# Programação Orientada a Objetos

## Stream de Dados + Pipe & Filter

André Santanchè
Laboratory of Information Systems – LIS
Instituto de Computação – UNICAMP
Maio 2015

# Arquitetura de Software

# Arquitetura de Software

- Trata de grandes estruturas do sistema
- Abstração – desconsidera detalhes de implementação, algoritmos e estruturas de dados
- Se concentra na interação de elementos do sistema como “caixas pretas”

(Bass, 2003)

# Arquitetura de Software
# Definição

- “A organização fundamental de um sistema personificado pelos seus componentes, seus relacionamentos entre si, e com o ambiente, e os princípios que guiam seu projeto e evolução.” (IEEE, 2007)

# Arquitetura de Software
# Definição

- componentes, relacionamentos entre si, princípios que guiam seu projeto e evolução

# Estilo Arquitetural

# Padrão ou Estilo Arquitetural

- Famílias de programas
  - conjuntos de programas que possuem tantas propriedades em comum, que torna-se mais vantajoso estudá-las a partir de suas similaridades, antes mesmo de analisar membros individuais [PARN76]
- Similaridades apontam para “padrões arquiteturais” ou “estilos arquiteturais”

# Padrão ou Estilo Arquitetural

- “Um padrão arquitetural é uma descrição de tipos de elementos e relações junto com um conjunto de restrições relativas a como eles podem ser usados.” (Bass, 2003)

Estilos Arquiteturais

Decomposição Modular

# Pipe & Filter

# *Pipe* & *Filter*

- Bastante popular em sistemas operacionais UNIX-like
- Processo incremental
  - vai gerando os dados de saída, sem esperar que a entrada de dados se complete (Garlan, 1993).
- Invariantes (Garlan, 1993)
  - entidades independentes
  - identidades de entrada e saída desconhecidas
  - especificação local

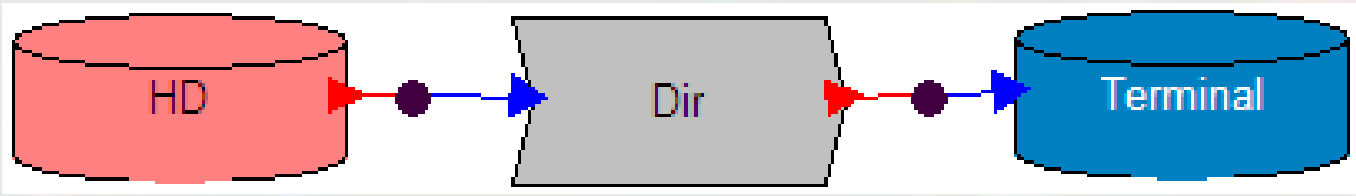
# *Pipe* & *Filter*

- *Filter* (componente)
  - Lê fluxos de dados de entrada e produz seus resultados como fluxos de dados de saída.


- *Pipe* (conector)
  - Conduzem o fluxo, conectando o fluxo de saída de um filter ao fluxo de entrada de outro filter.

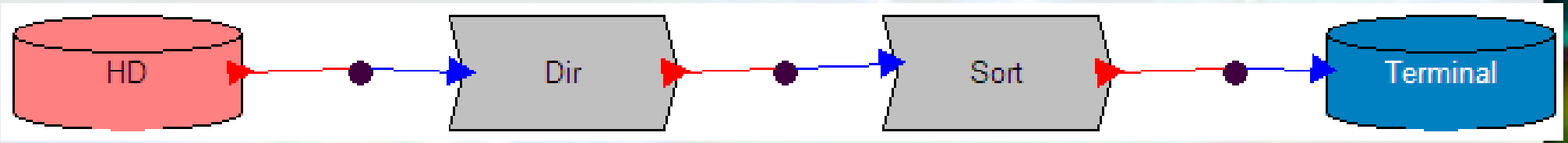
# Pipe & Filter UNIX-like

- Lista nome dos arquivos
  - dir /b



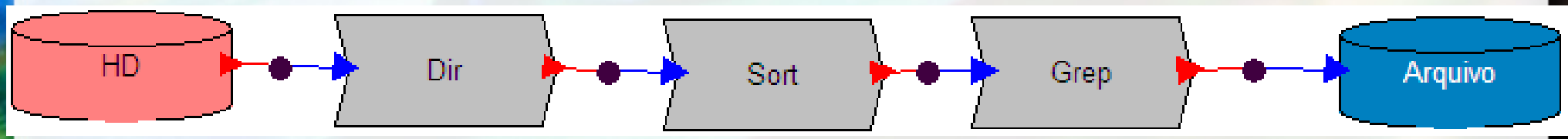
# Pipe & Filter UNIX-like

- Operador de pipe no DOS e Unix: |
- Lista nome dos arquivos “pipe” coloca em ordem alfabética
  - dir /b | sort



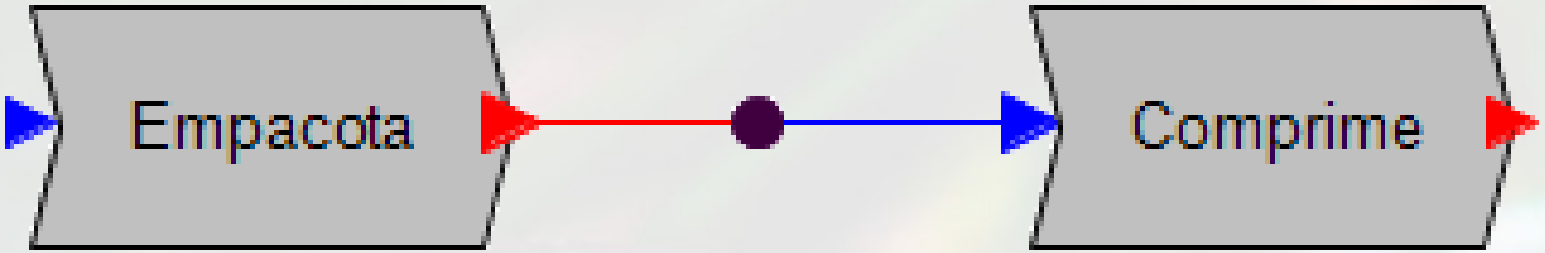
# Pipe & Filter Unix-like

- Lista nome dos arquivos “pipe” coloca em ordem alfabética “pipe” recorta aqueles que têm o trecho “Win”
  - dir /b | sort | grep "Win"
- Redireciona saída (pipe) no DOS: >
- Mesmo anterior com saída para arquivo “resultado.txt”
  - dir /b | sort | grep "Win" >resultado.txt

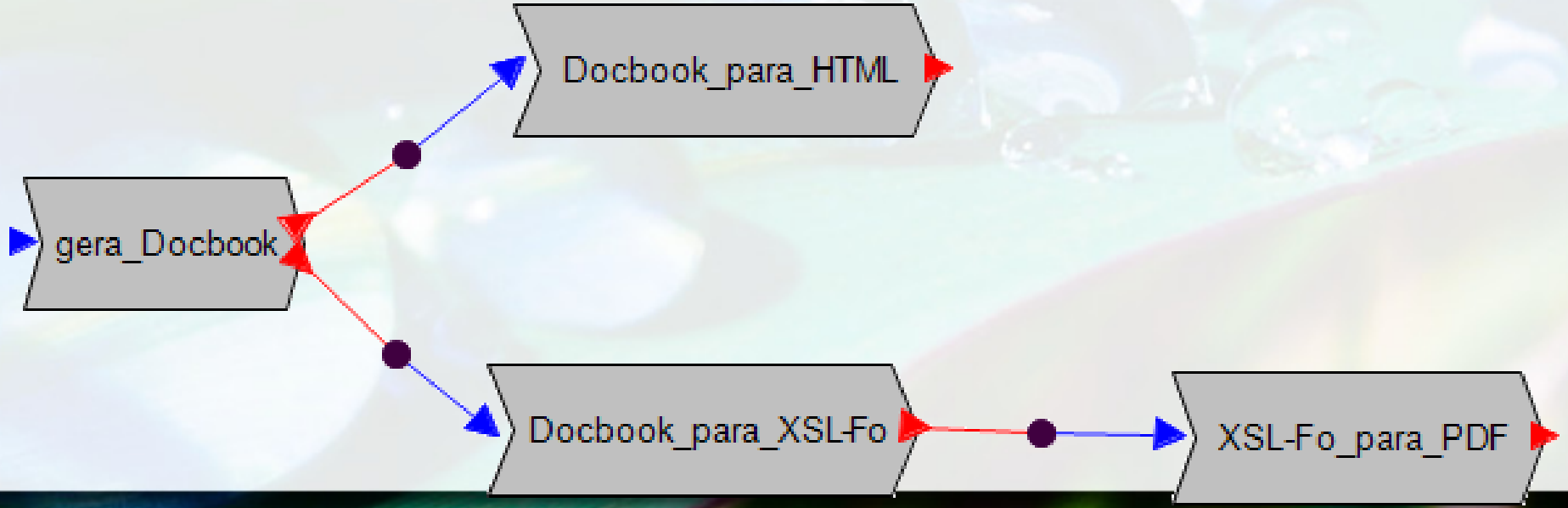


# *Pipe* & *Filter*
# Exemplos

- Empacotando e comprimindo




- Docbook



# Pipe & Filter - Implementação Java

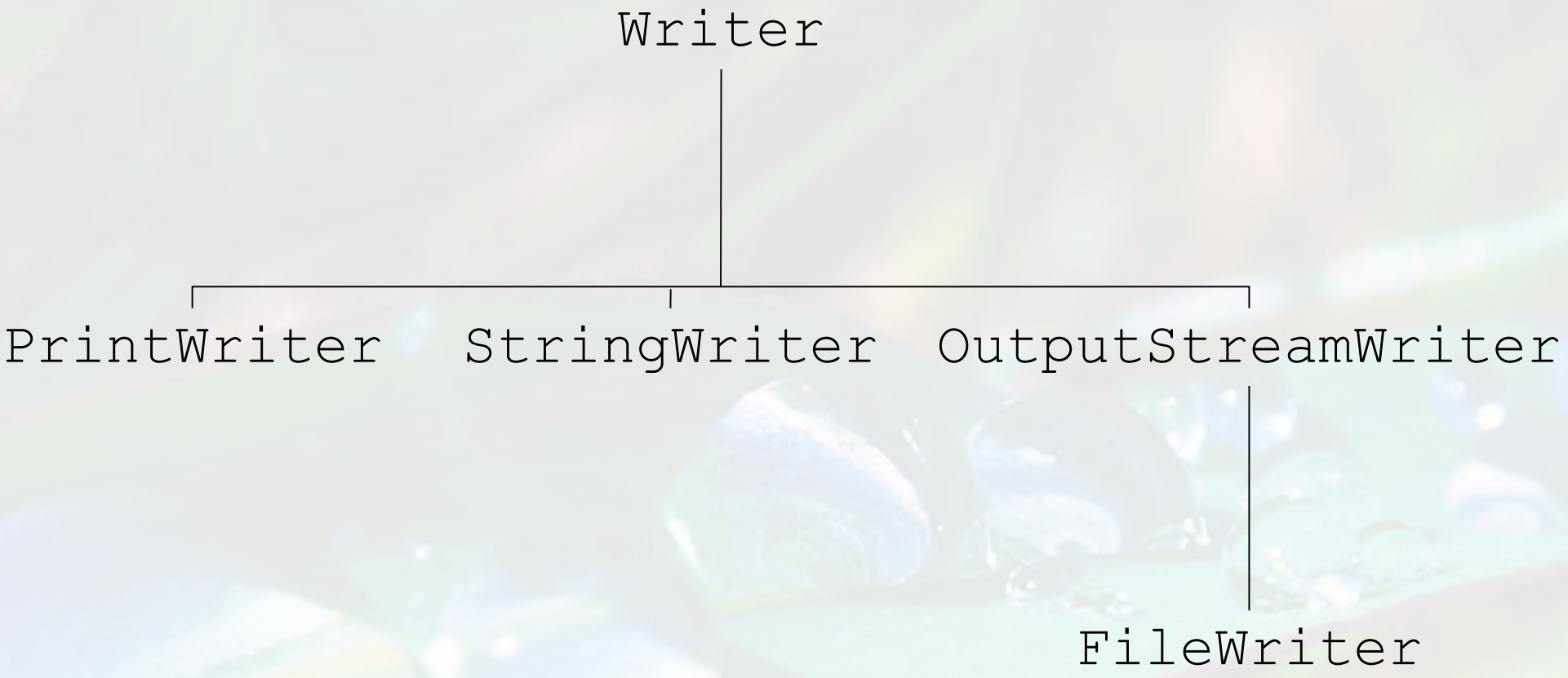
## Streams

- Envio e recuperação de dados para/de fontes externas (arquivos, dados pela rede etc.)
- Streams representam fluxos de informação de entrada ou saída
- As Streams são representadas genericamente por duas classes abstratas:

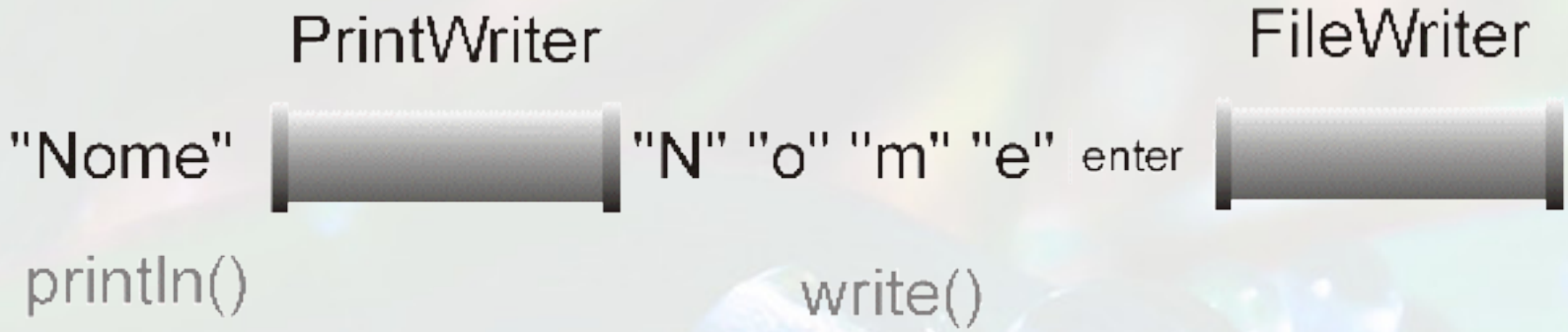
`Reader` - stream de entrada

`Writer` - stream de saída

# Hierarquia de Streams `Writer`



# Writer → Pipe & Filter



# Pipe & Filter em Java

1. Crie o Writer final

2. Crie o Writer inicial ligado ao final

3. Ao chamar o método do Writer inicial ele automaticamente canalizará para o final

4. Feche o Writer no final

```
FileWriter arquivo;
arquivo = new FileWriter("saida2.txt");
```

```
PrintWriter formatado;
formatado = new PrintWriter(arquivo);
```

```
formatado.println("Tecodonte");
```

```
formatado.close();
```

# Referências


- Abowd, G. D., Allen, R., Garlan, D. **Formalizing style to understand descriptions of software architecture**. ACM Trans. Softw. Eng. Methodol., ACM Press, 1995, 4, 319-364.
- Agenda OpenSystems. **COMPIERE – Smart Open Source ERP Software with integrated CRM Solutions**. Disponível em http://www.agenda.si/fileadmin/www.agenda.si/documents/Compiere.opis.pdf, acessado em 9/04/2010.
- Bass, L., Clements, P., Kazman, R. **Software Architecture in Practice**. Addison-Wesley, 2003.
- Clements, P. C., Northrop, L. M. **Software Architecture: An Executive Overview**. Technical Report – CMU/SEI-96-TR-003 – ESC-TR-96-003, Fevereiro 1996.

# Referências


- Garlan, D. et al. **Architectural Mismatch (Why It's Hard to Build Systems Out of Existing Parts)**. Proceedings, 17th Int. Conf. on Software Engineering. Seattle, WA, April 23-30, 1995.
- Garlan, D., Monroe, R. T., Wile, D. **Acme: Architectural Description of Component-Based Systems**. Foundations of Component-Based Systems, Cambridge University Press, 2000, 47-68.
- He, H. **What Is Service-Oriented Architecture**. Setembro 2003. Disponível em http://www.xml.com/pub/a/ws/2003/09/30/soa.html
- Houaiss, Instituto Antônio. **Dicionário Houaiss da língua portuguesa**. Editora Objetiva, Março 2006.
- Krasner, G., Pope, S. **A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 system**. Journal of Object Oriented Programming, 1988, 1, 26-49.

# Referências


- Krueger, C. W. **Software Reuse**. ACM Comput. Surv., ACM Press, 1992, 24, 131-183.
- Papazoglou, M. P., Georgakopoulos, D. **Service-oriented computing**. Commun. ACM, 2003, 46, 25-28.
- Parnas, D. **On the Design and Development of Program Families**. IEEE Transactions on Software Engineering SE-2, 1976, 1, 1-9.
- Pressman, R. (2006) Engenharia de Software, 6.ed. - São Paulo: McGraw-Hill.
- Shaw, M. **Abstraction Techniques in Modern Programming Languages**. IEEE Software, 1984, 1, 4, 10-26.
- Shaw, M., Garlan, D. **Software Architecture: Perspectives on an Emerging Discipline**. Prentice Hall, 1996.

# Referências


- Software Engineering Standards Committee of the IEEE Computer Society. **Systems and software engineering - Recommended practice for architectural description of software-intensive systems**, ISO/IEC 42010 IEEE Std 1471-2000 First edition 2007-07-15, Julho 2007.
- Sommerville, I. (2007) **Software Engineering**, 8th. ed. Addison Wesley.
- Taylor, R. N. , et al. **A Component- and Message-Based Architectural Style for GUI Software**. IEEE Trans. Software Engineering, IEEE Press, 1996, 22, 390-406.
- Wegner, P. **Varieties of reusability.** In Workshop on Reusability in Programming (Newport, R. I., Sept.). ITT Programming, Stratford, Corm., pp. 30–44, 1983.

# Agradecimentos

Fotografias de Simone Almeida Chaves Santanchè

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 Moyan Brenn [ http://www.flickr.com/photos/aigle_dore/] por sua fotografia “Dew drops” usada na capa e nos fundos, disponível em [ http://www.flickr.com/photos/aigle_dore/6225536653/] vide licença específica da fotografia.