

# Refactoring using Together

Amos You, Principal Consultant  
Borland Software China

# 议程

- Refactoring的相关概念
  - 定义，目标，核心内容，基本技巧，典型困难
- Together为Refactoring提供综合支撑环境
  - LiveSource Technology, Metrics, Refactoring Facilities, JUnit Framework, Version control
- 用Together有效进行典型的Refactoring工作
- 建议
  - 理解方法，掌握工具，加速成功

# Refactoring的相关概念

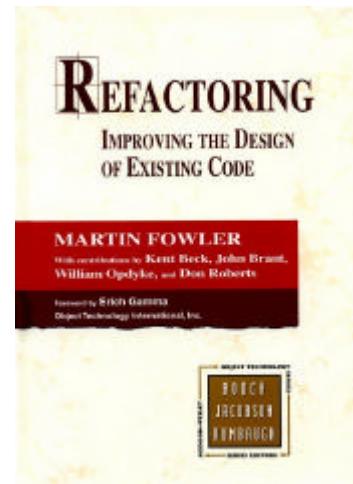
# Refactoring的相关概念

## ■ 定义

- Martin Fowler defines refactoring as:  
“...the process of changing a software system in such a way that it does not alter the external behavior of the code yet improves its internal structure.”
- A frequent result of code reviews

## ■ 目标

- To improve design, understandability and code quality
- To decrease the cost of maintenance



# Refactoring的相关概念

## ■ 核心内容

- 22 “bad smell” (into 7 categories), The Reasons
- 72 “Refactoring”, The corresponding Actions

## ■ 基本技巧

- Taking small steps
- Testing often

# Refactoring的相关概念

## ■ 典型困难

- 代码的直观性和概括性
- 发现“bad smell”的手段
- 局部改动的整体完整性
- 有效的集成单元测试设施
- 对代码调整的阶段可逆性保障

# Together为Refactoring 提供综合支撑环境

# Live Source Technology

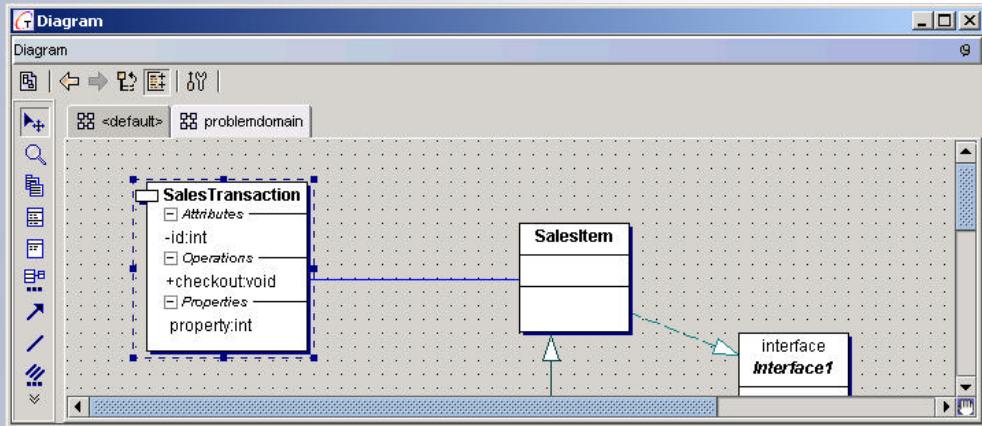
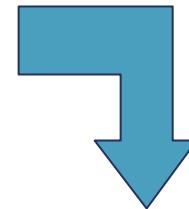


Diagram elements  
generates source



Source reflected  
in diagram

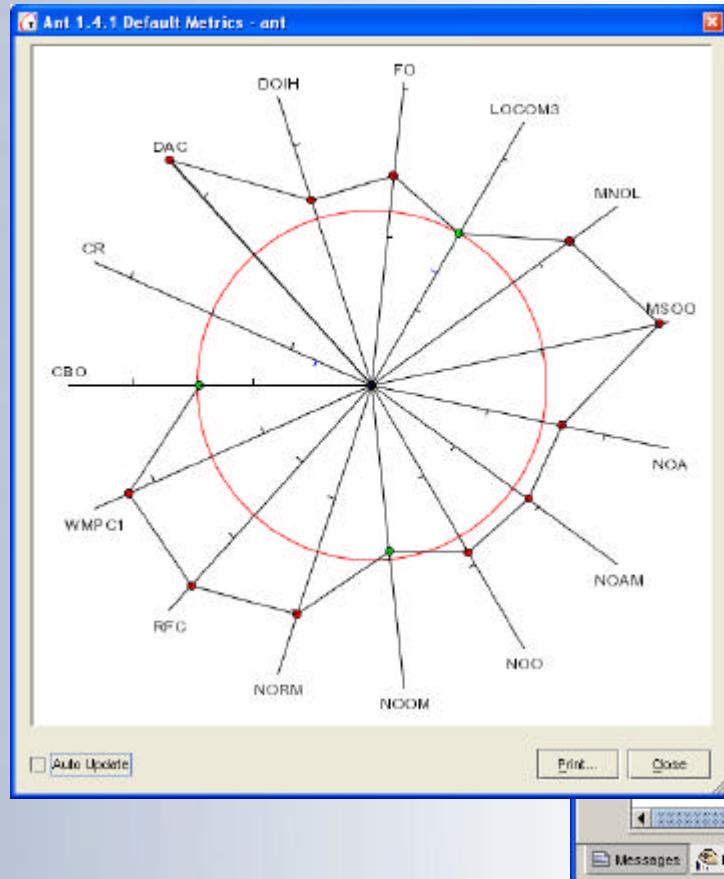


The code editor displays the generated Java source code for the SalesTransaction class:

```
/* Generated by Together */
package problemdomain;
public class SalesTransaction {
    public void checkout() {
    }
    public int getProperty(){
}
```

The file is named "SalesTransaction.java".

# Metrics



Metrics | Ant 1.4.1 Default Metrics

	AHF	AF	CBO	CC	M500	WMPC1	OF	CR	DAC	D0H	FO
ant	28	173	48	173				0	16	6	26
listener	5	21	4	21				32	0	1	3
taskdefs	28	173	48	173				0	16	6	26
types	11	70	14	70				0	3	4	11
util	12	47	15	47				38	0	1	11
AntClassLoader	15	66	8	66				38	4	3	15
BuildEvent	3	11	1	11				58	3	3	3
BuildException	3	16	2	16				58	1	5	3
BuildListener	1	7	1	7				81	0	2	1
BuildLogger	1	4	1	4				85	0	3	1
Constants	0	0	0	0				71	0	1	0
DefaultLogger	4	21	5	21				44	1	1	2
DemuxOutputStream	0	12	4	12				54	2	3	4
DesirableFilter	1	9	9	9				59	0	1	1
DirectoryScanner	3	138	35	138				40	2	1	3
EditException	0	2	1	2				78	0	8	0
FileScanner					DirectorScanner	WMPC1					
IntrospectionHelper					Ant 1.4 Default Metrics	45	80		0	2	12
Launcher	8	18	5	18				34	0	1	6

# Refactoring Facilities

Editor

```
9
10    public void checkout() {
11        for (int i = 0; i < items.size(); i++) {
12            LineItem unItem = (LineItem) items.elementAt(i);
13            InventoryManager.getInstance().update(unItem);
14        }
15    }
16
17
18    public void update(LineItem unItem) {
19        LineItem items;
20        int size = items.size();
21        for (int i = 0; i < size; i++) {
22            LineItem item = (LineItem) items.elementAt(i);
23            if (item.equals(unItem)) {
24                item.setQuantity(item.getQuantity() + unItem.getQuantity());
25            }
26        }
27    }
28
29    public void makeSale() {
30        LineItem items;
31        int size = items.size();
32        for (int i = 0; i < size; i++) {
33            LineItem item = (LineItem) items.elementAt(i);
34            item.setQuantity(item.getQuantity() - 1);
35        }
36    }
37
38    public void printDescription() {
39        LineItem items;
40        int size = items.size();
41        for (int i = 0; i < size; i++) {
42            LineItem item = (LineItem) items.elementAt(i);
43            System.out.println(item.getDescription());
44        }
45    }
46
47    public void printLineItem() {
48        LineItem items;
49        int size = items.size();
50        for (int i = 0; i < size; i++) {
51            LineItem item = (LineItem) items.elementAt(i);
52            System.out.println(item.getLineItem());
53        }
54    }
55
56    public void printInventory() {
57        LineItem items;
58        int size = items.size();
59        for (int i = 0; i < size; i++) {
60            LineItem item = (LineItem) items.elementAt(i);
61            System.out.println(item.getInventory());
62        }
63    }
64
65    public void printSales() {
66        LineItem items;
67        int size = items.size();
68        for (int i = 0; i < size; i++) {
69            LineItem item = (LineItem) items.elementAt(i);
70            System.out.println(item.getSales());
71        }
72    }
73
74    public void printTotalSales() {
75        LineItem items;
76        int size = items.size();
77        for (int i = 0; i < size; i++) {
78            LineItem item = (LineItem) items.elementAt(i);
79            System.out.println(item.getTotalSales());
80        }
81    }
82
83    public void printTotalInventory() {
84        LineItem items;
85        int size = items.size();
86        for (int i = 0; i < size; i++) {
87            LineItem item = (LineItem) items.elementAt(i);
88            System.out.println(item.getTotalInventory());
89        }
90    }
91
92    public void printTotalSalesAndInventory() {
93        LineItem items;
94        int size = items.size();
95        for (int i = 0; i < size; i++) {
96            LineItem item = (LineItem) items.elementAt(i);
97            System.out.println(item.getTotalSalesAndInventory());
98        }
99    }
100 }
```

Editor

```
7
8    public class SalesTransaction {
9
10        public void checkout() {
11            for (int i = 0; i < getItems().size(); i++) {
12                LineItem unItem = (LineItem) getItems().elementAt(i);
13                InventoryManager.getInstance().update(unItem);
14            }
15        }
16
17        public void update(LineItem unItem) {
18            LineItem items;
19            int size = items.size();
20            for (int i = 0; i < size; i++) {
21                LineItem item = (LineItem) items.elementAt(i);
22                if (item.equals(unItem)) {
23                    item.setQuantity(item.getQuantity() + unItem.getQuantity());
24                }
25            }
26        }
27
28        public void makeSale() {
29            LineItem items;
30            int size = items.size();
31            for (int i = 0; i < size; i++) {
32                LineItem item = (LineItem) items.elementAt(i);
33                item.setQuantity(item.getQuantity() - 1);
34            }
35        }
36
37        public void printDescription() {
38            LineItem items;
39            int size = items.size();
40            for (int i = 0; i < size; i++) {
41                LineItem item = (LineItem) items.elementAt(i);
42                System.out.println(item.getDescription());
43            }
44        }
45
46        public void printLineItem() {
47            LineItem items;
48            int size = items.size();
49            for (int i = 0; i < size; i++) {
50                LineItem item = (LineItem) items.elementAt(i);
51                System.out.println(item.getLineItem());
52            }
53        }
54
55        public void printInventory() {
56            LineItem items;
57            int size = items.size();
58            for (int i = 0; i < size; i++) {
59                LineItem item = (LineItem) items.elementAt(i);
60                System.out.println(item.getInventory());
61            }
62        }
63
64        public void printSales() {
65            LineItem items;
66            int size = items.size();
67            for (int i = 0; i < size; i++) {
68                LineItem item = (LineItem) items.elementAt(i);
69                System.out.println(item.getSales());
70            }
71        }
72
73        public void printTotalSales() {
74            LineItem items;
75            int size = items.size();
76            for (int i = 0; i < size; i++) {
77                LineItem item = (LineItem) items.elementAt(i);
78                System.out.println(item.getTotalSales());
79            }
80        }
81
82        public void printTotalInventory() {
83            LineItem items;
84            int size = items.size();
85            for (int i = 0; i < size; i++) {
86                LineItem item = (LineItem) items.elementAt(i);
87                System.out.println(item.getTotalInventory());
88            }
89        }
90
91        public void printTotalSalesAndInventory() {
92            LineItem items;
93            int size = items.size();
94            for (int i = 0; i < size; i++) {
95                LineItem item = (LineItem) items.elementAt(i);
96                System.out.println(item.getTotalSalesAndInventory());
97            }
98        }
99    }
100 }
```

Editor

Select in Diagram  
Browse Symbol

Cut Ctrl+X  
Copy Ctrl+C  
Paste Ctrl+V  
Select All Ctrl+A  
Collapse Lines  
Expand Lines  
Tools  
Toggle Breakpoint F5  
Enable Breakpoint  
Text Editor Options...  
Refactoring  
Bookmarks  
Override/Implement Method

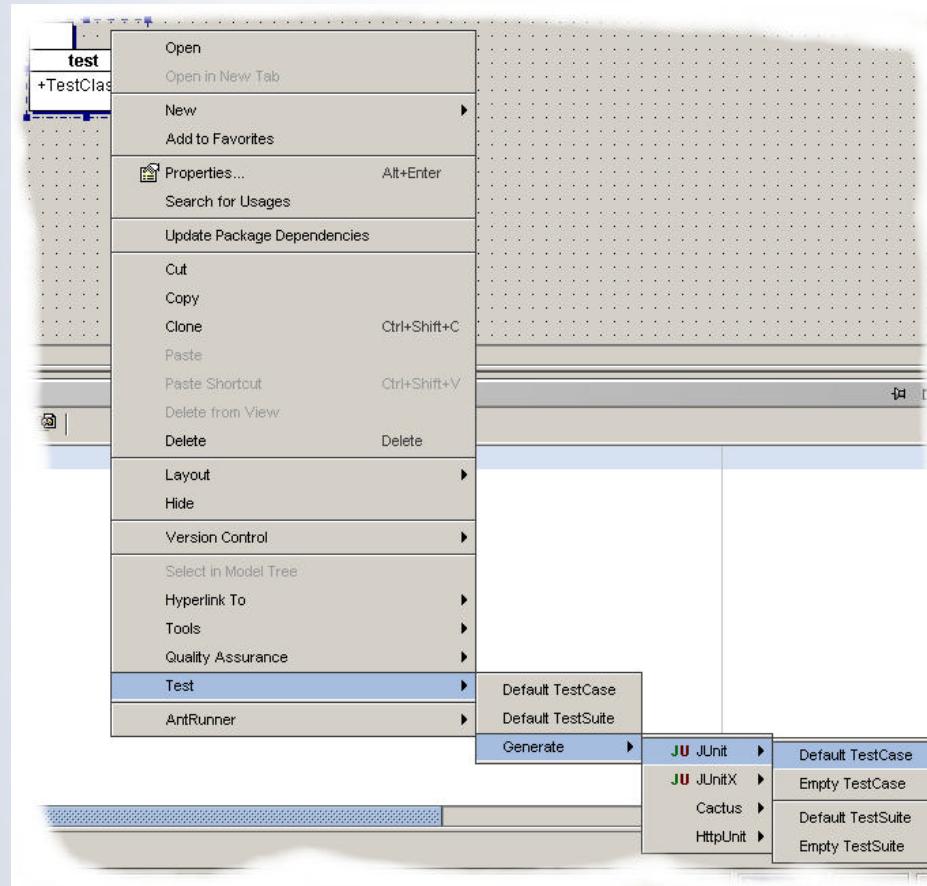
Rename Attribute... F2  
Extract Interface...  
Extract Superclass...  
Encapsulate Attribute...  
Push Down Attribute...

Direct attribute access

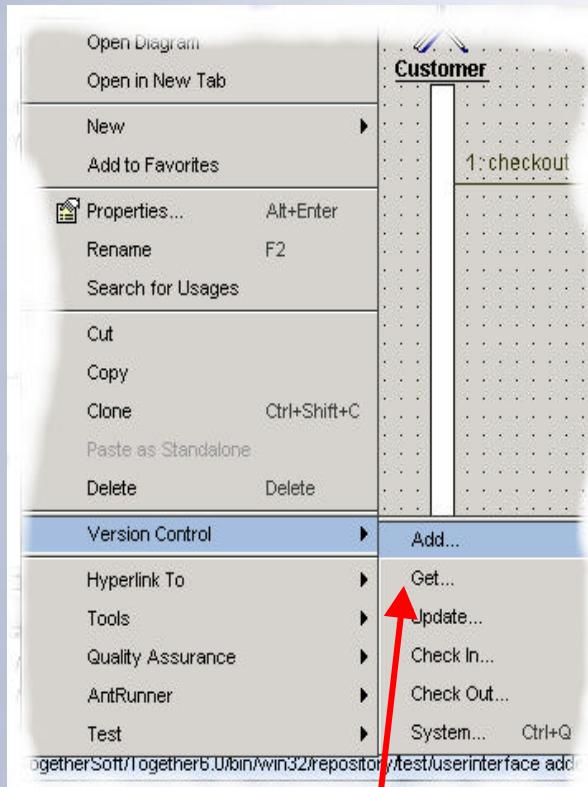
Re-factor attribute with encapsulate attribute pattern

Attribute access using get accessor

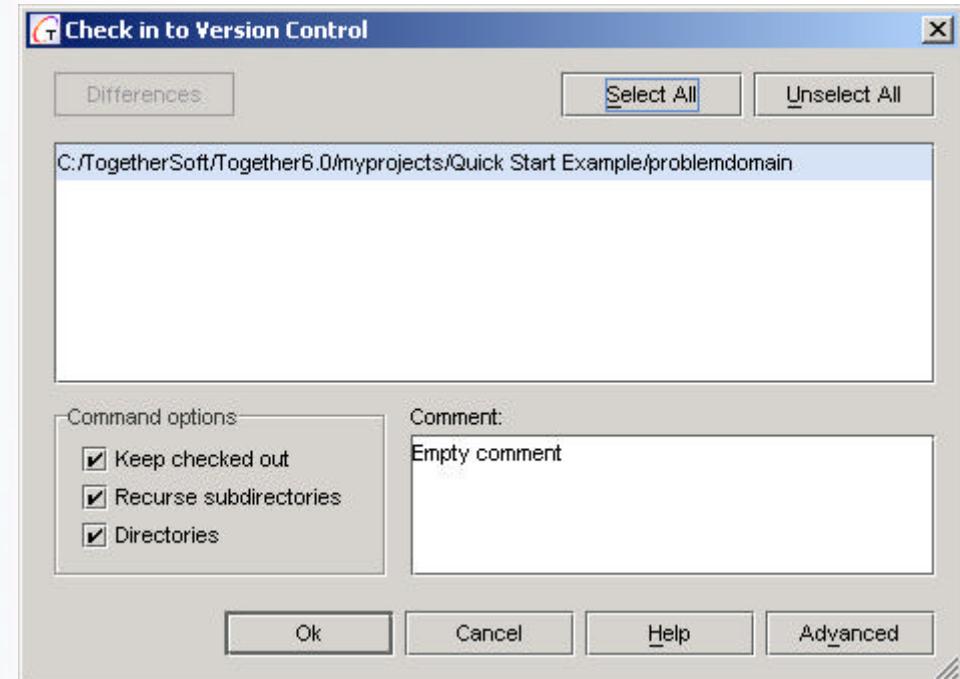
# JUnit Framework



# Version Control



Version control operations

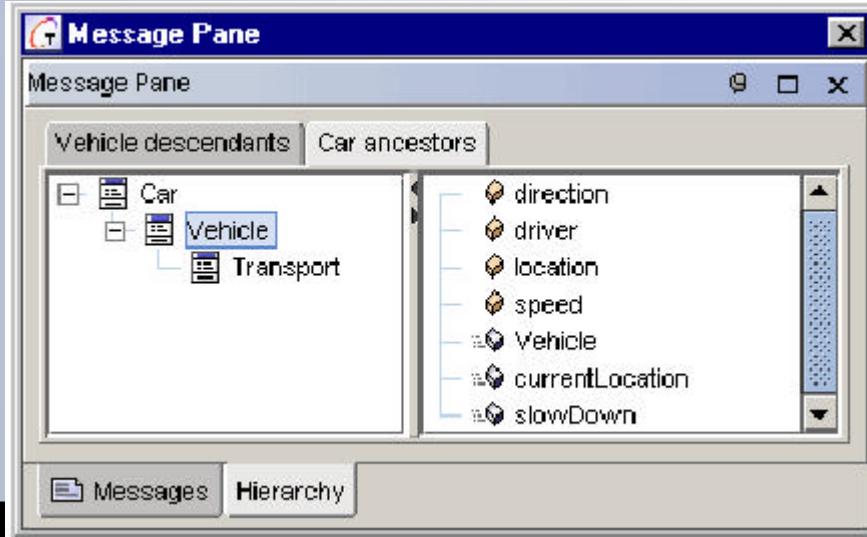


Check in dialog

# 用Together有效进行典型的 Refactoring工作

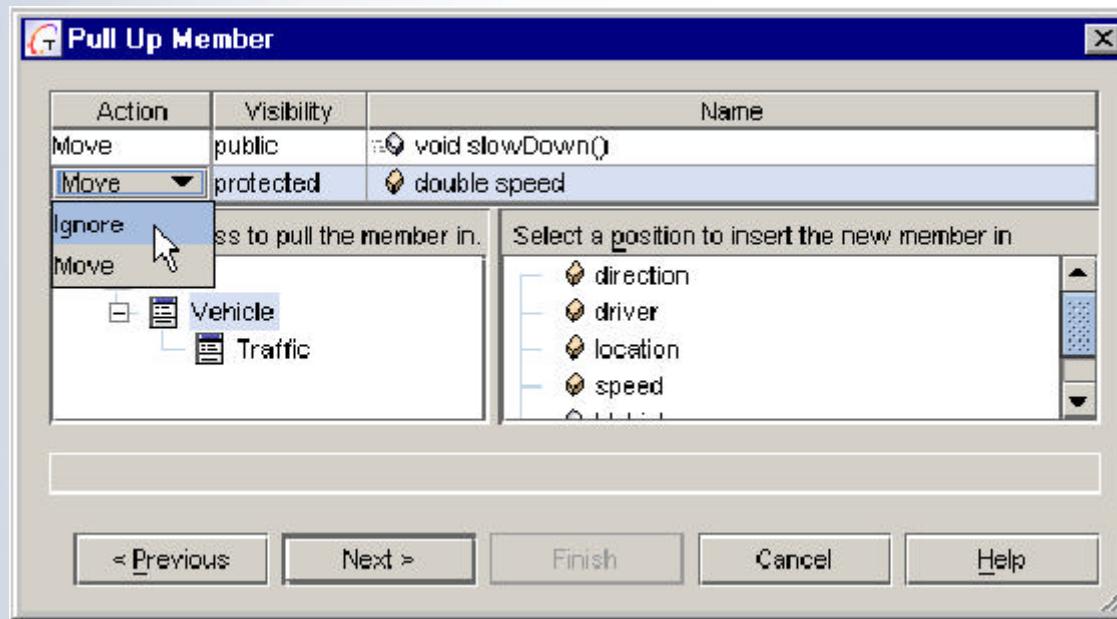
# Showing code structure

- Show Ancestors
- Show Descendants
- Show Implementing Classes
- Show Overrides

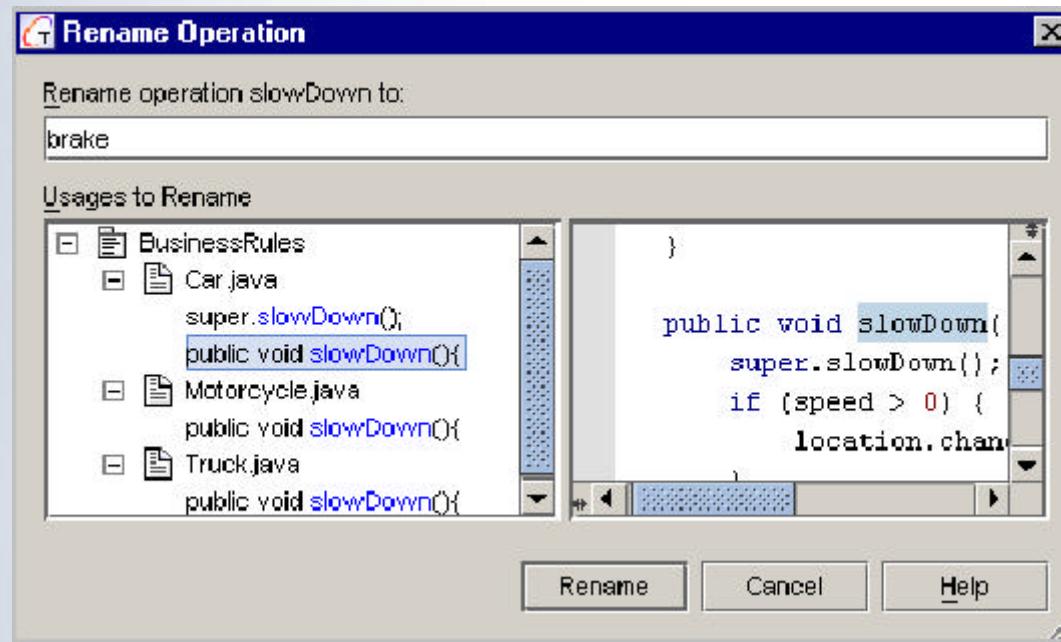


# Moving classes, interfaces, attributes, and operations

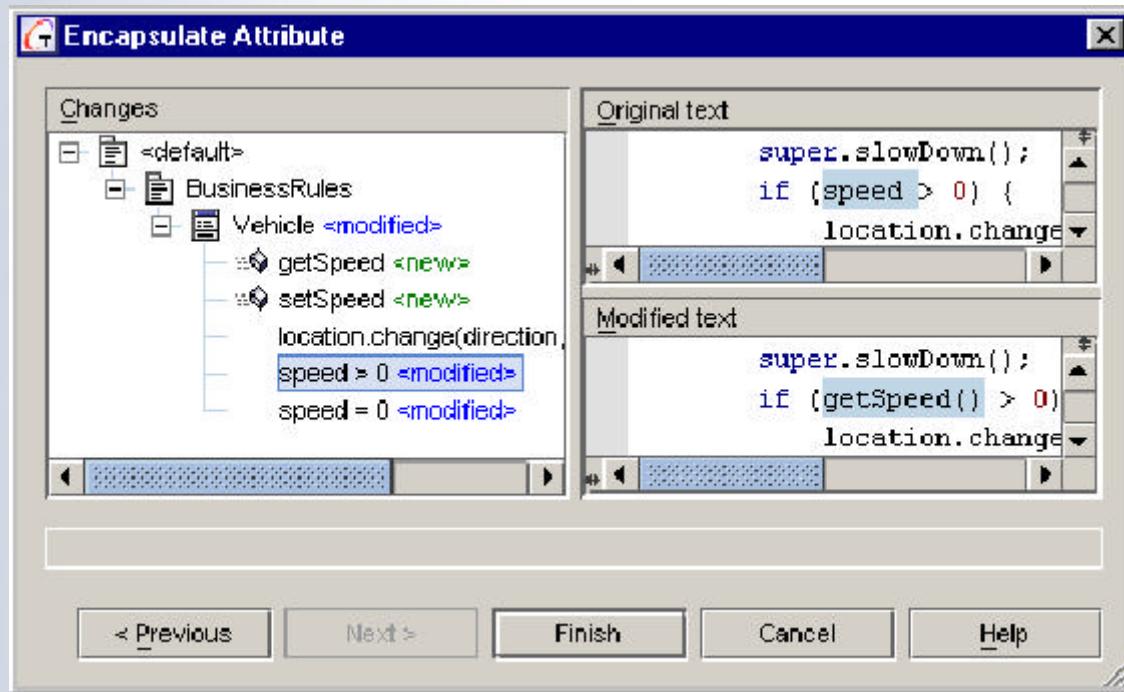
- Moving classes and interfaces
- Moving attributes and operations in the class hierarchy



# Renaming

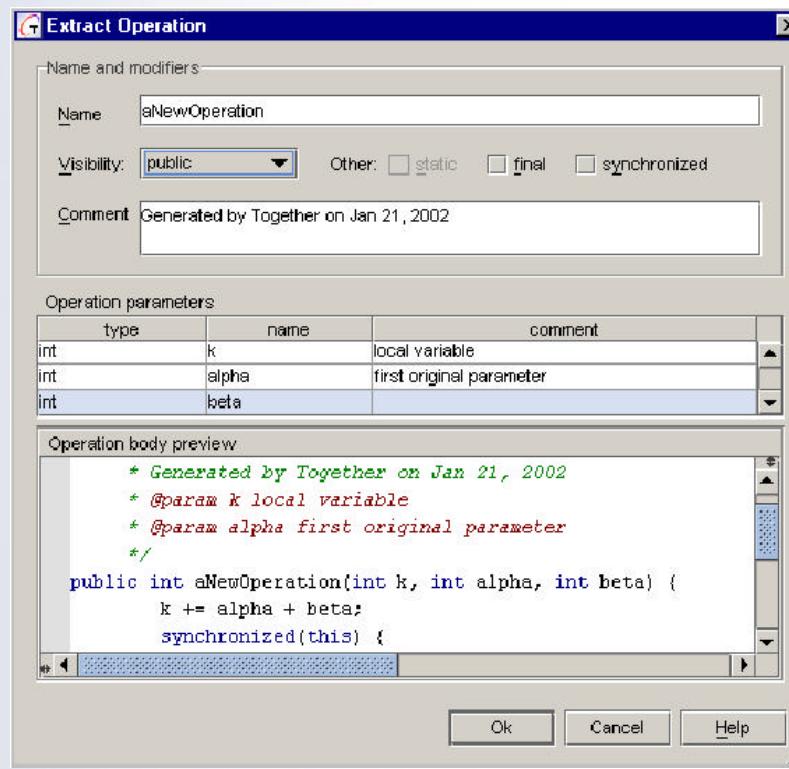


# Encapsulating attributes



# Extracting interfaces, superclasses, and operations

- Extracting interfaces and superclasses
- Extracting operations



# 建议

# 建议

## ■ 理解方法

- Basic catalogue available as book  
"Refactoring: Improving the design of existing code",  
Addison-Wesley, 1999
- Extended catalogue available online  
<http://www.refactoring.com/>

## ■ 掌握工具

- 下载工具: [www.borland.com](http://www.borland.com)
- 快速入门: Together Practical Guide
- White paper: Software Remodeling: Improving Design and Implementation Quality

## ■ 加速成功

- 选择Borland专业咨询与培训服务

谢谢

amos.you@borland.com