AspectJ: Declaração Intertipo

Eduardo Figueiredo

http://www.dcc.ufmg.br/~figueiredo

Declarações Intertipo

- Ponto de corte e adendo afetam o comportamento dinâmico do sistema
- AspectJ também fornece mecanismos para alterar a estrutura estática
- Tipos de declarações intertipo
  - Introduzir membros (métodos, atributos e construtores)
  - Modificar a hierarquia de herança

Exemplos: Introduzir Membros

```
private float Conta.chequeEspecial;
```

Introduz um novo atributo chamado `chequeEspecial` do tipo `float` na classe `Conta`

```
public static void Conta.main(String[] args){...}
```

Introduz um novo método `main` na classe `Conta`

Exemplos: Modificar a Herança

```
declare parents: Conta extends ContaAbstrata;
```

A classe `Conta` passa a estender `ContaAbstrata`

```
declare parents: Conta implements Serializable;
```

A classe `Conta` passa a implementar a interface `Serializable`

```
declare parents banco.entidades.*
                implements Serializable;
```

Toda classe e interface do pacote `banco.entidades` passa a estender/implementar `Serializable`

Herança de Aspectos

Aspecto

- Entidade modular semelhante a uma classe
  - Além de métodos e atributos, possui pontos de corte, adendos e declarações intertipo
- Um aspecto pode interceptar uma ou várias classes do sistema
- Tem como objetivo implementar um interesse transversal
  - Classes implementam os interesses centrais

Exemplo de Logging (Java)

Código da classe ATM que possui código relacionado a *Logging*

```
public class ATM {

   ...
   private int displayMainMenu() {
      screen.displayMessageLine( "\nMain menu:" );
      screen.displayMessageLine( "1 - View my balance" );
      screen.displayMessageLine( "2 - Withdraw cash" );
      screen.displayMessageLine( "3 - Deposit funds" );
      screen.displayMessageLine( "4 - Exit\n" );
      screen.displayMessage( "Enter a choice: " );
      int option = keypad.getInput();
      Logger.log("User option: " + option);
      return option;
   }
}
```

Aspecto *Logging*: Opção 1

Código parcial do aspecto *Logging*

```
public aspect Logging {

  ...

  public pointcut displayMainMenuPC() :
            call (private int ATM.displayMainMenu());

  after() returning (int option): displayMainMenuPC() {
    Logger.log("User option: " + option);
  }
}
```

O aspecto *Logging pega o retorno da chamada ao método displayMainMenu*

O que fazer com esta classe?

Código da classe Logger

```
public class Logger {

  private static Vector<String> logs = new Vector<String>();

  public static void log(String text) {
    logs.add(text);
  }

  public static void printLog() {
    for (int i=0; i<logs.size(); i++) System.out.println( logs.get(i) );
  }
}
```

Este código pode ser movido para o aspecto Logging, certo?

Aspecto *Logging*: Opção 2

Código parcial do aspecto *Logging*

```
public aspect Logging {

  private Vector<String> logs = new Vector<String>();
  private static final int LOG = 0;

  public void log(String text) {
    logs.add(text);
  }
  ...
  public pointcut displayMainMenuPC() :
            call (private int ATM.displayMainMenu());

  after() returning (int option): displayMainMenuPC() {
    log("User option: " + option);
  }
}
```

O código da classe *Logger* foi movido para o aspecto *Logging*

Aspecto *Logging*: Opção 3

Código parcial do aspecto *Logging*

```
public aspect Logging extends Logger {

  ...

  public pointcut displayMainMenuPC() :
            call (private int ATM.displayMainMenu());

  after() returning (int option): displayMainMenuPC() {
    log("User option: " + option);
  }
}
```

Aspectos podem herdar de outra classe ou de outro aspecto.

Bibliografia Principal

- R. LADDAD. **AspectJ in Action**, 2ª Ed. 2010.
  - Part 1 Understanding AOP and AspectJ

- Sergio Soares. **Programação Orientada a Aspectos com AspectJ**. Minicurso CBSoft 2010.