Tutorial 6
Aspect-oriented programming

How to program an aspect?
Tool support in aspectJ, AJDT
On Aspect-Orientation

• We explained the concepts of aspect-orientation: a recent paradigm in Software Engineering

• We shown a case study of aspect-oriented requirements engineering: (1) how to identify early aspects along with goal-oriented requirements analysis, (2) how the goal aspects are associated with the code aspects …

• What are code aspects?
Today…

This tutorial will cover code aspects:
1. AOP examples
2. aspectJ syntax
3. Eclipse/AJDT toolset
4. Relation to the OpenOME

5. How to deploy of our phase B results …
1. AspectJ

- It is one of the most mature aspect compilers [http://www.eclipse.org/aspectj](http://www.eclipse.org/aspectj)
- It has convenient tool support through Eclipse [http://www.eclipse.org/ajdt](http://www.eclipse.org/ajdt)
1.1 Example: the Hello class

See the online tutorial
1.2 Enable the World aspect
1.3 Run the woven code
2 The AspectJ syntax

- *aspect* vs. *class*
- *advice* vs. *method*
  - before(), after(), around()
  - allow formal parameters in regular expressions
  - special parameter joinpoint
  - allow “theJoinPoint” to access the call stack
  - around() can call the replaced method with proceed()
- (anonymous) pointcut vs. expression
  - just like any boolean expression, where the predicates are joinpoints
- joinpoints
  - call(), execute(): method calls (caller), execution (callee), try/catch blocks
  - set(), get(): field accesses
  - within(): to limit the scopes to method, class or packages
- declarations
  - Change the class structures: introducing fields, method, structural relations

Reference
The “quickref” document:

Spring 2005 ECE450H1S Software Engineering II
3 Some AJDT features

3.1 The tracer ex. without aspect
3.2 The Tracer example with aspect

```java
abstract pointcut myClass();
/**
 * The constructors in those classes.
 */

pointcut myConstructor(): myClass() & execution(new..);
/**
 * The methods of those classes.
 */

pointcut myMethod(): myClass() & execution(* *(..));
/**
 * Prints trace messages before and after executing constructors.
 */

before(): myConstructor() {
    traceEntry() + thisJoinPointStaticPart.getSignature();
}

after(): myConstructor() {
    traceExit() + thisJoinPointStaticPart.getSignature();
}
/**
 * Prints trace messages before and after executing methods.
 */

double tracing.TwoDShape.getY()
-->
double tracing.TwoDShape.getY()

<br>--- 4.0
```
3.3 A visualizer
4. Applying to the OpenOME

• I would show you an AOP example in OpenOME, however …

• AJDT requires more Memory resources, when I was compiling it with the complete OpenOME … Memory overflow problem

• Gold plating in AJDT?
The effects of the woven code
Visualizing the trace aspect
5. Hosting the course projects

- We have 11 teams, codenames are: mindz, photons, team2, xteam, lumiere solutions, mugqq, overnight enterprise, team7, websilon, team9, canadiantired
- Any windows ECF lab, use “terminal services”, the first server is dedicated for us, enter your team’s code name as username, “ece450” as password for now, you can access it, only within the lab for now
- The URL of the tomcat server is http://127.0.0.1:8080
- The deployed web service should be placed under c:\program files\apache software foundation\tomcat 5.5\webapps\axis\WEB-INF\classes\<codename>
- These directories are shared as \\128.100.36.17\<codename> so that you can copy the binary files into the place
- We also installed an Eclipse plugin for collecting metrics in the ECF windows lab