

Tutorial 6

Aspect-oriented programming

How to program an aspect?
Tool support in aspectJ, AJDT

Last lecture...

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...

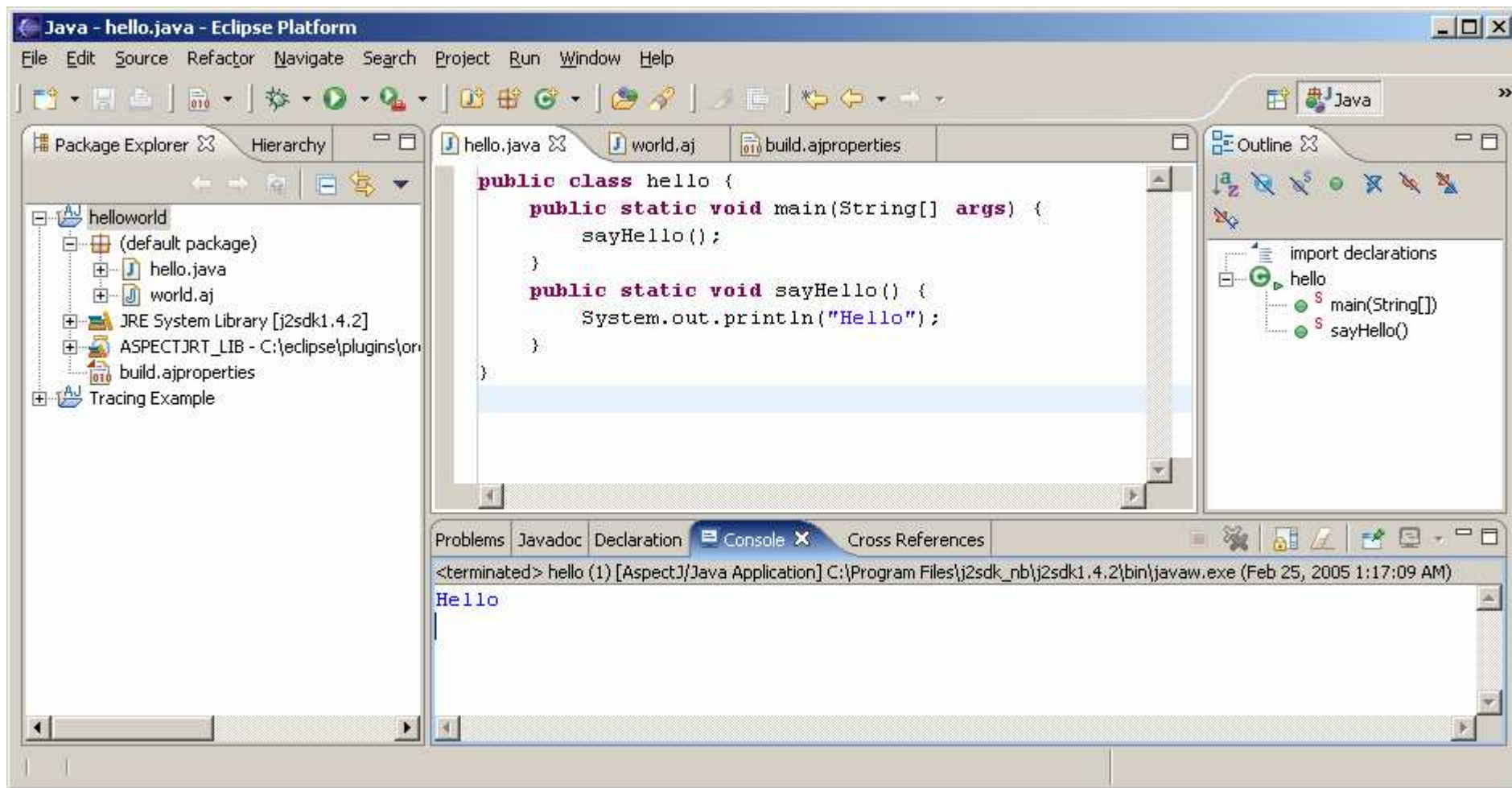
This tutorial will cover code aspects:

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

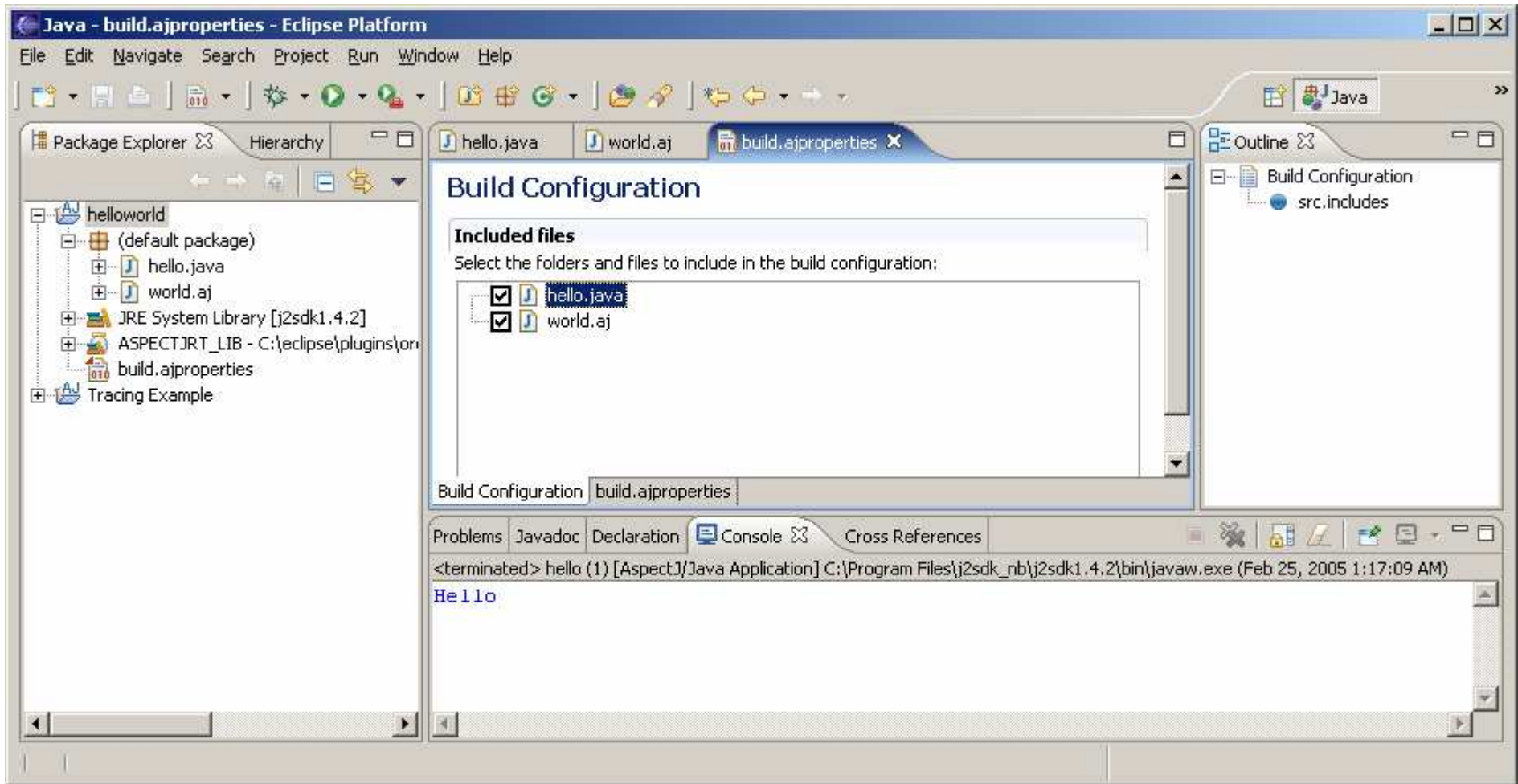
- It is one of the most mature aspect compilers <http://www.eclipse.org/aspectj>
- It has convenient tool support through Eclipse <http://www.eclipse.org/ajdt>

1.1 Example: the Hello class

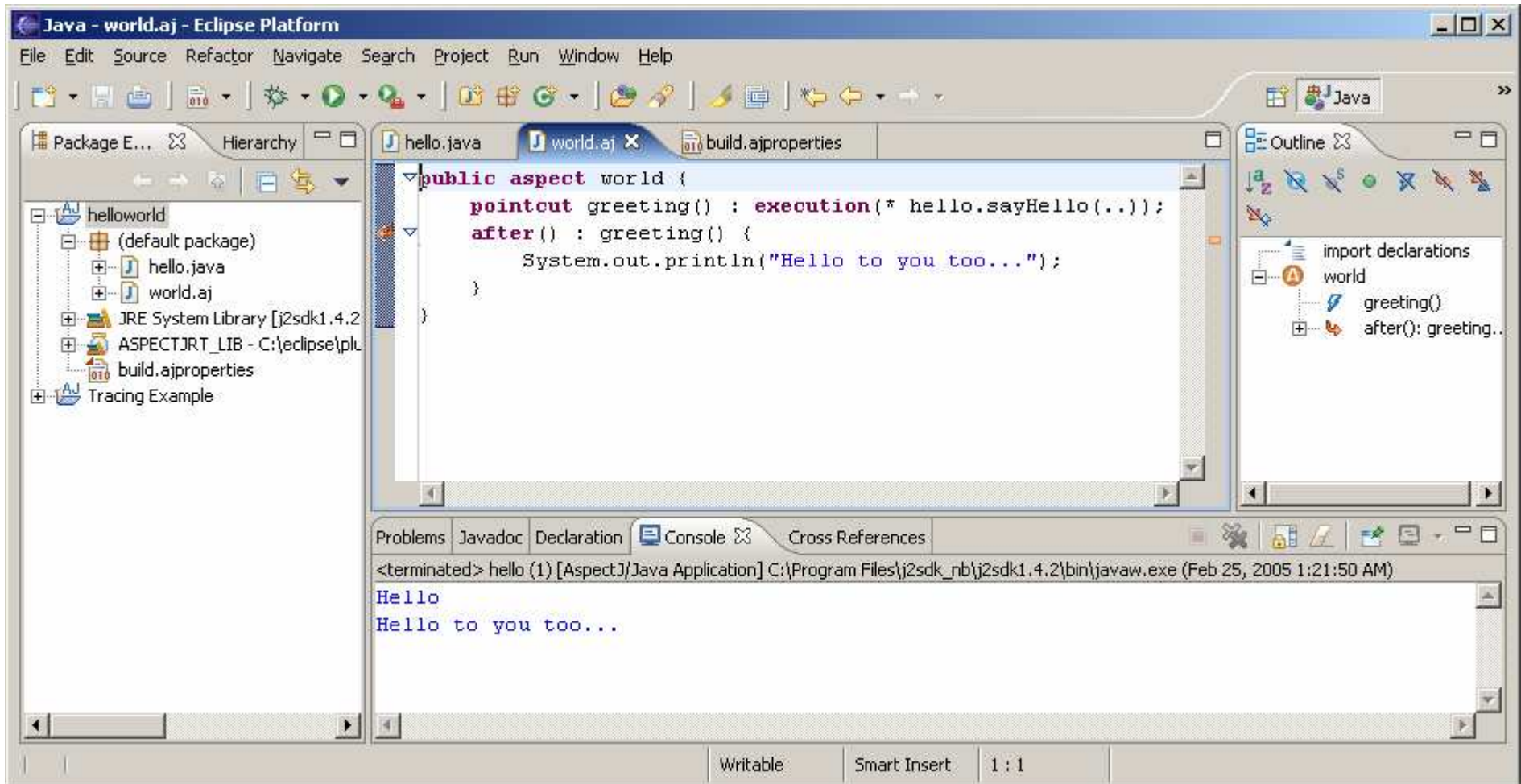


See the online tutorial

1.2 Enable the World aspect



1.3 Run the woven code



2 The AspectJ syntax

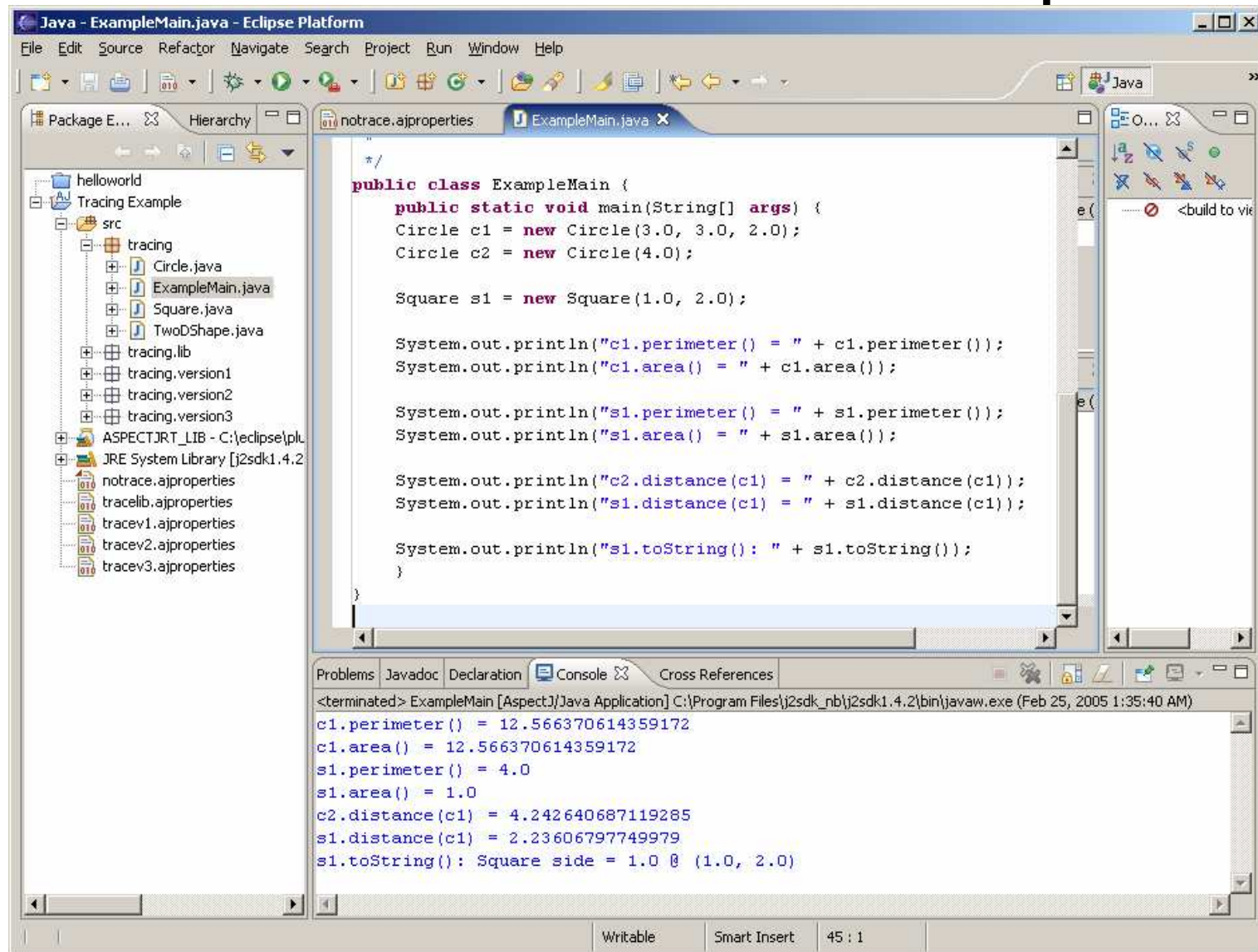
- *aspect vs. class*
- *advice vs. method*
 - **before()**, **after()**, **around()**
 - allow formal parameters in regular expressions
 - special parameter `joinpoint`
 - allow “`theJoinPoint`” to access the call stack
 - `around()` can call the replaced method with **proceed()**
- (anonymous) pointcut 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:

3 Some AJDT features

3.1 The tracer ex. without aspect



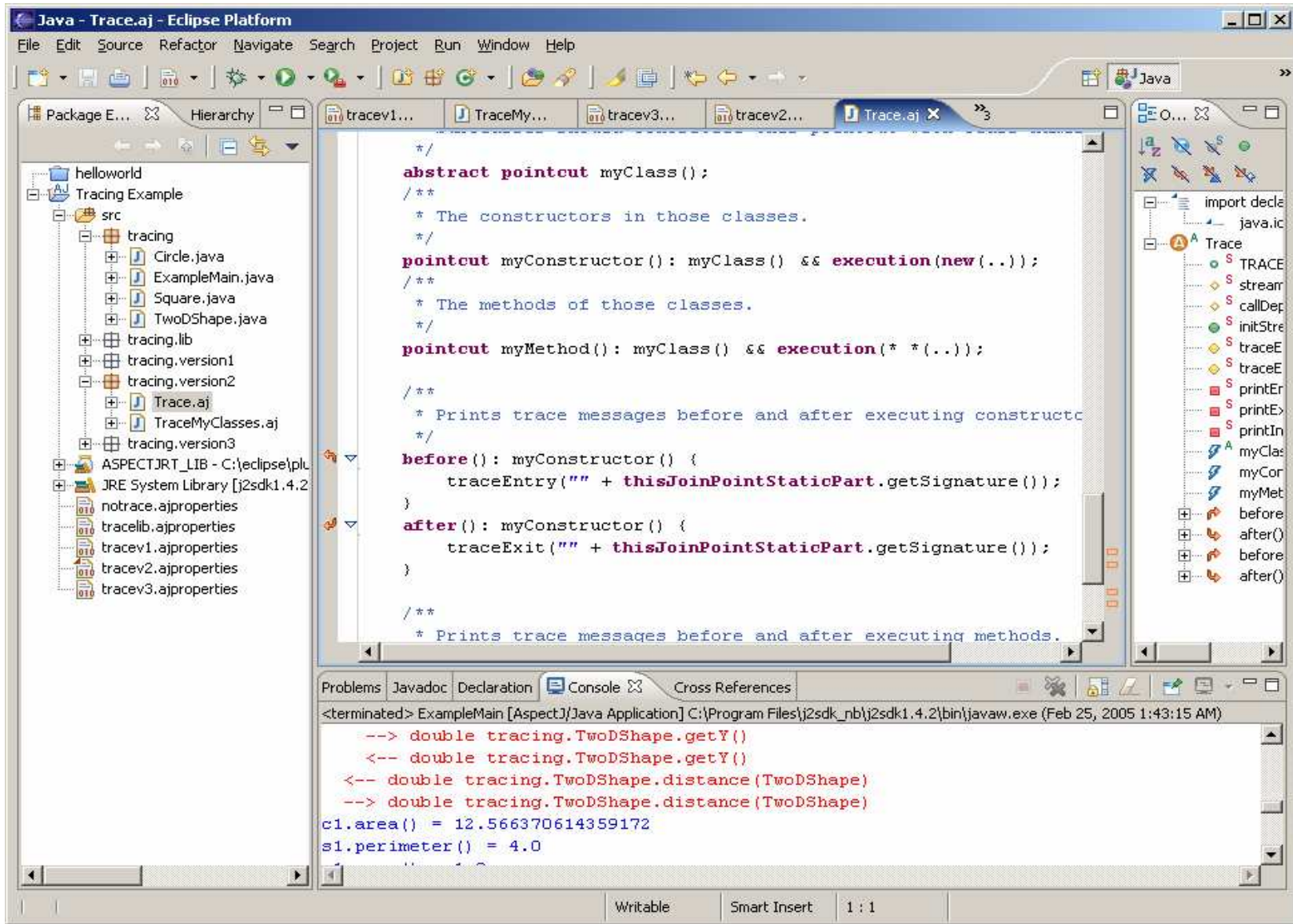
The screenshot shows the Eclipse IDE with the following components:

- Package Explorer:** Shows a project named 'helloworld' with a package 'Tracing Example' containing a 'src' folder. Inside 'src', there is a 'tracing' folder with files 'Circle.java', 'ExampleMain.java', 'Square.java', and 'TwoDShape.java'. There are also 'tracing.lib', 'tracing.version1', 'tracing.version2', and 'tracing.version3' folders. Below the project, there are several 'ajproperties' files: 'notrace.ajproperties', 'tracelib.ajproperties', 'tracev1.ajproperties', 'tracev2.ajproperties', and 'tracev3.ajproperties'.
- Editor:** Displays the source code of 'ExampleMain.java'. The code is as follows:

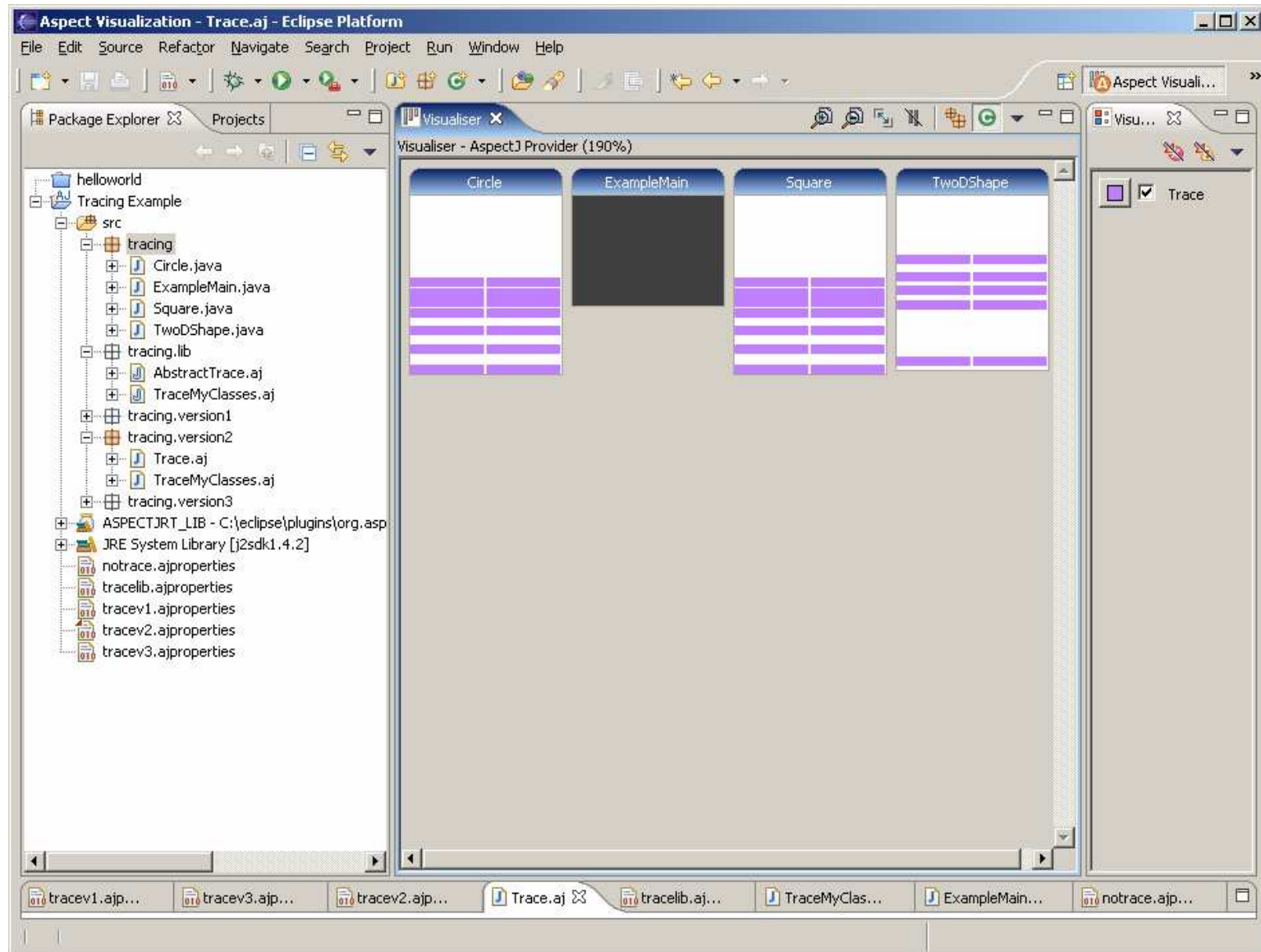
```
public class ExampleMain {  
    public static void main(String[] args) {  
        Circle c1 = new Circle(3.0, 3.0, 2.0);  
        Circle c2 = new Circle(4.0);  
  
        Square s1 = new Square(1.0, 2.0);  
  
        System.out.println("c1.perimeter() = " + c1.perimeter());  
        System.out.println("c1.area() = " + c1.area());  
  
        System.out.println("s1.perimeter() = " + s1.perimeter());  
        System.out.println("s1.area() = " + s1.area());  
  
        System.out.println("c2.distance(c1) = " + c2.distance(c1));  
        System.out.println("s1.distance(c1) = " + s1.distance(c1));  
  
        System.out.println("s1.toString(): " + s1.toString());  
    }  
}
```
- Console:** Shows the output of the program execution:

```
<terminated> ExampleMain [AspectJ]Java Application C:\Program Files\jdk1.4.2\bin\javaw.exe (Feb 25, 2005 1:35:40 AM)  
c1.perimeter() = 12.566370614359172  
c1.area() = 12.566370614359172  
s1.perimeter() = 4.0  
s1.area() = 1.0  
c2.distance(c1) = 4.242640687119285  
s1.distance(c1) = 2.23606797749979  
s1.toString(): Square side = 1.0 @ (1.0, 2.0)
```
- Bottom Bar:** Shows 'Writable', 'Smart Insert', and '45 : 1'.

3.2 The Tracer example with aspect

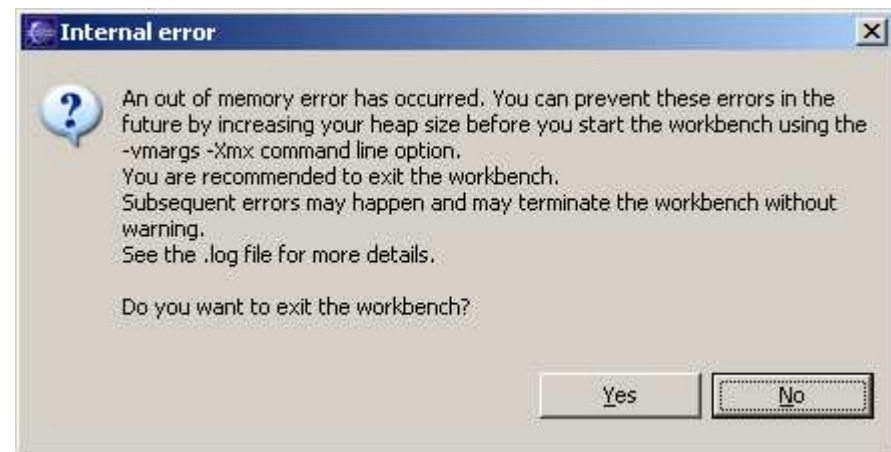


3.3 A visualizer



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

The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays a project structure for 'OpenOME' with various source files and binaries. The central 'Build Configuration' dialog for 'build.ajproperties' shows a tree view of included files, including the 'edu' package and its sub-packages 'stanford', 'toronto', and 'cs'. The 'toronto' package is expanded to show 'ome', 'aspects', and 'model' sub-packages, with several files like 'OMETab.java', 'model_trace.aj', 'tracing.aj', and 'controller' selected. The Console window at the bottom shows the output of a Java application, displaying a series of 'execution' and '<-- execution' messages for the 'TelosElement.getParent()' method, indicating the presence of woven code.

```
Build Configuration
```

Included files

Select the folders and files to include in the build configuration:

- edu
 - stanford
 - toronto
 - cs
 - ome
 - OMETab.java
 - aspects
 - model_trace.aj
 - tracing.aj
 - controller
 - model

Build Configuration build.ajproperties

```
aop [AspectJ/Java Application] C:\Program Files\j2sdk_nb\j2sdk1.4.2\bin\javaw.exe (Feb 25, 2005 1:52:50 PM)
--> execution(OMElement edu.toronto.cs.ome.model.TelosElement.getParent())
<-- execution(OMElement edu.toronto.cs.ome.model.TelosElement.getParent())
--> execution(OMElement edu.toronto.cs.ome.model.TelosElement.getParent())
<-- execution(OMElement edu.toronto.cs.ome.model.TelosElement.getParent())
--> execution(OMElement edu.toronto.cs.ome.model.TelosElement.getParent())
<-- execution(OMElement edu.toronto.cs.ome.model.TelosElement.getParent())
--> execution(OMElement edu.toronto.cs.ome.model.TelosElement.getParent())
<-- execution(OMElement edu.toronto.cs.ome.model.TelosElement.getParent())
--> execution(OMElement edu.toronto.cs.ome.model.TelosElement.getParent())
<-- execution(OMElement edu.toronto.cs.ome.model.TelosElement.getParent())
--> execution(OMElement edu.toronto.cs.ome.model.TelosElement.getParent())
<-- execution(OMElement edu.toronto.cs.ome.model.TelosElement.getParent())
--> execution(OMElement edu.toronto.cs.ome.model.TelosElement.getParent())
<-- execution(OMElement edu.toronto.cs.ome.model.TelosElement.getParent())
--> execution(OMElement edu.toronto.cs.ome.model.TelosElement.getParent())
<-- execution(OMElement edu.toronto.cs.ome.model.TelosElement.getParent())
```

Visualizing the trace aspect

The screenshot shows the Eclipse IDE interface for AspectJ visualization. The main window is titled "Visualiser - AspectJ Provider" and displays a series of vertical tracks for different classes: Ele..., TelosFra..., TelosFun..., TelosLink, TelosLink..., TelosModel, TelosObj..., TelosType, TelosVie..., and TwinElem. The tracks show execution traces with blue and black bars. A "tracing" checkbox is checked in the right-hand panel. The bottom of the IDE shows a "Build Configuration" window and a "Console" window with execution logs.

Build Configuration

Included files

Select the folders and files to include in the build configuration:

- edu
- stanford
- toronto

Build Configuration build.ajproperties

```
<terminated> aop [AspectJ]Java Application C:\Program Files\j2sdk_nb\j2sdk1.4.2\bin\javaw.exe (Fel
<-- execution(OMEElement edu.toronto.cs.ome.model.TelosEle
--> execution(OMEElement edu.toronto.cs.ome.model.TelosElem
<-- execution(OMEElement edu.toronto.cs.ome.model.TelosEle
--> execution(OMEElement edu.toronto.cs.ome.model.TelosElem
<-- execution(OMEElement edu.toronto.cs.ome.model.TelosEle
```

5. Hosting the course projects

- We have 11 teams, codenames are:
mindz, photons, team2, xteam, lumiere solutions, muggq, 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>\<code><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