Universidade de São Paulo
Instituto de Matemática e Estatística
Programação Orientada a Objetos - 2014

Programação Orientada a Aspectos (POA)

Diogo de Jesus Pina
Paulo Mei
Thiago Dias Simão

Introdução

```
public class RegraNegocio {
    ... Dados centrais
    ... Log stream
    ... Concorrecia control lock

    public void algumaOperacao(parametros) {
        ... Garanta autorização usando as credenciais
        ... Feche o lock para garantir a restrição de acesso a região crítica
        ... Comece a transação
        ... Registre o início da operação
        ... Execute a operação principal
        ... Registre o fim da operação
        ... Complete a transação ou a desfaça
        ... Abra o lock para permitir que a entrada na região crítica
    }
}
```

Introdução

```
public class RegraNegocio {
        ... Dados centrais

        public void algumaOperacao(parametros) {
                ... Execute a operação principal
        }
}
```

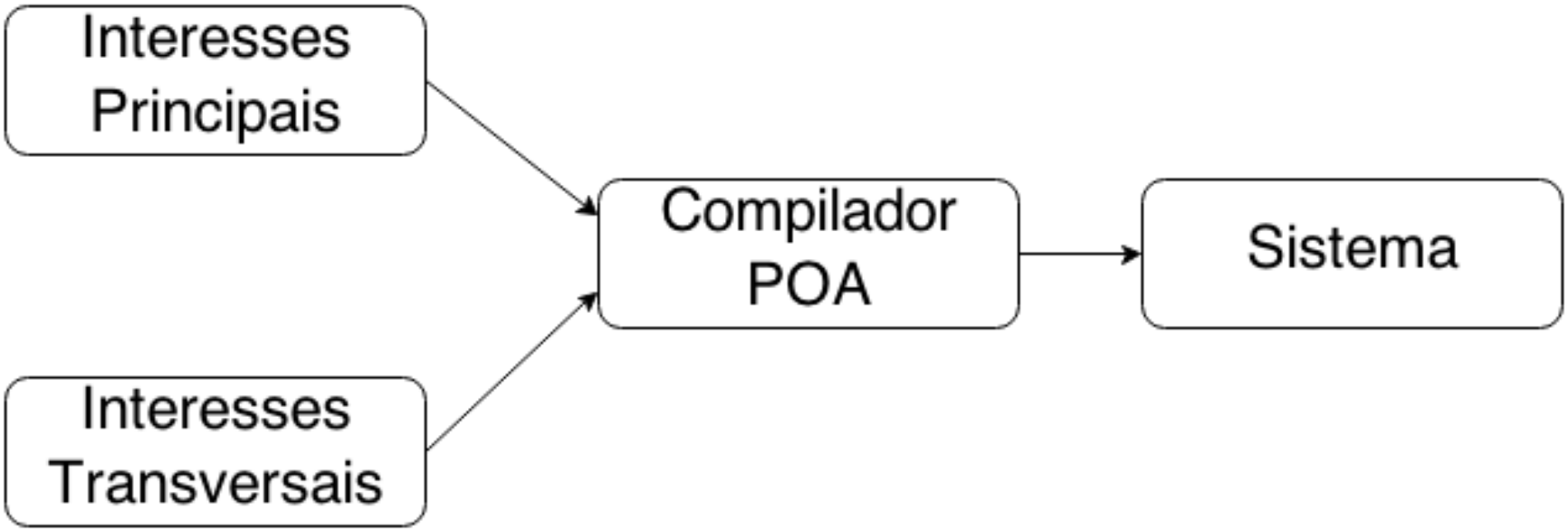
História

- Cristina Lopes
- Gregor Kiczales
- Xerox PARC
- 1996
- AspectJ

Anatomia de uma Linguagem POA

- Linguagem de especificação
  - Dados e Comportamento
    - C, C++ e Java
- Linguagem de implementação
  - Regras de Especificação de Tecelagem

Implementação de POA



Conceitos Fundamentais

- Identificar *join points*
- Criar *pointcuts*
- Alterar o comportamento do programa
  - *Advice*
- Alterar a estrutura estática do sistema
- Módulo para expressar todos os construtores transversais
  - Aspecto

POA por Analogia

- HTML e CSS
- Sistemas de Banco de Dados
  - SQL e *Pointcuts*
  - *Triggers* e *Advice*
- Programação Orientada a Eventos

Implementações de POA

- AspectJ
- Spring AOP
- JBoss (*pointcuts*)
- AspectS
- PHANtom for Pharo smalltalk
- Aquarium for Ruby
- AspectC++
- PostSharp
- Aspect#

Alternativas para POA

- Arcabouço
- Geração de Código
- Padrões de Design
  - *Observer*
  - *Chain of Responsibility*
  - *Decorator* e *Proxy*
  - *Interceptor*
- Linguagens Dinâmicas

Custos

- Curva de aprendizagem
- Padronizar adoção para evitar um uso abusivo
- Modificar o processo de *build* e outros
- Lidar com as ferramentas disponíveis
- Necessidades de maiores habilidades
- Fluxo complexo do programa

Benefícios

- Simplicidade de Design
- Implementação mais Limpa
- Melhor Reuso de Código

AspectS

Abordagem POA no ambiente Squeak.

AspectJ + MethodWrappers

AspectS

1. *Join point*
2. *Pointcut*
3. *Advice*

AspectS

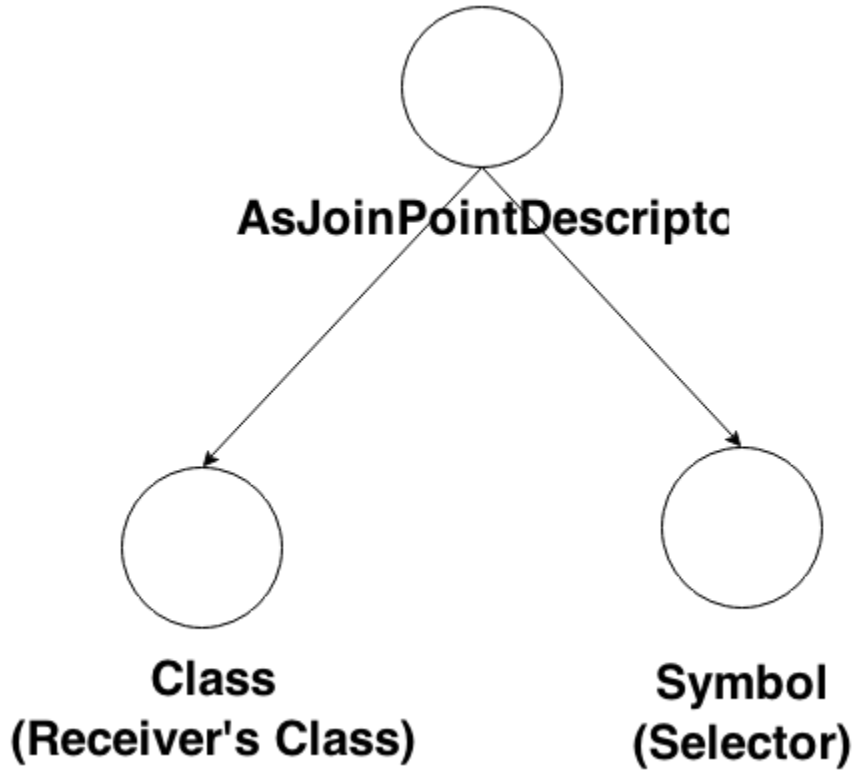
Aspects: AsAspect

Exemplo:

```
AsAspect subclass: #TimestampedTranscriptAspect
    instanceVariableNames: ''
    classVariableNames: ''
    poolDictionaries: ''
    category: 'AspectS-Examples TimestampedTranscript'
```

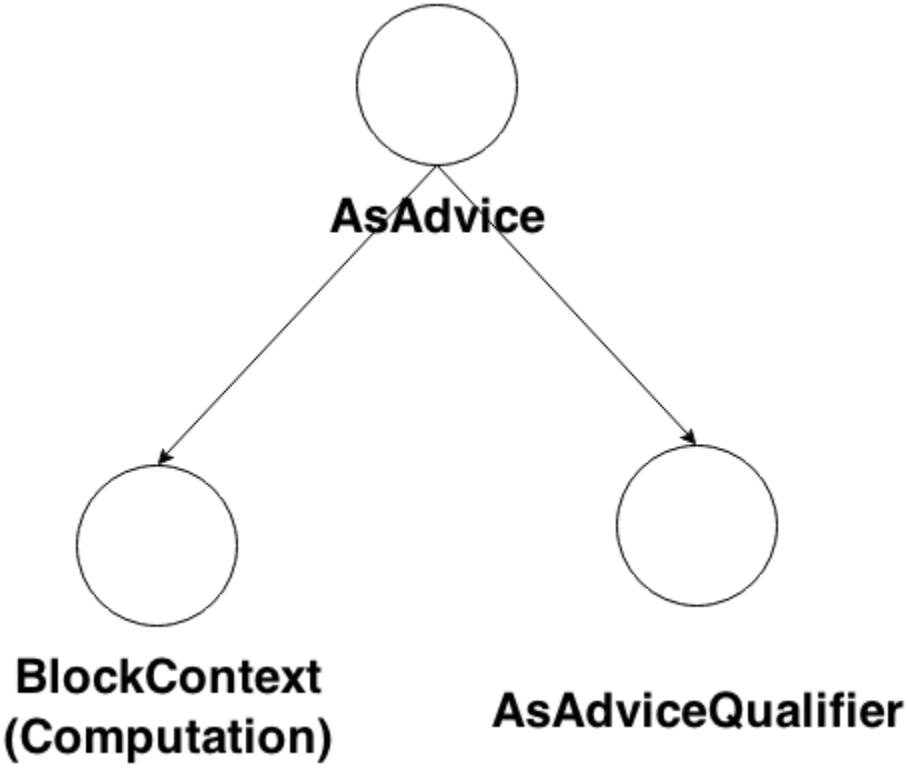
AspectS

Pointcut e Join Points



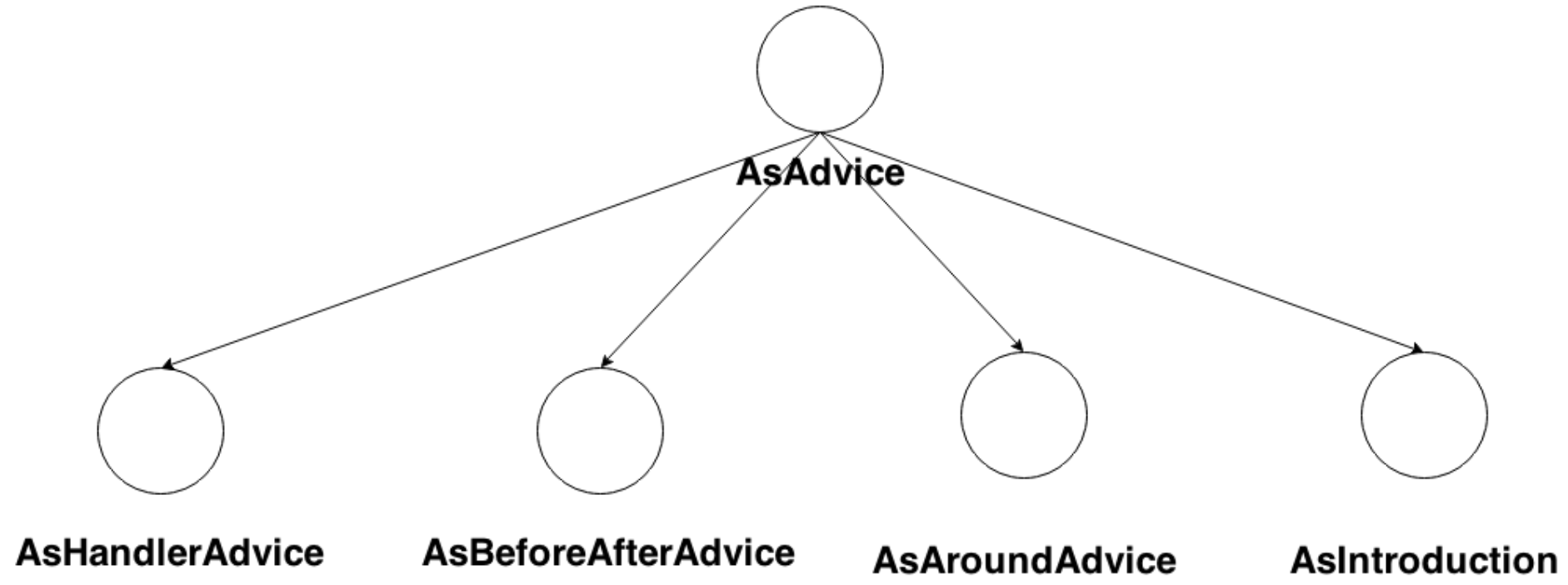
AspectS

Advice



AspectS

Tipos de Advices



AspectS

Exemplo:

```
TimestampedTranscriptAspect >> adviceTranscriptStreamShow
    ^ AsBeforeAfterAdvice
        qualifier: (AsAdviceQualifier
            attributes: { #receiverClassSpecific. #cfFirstClass. })
        pointcut: [{
            AsJoinPointDescriptor
                targetClass: TranscriptStream
                targetSelector: #show:. }]
        beforeBlock: [:receiver :arguments :aspect :client |
            Transcript show: '[', Time now printString, '] ' ]
```

AspectJ

Extensão orientada a aspectos para a linguagem de programação Java

AspectJ

1. *Join point*
2. *Pointcut*
3. *Advice*

AspectJ

***Join Point**:*

Ponto da execução de um programa que pode ser identificado

```
public class Account {
        static int _balance;
        void credit(float amount){
                _balance += amount;
        }
}
```

AspectJ

***Pointcut**:*

Um meio de selecionar *join points*

```
import main.Account;
public aspect Example {

    pointcut creditExecution(): execution(void Account.credit(float));

    pointcut setCredit(): set(private float Account._balance);
}
```

AspectJ

***Advice**:*

Código a ser executado

```
before() : creditExecution() {
    authenticator.authenticate();
}
after() : creditExecution() {
    System.out.println("credit finished");
}
```

AspectJ

Aspect:

```
import security.Authenticator;
import main.Account;
public aspect Example {
    pointcut creditExecution():
        execution(void Account.credit(float));

    private Authenticator authenticator = new Authenticator();

    before() : creditExecution() {
        authenticator.authenticate();
    }
    after() : creditExecution() {
        System.out.println("credit finished");
    }
}
```

AspectJ

Exemplos:

1. **Segurança**
2. **Log**
3. **Otimização**
4. **Reutilização de código**
5. **Acesso a parâmetros**

Referências

- **AspectJ in Action, Second Edition,** Enterprise AOP with Spring Applications - Ramnivas Laddad, Foreword by Rod Johnson. Setembro de 2009, Manning Publications Co.
- **Aspect-Oriented Programming** - Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Videira Lopes, Jean-Marc Loingtier, John Irwin. Junho de 1997. European Conference on Object-Oriented Programming (ECOOP)
- **Programação Orientada a Aspectos, Uma Visão Geral** - Alexandre Henrique Vieira Soares, Anderson de Rezende Rocha, Flávio Luis Alves, Júlio César Alves. Departamento de Ciência da Computação, Universidade Federal de Lavras
- **AspectJ em 20 minutos** - Diogo Vinícius Winck, Vicente Goetten. Javafree. org
- **AspectS – Aspect-Oriented Programming with Squeak** - Robert Hirschfeld. DoCoMo Communications Laboratories Europe
- **Extending Advice Activation in AspectS** - Robert Hirschfeld, Pascal Costanza.

Obrigado!!!

Dúvidas?