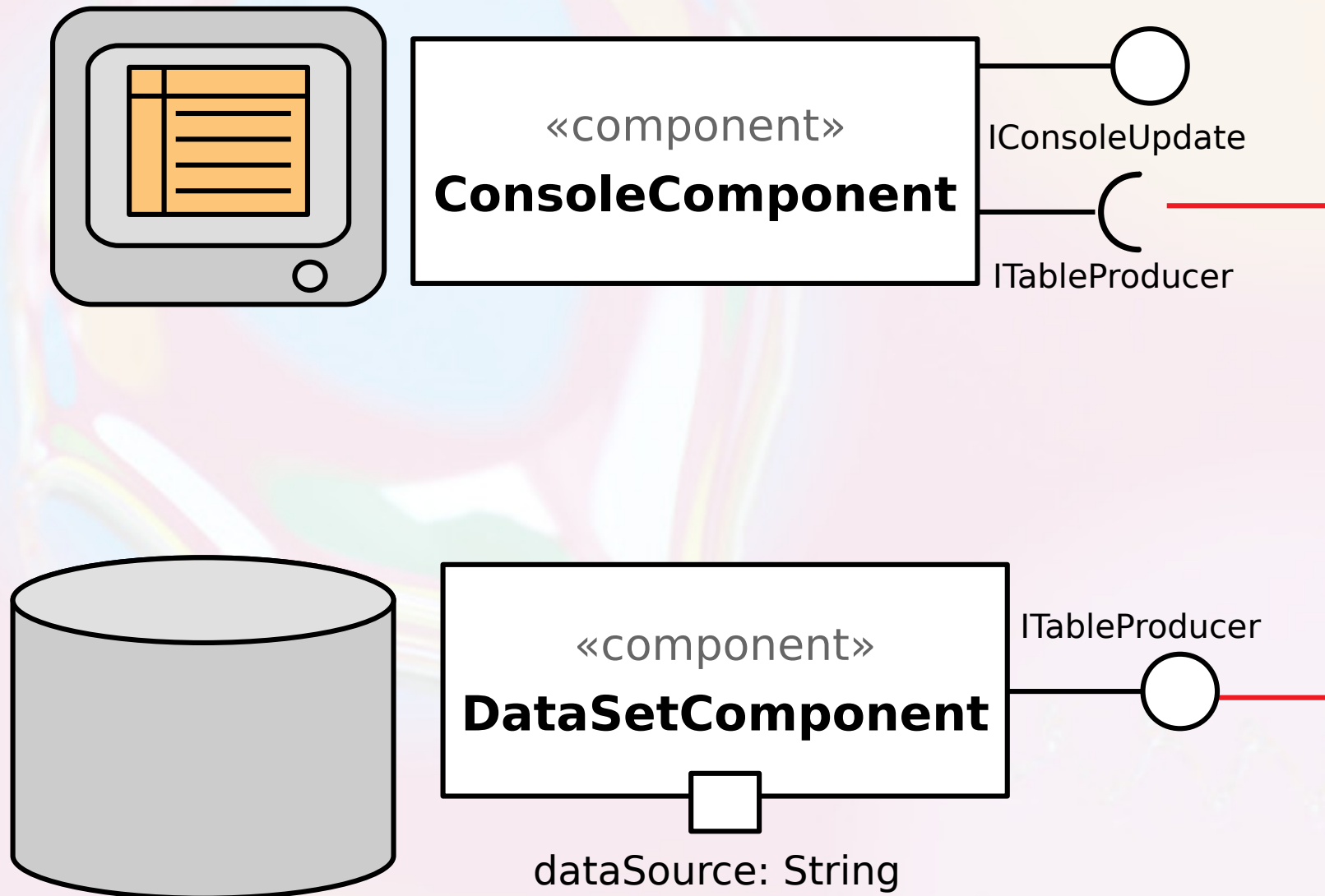
Conectando Componentes



Conectando Componentes



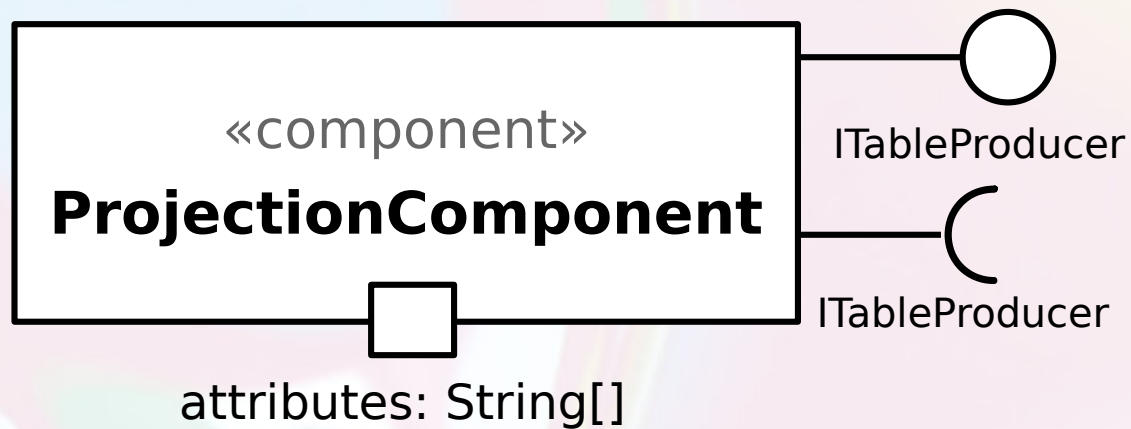
Conectando Componentes



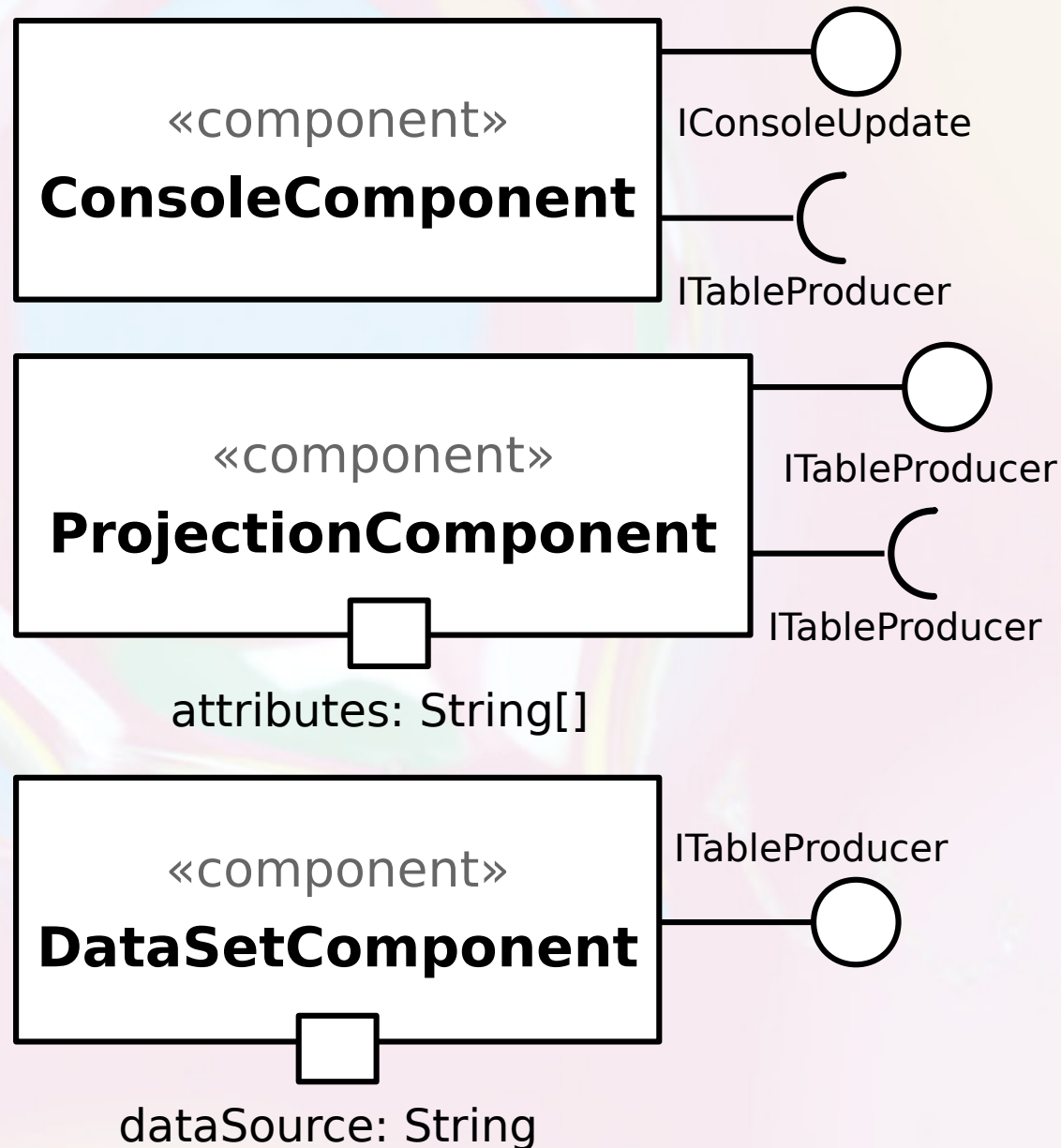
Exercício 3

- Faça um diagrama UML de como seria a interface de um componente que realize uma filtragem da coluna de “name” da tabela.

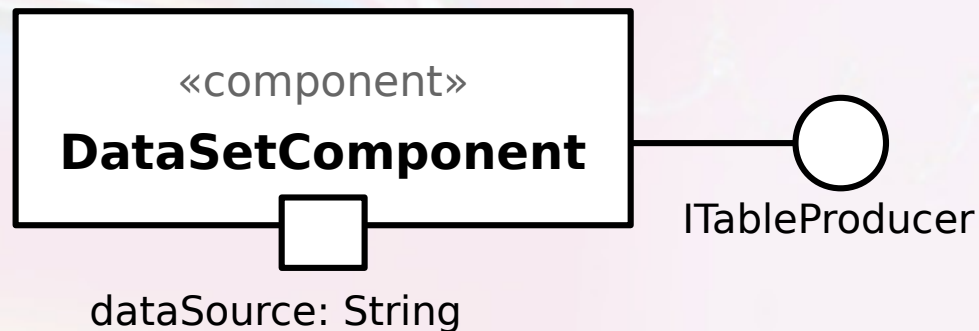
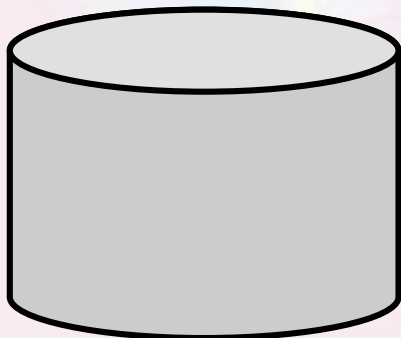
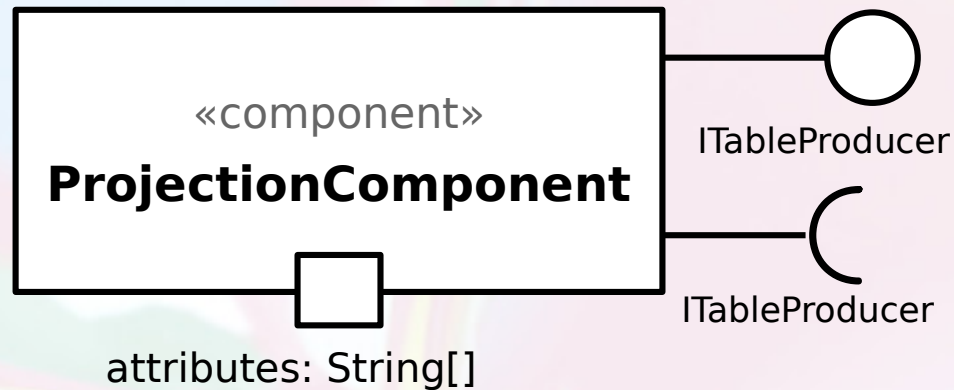
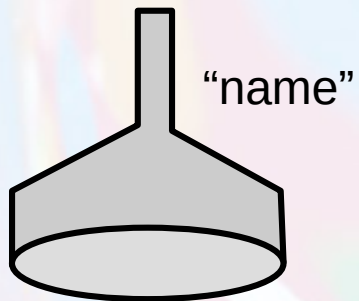
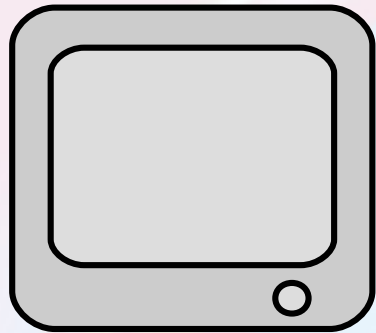
Componente Projection



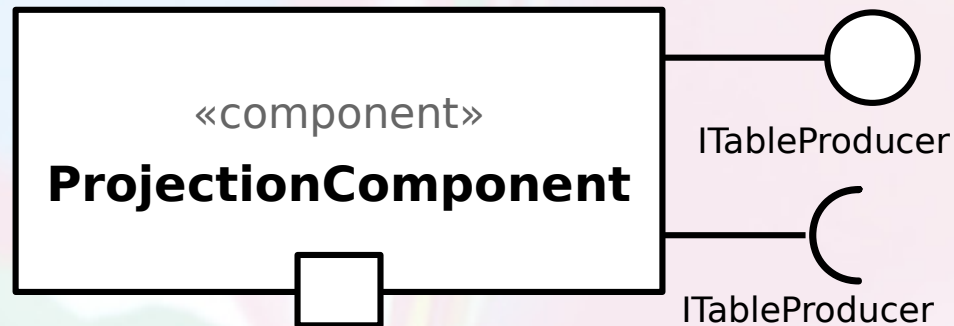
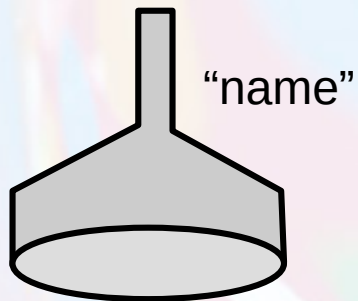
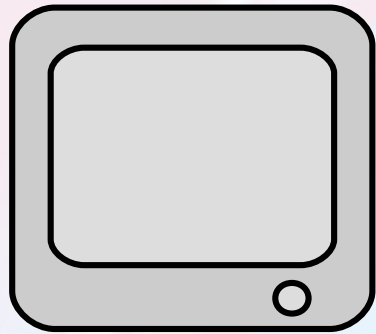
Conectando Três Componentes



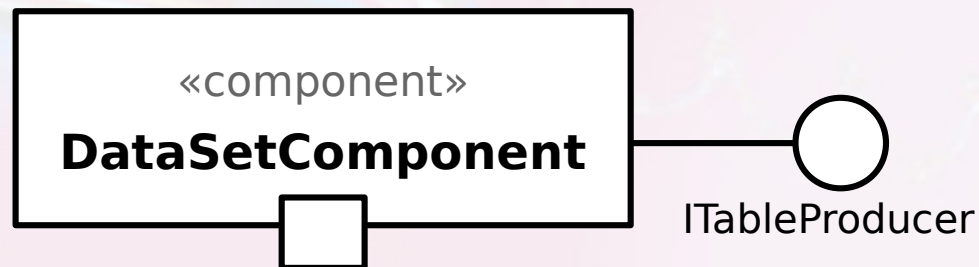
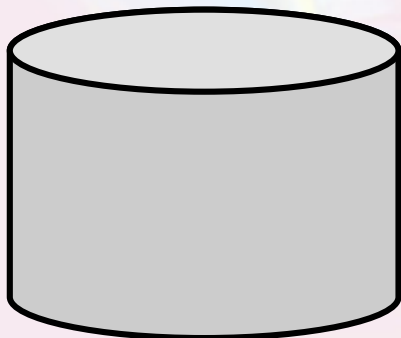
Conectando Três Componentes



Conectando Três Componentes

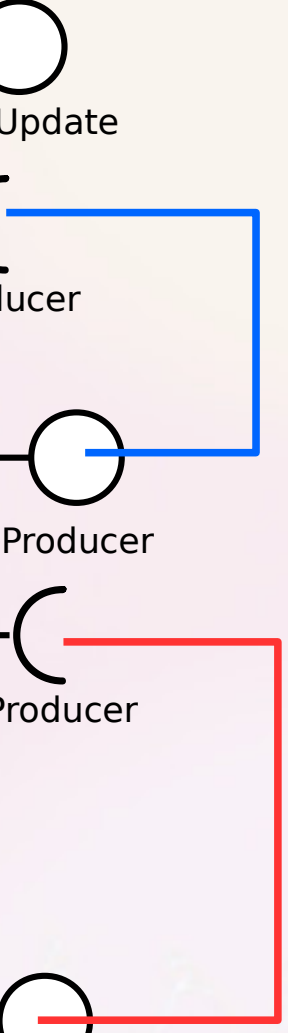
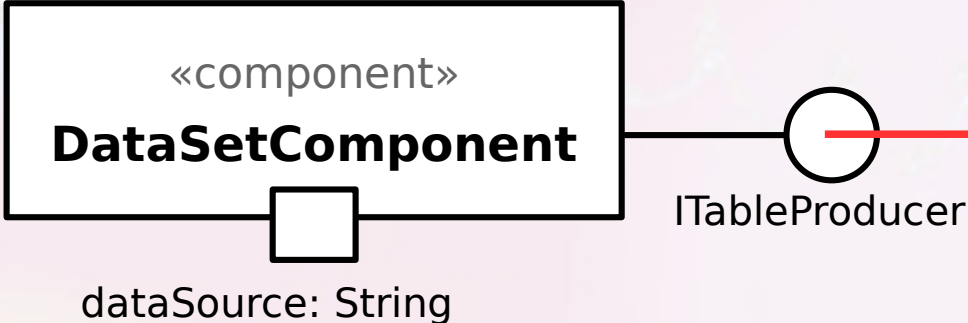
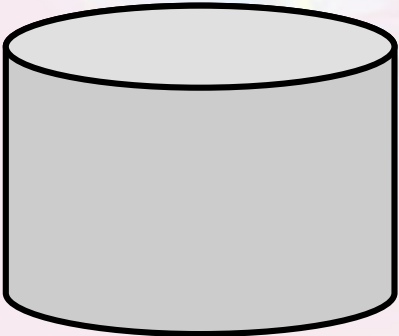
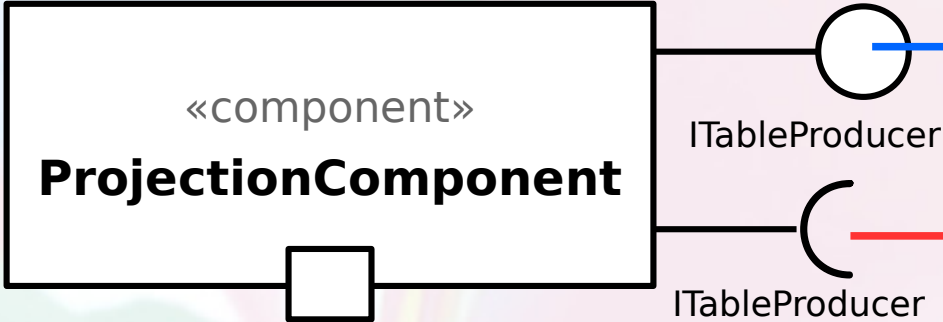
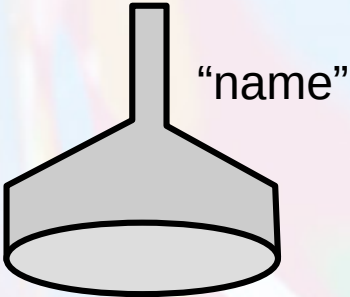
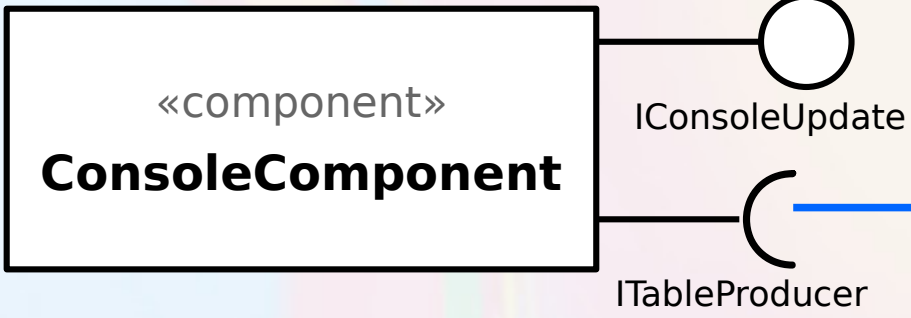
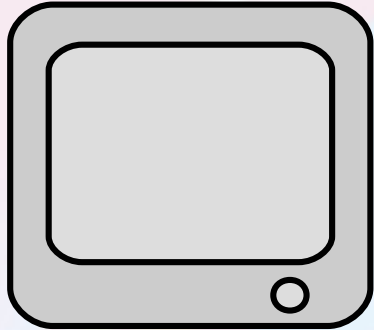


attributes: String[]
"name"

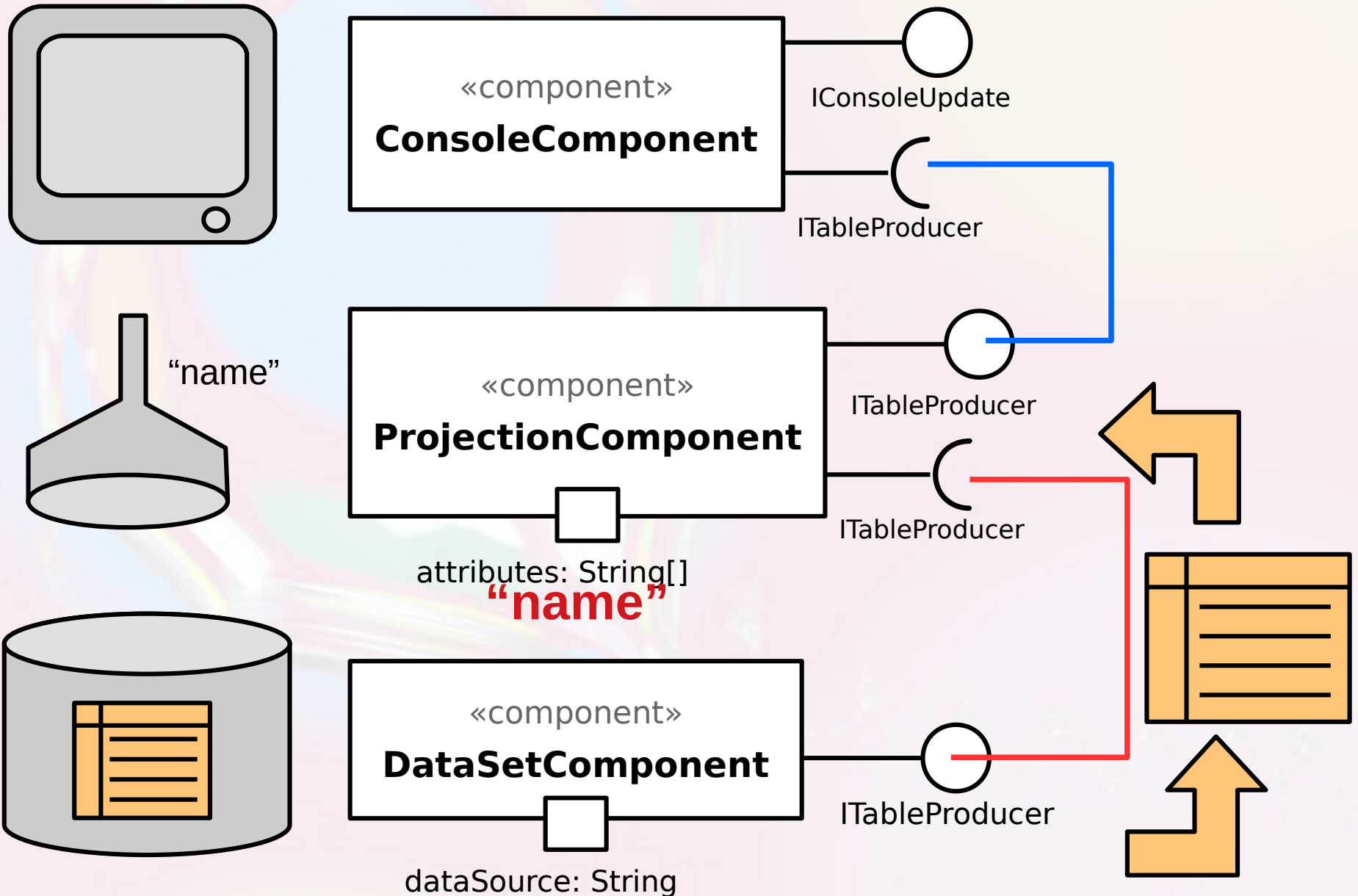


dataSource: String

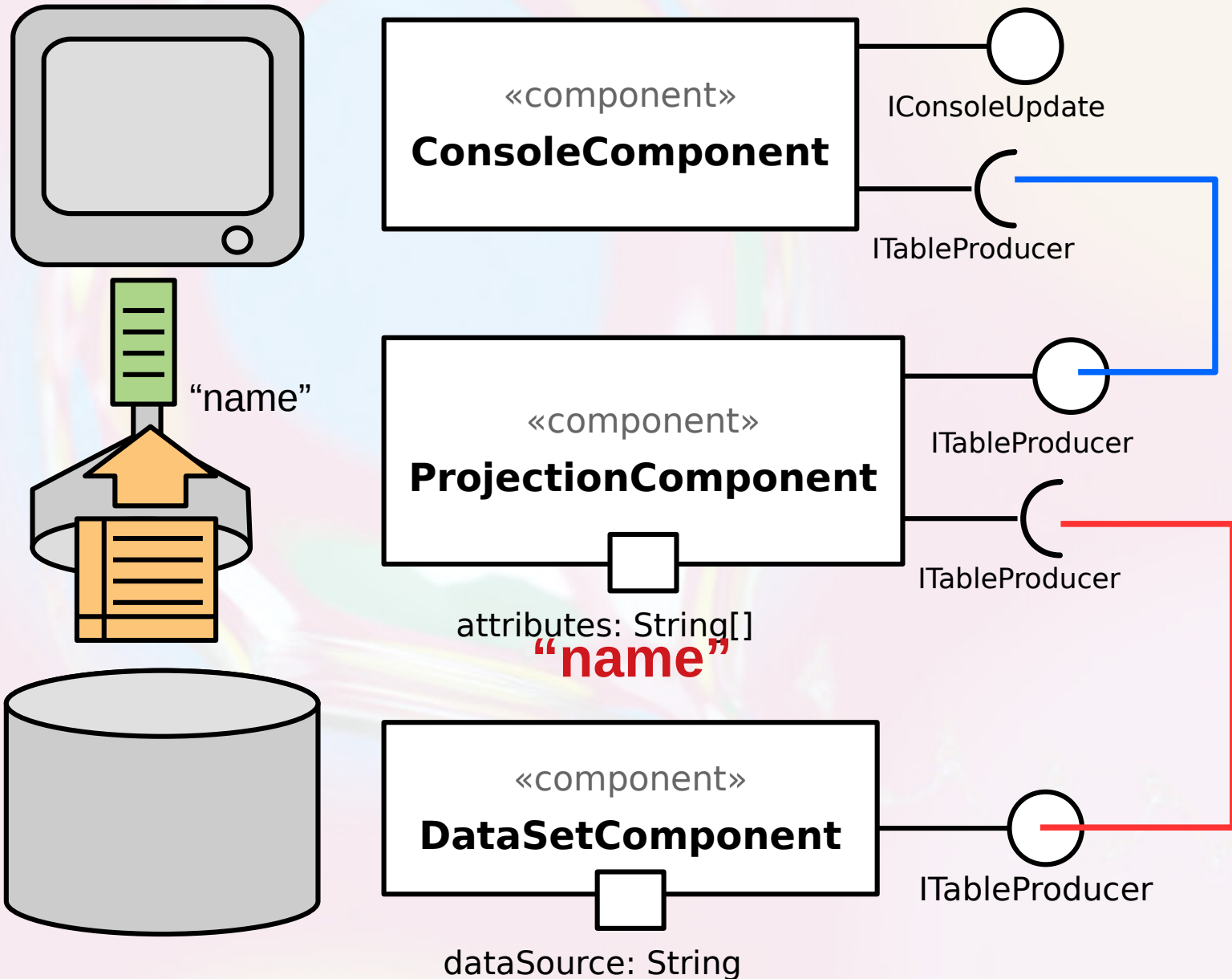
Conectando Três Componentes



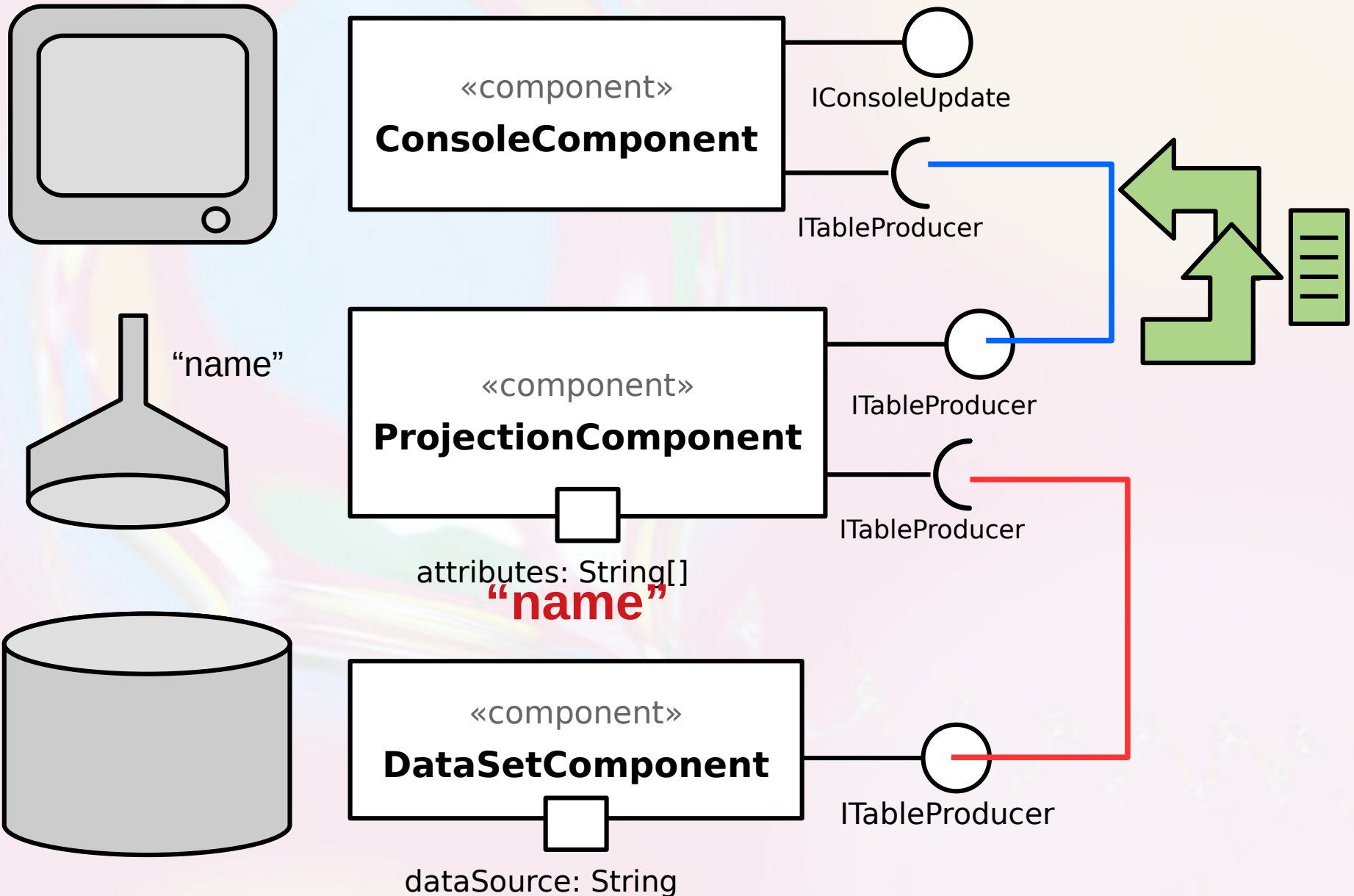
Conectando Três Componentes



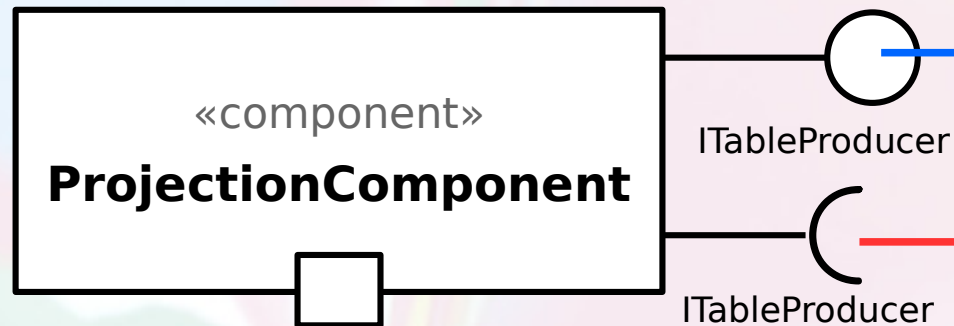
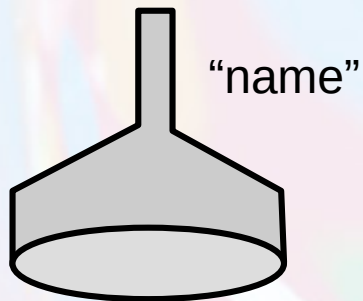
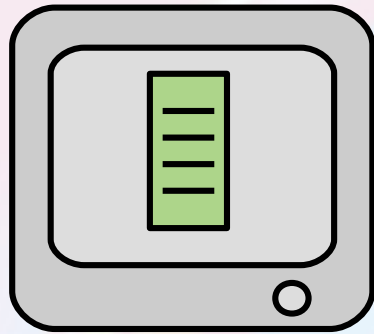
Conectando Três Componentes



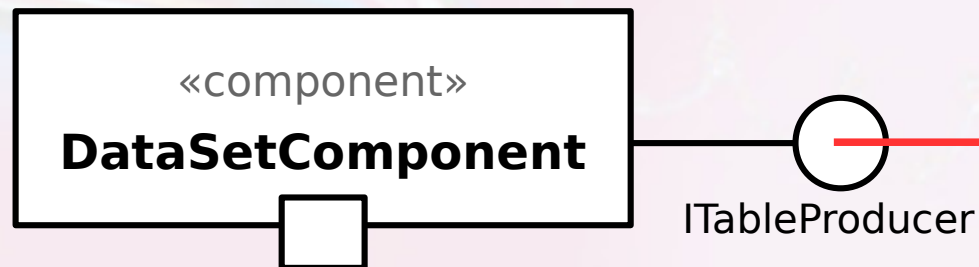
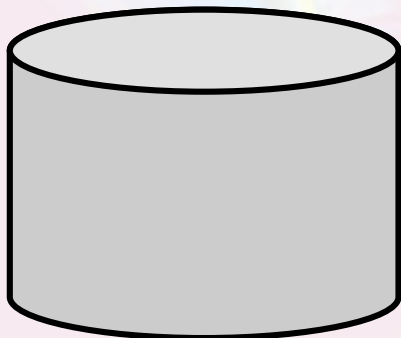
Conectando Três Componentes



Conectando Três Componentes



attributes: String[]
"name"



dataSource: String

Componente Chart

«interface»
IRun

+ start()
+ stop()

«component»

ChartBubbleComponent

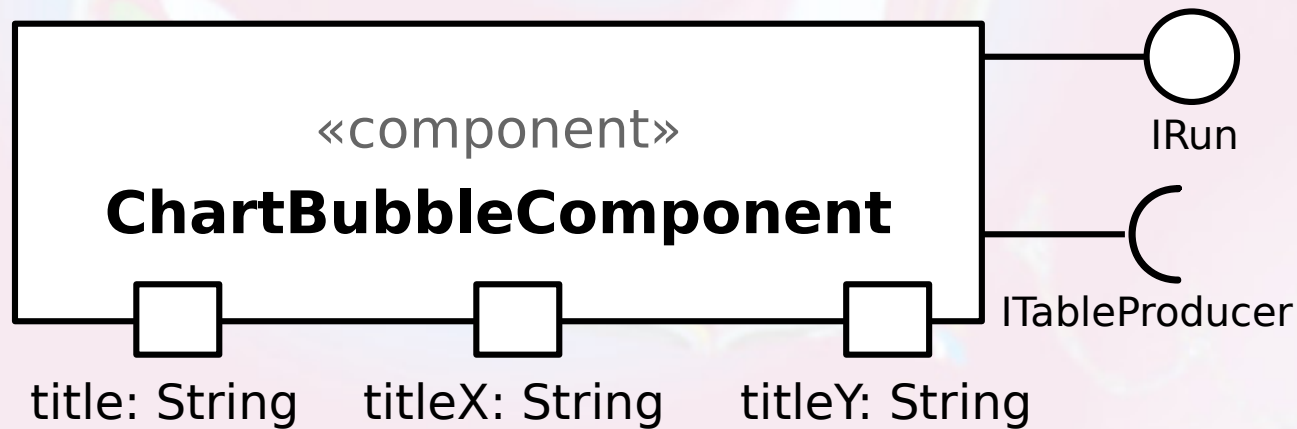
title: String

titleX: String

titleY: String

IRun

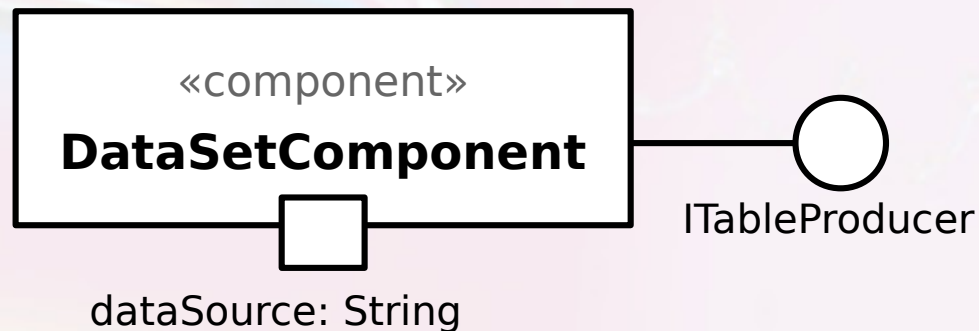
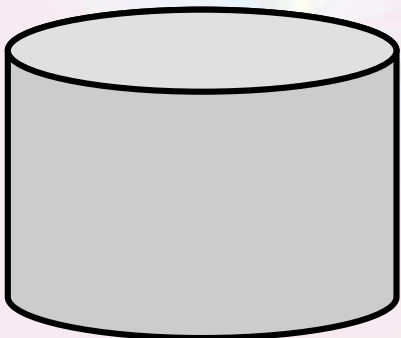
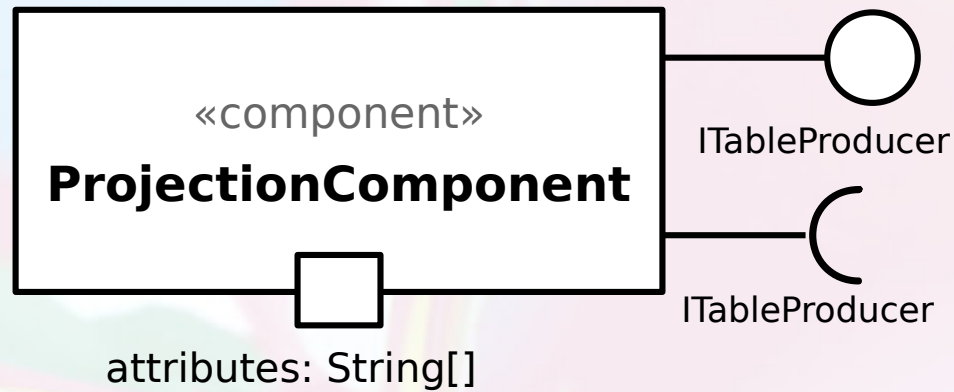
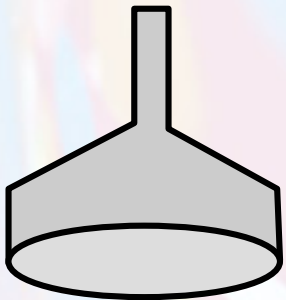
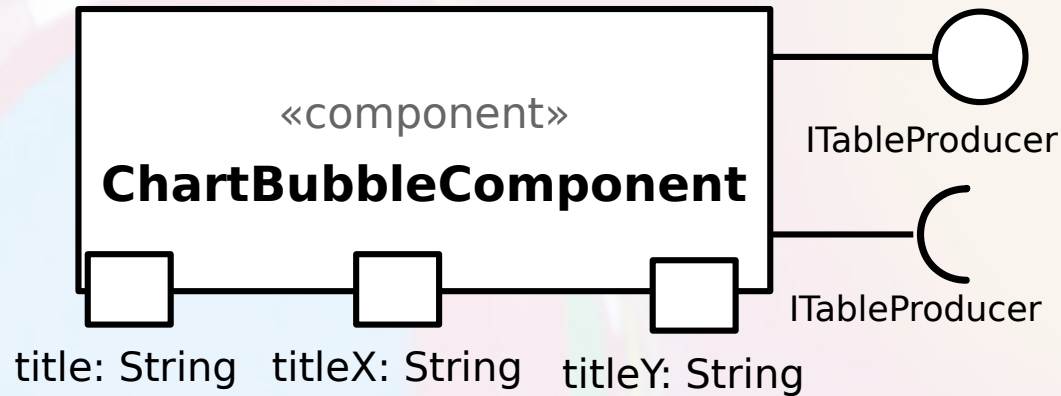
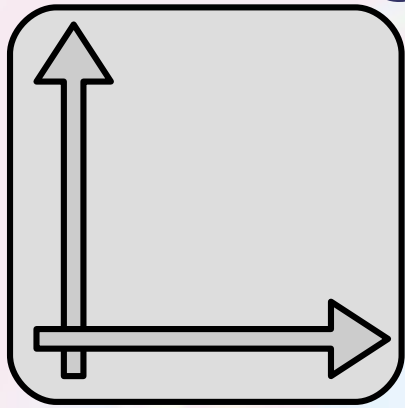
ITableProducer



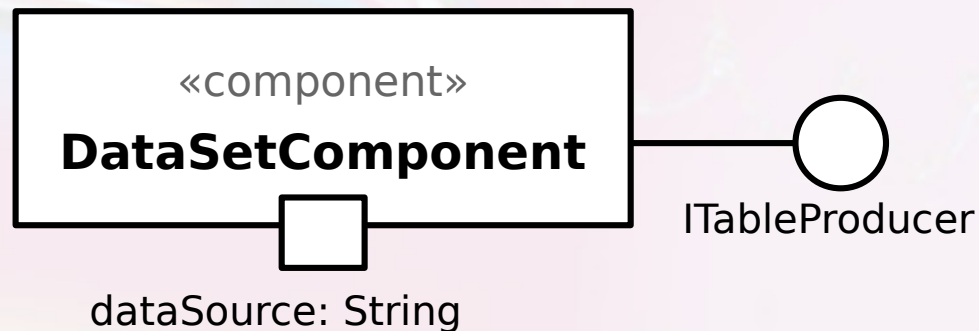
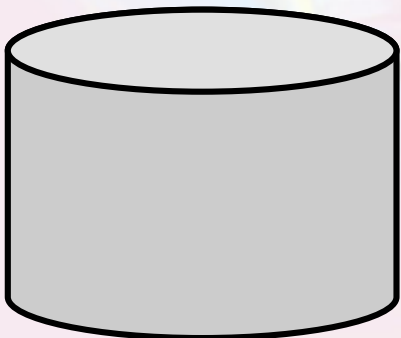
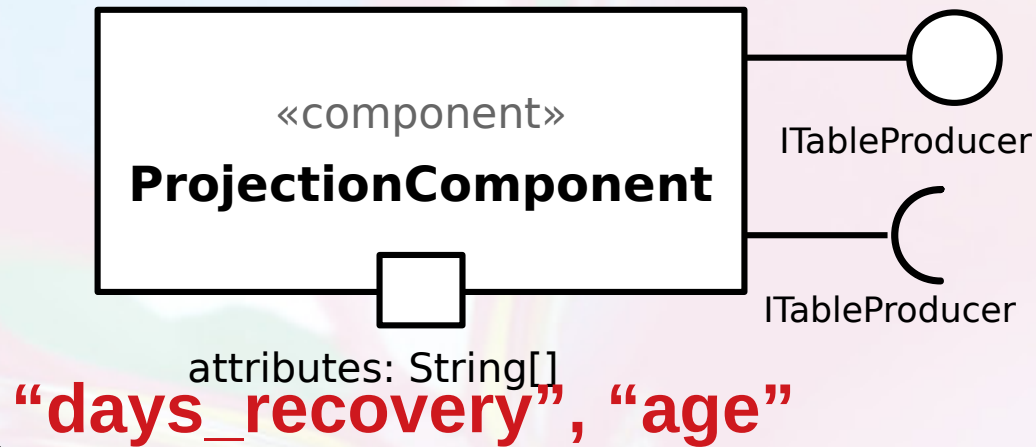
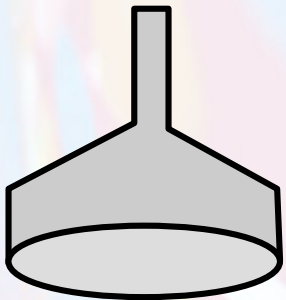
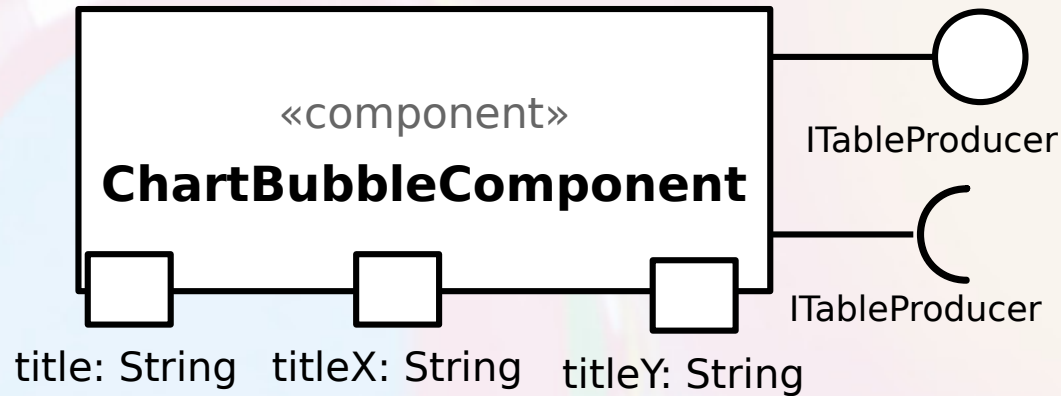
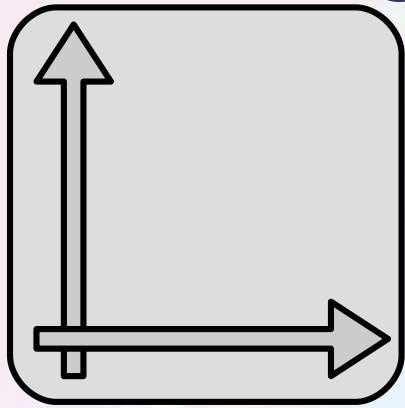
Exercício 4

- Apresente um gráfico comparativo entre idade e tempo de recuperação dos zumbis.

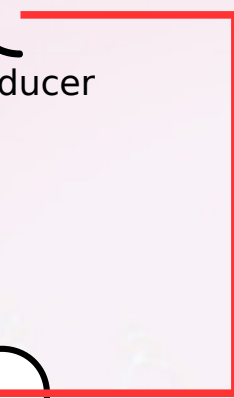
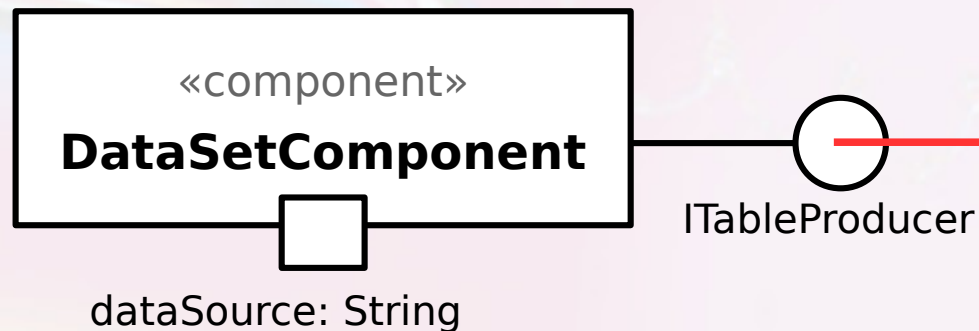
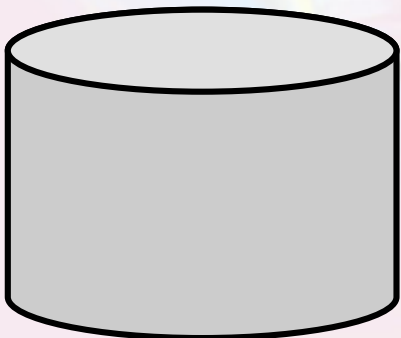
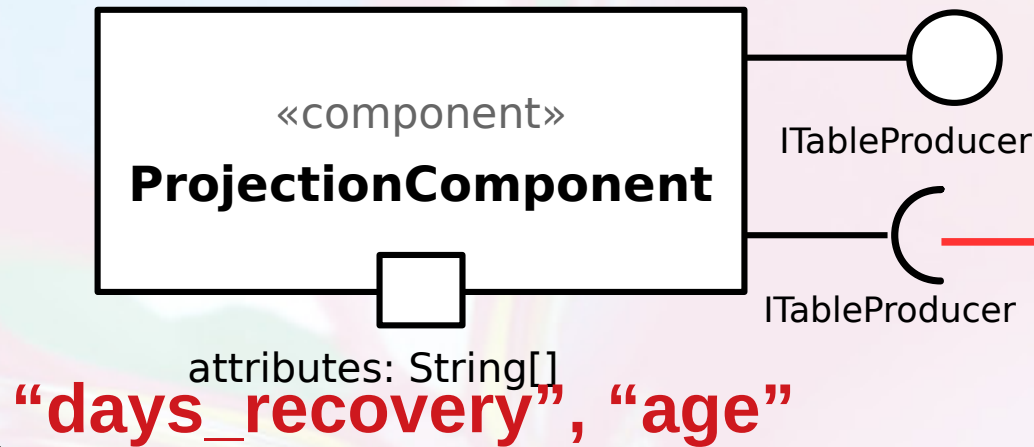
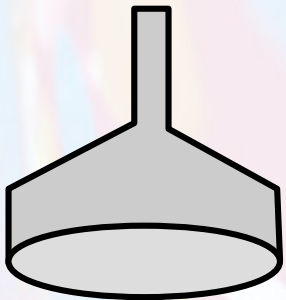
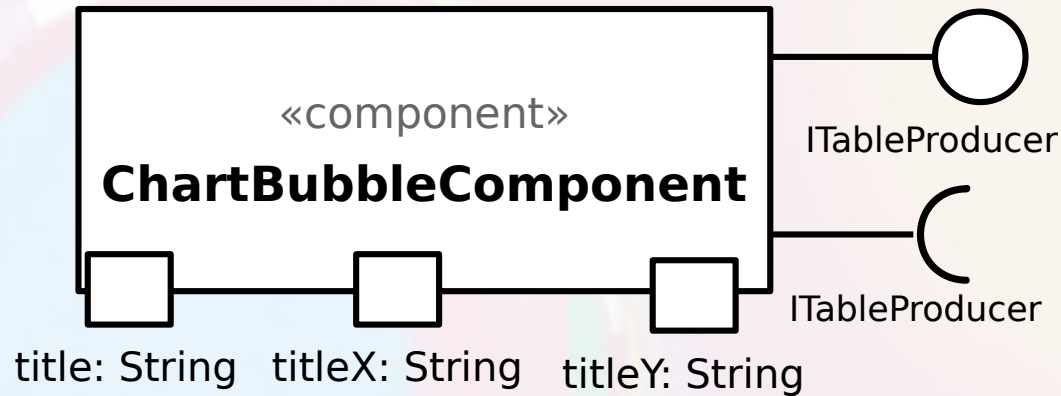
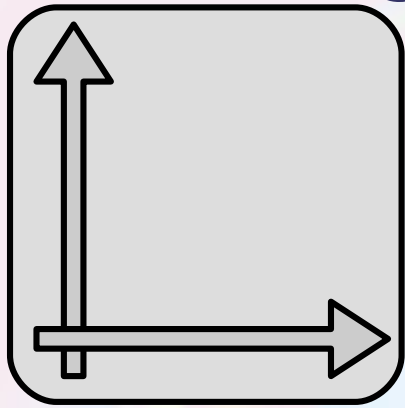
Conectando com Chart



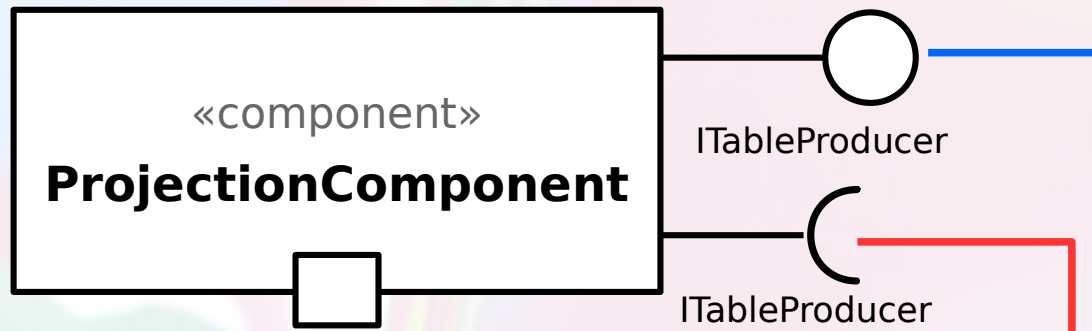
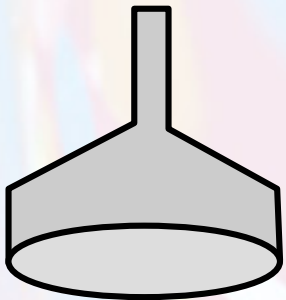
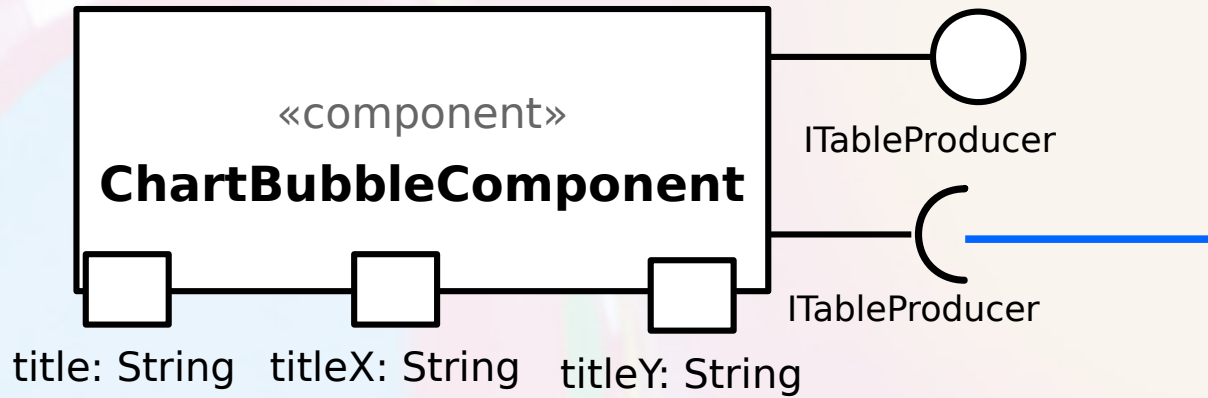
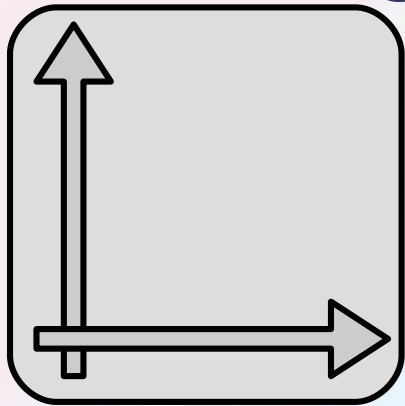
Conectando com Chart



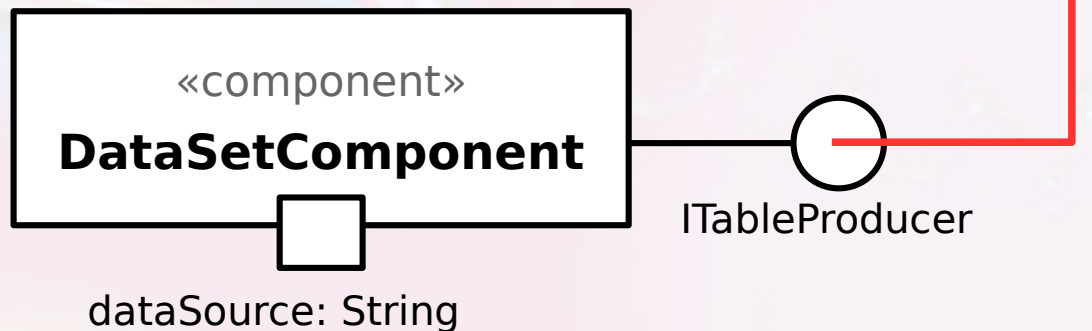
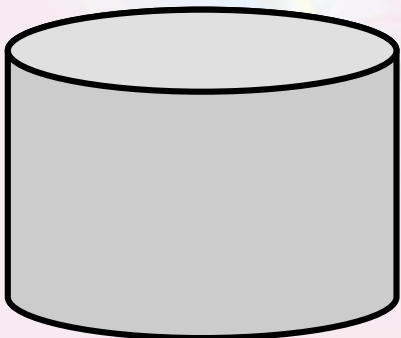
Conectando com Chart



Conectando com Chart



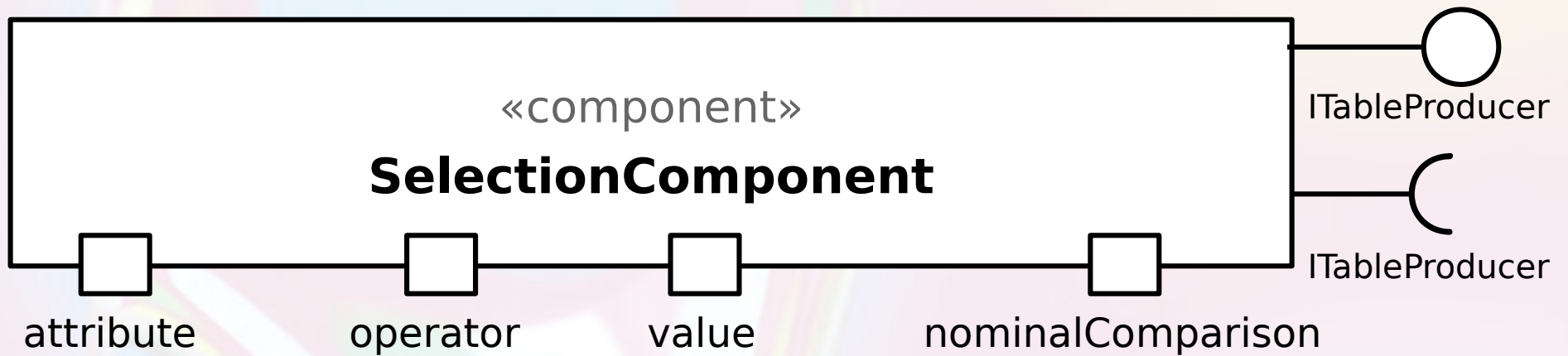
“days_recovery”, “age”



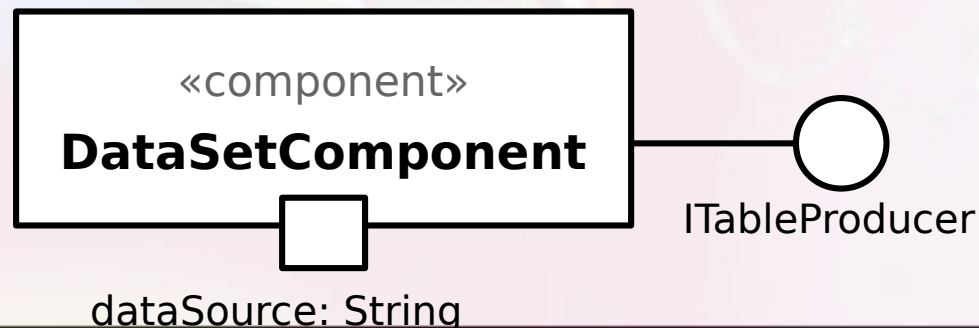
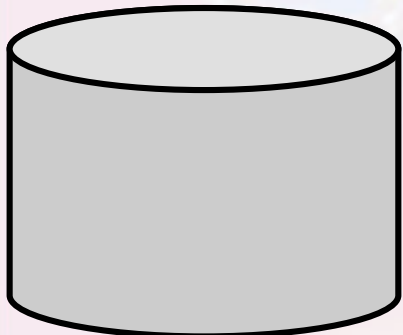
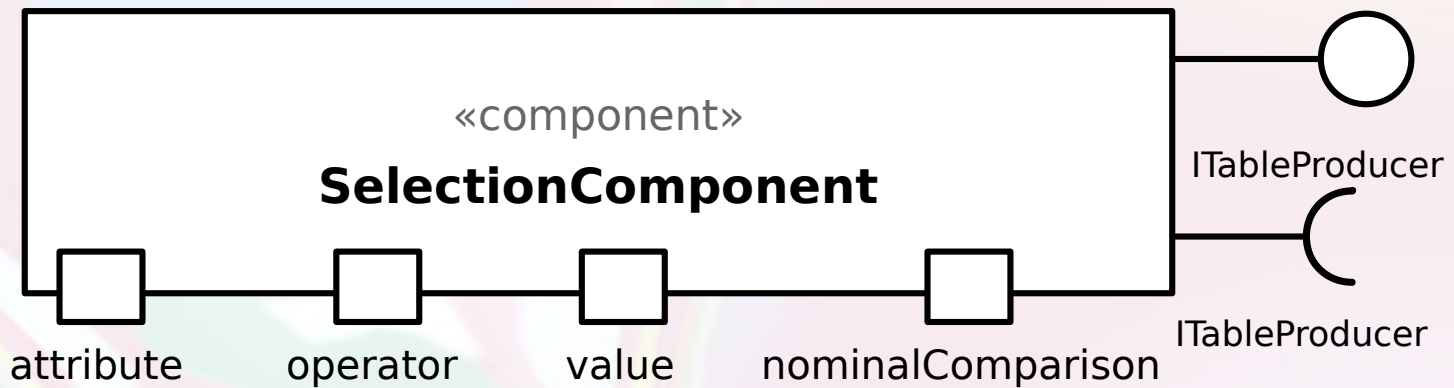
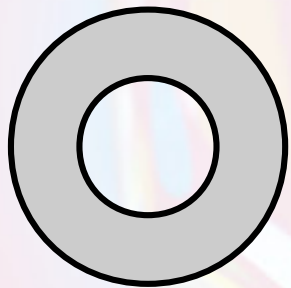
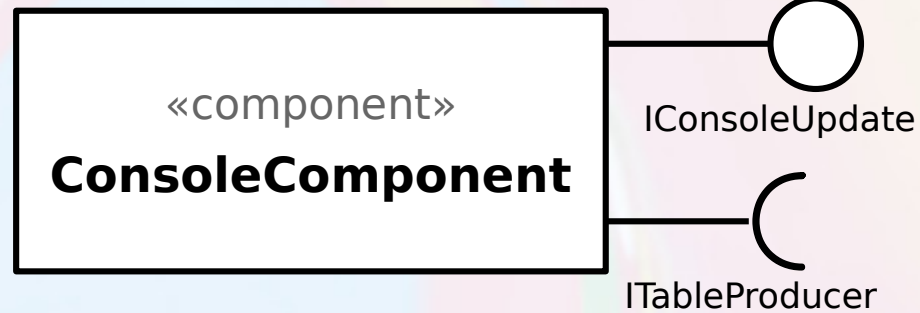
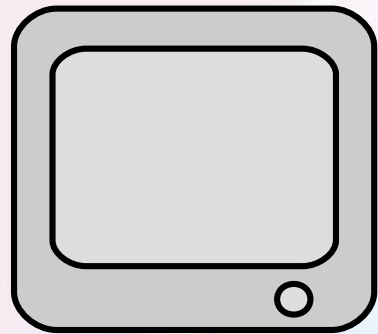
Exercício 5

- Apresente o DataSet filtrando a doença “bacterial_infection”.

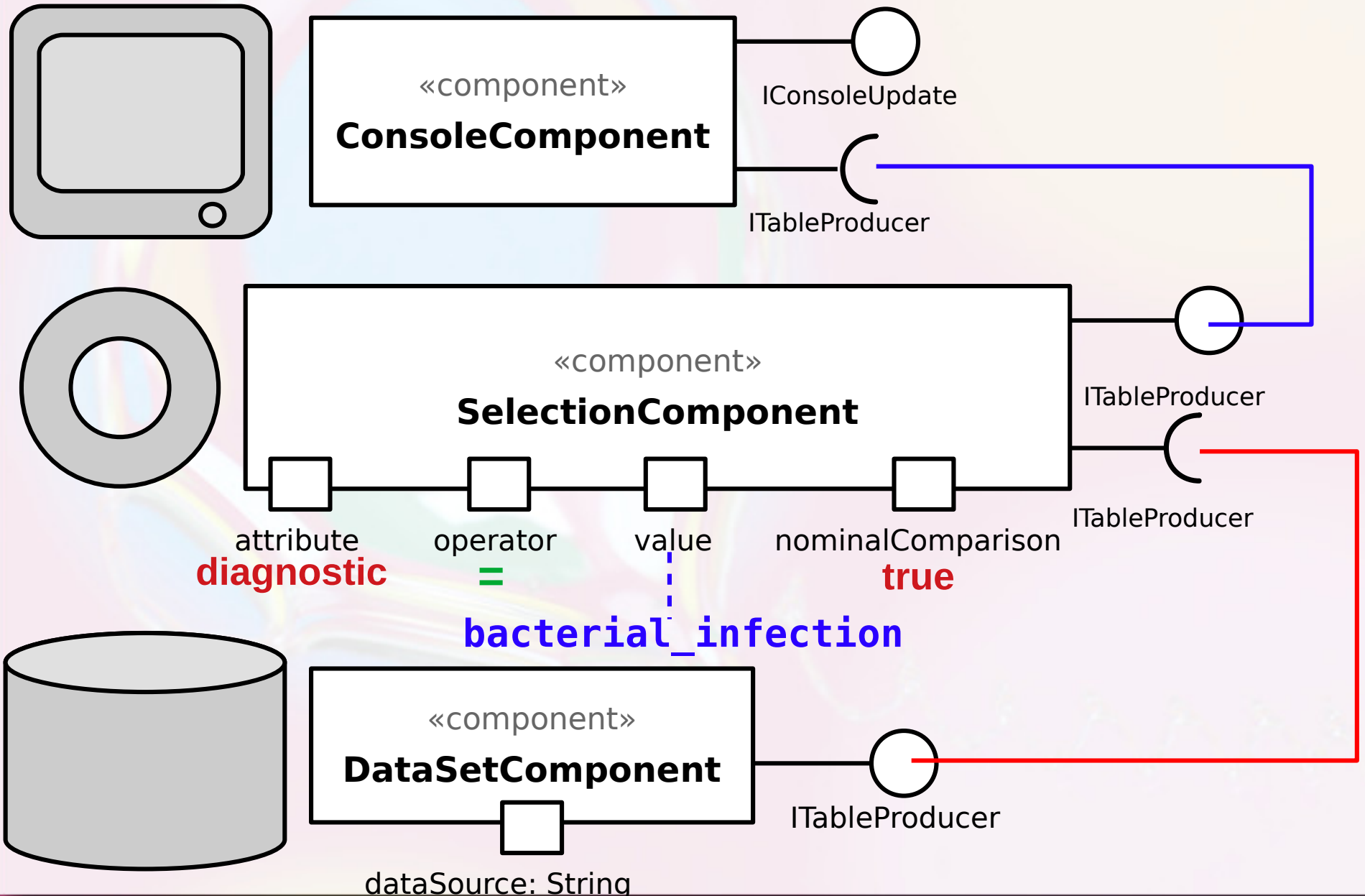
Componente Selection



Conectando com Selection

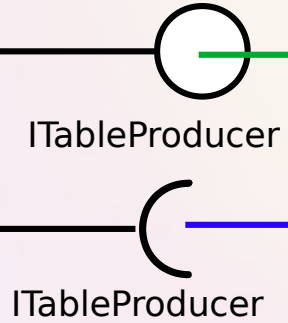
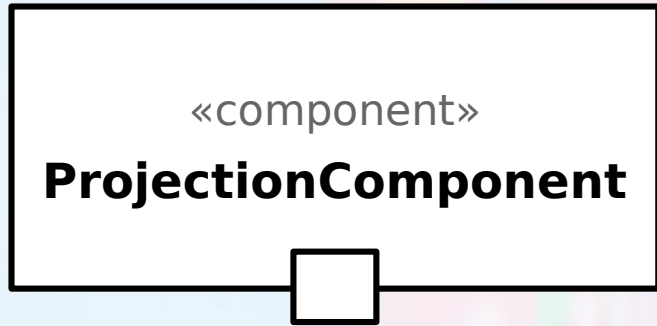
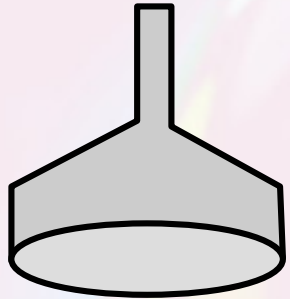
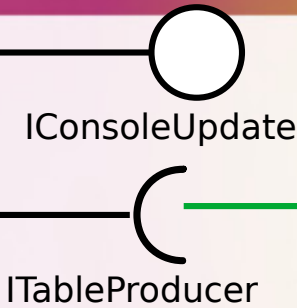
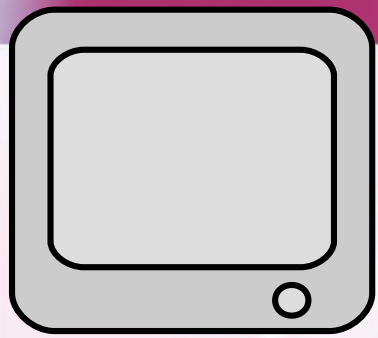


Conectando com Selection

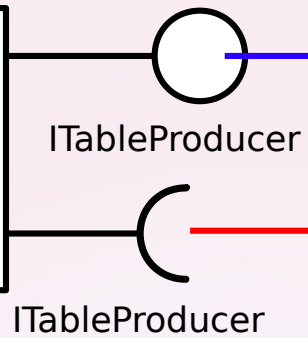
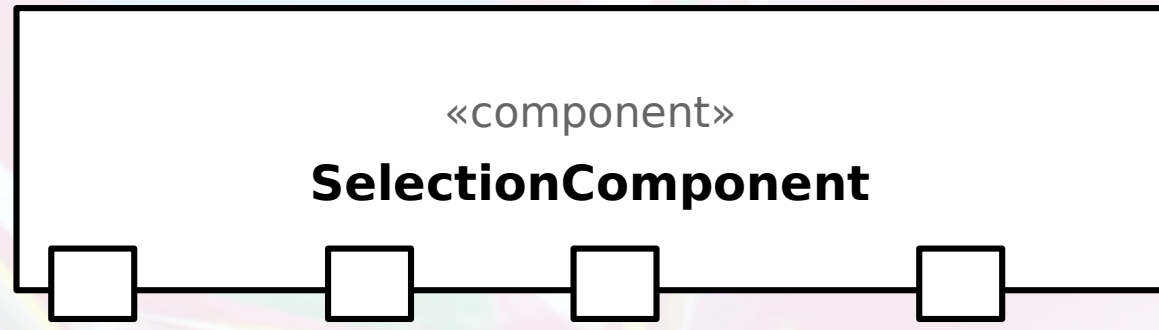
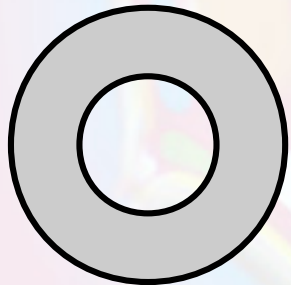


Exercício 6

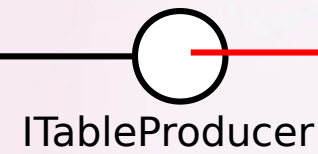
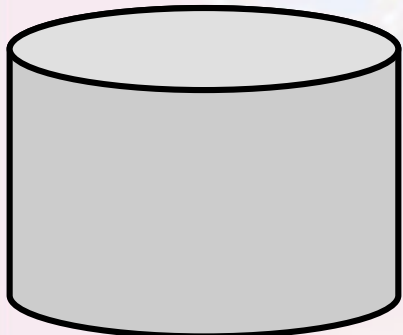
- Apresente um gráfico comparativo entre idade e tempo de recuperação, filtrando apenas a doença “bacterial_infection”.



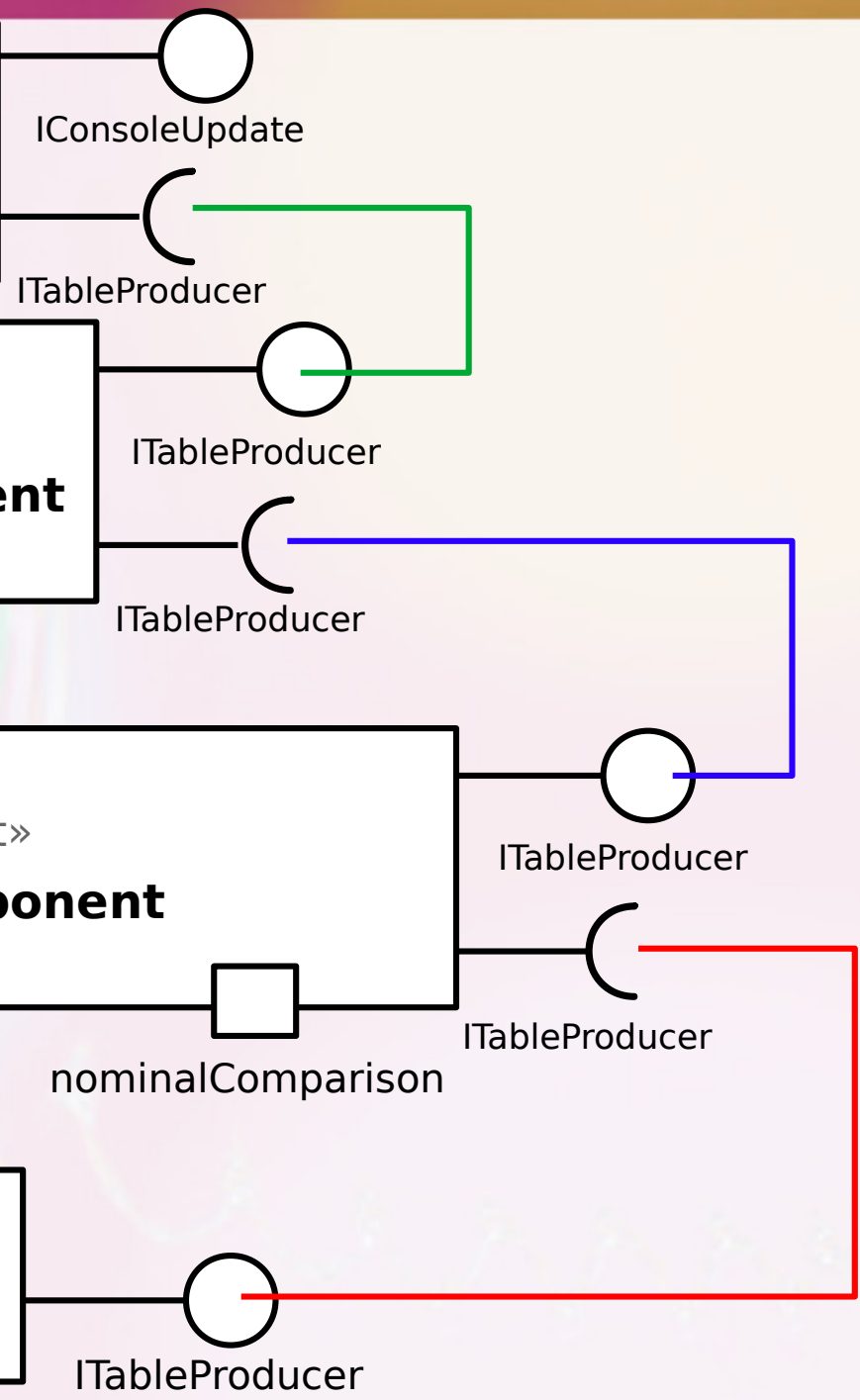
attributes: String[]

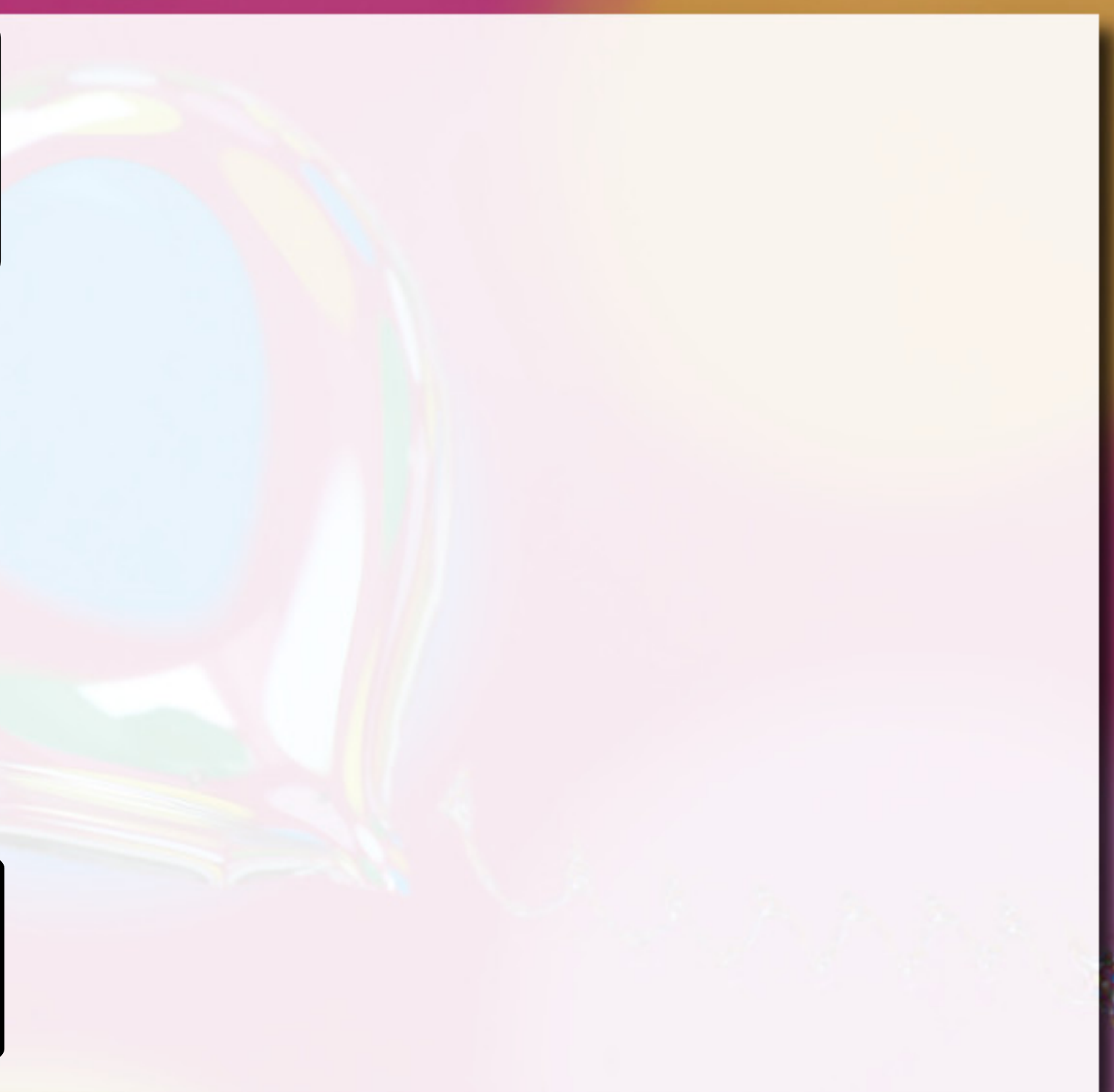
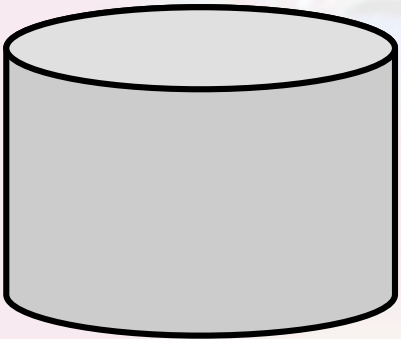
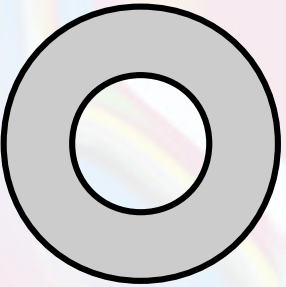
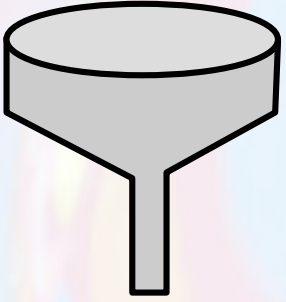
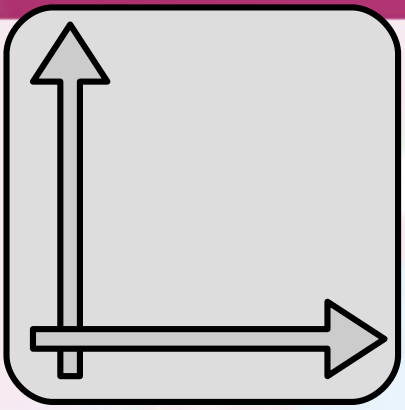


attribute operator value nominalComparison



dataSource: String





Visão Externa

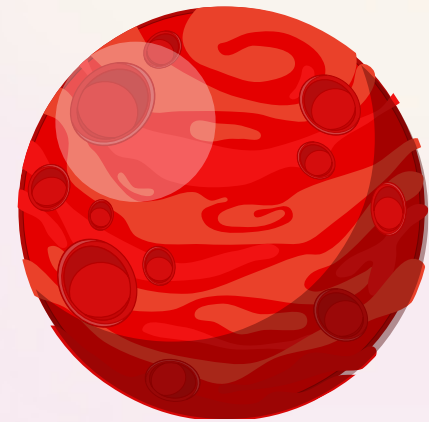
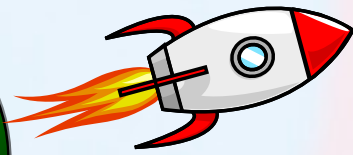
- Visão Externa (esta aula)
 - Foco: blackbox
 - Abstração das funcionalidades de um componente vendo-o externamente através de suas interfaces
 - Uso de componentes → Composição
- Visão Interna (próxima aula)
 - Foco: whitebox
 - Como um componente é implementado internamente

Exercício 2

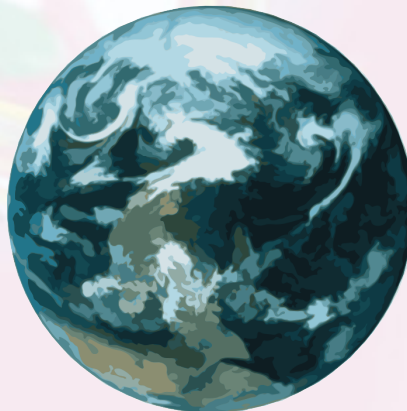
Mercante Interplanetário



Pindora



Bantor



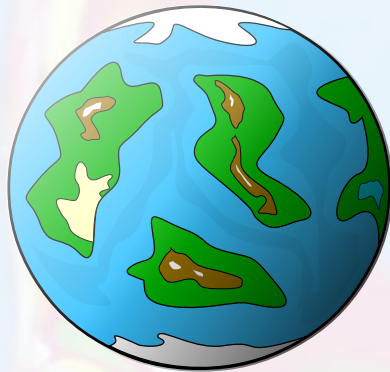
Zeta

Exercício 2

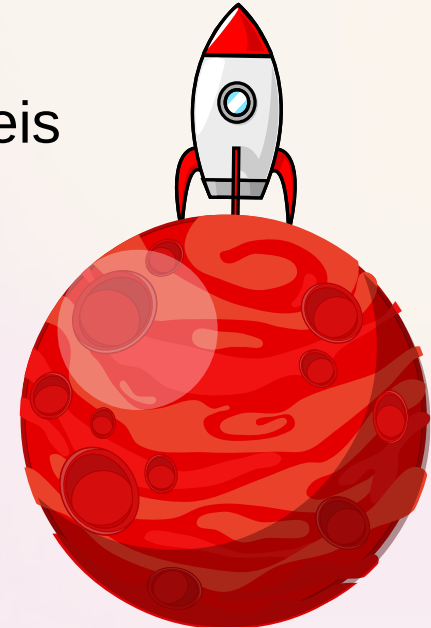
Componente Mercado Planeta

Operações:

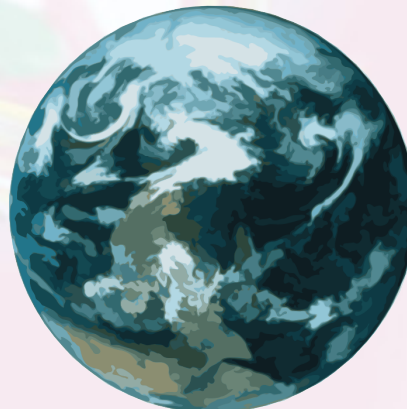
- Verificar mercadorias disponíveis
- Comprar mercadorias
- Vender mercadorias



Pindora



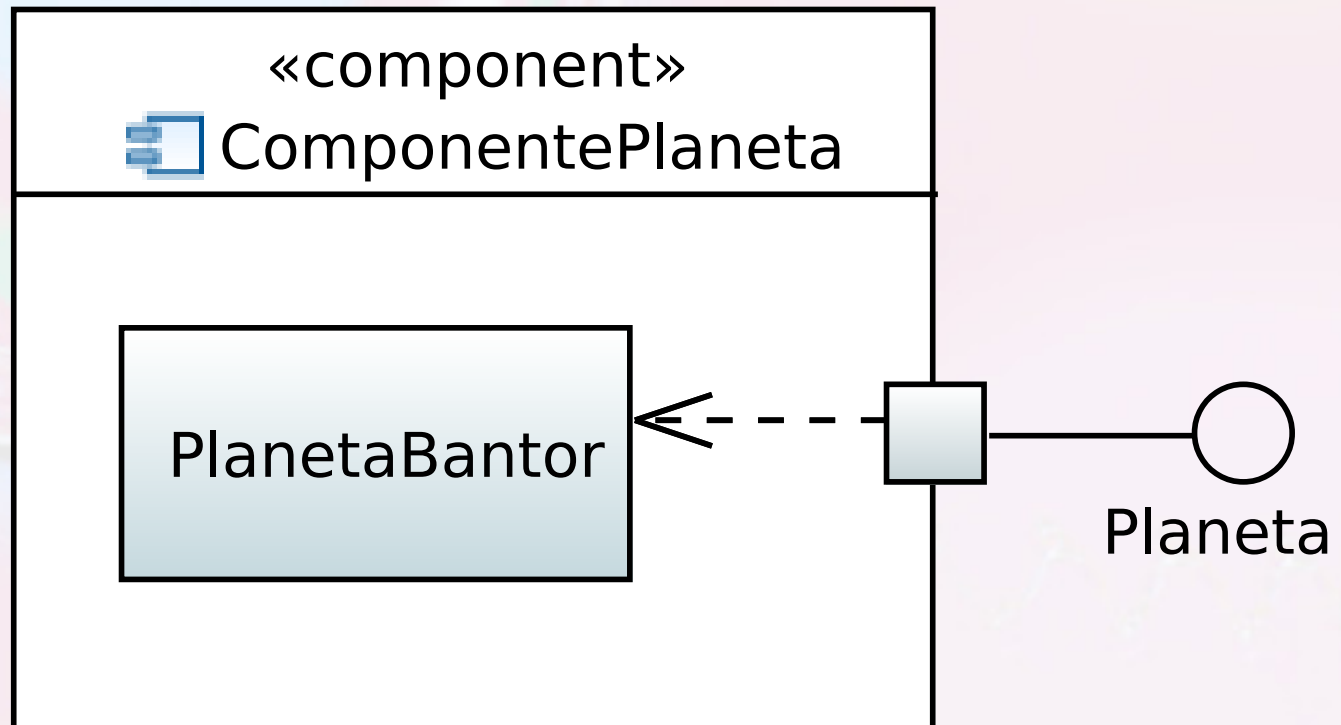
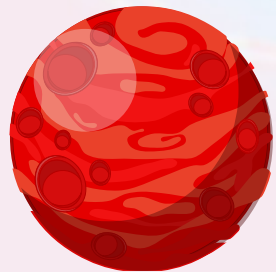
Bantor



Zeta

Exercício 2

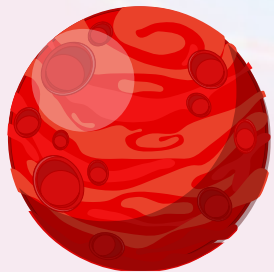
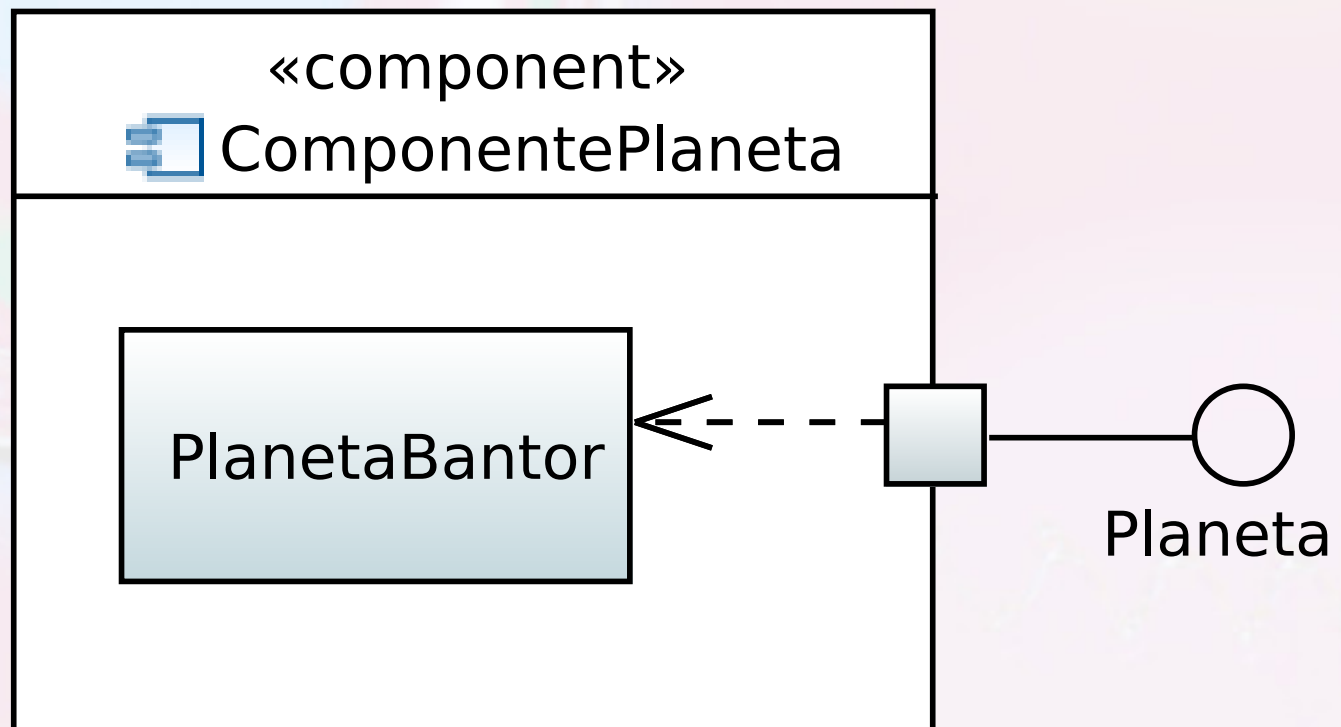
- Instancie o componente a seguir e atribua a uma variável.



Exercício 2

- Instancie o componente a seguir e atribua a uma variável:

```
Planeta p = new PlanetaBantor();
```

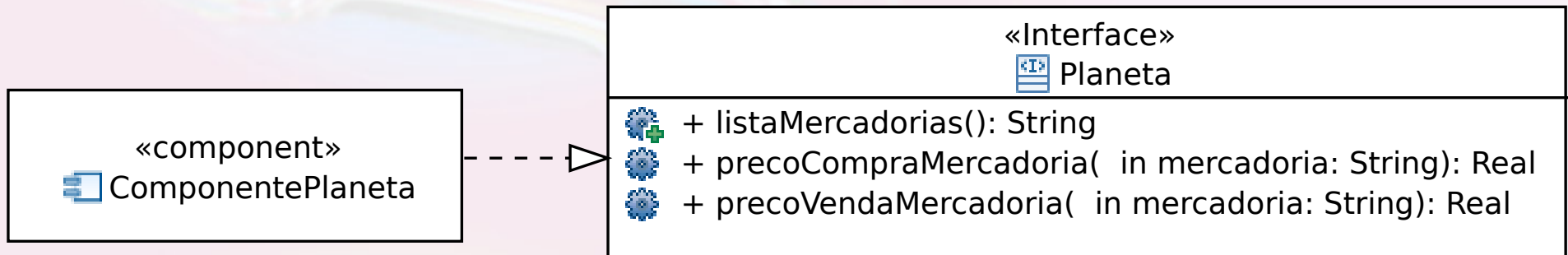
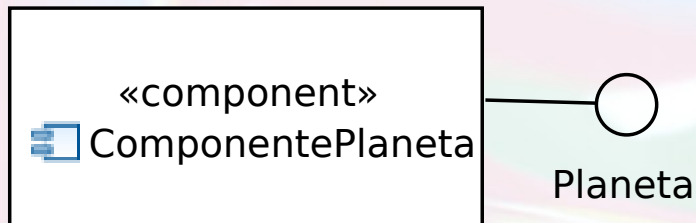


Exercício 2.5

- Escreva duas linhas de código para listar todas as mercadorias e verificar o preço de compra da mercadoria “Pistola Xist Paralisante”.

Operações:

- Verificar mercadorias disponíveis
- Verificar preço de compra de uma mercadoria
- Verificar preço de venda de uma mercadoria

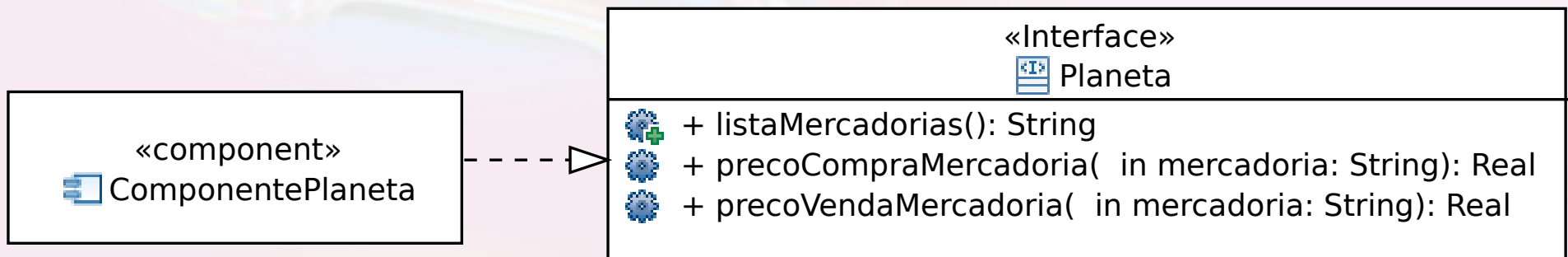
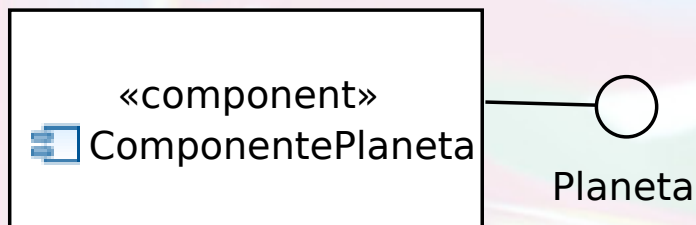


Exercício 2.5

- Escreva duas linhas de código para listar todas as mercadorias e verificar o preço de compra da mercadoria “Pistola Xist Paralisante”.
- ```
System.out.println(p.listaMercadorias());
System.out.println(p.precoCompraMercadoria(“Pistola Xist ...”));
```

## Operações:

- Verificar mercadorias disponíveis
- Verificar preço de compra de uma mercadoria
- Verificar preço de venda de uma mercadoria



# Referências

- Caires, Luis. **Fundamentos e Tecnologias de Componentes (slides)**. Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, 2002.
- Cheesman, J., & Daniels, J. (2000). **UML Components: A simple process for specifying component-based software**. Addison-Wesley.
- Cook, S., Bock, C., Rivett, P., Rutt, T., Seidewitz, E., Selic, B., & Tolbert, D. (2015). **OMG Unified Modeling Language (OMG UML) - version 2.5**. Needham. Retrieved from <http://www.omg.org/spec/UML/2.5/>
- Szyperski, C. **Component Software: Beyond Object-Oriented Programming**. Addison-Wesley Longman Publishing Co., Inc., 2002.

# Referências

- Comella-Dorda, S. **Component Object Model (COM), DCOM, and Related Capabilities**. Carnegie Mellon University, março de 2001.
- Cook, S., Bock, C., Rivett, P., Rutt, T., Seidewitz, E., Selic, B., & Tolbert, D. (2015). **OMG Unified Modeling Language (OMG UML) - version 2.5**. Needham. Retrieved from <http://www.omg.org/spec/UML/2.5/>
- Gamma, E. Helm, R. Johnson, R. Vlissides, J. **Design Patterns: Elements of Reusable Object-Oriented Software**. Addison-Wesley, 1995.
- Martin, R. C. **Design Principles and Design Patterns**. Object Mentor, 2000.
- Parrish, R. **XPCOM Part 1: An introduction to XPCOM**. DeveloperWorks, fevereiro de 2001, on-line:  
<http://www.ibm.com/developerworks/webservices/library/co-xpcom.html>
- Williams, S. & Kindel, C. **The Component Object Model: A Technical Overview**. Microsoft Corporation, 1994

**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 Doug Wheller [<http://www.flickr.com/photos/doug88888/>] por sua fotografia “Water drop” usada na capa e nos fundos, disponível em [<http://www.flickr.com/photos/doug88888/7032440831/>] vide licença específica da fotografia.