

Aspect-Oriented Programming with AspectJ™

“Those are not independent coincidences.”
-- A Tangled Tale, L.C.

credits

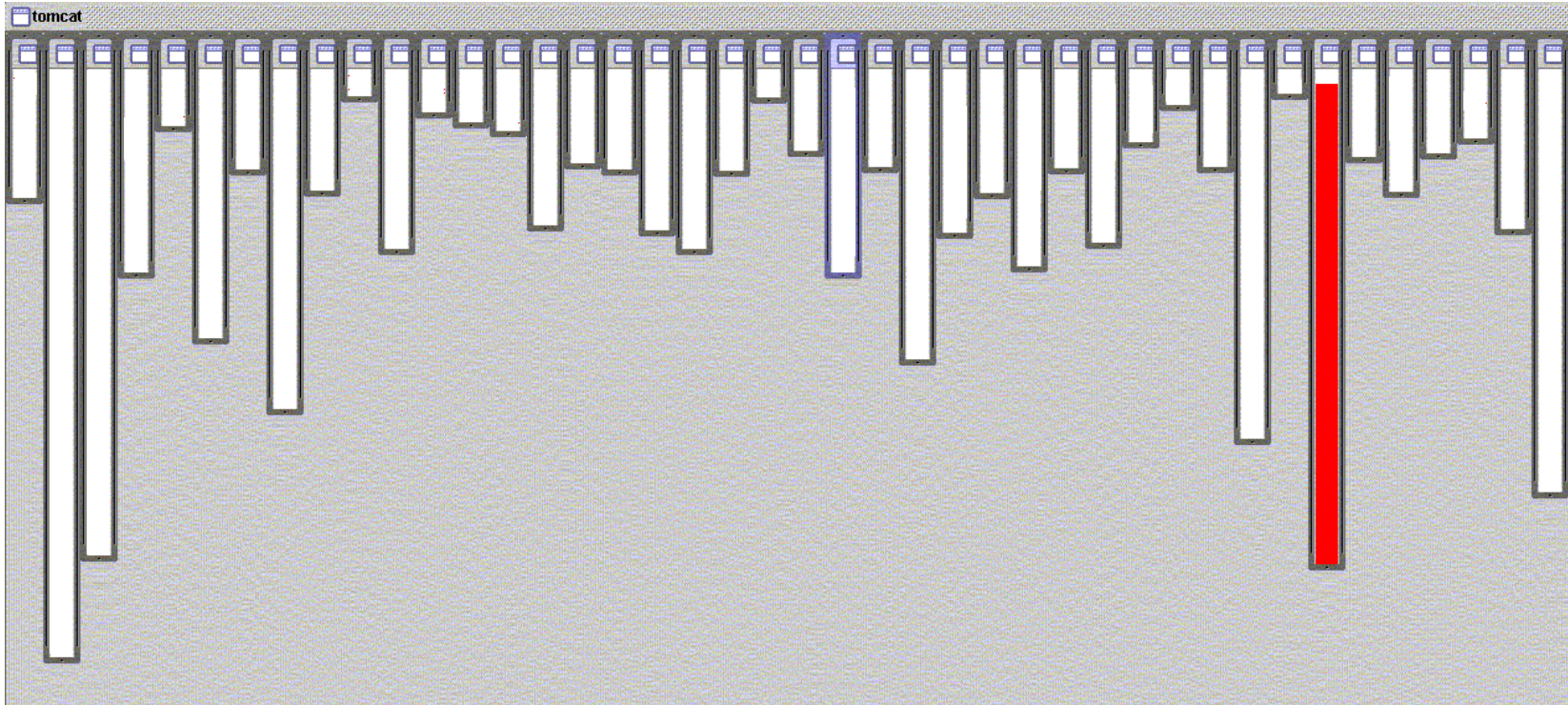
This talk is based on work done by the AspectJ team at Xerox PARC.

**Bill Griswold, Erik Hilsdale, Jim Hugunin,
Mik Kersten, Gregor Kiczales, Jeffrey Palm**

slides, compiler, tools & documentation are available at aspectj.org

good modularity

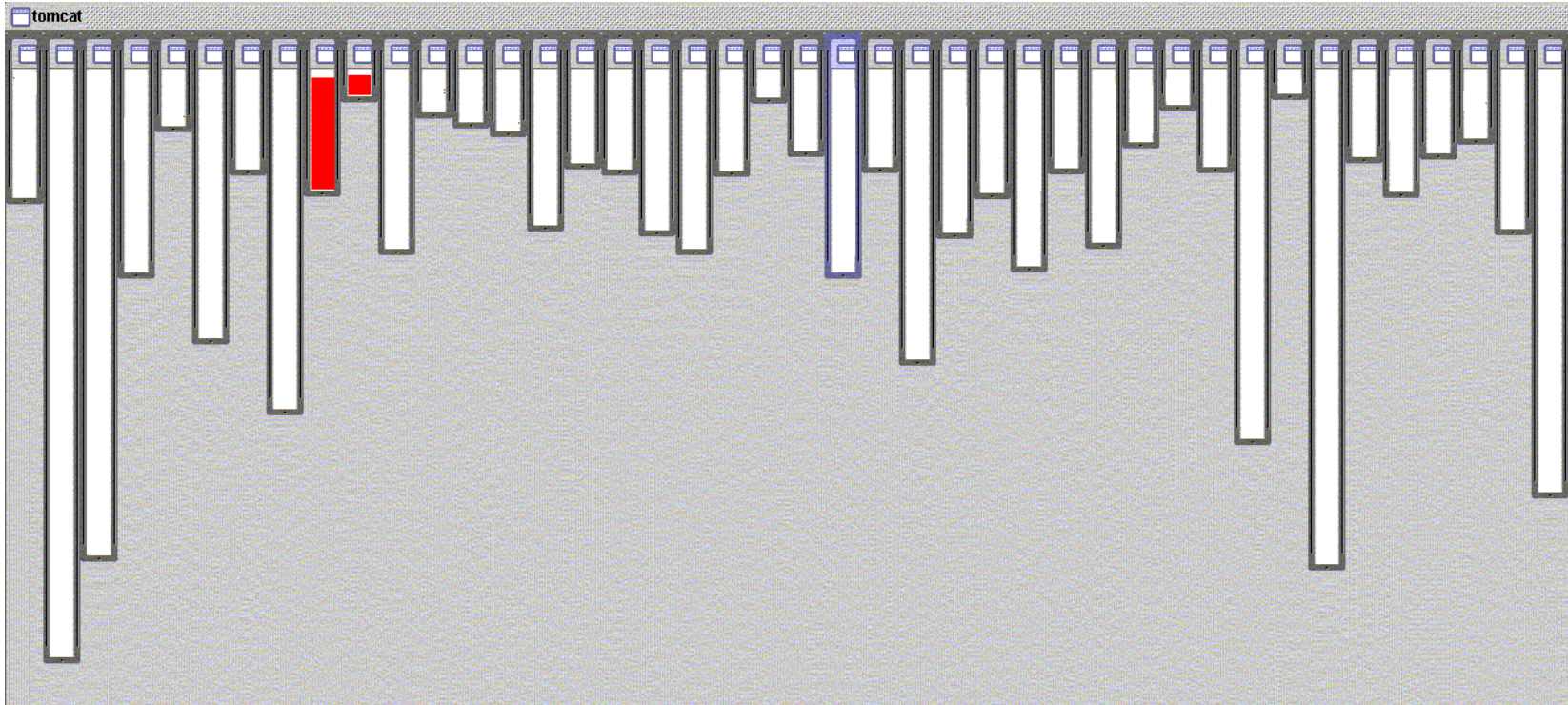
XML parsing



- **XML parsing in org.apache.tomcat**
 - red shows relevant lines of code
 - nicely fits in one box

good modularity

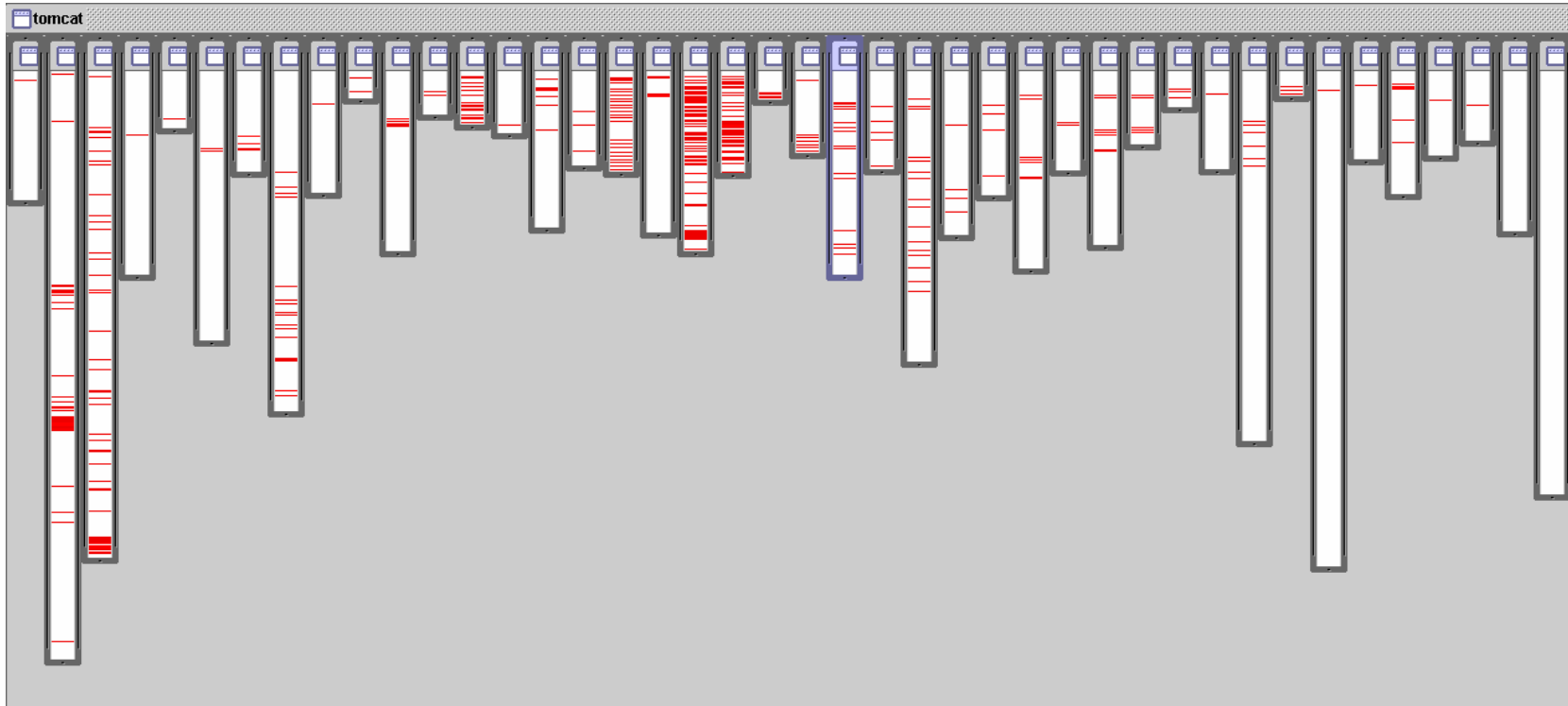
URL pattern matching



- **URL pattern matching in org.apache.tomcat**
 - red shows relevant lines of code
 - nicely fits in two boxes (using inheritance)

modularity problems

logging is not modularized



- **where is logging in org.apache.tomcat**
 - red shows lines of code that handle logging
 - not in just one place
 - not even in a small number of places

modularity problems

session expiration is not modularized

ApplicationSession

[illegible]

StandardSession

[illegible]

ServerSession

[illegible]

SessionInterceptor

[illegible]

StandardManager

[illegible]

StandardSessionManager

```

// Update the log file with the new status
void LogUpdate(const string& status) {
    // Open the log file for writing
    ofstream logFile("log.txt", ios::app);
    if (!logFile.is_open()) {
        cout << "Error: Unable to open log file." << endl;
        return;
    }
    // Write the status to the log file
    logFile << status << endl;
    logFile.close();
}

// Main function
int main() {
    // Create a new instance of the class
    MyClass obj;
    // Call the method to update the log file
    obj.LogUpdate("New status");
    return 0;
}

```

1. The code defines a class `MyClass` with a public method `LogUpdate` that takes a string parameter `status`.

2. The `LogUpdate` method opens a file named `log.txt` in append mode (`ios::app`) using an `ofstream` object named `logFile`.

3. It checks if the file was successfully opened. If not, it prints an error message and returns.

4. If the file is open, it writes the `status` string to the file followed by a newline character (`<< endl`).

5. Finally, it closes the file (`logFile.close()`).

6. The `main` function creates an instance of `MyClass` named `obj` and calls the `LogUpdate` method with the string `"New status"`.

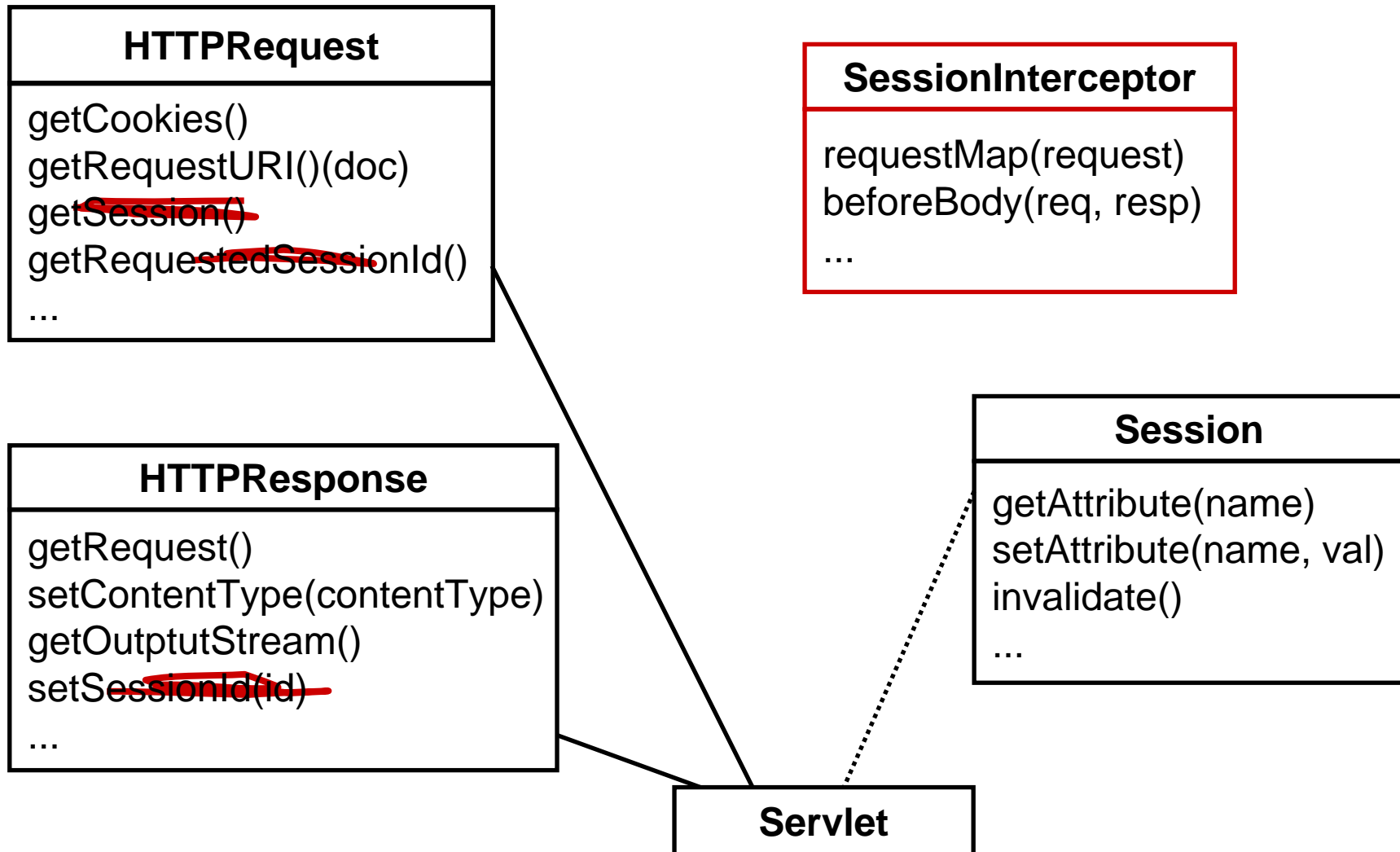
7. The program returns 0, indicating successful execution.

ServerSessionManager

[illegible]

modularity problems

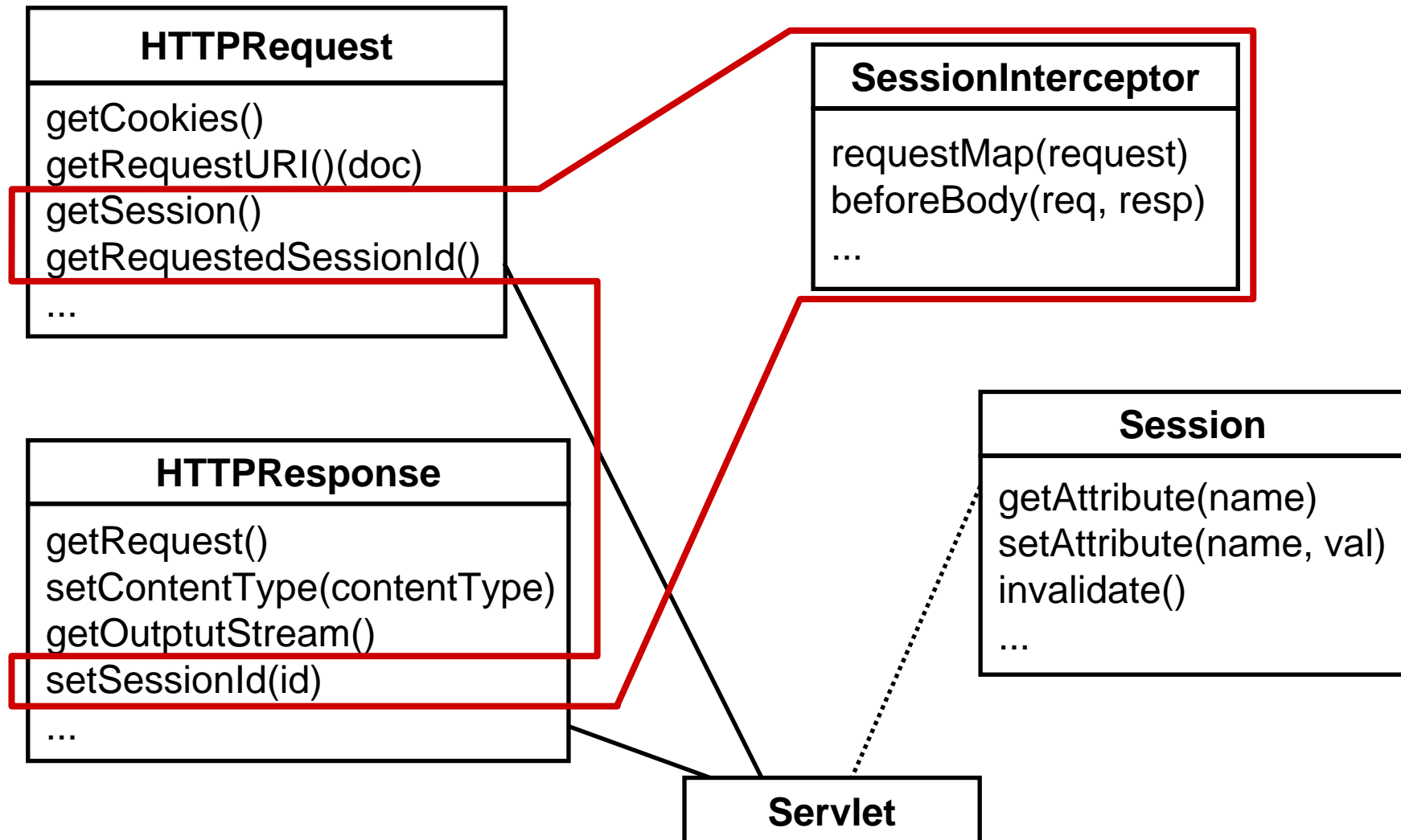
session tracking is not modularized



the cost of tangled code

- **redundant code**
 - same fragment of code in many places
- **difficult to reason about**
 - non-explicit structure
 - the big picture of the tangling isn't clear
- **difficult to change**
 - have to find all the code involved
 - and be sure to change it consistently
 - and be sure not to break it by accident

crosscutting concerns

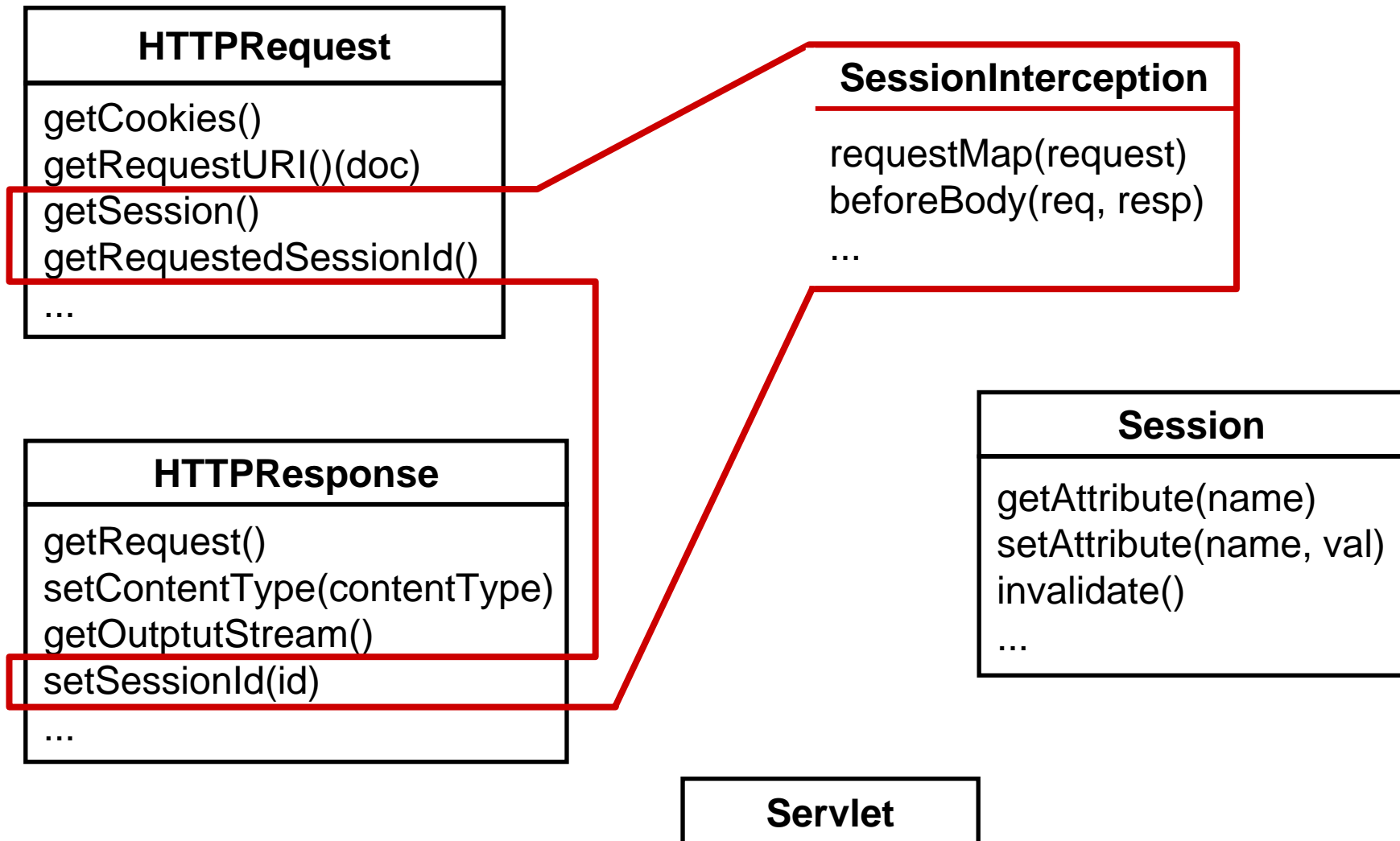


the AOP idea

aspect-oriented programming

- **crosscutting is inherent in complex systems**
- **crosscutting concerns**
 - have a clear purpose
 - have a natural structure
 - defined set of methods, module boundary crossings, points of resource utilization, lines of dataflow...
- **so, let's capture the structure of crosscutting concerns explicitly...**
 - in a modular way
 - with linguistic and tool support
- **aspects are**
 - well-modularized crosscutting concerns

modular aspects

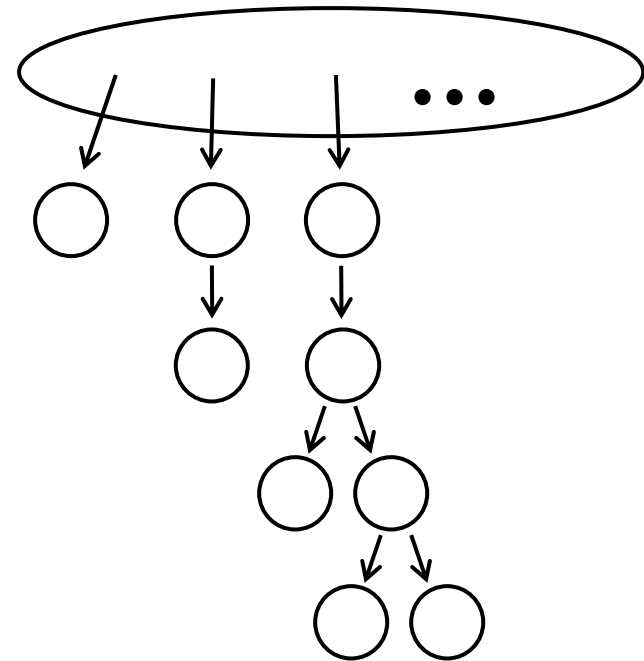


talk contents

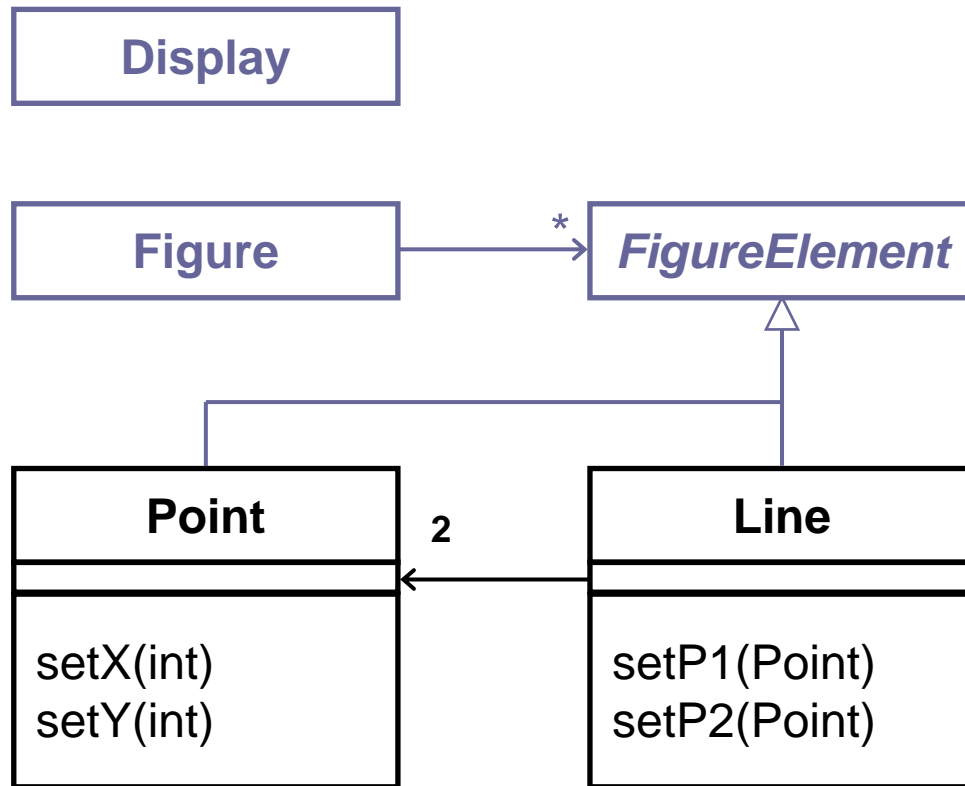
- **aspect-oriented programming (AOP)**
 - crosscutting concerns and aspects
- **AspectJ™**
 - simple AOP extension to the Java programming language
- **how AspectJ works to solve**
 - difficult program modularity problems
 - structure, code “tangling”, reusability
 - program development problems
 - design, coding, debugging
- **a technical talk**
 - will show code, language details, demo tools

problem 1 – change tracking

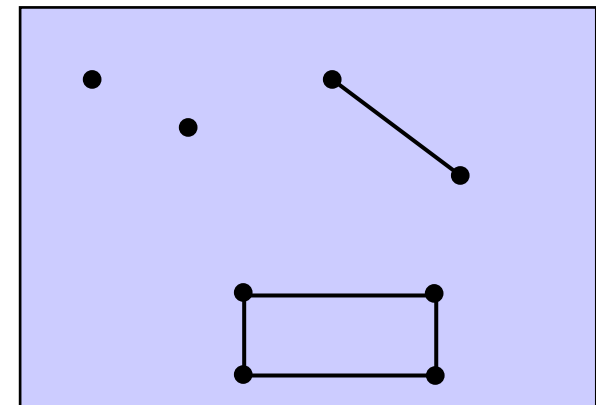
- **collection of objects**
 - that change periodically
 - must monitor changes to maintain centralized state
 - collection can be complex
 - hierarchical
 - asynchronous events
- **for example**
 - session liveness
 - image redisplay
 - value caching



a simple figure editor



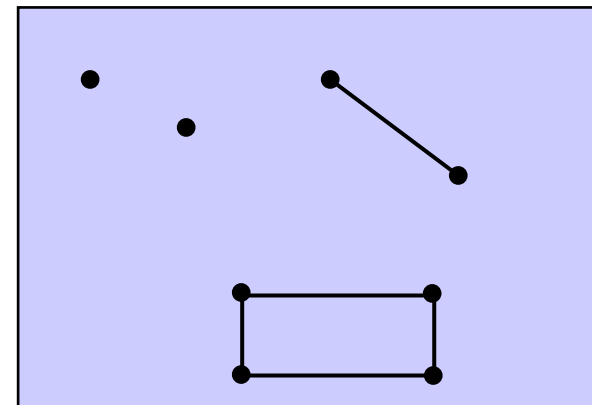
display must be
updated when
objects move



a simple figure editor

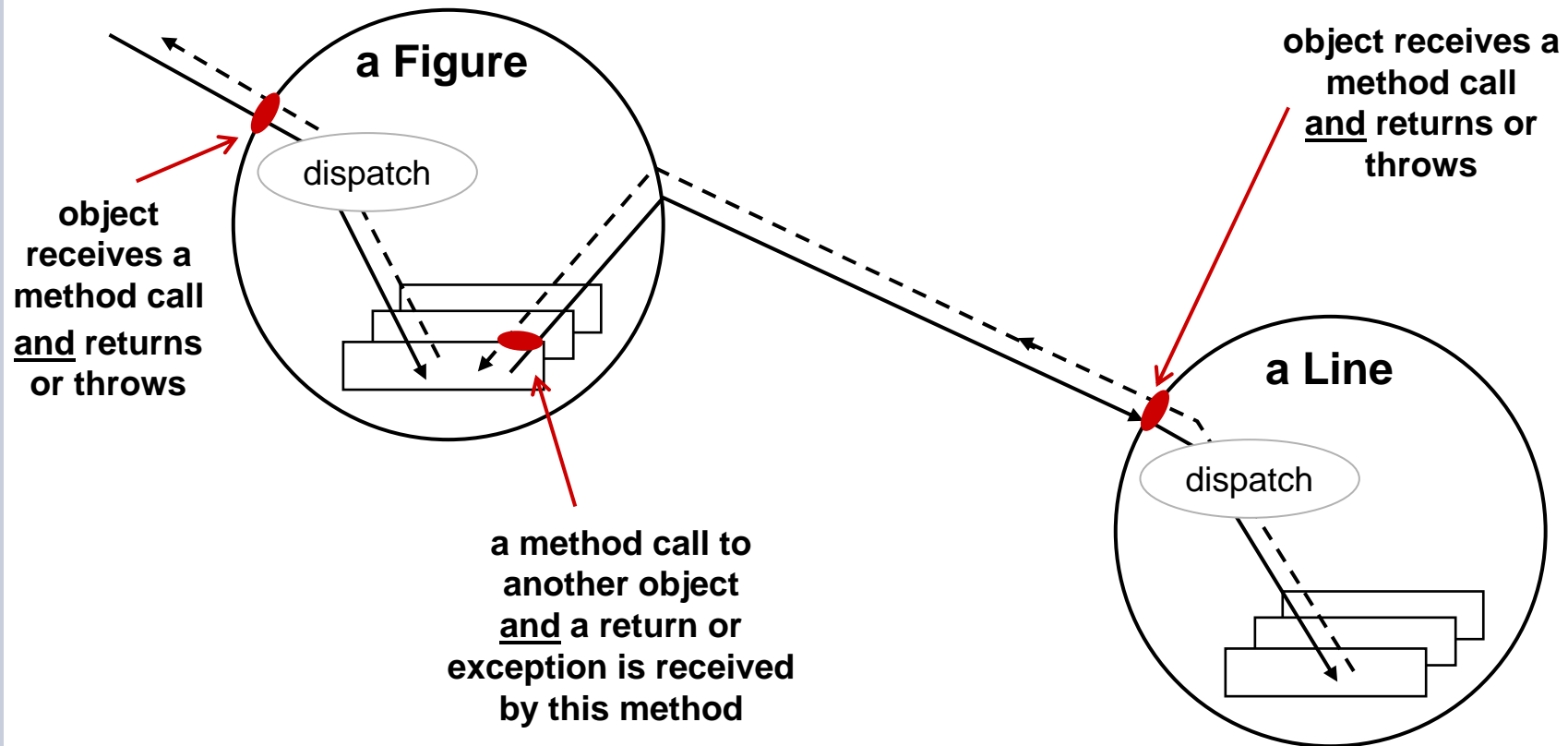
```
class Line {  
    private Point _p1, _p2;  
  
    Point getP1() { return _p1; }  
    Point getP2() { return _p2; }  
  
    void setP1(Point p1) { _p1 = p1; }  
    void setP2(Point p2) { _p2 = p2; }  
}  
  
class Point {  
    private int _x = 0, _y = 0;  
  
    int getX() { return _x; }  
    int getY() { return _y; }  
  
    void setX(int x) { _x = x; }  
    void setY(int y) { _y = y; }  
}
```

display must be
updated when
objects move



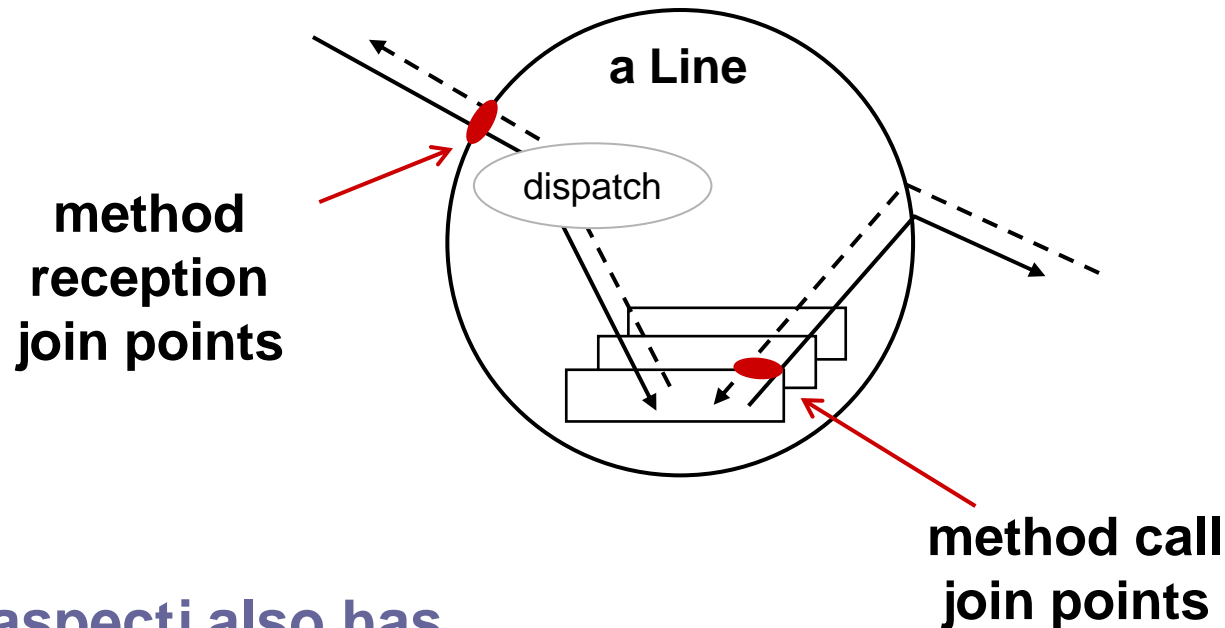
join points

key points in dynamic call graph



join point terminology

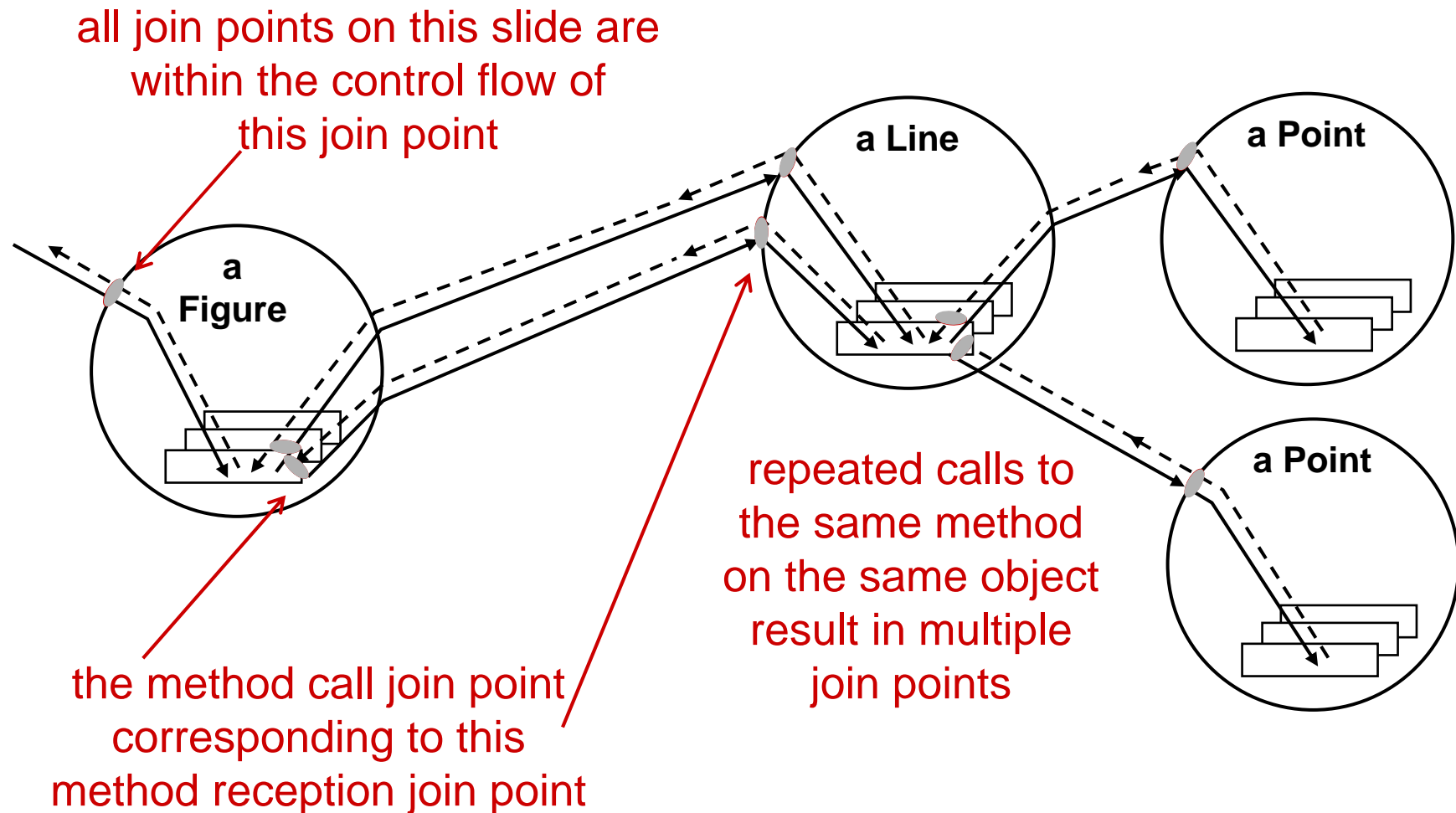
key points in dynamic call graph



- **aspectj also has**
 - method execution join points
 - constructor call reception join points
 - constructor execution join points
 - field access join points
 - exception handler execution join points

join point terminology

key points in dynamic call graph



the pointcut construct

names certain join points

each time a Line receives void setP1(Point)” or
“void setP2(Point)” method calls

name and parameters

reception of a “void Line.setP1(Point)” call

`pointcut` moves():

`call(void Line.setP1(Point)) ||`

`call(void Line.setP2(Point));`

or

reception of a “void Line.setP2(Point)” call

pointcut designators

user-defined pointcut designator

```
pointcut moves():  
    call(void Line.setP1(Point)) ||  
    call(void Line.setP2(Point));
```

primitive pointcut designator
also:

- calls, executions
- gets, sets
- handles
- instanceof,
- within, withincode
- cflow

after advice

action to take after
computation under join points

```
pointcut moves():  
    call(void Line.setP1(Point)) ||  
    call(void Line.setP2(Point));
```

```
after(): moves() {  
    <runs after moves>  
}
```

a simple aspect

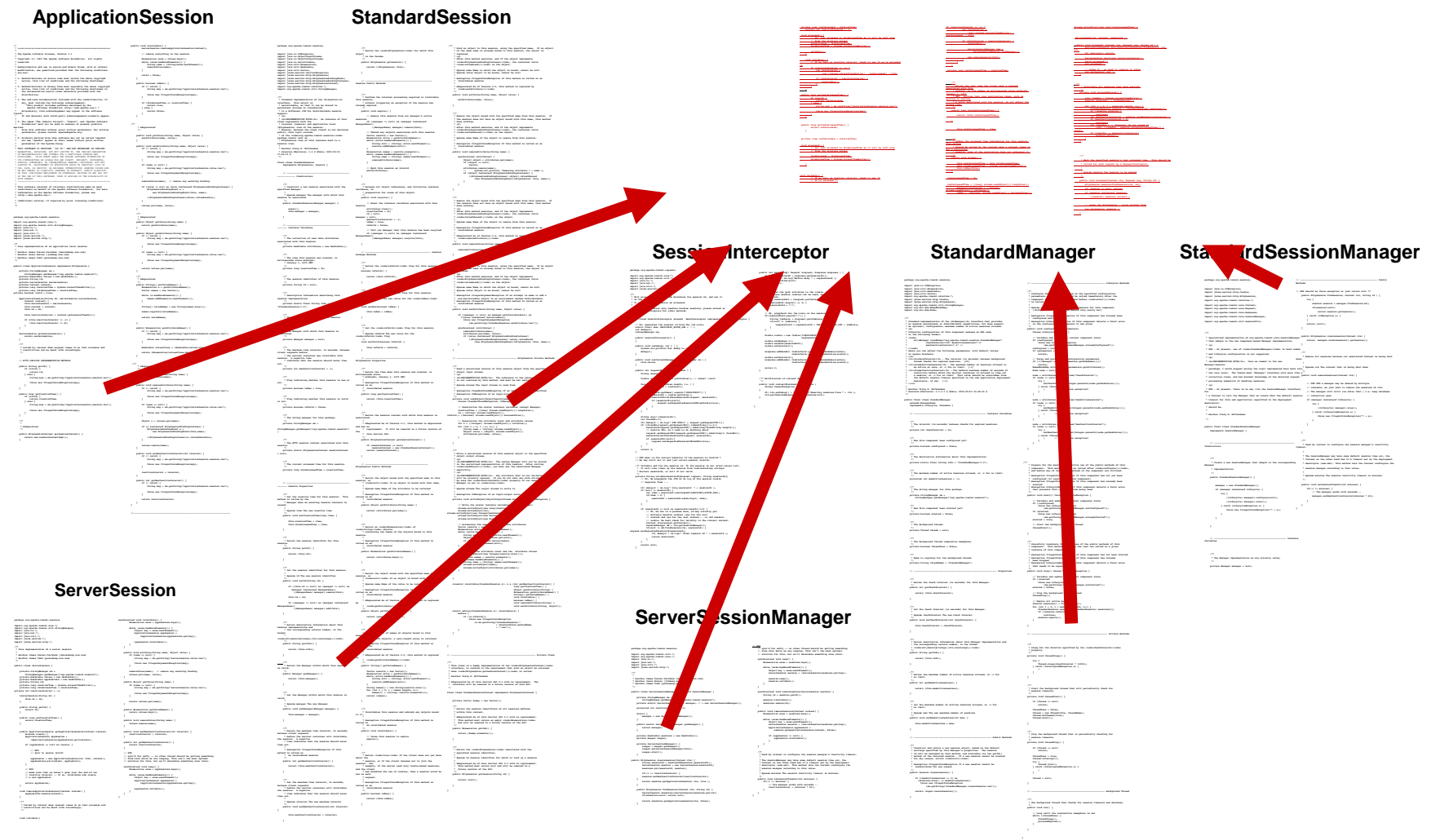
MoveTracking v1

```
aspect TrackMoves {  
    private static boolean _flag = false;  
    public static boolean testAndClear() {  
        boolean result = _flag;  
        _flag = false;  
        return result;  
    }  
  
    pointcut moves():  
        call(void Line.setP1(Point)) ||  
        call(void Line.setP2(Point));  
  
    after(): moves() {  
        _flag = true;  
    }  
}
```

← aspect defines a
special class that can
crosscut other classes

box means complete running code ↗

language support to...



language support to...

ApplicationSession

[illegible][illegible]

StandardSession

[illegible]

```

1  # Create the independent variable, the adult size
2  # in the female
3  #
4  #
5  #
6  #
7  #
8  #
9  #
10 #
11 #
12 #
13 #
14 #
15 #
16 #
17 #
18 #
19 #
20 #
21 #
22 #
23 #
24 #
25 #
26 #
27 #
28 #
29 #
30 #
31 #
32 #
33 #
34 #
35 #
36 #
37 #
38 #
39 #
40 #
41 #
42 #
43 #
44 #
45 #
46 #
47 #
48 #
49 #
50 #
51 #
52 #
53 #
54 #
55 #
56 #
57 #
58 #
59 #
60 #
61 #
62 #
63 #
64 #
65 #
66 #
67 #
68 #
69 #
70 #
71 #
72 #
73 #
74 #
75 #
76 #
77 #
78 #
79 #
80 #
81 #
82 #
83 #
84 #
85 #
86 #
87 #
88 #
89 #
90 #
91 #
92 #
93 #
94 #
95 #
96 #
97 #
98 #
99 #
100 #
101 #
102 #
103 #
104 #
105 #
106 #
107 #
108 #
109 #
110 #
111 #
112 #
113 #
114 #
115 #
116 #
117 #
118 #
119 #
120 #
121 #
122 #
123 #
124 #
125 #
126 #
127 #
128 #
129 #
130 #
131 #
132 #
133 #
134 #
135 #
136 #
137 #
138 #
139 #
140 #
141 #
142 #
143 #
144 #
145 #
146 #
147 #
148 #
149 #
150 #
151 #
152 #
153 #
154 #
155 #
156 #
157 #
158 #
159 #
160 #
161 #
162 #
163 #
164 #
165 #
166 #
167 #
168 #
169 #
170 #
171 #
172 #
173 #
174 #
175 #
176 #
177 #
178 #
179 #
180 #
181 #
182 #
183 #
184 #
185 #
186 #
187 #
188 #
189 #
190 #
191 #
192 #
193 #
194 #
195 #
196 #
197 #
198 #
199 #
200 #
201 #
202 #
203 #
204 #
205 #
206 #
207 #
208 #
209 #
210 #
211 #
212 #
213 #
214 #
215 #
216 #
217 #
218 #
219 #
220 #
221 #
222 #
223 #
224 #
225 #
226 #
227 #
228 #
229 #
230 #
231 #
232 #
233 #
234 #
235 #
236 #
237 #
238 #
239 #
240 #
241 #
242 #
243 #
244 #
245 #
246 #
247 #
248 #
249 #
250 #
251 #
252 #
253 #
254 #
255 #
256 #
257 #
258 #
259 #
260 #
261 #
262 #
263 #
264 #
265 #
266 #
267 #
268 #
269 #
270 #
271 #
272 #
273 #
274 #
275 #
276 #
277 #
278 #
279 #
280 #
281 #
282 #
283 #
284 #
285 #
286 #
287 #
288 #
289 #
290 #
291 #
292 #
293 #
294 #
295 #
296 #
297 #
298 #
299 #
300 #
301 #
302 #
303 #
304 #
305 #
306 #
307 #
308 #
309 #
310 #
311 #
312 #
313 #
314 #
315 #
316 #
317 #
318 #
319 #
320 #
321 #
322 #
323 #
324 #
325 #
326 #
327 #
328 #
329 #
330 #
331 #
332 #
333 #
334 #
335 #
336 #
337 #
338 #
339 #
340 #
341 #
342 #
343 #
344 #
345 #
346 #
347 #
348 #
349 #
350 #
351 #
352 #
353 #
354 #
355 #
356 #
357 #
358 #
359 #
360 #
361 #
362 #
363 #
364 #
365 #
366 #
367 #
368 #
369 #
370 #
371 #
372 #
373 #
374 #
375 #
376 #
377 #
378 #
379 #
380 #
381 #
382 #
383 #
384 #
385 #
386 #
387 #
388 #
389 #
390 #
391 #
392 #
393 #
394 #
395 #
396 #
397 #
398 #
399 #
400 #
401 #
402 #
403 #
404 #
405 #
406 #
407 #
408 #
409 #
410 #
411 #
412 #
413 #
414 #
415 #
416 #
417 #
418 #
419 #
420 #
421 #
422 #
423 #
424 #
425 #
426 #
427 #
428 #
429 #
430 #
431 #
432 #
433 #
434 #
435 #
436 #
437 #
438 #
439 #
440 #
441 #
442 #
443 #
444 #
445 #
446 #
447 #
448 #
449 #
450 #
451 #
452 #
453 #
454 #
455 #
456 #
457 #
458 #
459 #
460 #
461 #
462 #
463 #
464 #
465 #
466 #
467 #
468 #
469 #
470 #
471 #
472 #
473 #
474 #
475 #
476 #
477 #
478 #
479 #
480 #
481 #
482 #
483 #
484 #
485 #
486 #
487 #
488 #
489 #
490 #
491 #
492 #
493 #
494 #
495 #
496 #
497 #
498 #
499 #
500 #
501 #
502 #
503 #
504 #
505 #
506 #
507 #
508 #
509 #
510 #
511 #
512 #
513 #
514 #
515 #
516 #
517 #
518 #
519 #
520 #
521 #
522 #
523 #
524 #
525 #
526 #
527 #
528 #
529 #
530 #
531 #
532 #
533 #
534 #
535 #
536 #
537 #
538 #
539 #
540 #
541 #
542 #
543 #
544 #
545 #
546 #
547 #
548 #
549 #
550 #
551 #
552 #
553 #
554 #
555 #
556 #
557 #
558 #
559 #
560 #
561 #
562 #
563 #
564 #
565 #
566 #
567 #
568 #
569 #
570 #
571 #
572 #
573 #
574 #
575 #
576 #
577 #
578 #
579 #
580 #
581 #
582 #
583 #
584 #
585 #
586 #
587 #
588 #
589 #
590 #
591 #
592 #
593 #
594 #
595 #
596 #
597 #
598 #
599 #
600 #
601 #
602 #
603 #
604 #
605 #
606 #
607 #
608 #
609 #
610 #
611 #
612 #
613 #
614 #
615 #
616 #
617 #
618 #
619 #
620 #
621 #
622 #
623 #
624 #
625 #
626 #
627 #
628 #
629 #
630 #
631 #
632 #
633 #
634 #
635 #
636 #
637 #
638 #
639 #
640 #
641 #
642 #
643 #
644 #
645 #
646 #
647 #
648 #
649 #
650 #
651 #
652 #
653 #
654 #
655 #
656 #
657 #
658 #
659 #
660 #
661 #
662 #
663 #
664 #
665 #
666 #
667 #
668 #
669 #
670 #
671 #
672 #
673 #
674 #
675 #
676 #
677 #
678 #
679 #
680 #
681 #
682 #
683 #
684 #
685 #
686 #
687 #
688 #
689 #
690 #
691 #
692 #
693 #
694 #
695 #
696 #
697 #
698 #
699 #
700 #
701 #
702 #
703 #
704 #
705 #
706 #
707 #
708 #
709 #
710 #
711 #
712 #
713 #
714 #
715 #
716 #
717 #
718 #
719 #
720 #
721 #
722 #
723 #
724 #
725 #
726 #
727 #
728 #
729 #
730 #
731 #
732 #
733 #
734 #
735 #
736 #
737 #
738 #
739 #
740 #
741 #
742 #
743 #
744 #
745 #
746 #
747 #
748 #
749 #
750 #
751 #
752 #
753 #
754 #
755 #
756 #
757 #
758 #
759 #
760 #
761 #
762 #
763 #
764 #
765 #
766 #
767 #
768 #
769 #
770 #
771 #
772 #
773 #
774 #
775 #
776 #
777 #
778 #
779 #
780 #
781 #
782 #
783 #
784 #
785 #
786 #
787 #
788 #
789 #
790 #
791 #
792 #
793 #
794 #
795 #
796 #
797 #
798 #
799 #
800 #
801 #
802 #
803 #
804 #
805 #
806 #
807 #
808 #
809 #
810 #
811 #
812 #
813 #
814 #
815 #
816 #
817 #
818 #
819 #
820 #
821 #
822 #
823 #
824 #
825 #
826 #
827 #
828 #
829 #
830 #
831 #
832 #
833 #
834 #
835 #
836 #
8
```

[illegible]

SessionInterceptor

[illegible][illegible]

StandardManager

[illegible][illegible]

StandardSessionManager

[illegible]

ServerSession

[illegible]

```

1  # Import the necessary modules
2  import pandas as pd
3  import numpy as np
4  from sklearn.preprocessing import StandardScaler
5  from sklearn.model_selection import train_test_split
6  from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
7  from sklearn.svm import SVC
8  from sklearn.ensemble import RandomForestClassifier
9  from sklearn.neural_network import MLPClassifier
10
11 # Load the dataset
12 data = pd.read_csv('data.csv')
13
14 # Split the data into training and testing sets
15 X_train, X_test, y_train, y_test = train_test_split(data, data['category'],
16                                                    test_size=0.2,
17                                                    random_state=42)
18
19 # Standardize the features
20 scaler = StandardScaler()
21 X_train = scaler.fit_transform(X_train)
22 X_test = scaler.transform(X_test)
23
24 # Train the models
25 svm = SVC(kernel='rbf')
26 svm.fit(X_train, y_train)
27
28 rf = RandomForestClassifier(n_estimators=100)
29 rf.fit(X_train, y_train)
30
31 mlp = MLPClassifier(hidden_layer_sizes=(100, 100))
32 mlp.fit(X_train, y_train)
33
34 # Evaluate the models
35 svm_pred = svm.predict(X_test)
36 rf_pred = rf.predict(X_test)
37 mlp_pred = mlp.predict(X_test)
38
39 # Print the accuracy scores
40 print('SVM Accuracy: %.2f' % accuracy_score(y_test, svm_pred))
41 print('RF Accuracy: %.2f' % accuracy_score(y_test, rf_pred))
42 print('MLP Accuracy: %.2f' % accuracy_score(y_test, mlp_pred))
43
44 # Print the confusion matrices and classification reports
45 print('SVM Confusion Matrix: \n', confusion_matrix(y_test, svm_pred))
46 print('SVM Classification Report: \n', classification_report(y_test, svm_pred))
47 print('RF Confusion Matrix: \n', confusion_matrix(y_test, rf_pred))
48 print('RF Classification Report: \n', classification_report(y_test, rf_pred))
49 print('MLP Confusion Matrix: \n', confusion_matrix(y_test, mlp_pred))
50 print('MLP Classification Report: \n', classification_report(y_test, mlp_pred))
51
52 # Save the trained models
53 svm.save('svm_model.pkl')
54 rf.save('rf_model.pkl')
55 mlp.save('mlp_model.pkl')
56
57 # End of the script

```

[illegible][illegible][illegible]

ServerSessionManager

[illegible][illegible]

```

1  // QUESTION
2  // Given an array of integers, return the maximum value of the sum of a
3  // subarray of the array. The subarray must be non-empty.
4  // Example:
5  // Input: [-2,1,-3,4,-1,2,1,-5,4]
6  // Output: 6
7  // Explanation: The subarray [4,-1,2,1] has the maximum sum of 6.
8  // Constraints:
9  // 1 <= nums.length <= 10^5
10 // -100 <= nums[i] <= 100
11 // SOLUTION
12 // This is a classic problem that can be solved using Kadane's algorithm.
13 // The idea is to iterate through the array and keep track of the current
14 // sum of the subarray. If the current sum is negative, we reset it to 0.
15 // Otherwise, we add the current element to the sum. The maximum sum
16 // encountered during the iteration is the answer.
17 // Time complexity: O(n)
18 // Space complexity: O(1)
19 // Code
20 // C++
21 int maxSubArray(vector<int> &nums) {
22     int n = nums.size();
23     if (n == 0) return 0;
24     int maxSum = nums[0];
25     int currentSum = nums[0];
26     for (int i = 1; i < n; i++) {
27         currentSum = max(nums[i], currentSum + nums[i]);
28         maxSum = max(maxSum, currentSum);
29     }
30     return maxSum;
31 }
32 // Python
33 def maxSubArray(nums):
34     n = len(nums)
35     if n == 0:
36         return 0
37     maxSum = nums[0]
38     currentSum = nums[0]
39     for i in range(1, n):
40         currentSum = max(nums[i], currentSum + nums[i])
41         maxSum = max(maxSum, currentSum)
42     return maxSum
43 
```

[illegible]

Aspect processing (Weaving)

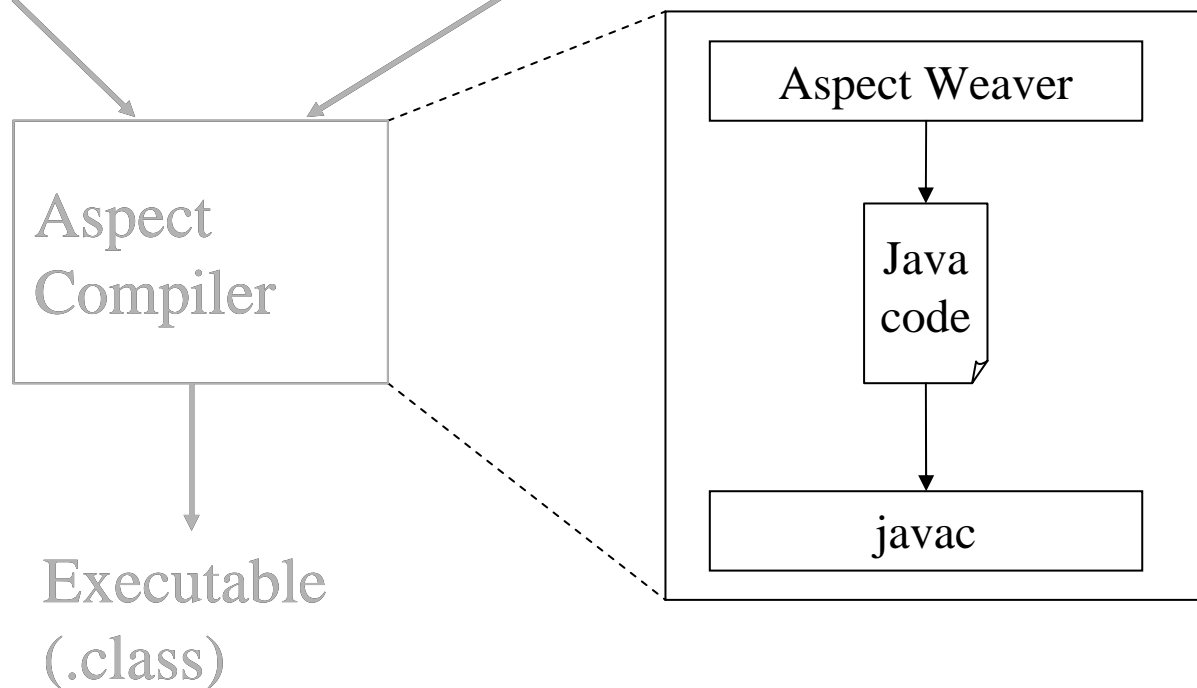
Source
(.java)

Classes

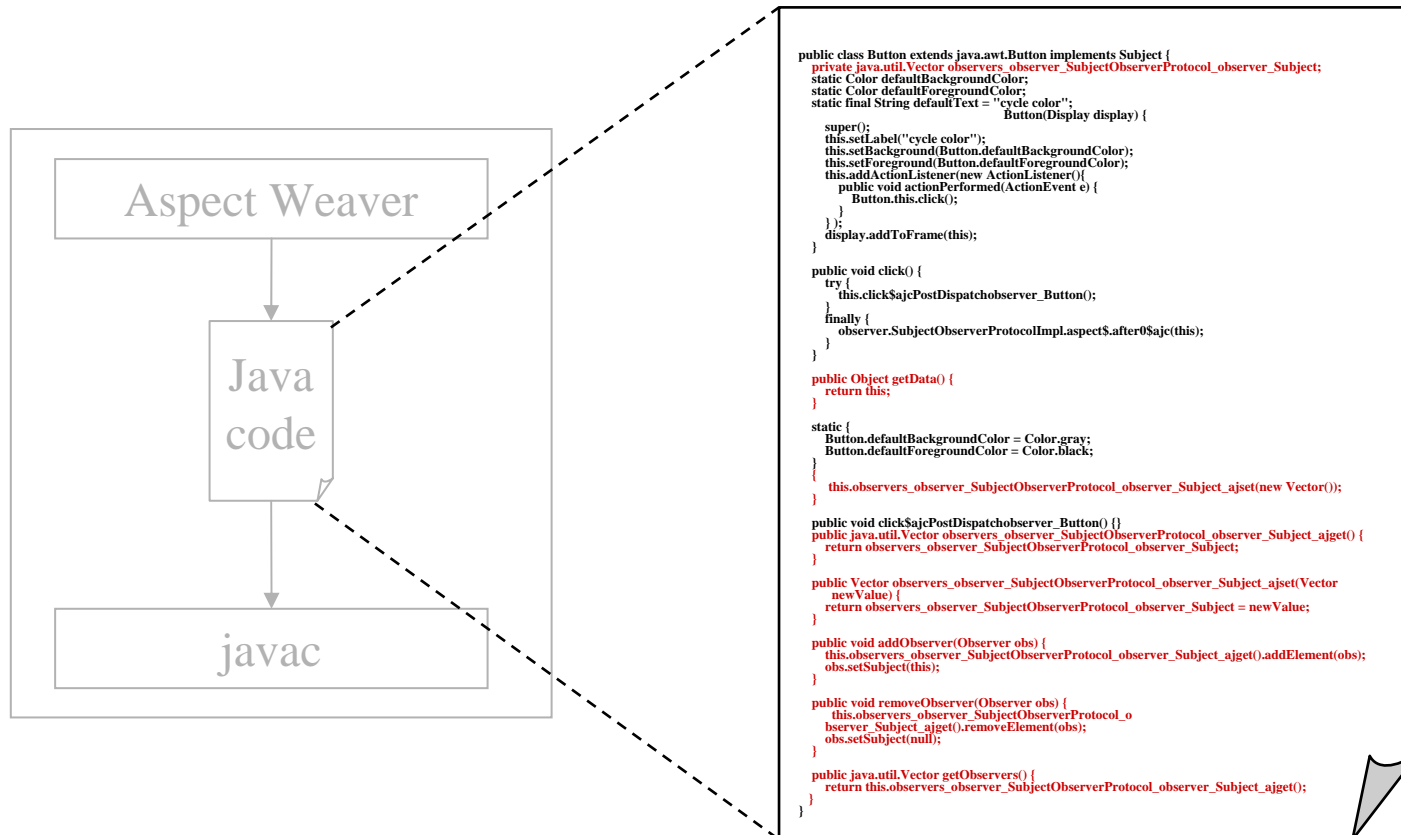
```
class Button extends JButton implements SubjectObserverProtocol.Observer {
    static final Color defaultBackgroundColor = Color.gray;
    static final Color defaultForegroundColor = Color.black;
    static final String defaultText = "click me";
    Button(Display display) {
        super();
        setDefaults();
        setVerticalTextPosition(AbstractButton.CENTER);
        setHorizontalTextPosition(AbstractButton.CENTER);
        setActionCommand("start");
        display.addToFrame(this);
    }
    void setDefaults() {
        setText(defaultText);
        setBackground(defaultBackgroundColor);
        setForeground(defaultForegroundColor);
    }
}
```

Aspects

```
class ColoredNumberAsSubject extends SubjectObserverProtocol
{
    introduction ColoredNumber {
        public Object getData () {
            Object o = null;
            try { o = thisObject.clone(); }
            catch (CloneNotSupportedException e) { System.out.println(e.toString()); }
            return o;
        }
    }
    crosscut stateChanges(): void setValue(..) | void setColor(..);
}
```



Aspect processing (Weaving)



the pointcut construct

can cut across multiple classes

```
pointcut moves():
```

```
    call(void Line.setP1(Point)) ||  
    call(void Line.setP2(Point)) ||  
    call(void Point.setX(int)) ||  
    call(void Point.setY(int));
```


a multi-class aspect


MoveTracking v2

```
aspect MoveTracking {  
    private static boolean _flag = false;  
    public static boolean testAndClear() {  
        boolean result = _flag;  
        _flag = false;  
        return result;  
    }  
  
    pointcut moves():  
        call(void Line.setP1(Point)) ||  
        call(void Line.setP2(Point)) ||  
        call(void Point.setX(int)) ||  
        call(void Point.setY(int));  
  
    after(): moves() {  
        _flag = true;  
    }  
}
```

using context in advice

- pointcut can explicitly expose certain values
- advice can use value

typed variable in place of type name



```
pointcut moves(FigureElement figElt):  
    instanceof(figElt) &&  
    (call(void Line.setP1(Point)) ||  
     call(void Line.setP2(Point)) ||  
     call(void Point.setX(int)) ||  
     call(void Point.setY(int)));
```

any join point
where “this” is
instanceof figElt



```
after(FigureElement fe): moves(fe) {  
    <fe is bound to the figure element>  
}
```

multi-class aspect with context

MoveTracking v3

```
aspect MoveTracking {
    private static Set _movees = new HashSet();
    public static Set getMovees() {
        Set result = _movees;
        _movees = new HashSet();
        return result;
    }

    pointcut moves(FigureElement figElt):
        instanceof(figElt) &&
        (call(void Line.setP1(Point)) ||
         call(void Line.setP2(Point)) ||
         call(void Point.setX(int)) ||
         call(void Point.setY(int)));

    after(FigureElement fe): moves(fe) {
        _movees.add(fe);
    }
}
```

without AspectJ

```
class Line {  
    private Point _p1, _p2;  
  
    Point getP1() { return _p1; }  
    Point getP2() { return _p2; }  
  
    void setP1(Point p1) {  
        _p1 = p1;  
    }  
    void setP2(Point p2) {  
        _p2 = p2;  
    }  
}
```

```
class Point {  
    private int _x = 0, _y = 0;  
  
    int getX() { return _x; }  
    int getY() { return _y; }  
  
    void setX(int x) {  
        _x = x;  
    }  
    void setY(int y) {  
        _y = y;  
    }  
}
```

without AspectJ

TrackMoves v1

```
class Line {
    private Point _p1, _p2;

    Point getP1() { return _p1; }
    Point getP2() { return _p2; }

    void setP1(Point p1) {
        _p1 = p1;
        TrackMoves.setFlag();
    }
    void setP2(Point p2) {
        _p2 = p2;
        TrackMoves.setFlag();
    }
}
```

```
class Point {
    private int _x = 0, _y = 0;

    int getX() { return _x; }
    int getY() { return _y; }

    void setX(int x) {
        _x = x;
    }
    void setY(int y) {
        _y = y;
    }
}
```

```
class TrackMoves {
    private static boolean _flag = false;

    public static void setFlag() {
        _flag = true;
    }

    public static boolean testAndClear() {
        boolean result = _flag;
        _flag = false;
        return result;
    }
}
```

without AspectJ

TrackMoves v2

```
class Line {
    private Point _p1, _p2;

    Point getP1() { return _p1; }
    Point getP2() { return _p2; }

    void setP1(Point p1) {
        _p1 = p1;
        TrackMoves.setFlag();
    }
    void setP2(Point p2) {
        _p2 = p2;
        TrackMoves.setFlag();
    }
}
```

```
class Point {
    private int _x = 0, _y = 0;

    int getX() { return _x; }
    int getY() { return _y; }

    void setX(int x) {
        _x = x;
        TrackMoves.setFlag();
    }
    void setY(int y) {
        _y = y;
        TrackMoves.setFlag();
    }
}
```

```
class TrackMoves {
    private static boolean _flag = false;

    public static void setFlag() {
        _flag = true;
    }

    public static boolean testAndClear() {
        boolean result = _flag;
        _flag = false;
        return result;
    }
}
```

without AspectJ

TrackMoves v3

```
class Line {
    private Point _p1, _p2;

    Point getP1() { return _p1; }
    Point getP2() { return _p2; }

    void setP1(Point p1) {
        _p1 = p1;
        TrackMoves.collectOne(this);
    }
    void setP2(Point p2) {
        _p2 = p2;
        TrackMoves.collectOne(this);
    }
}
```

```
class Point {
    private int _x = 0, _y = 0;

    int getX() { return _x; }
    int getY() { return _y; }

    void setX(int x) {
        _x = x;
        TrackMoves.collectOne(this);
    }
    void setY(int y) {
        _y = y;
        TrackMoves.collectOne(this);
    }
}
```

```
class TrackMoves {
    private static Set _movers = new HashSet();

    public void collectOne(Object o) {
        _movers.add(o);
    }

    public static Set getMovers() {
        Set result = _movers;
        _movers = new HashSet();
        return result;
    }
}
```

- **evolution is cumbersome**
 - changes in all three classes
 - have to track all callers
 - change method name
 - add argument

with AspectJ

```
class Line {  
    private Point _p1, _p2;  
  
    Point getP1() { return _p1; }  
    Point getP2() { return _p2; }  
  
    void setP1(Point p1) {  
        _p1 = p1;  
    }  
    void setP2(Point p2) {  
        _p2 = p2;  
    }  
}
```

```
class Point {  
    private int _x = 0, _y = 0;  
  
    int getX() { return _x; }  
    int getY() { return _y; }  
  
    void setX(int x) {  
        _x = x;  
    }  
    void setY(int y) {  
        _y = y;  
    }  
}
```


with AspectJ

MoveTracking v1

```
class Line {
    private Point _p1, _p2;

    Point getP1() { return _p1; }
    Point getP2() { return _p2; }

    void setP1(Point p1) {
        _p1 = p1;
    }
    void setP2(Point p2) {
        _p2 = p2;
    }
}

class Point {
    private int _x = 0, _y = 0;

    int getX() { return _x; }
    int getY() { return _y; }

    void setX(int x) {
        _x = x;
    }
    void setY(int y) {
        _y = y;
    }
}
```

```
aspect MoveTracking {
    private static boolean _flag = false;
    public static boolean testAndClear() {
        boolean result = _flag;
        _flag = false;
        return result;
    }

    pointcut moves():
        call(void Line.setP1(Point)) ||
        call(void Line.setP2(Point));

    after(): moves() {
        _flag = true;
    }
}
```

with AspectJ

MoveTracking v2

```
class Line {
    private Point _p1, _p2;

    Point getP1() { return _p1; }
    Point getP2() { return _p2; }

    void setP1(Point p1) {
        _p1 = p1;
    }
    void setP2(Point p2) {
        _p2 = p2;
    }
}
```

```
class Point {
    private int _x = 0, _y = 0;

    int getX() { return _x; }
    int getY() { return _y; }

    void setX(int x) {
        _x = x;
    }
    void setY(int y) {
        _y = y;
    }
}
```

```
aspect MoveTracking {
    private static boolean _flag = false;
    public static boolean testAndClear() {
        boolean result = _flag;
        _flag = false;
        return result;
    }

    pointcut moves():
        call(void Line.setP1(Point)) ||
        call(void Line.setP2(Point)) ||
        call(void Point.setX(int)) ||
        call(void Point.setY(int));

    after(): moves() {
        _flag = true;
    }
}
```

with AspectJ

MoveTracking v3

```
class Line {
    private Point _p1, _p2;

    Point getP1() { return _p1; }
    Point getP2() { return _p2; }

    void setP1(Point p1) {
        _p1 = p1;
    }
    void setP2(Point p2) {
        _p2 = p2;
    }
}

class Point {
    private int _x = 0, _y = 0;

    int getX() { return _x; }
    int getY() { return _y; }

    void setX(int x) {
        _x = x;
    }
    void setY(int y) {
        _y = y;
    }
}
```

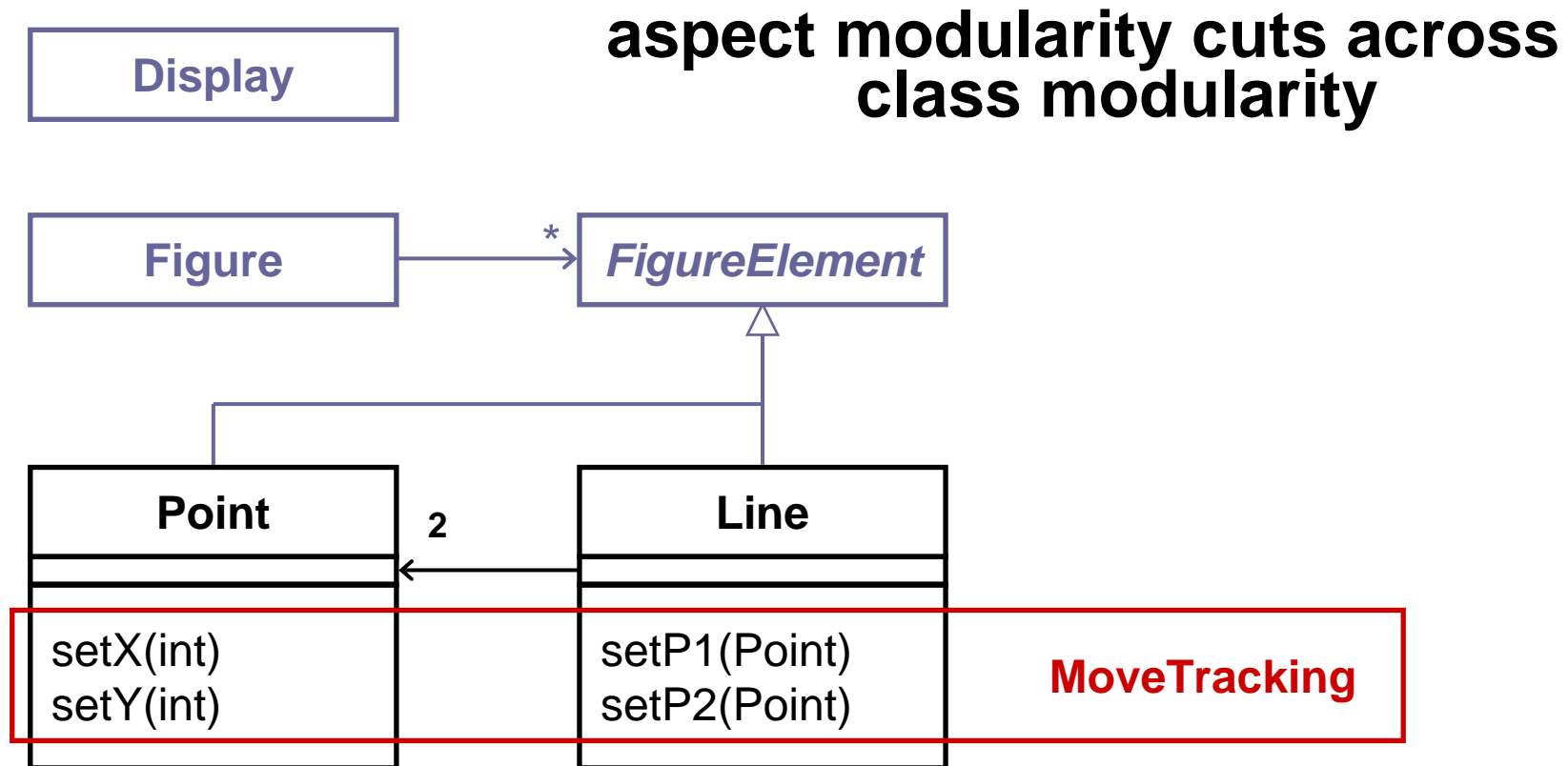
```
aspect MoveTracking {
    private static Set _movees = new HashSet();
    public static Set getmovees() {
        Set result = _movees;
        _movees = new HashSet();
        return result;
    }

    pointcut moves(FigureElement figElt):
        instanceof(figElt) &&
        (call(void Line.setP1(Point)) ||
         call(void Line.setP2(Point)) ||
         call(void Point.setX(int)) ||
         call(void Point.setY(int)));

    after(FigureElement fe): moves(fe) {
        _movees.add(fe);
    }
}
```

- evolution is more modular
 - all changes in single aspect

aspects crosscut classes



context sensitive aspects

MoveTracking v4

```
aspect MoveTracking {
    ...

    pointcut moveCalls(Object mover, FigureElement movee):
        instanceof(mover) &&
        (lineMoveCalls(movee) || pointMoveCalls(movee));

    pointcut lineMoveCalls(Line ln):
        call(void ln.setP1(Point)) || call(void ln.setP2(Point));

    pointcut pointMoveCalls(Point pt):
        call(void pt.setX(int)) || call(void pt.setY(int));

    after(Object mover, FigureElement movee):
        moveCalls(mover, movee) {
            _movers.add(mover);
            _movees.add(movee);
        }
}
```

without AspectJ

MoveTracking v4

```
class Line {
    private Point _p1, _p2;

    Point getP1() { return _p1; }
    Point getP2() { return _p2; }

    void setP1(Point p1) {
        _p1 = p1;
        MoveTracking.collectOne(this);
    }
    void setP2(Point p2) {
        _p2 = p2;
        MoveTracking.collectOne(this);
    }
}
```

```
class Point {
    private int _x = 0, _y = 0;

    int getX() { return _x; }
    int getY() { return _y; }

    void setX(int x) {
        _x = x;
        MoveTracking.collectOne(this);
    }
    void setY(int y) {
        _y = y;
        MoveTracking.collectOne(this);
    }
}
```

```
class MoveTracking {
    private static Set _movees = new HashSet();
    private static Set _movers = new HashSet();

    public static void collectMove(Object o1, Object o2) {
        _movees.add(o1);
        _movers.add(o2);
    }

    public static Set getmovees() {
        Set result = _movees;
        _movees = new HashSet();
        return result;
    }
    ...
}
```

```
class Foo {
    ...

    public void mumble(Line l) {
        MoveTracking.collectMove(this, l);
        l.setP1(new Point());
        ...
    }
    ...
}
```

without AspectJ

MoveTracking v4

```
class Line {  
    private Point _p1, _p2;
```

```
    class XYZZY {  
        ...
```

```
        public void moveit(Point p) {  
            MoveTracking.collectMove(this, p);
```

```
            p.setX  
            ...  
        }  
        ...  
    }  
}
```

```
class Bar {  
    ...
```

```
    public void mumble(Line l) {  
        MoveTracking.collectMove(this, l);  
        l.setP1(new Point());  
        ...  
    }  
    ...  
}
```

```
class Bar {  
    ...
```

```
    public void mumble(Line l) {  
        MoveTracking.collectMove(this, l);  
        l.setP1(new Point());  
        ...  
    }  
    ...  
}
```

```
    void set  
        _y = y  
        -MoveTr  
    }  
}
```

```
        MoveTracking.collectMove(this, l);  
        l.setP1(new Point());  
        ...  
    }  
    ...  
}
```

```
class MoveTracking {  
    private static Set _movees = new HashSet();  
    private static Set _movers = new HashSet();
```

```
    public static void collectMove(Object o1, Object o2) {  
        _movees.add(o1);  
        _movers.add(o2);  
    }
```

```
    static Set getmovees() {
```

```
        result = _movees;  
        s = new HashSet();  
        result;
```

```
{
```

```
    public void mumble(Line l) {  
        MoveTracking.collectMove(thi  
        new Point();
```

```
class AnotherClass {  
    ...
```

```
    public void frotz(Point p) {  
        MoveTracking.collectMove(this, p);  
        p.setY(42);  
        ...  
    }  
    ...  
}
```

```
class Foo {  
    ...
```

```
    public void mbar(Line l) {  
        MoveTracking.collectMove(this, l);  
        l.setP1(new Point());  
        ...  
    }  
    ...  
}
```

context sensitive aspects

MoveTracking v5

```
aspect MoveTracking {

    private static Set _movees = new HashSet();
    public static Set getMovees() {
        Set result = _movees;
        _movees = new HashSet();
        return result;
    }

    pointcut moves(FigureElement figElt):
        instanceof(figElt) &&
        (receptions(void Line.setP1(Point)) ||
         receptions(void Line.setP2(Point)) ||
         receptions(void Point.setX(int)) ||
         receptions(void Point.setY(int)));

    pointcut topLevelMoves(FigureElement figElt):
        moves(figElt) && !cflow(moves(figElt));

    after(FigureElement fe): topLevelMoves(fe) {
        _movees.add(fe);
    }
}
```


problem 2 – pre- and/or post-conditions

- **pre-conditions**
 - test values of parameters
 - are point coordinates legal?
 - test object and/or global state
- **post-conditions**
 - test whether operation worked properly
- **condition enforcement**
 - coerce parameters to be legal

advice is

additional action to take at join points

- **before** before proceeding at join point
- **after returning** a value to join point
- **after throwing** a throwable to join point
- **after** returning to join point either way
- **around** on arrival at join point gets explicit control over when&if program does proceed

pre-condition

using before advice

```
aspect PointBoundsPreCondition {

    before(Point p, int newX):
        receptions(void p.setX(newX)) {
        assert(newX >= MIN_X);
        assert(newX <= MAX_X);
    }

    before(Point p, int newY):
        receptions(void p.setY(newY)) {
        assert(newY >= MIN_Y);
        assert(newY <= MAX_Y);
    }

    static void assert(boolean v) {
        if ( !v )
            throw new RuntimeException();
    }
}
```

post-condition

using after advice

```
aspect PointBoundsPostCondition {

    after(Point p, int newX):
        receptions(void p.setX(newX)) {
            assert(p.getX() == newX);
        }

    after(Point p, int newY):
        receptions(void p.setY(newY)) {
            assert(p.getY() == newY);
        }

    static void assert(boolean v) {
        if ( !v )
            throw new RuntimeException();
    }
}
```

condition enforcement

using around advice

```
aspect PointBoundsEnforcement {  
  
    around(Point p, int newX) returns void:  
        receptions(void p.setX(newX)) {  
            proceed(p, clip(newX, MIN_X, MAX_X));  
        }  
  
    around(Point p, int newY) returns void:  
        receptions(void p.setY(newY)) {  
            proceed(p, clip(newY, MIN_Y, MAX_Y));  
        }  
  
    static int clip(int val, int min, int max) {  
        return Math.max(min, Math.min(max, val));  
    }  
}
```

problem 3 – debugging support

//From ContextManager

```
public void service( Request rrequest, Response rresponse ) {  
    // log( "New request " + rrequest );  
    try {  
        // System.out.print("A");  
        rrequest.setContextManager( this );  
        rrequest.setResponse(rresponse);  
        rresponse.setRequest(rrequest);  
  
        // wront request - parsing error  
        int status=rresponse.getStatus();  
  
        if( status < 400 )  
            status= processRequest( rrequest );  
  
        if(status==0)  
            status=authenticate( rrequest, rresponse );  
        if(status == 0)  
            status=authorize( rrequest, rresponse );  
        if( status == 0 ) {  
            rrequest.getWrapper().handleRequest(rrequest,  
                rresponse);  
        } else {  
            // something went wrong  
            handleError( rrequest, rresponse, null, status );  
        }  
    } catch (Throwable t) {  
        handleError( rrequest, rresponse, t, 0 );  
    }  
    // System.out.print("B");  
    try {  
        rresponse.finish();  
        rrequest.recycle();  
        rresponse.recycle();  
    } catch ( Throwable ex ) {  
        if(debug>0) log( "Error closing request " + ex);  
    }  
    // log( "Done with request " + rrequest );  
    // System.out.print("C");  
    return;  
}
```

// log("New request " + rrequest);

// System.out.print("A");

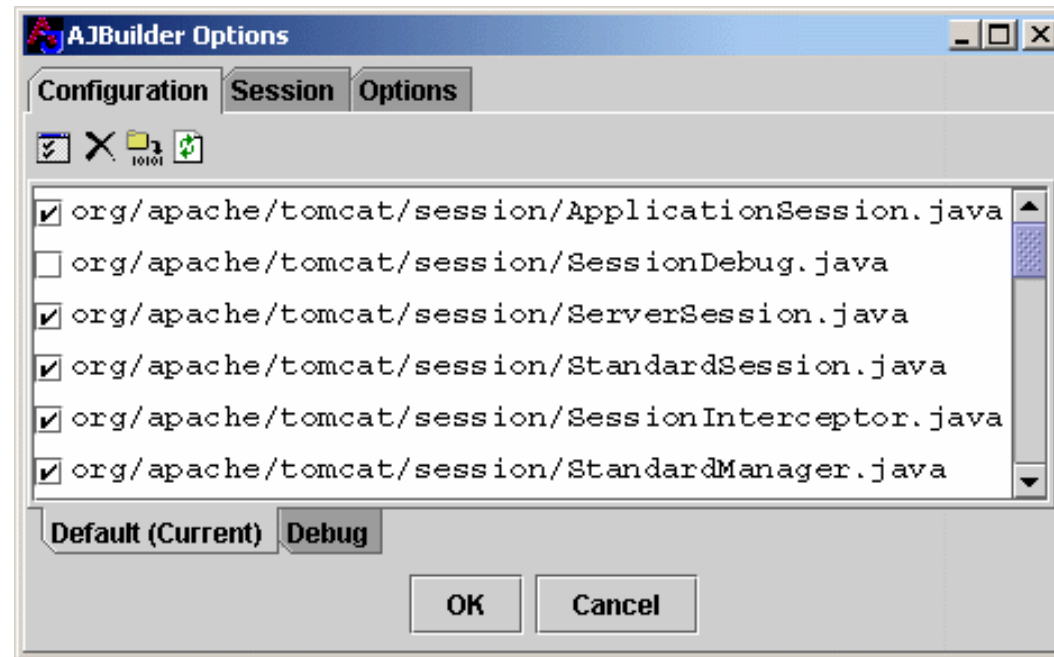
// System.out.print("B");

if(debug>0)
 log("Error closing request " + ex);

// log("Done with request " + rrequest);

// System.out.print("C");

pluggable aspects



- **plug and debug**
 - can be used for profiling, logging
 - easy to be sure it is off

property-based crosscutting

```
package my.package;  
public class C1 {  
    ...  
    public void foo() {  
        A.doSomething(...);  
    }  
    ...  
}
```

```
package my.package;  
public class C2 {  
    ...  
    public int frotz() {  
        A.doSomething(...);  
    }  
    ...  
    public int bar() {  
        A.doSomething(...);  
    }  
    ...  
}
```

```
package my.package;  
public class C3 {  
    ...  
    public String s1() {  
        A.doSomething(...);  
    }  
    ...  
}
```

- **crosscuts of methods with a common property**
 - public/private, return a certain value, in a particular package
- **logging, debugging, profiling**
 - log on entry to every public method

wildcarding in pointcuts

```
instanceof(Point)
instanceof(graphics.geom.Point)
instanceof(graphics.geom.*)
instanceof(graphics..*)
```

“*” is wild card

“..” is multi-part wild card

any type in graphics.geom
any type in any sub-package
of graphics

```
receptions(void Point.setX(int))
receptions(public * Point.*(..))
receptions(public * *.*.*(..))
```

any public method on Point
any public method on any type

```
receptions(void getX())
receptions(void getY())
receptions(void get*())
```

any getter

```
receptions(new(int, int))
receptions(new(..))
```

any constructor

property-based crosscutting

```
aspect PublicErrorLogging {  
    static Log log = new Log();  
  
    pointcut publicInterface ():  
        receptions(public * org.aspectj..*.*(..));  
  
    after() throwing (Exception e): publicInterface() {  
        log.write(e);  
    }  
}
```

neatly captures public
interface of mypackage



consider code maintenance

- **another programmer adds a public method**
 - i.e. extends public interface – this code will still work
- **another programmer reads this code**
 - “what’s really going on” is explicit
- **decision to disable logging**

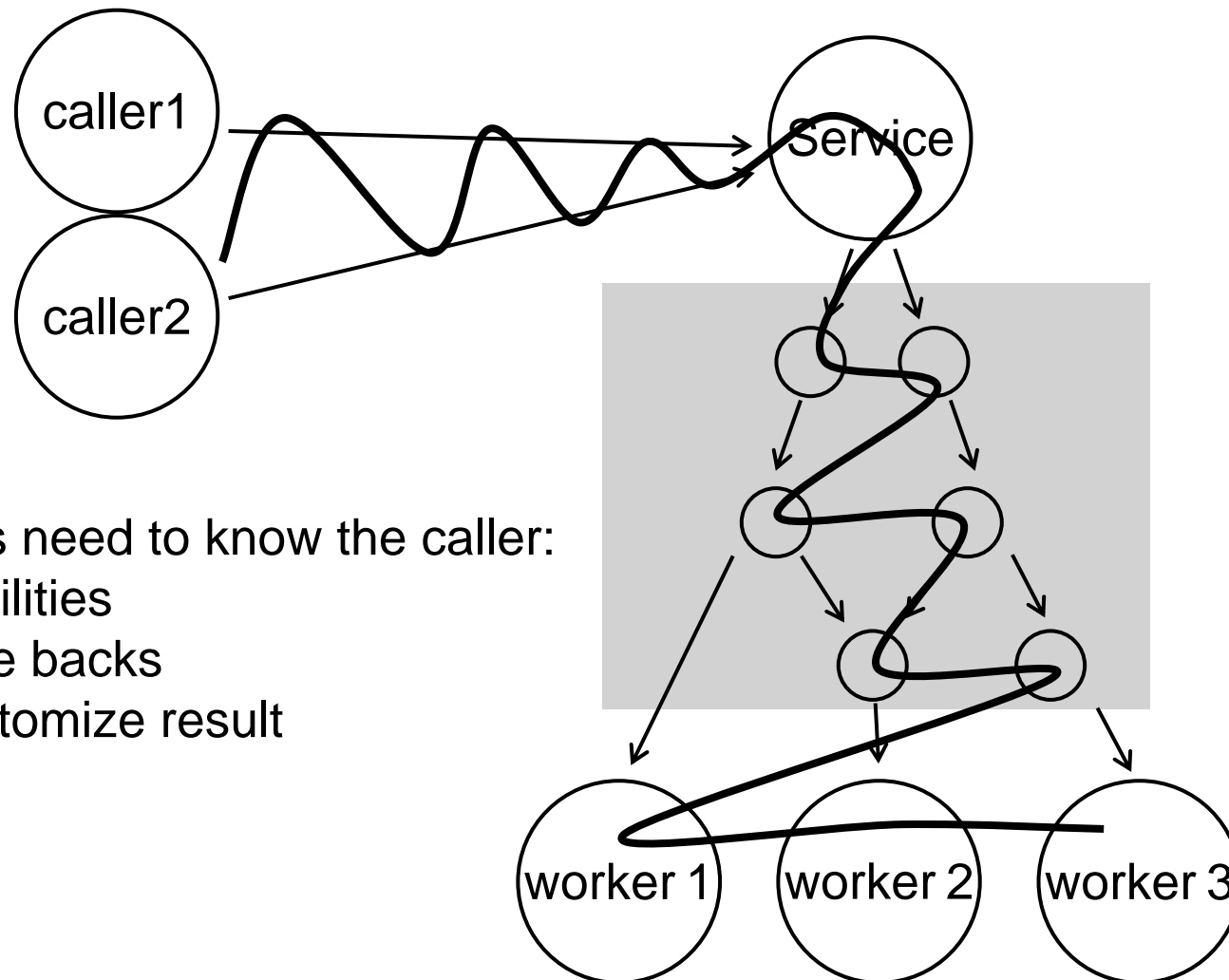
modular crosscutting

```
aspect PublicErrorLogging {  
  
    static Log log = new Log();  
  
    pointcut publicInterface ():  
        receptions(public * com.xerox..*.*(..));  
  
    after() throwing (Error e): publicInterface() {  
        log.write(e);  
    }  
}
```

- **crosscutting concerns**
 - tangled implementation → complex, difficult to maintain
 - modular implementation → can be clear, easy to maintain
- **crosscutting concerns per se are not complicated!**

example 4

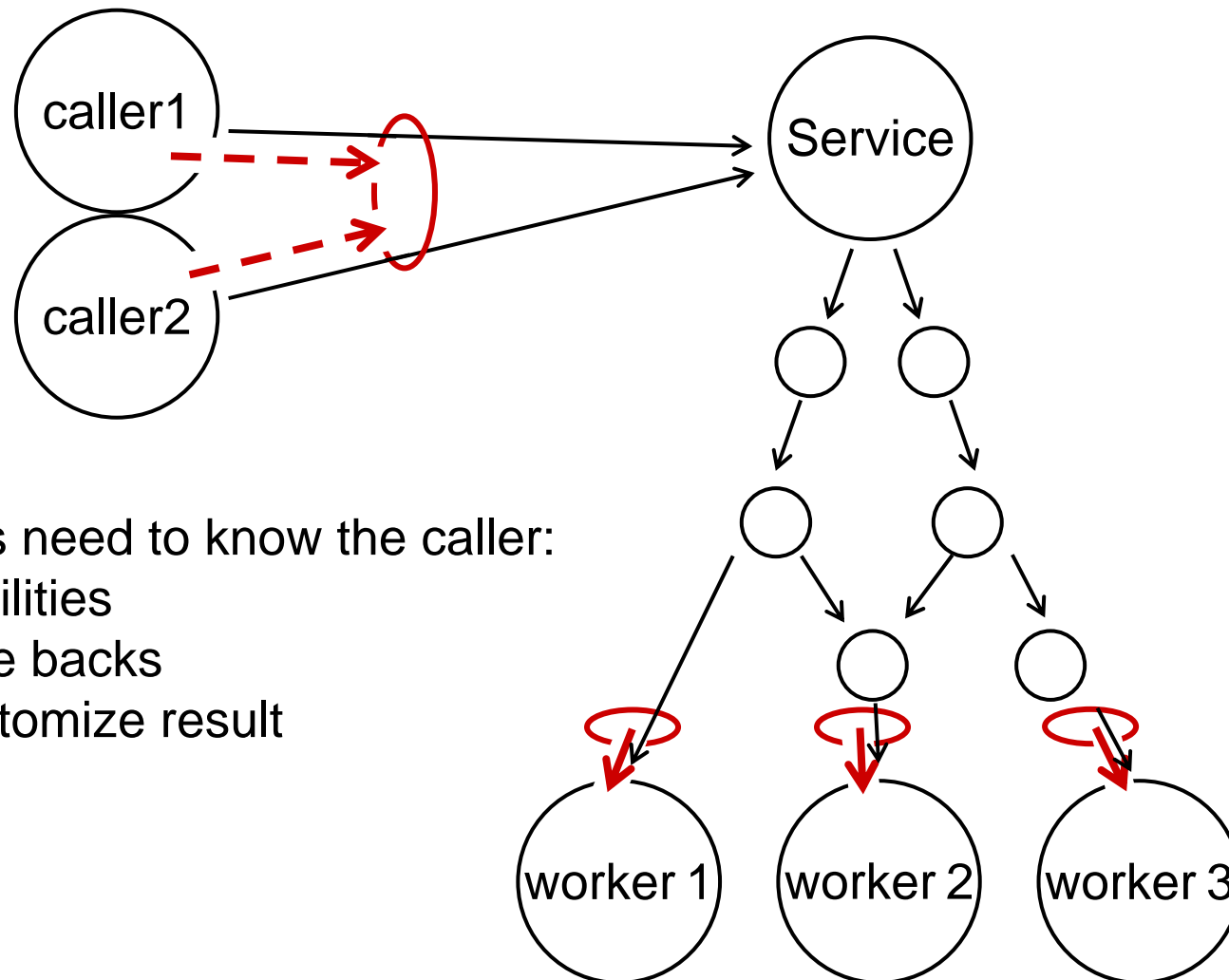
context-passing aspects



workers need to know the caller:

- capabilities
- charge backs
- to customize result

context passing aspects

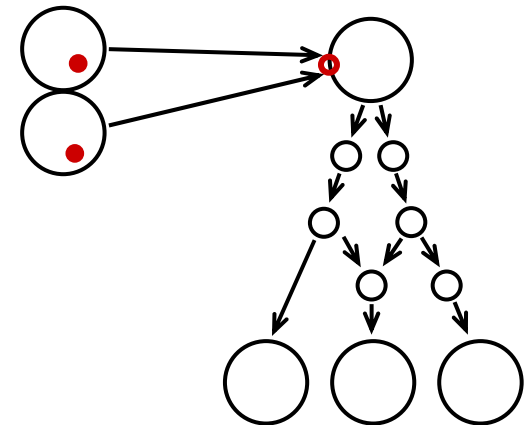


workers need to know the caller:

- capabilities
- charge backs
- to customize result

context passing aspects

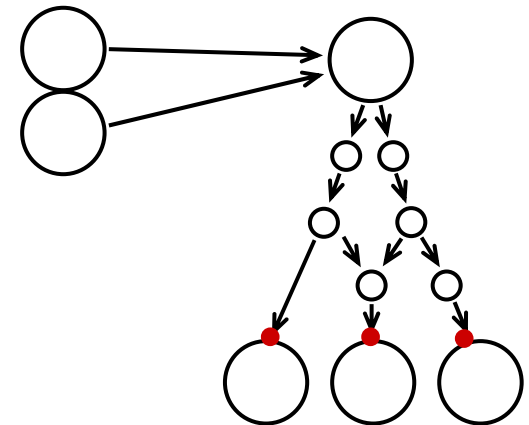
```
pointcut invocations(Caller c):  
    instanceof(c) && calls(void Service.doService(String));
```



context passing aspects

```
pointcut invocations(Caller c):
    instanceof(c) && calls(void Service.doService(String));
```

```
pointcut workPoints(Worker w):
    receptions(void w.doTask(Task));
```

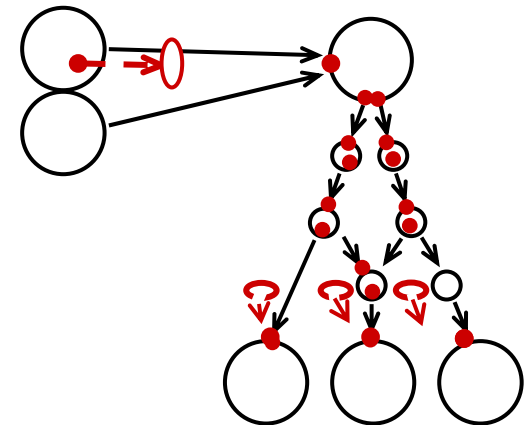


context passing aspects

```
pointcut invocations(Caller c):  
    instanceof(c) && calls(void Service.doService(String));
```

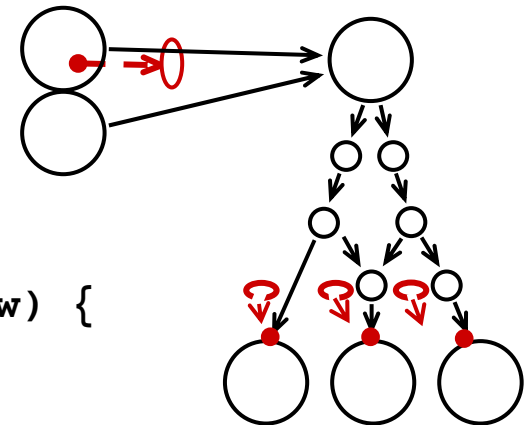
```
pointcut workPoints(Worker w):  
    receptions(void w.doTask(Task));
```

```
pointcut perCallerWork(Caller c, Worker w):  
    cflow(invocations(c)) && workPoints(w);
```



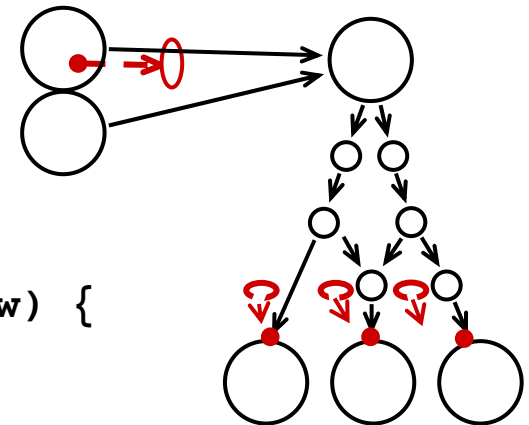
context passing aspects

```
aspect CapabilityChecking {  
  
    pointcut invocations(Caller c):  
        instanceof(c) && calls(void Service.doService(String));  
  
    pointcut workPoints(Worker w):  
        receptions(void w.doTask(Task));  
  
    pointcut perCallerWork(Caller c, Worker w):  
        cflow(invocations(c)) && workPoints(w);  
  
    before (Caller c, Worker w): perCallerWork(c, w) {  
        w.checkCapabilities(c);  
    }  
}
```



context passing aspects

```
aspect CapabilityChecking {  
  
    pointcut invocations(Caller c):  
        instanceof(c) && calls(void Service.doService(String));  
  
    pointcut workPoints(Worker w):  
        receptions(void w.doTask(Task));  
  
    pointcut perCallerWork(Caller c, Worker w):  
        cflow(invocations(c)) && workPoints(w);  
  
    before (Caller c, Worker w): perCallerWork(c, w) {  
        w.checkCapabilities(c);  
    }  
}
```



inheritance & specialization

- **pointcuts can have additional advice**
 - aspect with
 - concrete pointcut
 - perhaps no advice on the pointcut
 - in figure editor
 - moves() can have advice from multiple aspects
 - module can expose certain well-defined pointcuts
- **abstract pointcuts can be specialized**
 - aspect with
 - abstract pointcut
 - concrete advice on the abstract pointcut

a shared pointcut

```
public class FigureEditor {
    public pointcut moves(FigureElement figElt):
        instanceof(figElt) &&
        (receptions(void Line.setP1(Point)) ||
         receptions(void Line.setP2(Point)) ||
         receptions(void Point.setX(int)) ||
         receptions(void Point.setY(int)));
    ...
}

aspect TrackMoves {
    static after(FigureElement fe):
        FigureEditor.moves(fe) { ... }
    ...
}
```

a reusable exception handling aspect

```
abstract public aspect LogRemoteExceptions {  
  
    abstract pointcut rcptns(); ← abstract  
  
    static after() throwing (RemoteException e): rcptns() {  
        log.println("Remote call failed in: " +  
                    thisJoinPoint.toString() +  
                    "(" + e + ").");  
    }  
}
```

```
public aspect JWAMRemoteExceptionHandler  
    extends LogRemoteExceptions {  
    pointcut rcptns():  
        receptions(* RegistryServer.*(..)) ||  
        executions(private * RMIMessageBrokerImpl.*(..);  
}
```

common questions

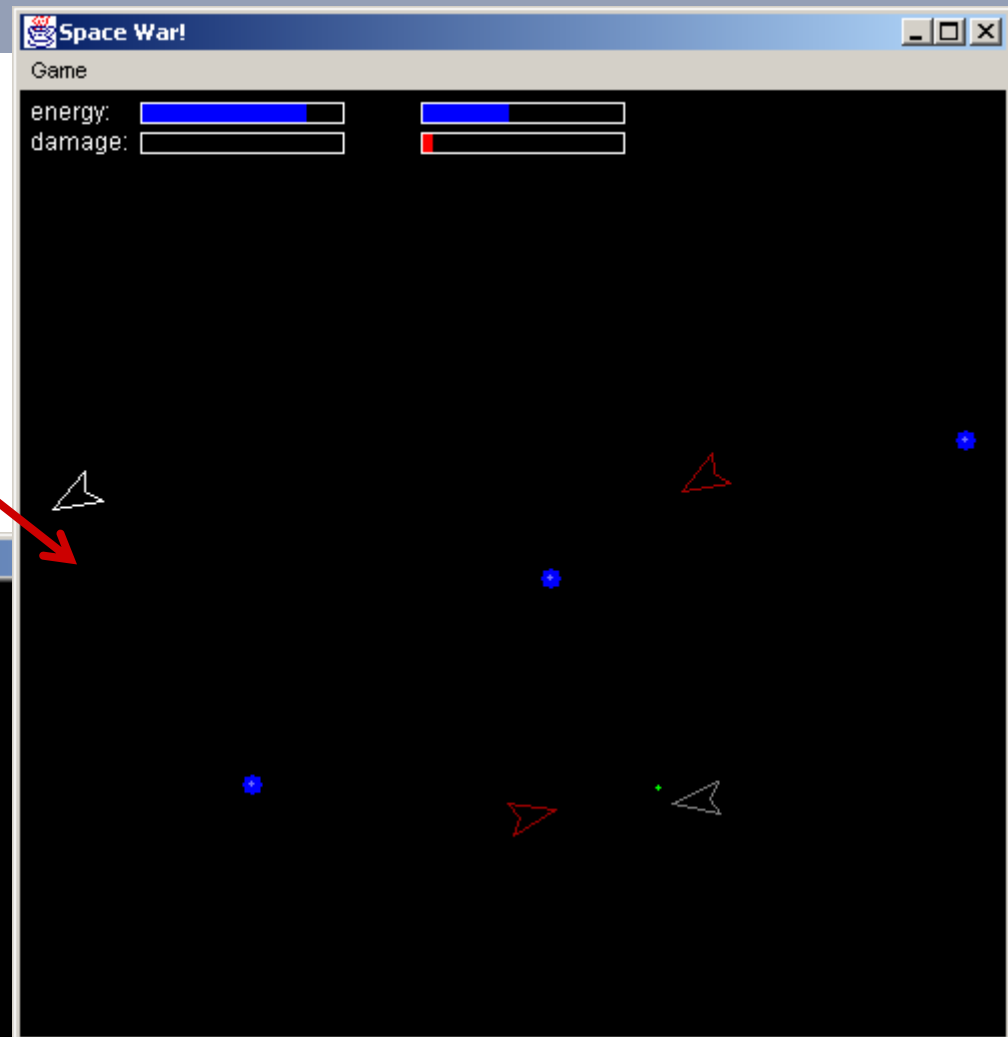
- **Well if I use AOP, how will I know when I'm looking at the code what aspects apply?**
 - IDE support

IDE demos

- **basic support, in emacs, JBuilder™, Forte™4J**
 - browsing program structure
 - editing
 - compiling
- **debugger support**
- **extension to javadoc**
- **will expand by 1.0**
 - more IDEs
 - more sophisticated browsing

demonstration...

- classic spacewar game
- ships move around and fire at each other

[illegible]

Speedbar

- [+] EnergyPacket.java #
- [+] EnergyPacketProducer.java\$
- [+] EnsureShipIsAlive.java #
- [+] Game.java #
- [+] Makefile
- [+] Pilot.java #
- [+] Player.java #
- [+] README.html
- [+] Registry.java #
- [+] Robot.java #
- [+] SWFrame.java #
- [-] Ship.java #
 - (-) Ship
 - (+) ACCELERATION_COST_FACTO\$
 - (-) e to r
 - expendEnergy(double)
 - energy
 - fire()
 - EnsureS...: static aro\$
 - getDamageLevel()
 - getEnergyLevel()
 - getPilot()
 - getRacc()
 - getOrientation()
 - getDamage()
 - getEnergy()
 - getSize()
 - helmCommandsCut(Ship):\$
 - handleCollision(SpaceO\$
 - inflictDamage(double)
 - + new(Game, double, doub\$
 - orientation
 - pilot

<< SPEEDBAR 8 >>

Ship.java

Buffers Files Tools Edit Search Mule Classes JDE AspectJ Java Help

```

void fire() {
    // firing a shot takes energy
    if (!expendEnergy(BULLET_ENERGY))
        return;

    double xV = getXVel() + BULLET_SPEED *
        (Math.cos(orientation));
    double yV = getYVel() + BULLET_SPEED *
        (Math.sin(orientation));

    // create the actual bullet
    new Bullet(getGame(),
        (getXPos() + ((getSize()/2 + 2) *
            (Math.cos(orientation))) + xV),
        (getYPos() + ((getSize()/2 + 2) *
            (Math.sin(orientation))) + yV),
        xV,
        yV,
        orientation);
}

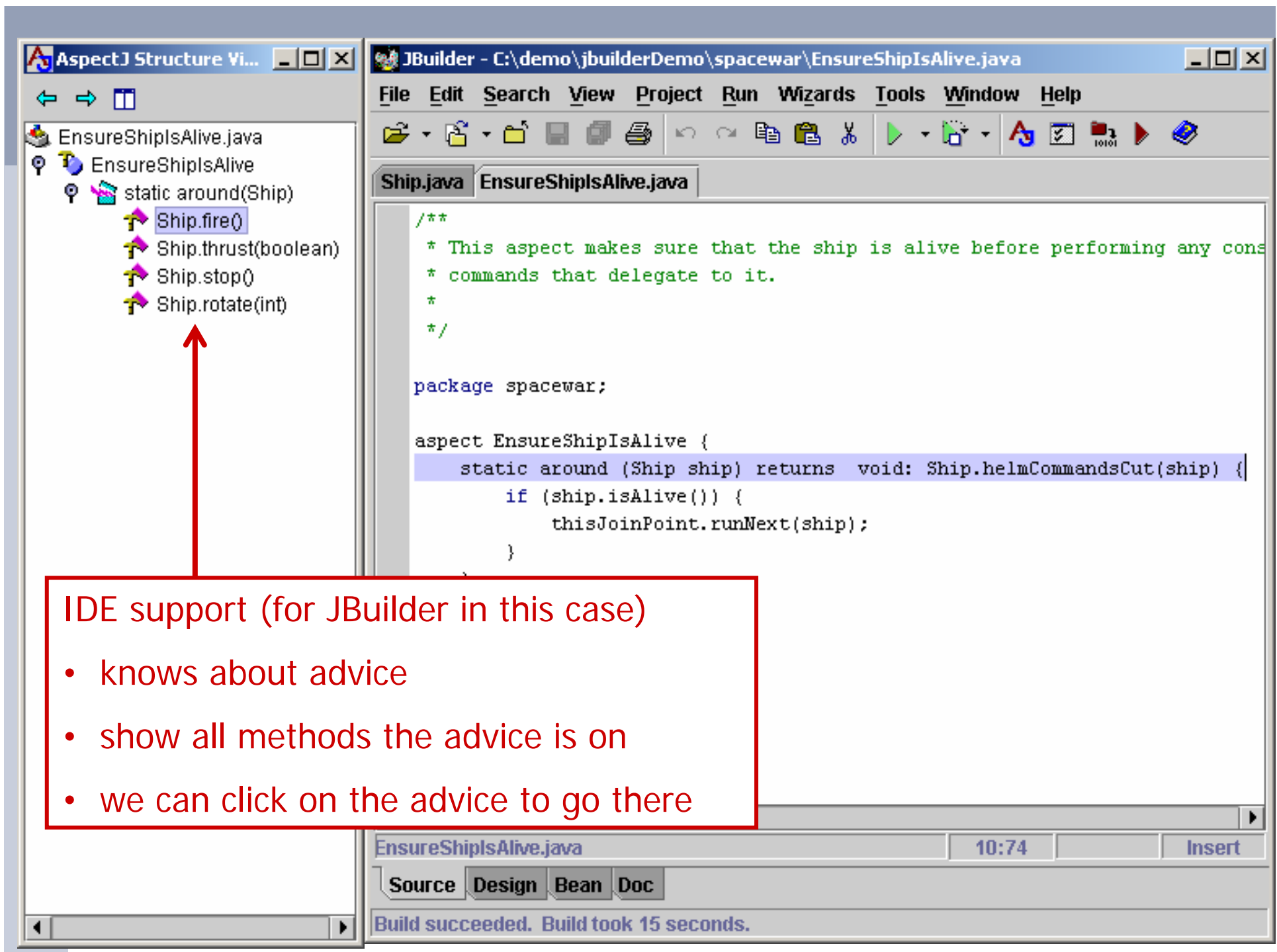
void handleCollision(SpaceObject obj) {
    if (obj instanceof Ship) {
        // should never be called. ship - ship collisions are h
    }
}

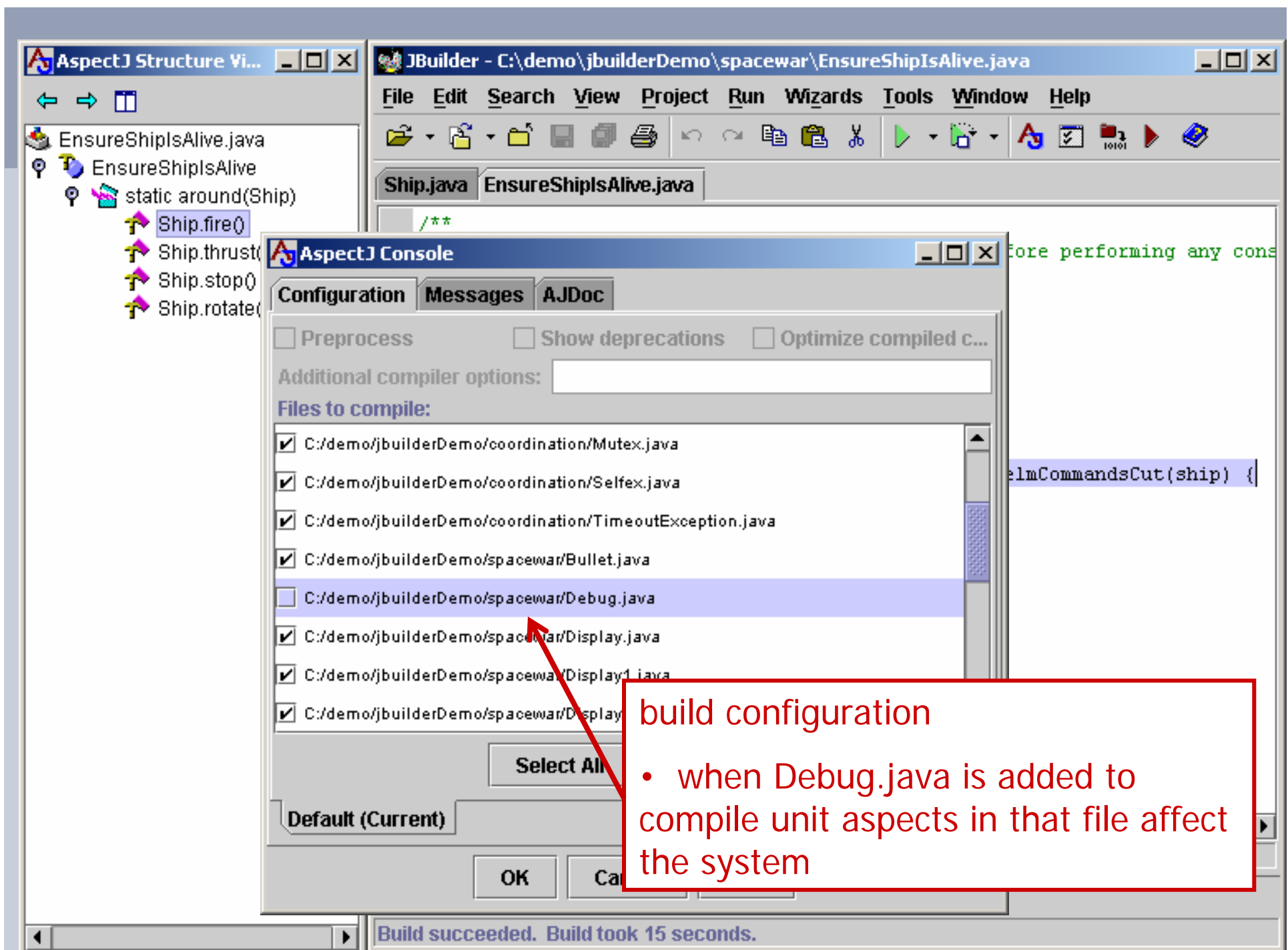
```

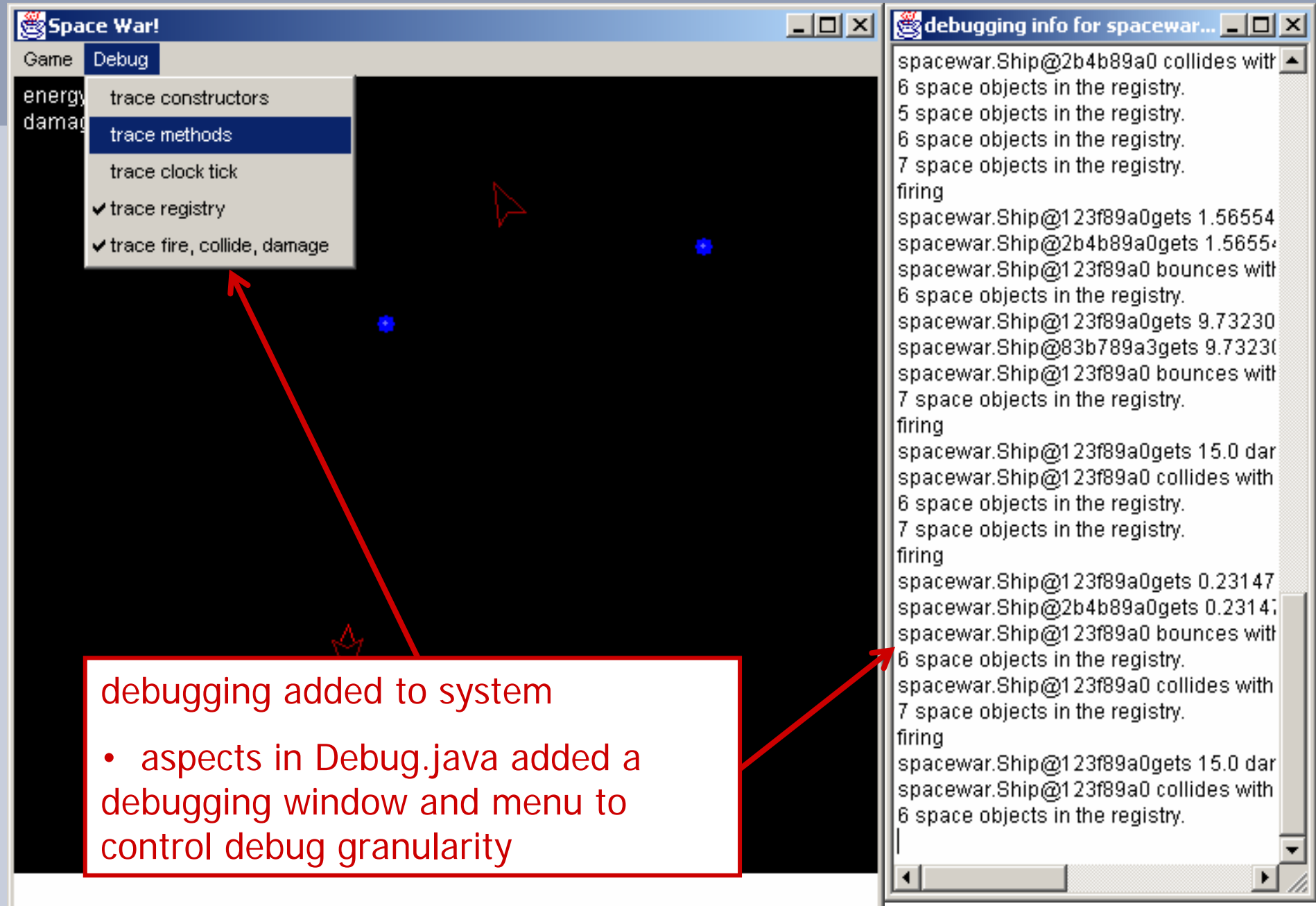
--** Ship.java (JDE AspectJ)--L210--71%--[C:Ship]-----

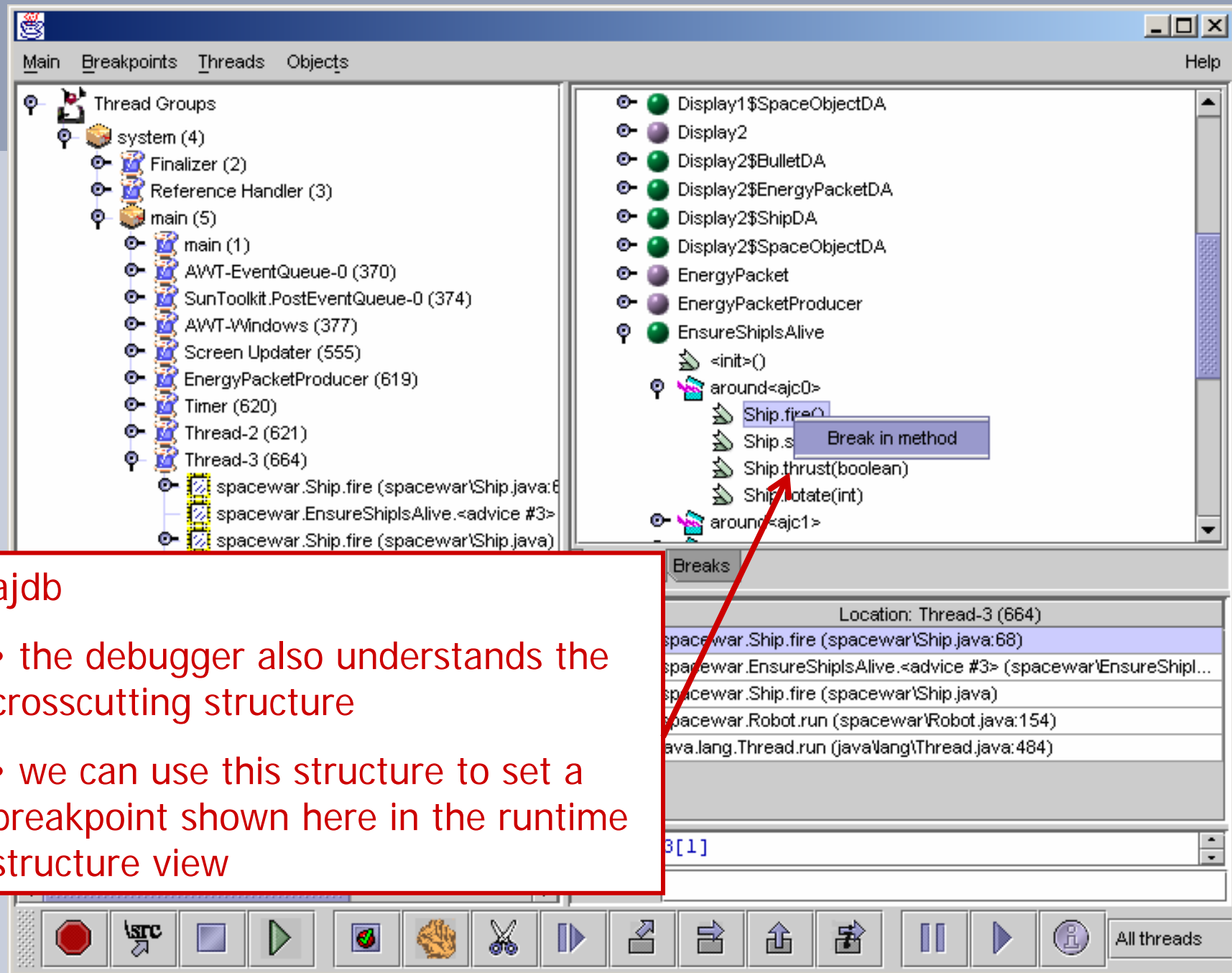
IDE support (for Emacs in this case)

- knows about advice
- shows one advice on fire method
- we can click on the advice to go there









Generated Documentation (Untitled) - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites History

Address C:\demo\emacsDemo\doc\index.html Go Links

[All Classes](#)

Packages

[coordination](#)

[spacewar](#)

Method

[MethodState](#)

[Mutex](#)

[Pilot](#)

[Player](#)

[Registry](#)

[RegistryCoordinat](#)

[Robot](#)

[Selfex](#)

[Ship](#)

[SpaceObject](#)

[SWFrame](#)

[TimeoutException](#)

[Timer](#)

Method Summary

(package private) static void	bounce (Ship shipA, Ship shipB)
void	clockTick ()
(package private) boolean	expendEnergy (double amount)
(package private) void	fire ()
	Advised by: spacewar.EnsureShipIsAlive
(package private) double	getDamage ()
(package private) float	getDamageLevel ()
(package private) double	get
(package private) float	get

ajdoc

- explicit crosscutting structure lets us navigate to the documentation of an aspect

when are aspects appropriate?

- **is there a concern that:**
 - crosscuts the structure of several objects or operations
 - is beneficial to separate out

common questions

- **does this really help?**
 - these examples are all simple

Case Study

- exception handling case study*
- framework w/ 600 classes, 44 KLOC
- reengineered with AspectJ 0.4

+		
	without aspects	with aspects
exception detection	2.1kloc pre-conditions .6kloc post-conditions	.6 KLOC pre-conditions .3 KLOC post-conditions
exception handling	2.1 kloc (414 catch statements)	.2 kloc (31 after throwing advice)
<i>% of total loc</i>	10.9%	2.9%

* appeared in ICSE'2000

good designs

summary

- **capture “the story” well**
- **may lead to good implementations, measured by**
 - code size
 - tangling
 - coupling
 - etc.

learned through
experience, influenced
by taste and style

expected benefits of using AOP

- **good modularity, even in the presence of crosscutting concerns**
 - less tangled code, more natural code, smaller code
 - easier maintenance and evolution
 - easier to reason about, debug, change
 - more reusable
 - more possibilities for plug and play
 - abstract aspects

more info

**more information on AOP and the
AspectJ™ programming language can
be found at:**

[**http://aspectj.org/**](http://aspectj.org/)

[**http://www.parc.xerox.com/aop/**](http://www.parc.xerox.com/aop/)

