

Programming Tools

David Rabinowitz

This Lecture

- Personal Productivity Tools
 - And how to use them
- Refactoring
- Static Analysis & Metrics
- Profiling

March 3rd, 2004

Object Oriented Design Course

2

Refactoring

- Improving the design of existing code, without changing its observable behavior
- Here's the *Extract Method* refactoring:

```
Before:                                     After:
void f(int[] a) {                             void f() {
  ...                                         ...
  // compute score                          computeScore();
  score = initial_score;                    }
  for (int i=0; i<a.length; i++)            computeScore(int[] a) {
    score += a[i] * delta;                  // code cut & pasted here
  }                                         }
}
```

March 3rd, 2004

Object Oriented Design Course

3

Why?

- Why Refactor?
 - Improve software design
 - Make software easier to understand
 - Help find bugs
 - Help program faster
- Preconditions
 - Working code
 - Good set of unit tests

March 3rd, 2004

Object Oriented Design Course

4

When?

- When to refactor
 - Before adding functionality
 - Before fixing a bug
 - During code review
- When not to refactor
 - During adding functionality
 - During fixing a bug
 - No good set of unit tests
 - Small programs (usually)

March 3rd, 2004

Object Oriented Design Course

5

Code Smells

- "If it stinks, change it"
 - Duplicate code
 - Switch statements
 - Long method
 - Data class
 - Long parameter list
 - Primitive obsession
 - Temporary field
 - ...

March 3rd, 2004

Object Oriented Design Course

6

Documented Refactorings

- There's a catalog
 - Fowler's book
 - www.refactoring.com/catalog
- There are many others
- Way to learn good OOD principles
- Pay attention to the mechanics

March 3rd, 2004

Object Oriented Design Course

7

Automated Refactorings

- Eclipse's 'Refactor' menu automates thing
 - Undo & Redo
 - Physical Structure
 - Class Level Structure
 - Structure Inside a Class
- Wizards & Preview Windows Included
- Other tools exist: see refactorit.com

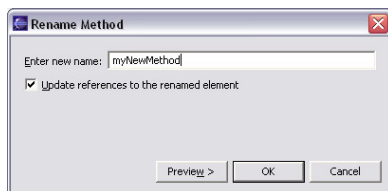
March 3rd, 2004

Object Oriented Design Course

8

Automated Refactoring Example

- The 'Rename' Refactoring renames any Java element, and references to it:



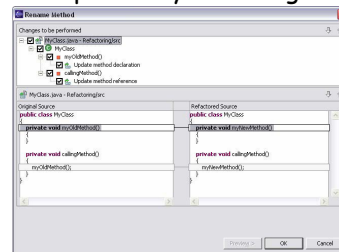
March 3rd, 2004

Object Oriented Design Course

9

Automated Refactoring Example II

- You can preview your changes:



March 3rd, 2004

Object Oriented Design Course

10

Encapsulate Field

Before:

```
public String name;
```

After:

```
private String name;
```

```
public String getName() {  
    return name;  
}
```

```
public void setName(String n) {  
    name = n;  
}
```

March 3rd, 2004

Object Oriented Design Course

11

Encapsulate Field in Eclipse



March 3rd, 2004

Object Oriented Design Course

12

Introduce Null Object

Before:

```
if (project == null)
    plan = Plan.default();
else
    plan = project.getPlan();
```

After:

```
class NullProject
    implements Project {
    public Plan getPlan() {
        return Plan.default();
    }
    // other Project methods
}
```

• This is the *Null Object Pattern*

March 3rd, 2004

Object Oriented Design Course

13

Parameterize Method

Before:

```
class Server {
    handleGet(...)
    handlePut(...)
    handleSet(...)
}
```

After:

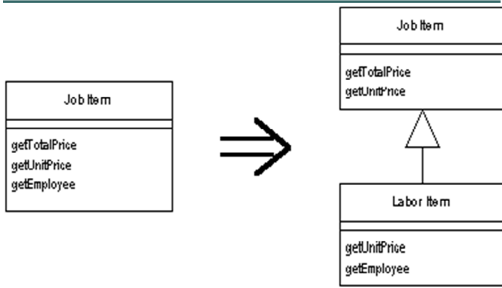
```
class Server {
    handle(EventType et, ...)
}
```

March 3rd, 2004

Object Oriented Design Course

14

Extract Subclass

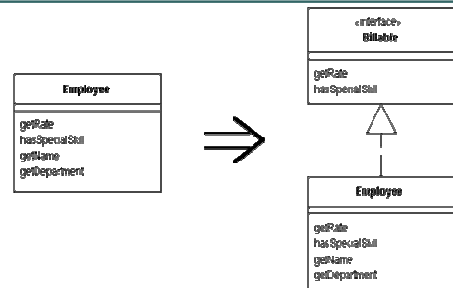


March 3rd, 2004

Object Oriented Design Course

15

Extract Interface

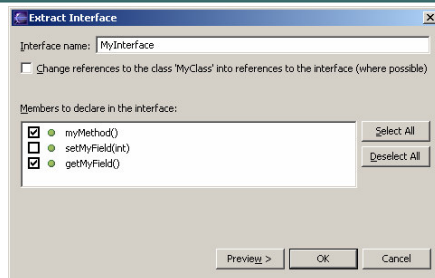


March 3rd, 2004

Object Oriented Design Course

16

Extract Interface in Eclipse

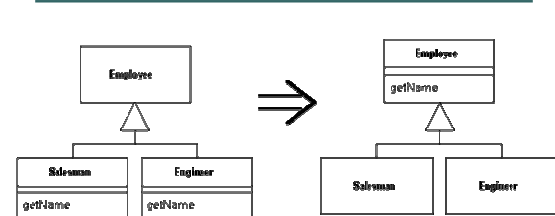


March 3rd, 2004

Object Oriented Design Course

17

Pull Up Method

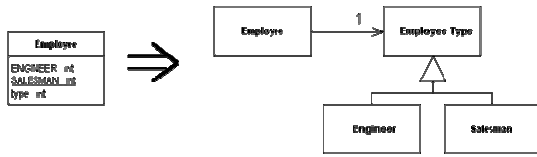


March 3rd, 2004

Object Oriented Design Course

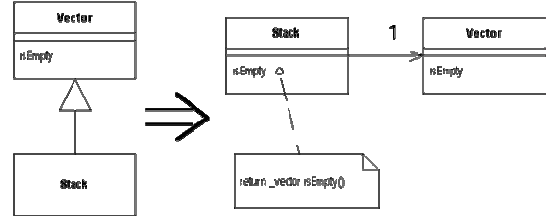
18

Replace Type Code with State/Strategy



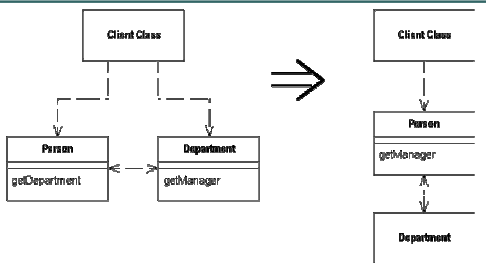
March 3rd, 2004 Object Oriented Design Course 19

Replace Inheritance with Delegation



March 3rd, 2004 Object Oriented Design Course 20

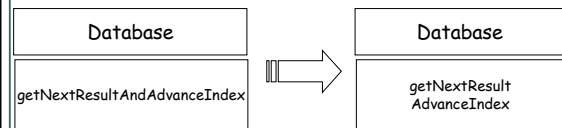
Hide Delegate



• Obeys the *Law of Demeter*

March 3rd, 2004 Object Oriented Design Course 21

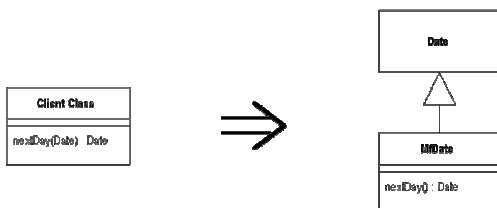
Separate Query from Modifier



• Obeys *Command-Query Separation*

March 3rd, 2004 Object Oriented Design Course 22

Introduce Local Extension



• Alternative: Introduce Foreign Method

March 3rd, 2004 Object Oriented Design Course 23

The opposites are there too

- Inline method (extract method)
- Replace Parameter with Explicit Methods (Parameterize Method)
- Collapse Hierarchy (Extract subclass)
- Remove middle man (Hide delegate)
- Push down method (pull up method)
- Replace delegation with inheritance

March 3rd, 2004 Object Oriented Design Course 24

More useful Refactorings in Eclipse

- Rename
- Move
- Change Method Signature
- Use Supertype where possible
- Extract Constant
- Introduce Factory
- ...

March 3rd, 2004

Object Oriented Design Course

25

How to Refactor

- Recognize the smells
- Refactor in small discrete steps
- Test after each step
- Refactor in pairs
- Use documented refactorings
- Don't mix with adding functionality or fixing a bug

March 3rd, 2004

Object Oriented Design Course

26

Static Code Analysis

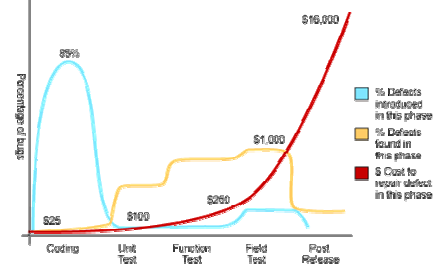
- Programs that help gain understanding of your code
- Find areas in the code with
 - Possible Bugs
 - "Fishy" Design
 - Inconsistent Style
- It's no replacement for testing
 - Finding (non-trivial) bugs is undecidable

March 3rd, 2004

Object Oriented Design Course

27

Why is it so important?



Source: Applied Software Measurement, Capers Jones, 1996

March 3rd, 2004

Object Oriented Design Course

28

Available Tools

- Commercial
 - Lint for C and C++ (see gimpel.com)
 - JTest (parasoft.com)
- Free Eclipse Plugins
 - JLint (arho.com)
 - CPD - Copy Paste Detector
 - PMD
 - CheckStyle
 - JDepend - Metrics

March 3rd, 2004

Object Oriented Design Course

29

Lint

- Looks for over 800 C/C++ Issues
- Things that compilers either miss or allow
- Specific C++ Errors, for example:
 - Throwing from a destructor
 - Not checking for NULL argument in 'delete'
 - Order of initializations / constructors
 - Non-virtual over-riden methods
- Macro scanning
 - Incorrect parameter passing, Side effects, ...

March 3rd, 2004

Object Oriented Design Course

30

Lint II

- Value Tracking
 - Division by zero, null dereference, out-of-bounds, memory leaks, double deallocation, ...
- Casting & Values
 - Loss of sign, truncations, Assignment in 'if', ...
- Specific C Issues
 - printf() arguments, order of evaluation: a[i] = i++;
- Style
 - Indentation, suspicious semi-colons (a > b); , ...
- Hundreds of other issues

March 3rd, 2004

Object Oriented Design Course

31

JTest

- Checks for 380 Java & Style Issues
- Can automatically correct 160 of these
- Extensible by user-defined issues
- Supports metrics as well
 - Number of bytes, classes, lines, methods, ...
 - Issue = Deviation from acceptable metric range
- Some issues are shared with C/C++
 - Values, Casting, Unreachable code, Indentation, Comments, Initialization, Exceptions, ...

March 3rd, 2004

Object Oriented Design Course

32

JTest II

- Other Java Specific Issues
 - Portability
 - Security
 - Optimization
 - Garbage Collection
 - Threads and Synchronization
 - Internationalization
 - Servlets / EJBs
 - Naming Conventions
 - ...

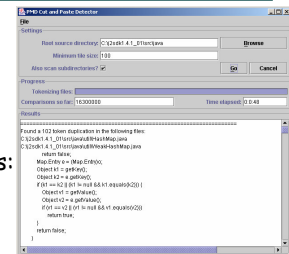
March 3rd, 2004

Object Oriented Design Course

33

CPD - Copy Paste Detector

- Works with Java, C, C++ and PHP
- <http://pmd.sourceforge.net/cpd.html>
- From the examples:
 - A 307 lines(!) of duplicated code in Apache 2



March 3rd, 2004

Object Oriented Design Course

34

PMD



- For Java code
- Checks
 - Unused local variables / parameters / private methods
 - Empty catch blocks
 - Empty 'if' statements
 - Duplicate import statements
 - Classes which could be Singletons
 - Short/long variable and method names
 - And many many more ...

March 3rd, 2004

Object Oriented Design Course

35

CheckStyle

- Similar to PMD
- Javadoc Comments, Naming Conventions, Headers, Imports, Size Violations, Whitespace, Modifiers, Blocks, Coding Problems, Class Design, Duplicate Code

Icon	Description	Resource	In Folder	Location
Q	Name 'task' field name must match pattern ''([a-z]+)([0-9-2_][a-z-2_])''	FieldNames.java	JavaTest\test\src\main\java\...	line 2
I	Missing a Javadoc comment.	FieldNames.java	JavaTest\test\src\main\java\...	line 5
I	'/' should be on the previous line.	FieldNames.java	JavaTest\test\src\main\java\...	line 6
I	Missing a Javadoc comment.	FieldNames.java	JavaTest\test\src\main\java\...	line 7
I	Missing a Javadoc comment.	FieldNames.java	JavaTest\test\src\main\java\...	line 8
Q	Name 'NON_CONSTANT' must match pattern ''[A-Z]_[A-Z0-9]*''	FieldNames.java	JavaTest\test\src\main\java\...	line 9
I	Missing a Javadoc comment.	FieldNames.java	JavaTest\test\src\main\java\...	line 10
Q	Name 'NON_CONSTANT' must match pattern ''[A-Z]_[A-Z0-9]*''	FieldNames.java	JavaTest\test\src\main\java\...	line 10
A	Variable 'NON_CONSTANT' must be private and have accessor method.	FieldNames.java	JavaTest\test\src\main\java\...	line 10
I	Missing a Javadoc comment.	FieldNames.java	JavaTest\test\src\main\java\...	line 11
A	Variable 'NON_CONSTANT' must be private and have accessor method.	FieldNames.java	JavaTest\test\src\main\java\...	line 11

March 3rd, 2004

Object Oriented Design Course

36

JDepend

- Calculates metrics for java packages
- Calculated metrics
- **CC** - Concrete Class Count
 - The number of concrete classes in this package.
- **AC** - Abstract Class Count
 - The number of abstract classes or interfaces in this package.

March 3rd, 2004

Object Oriented Design Course

37

JDepend (2)

- **Ca** - Affferent Couplings
 - The number of packages that depend on classes in this package.
 - "How will changes to me impact the rest of the project?"
- **Ce** - Efferent Couplings
 - The number of other packages that classes in this package depend upon.
 - "How sensitive am I to changes in other packages in the project?"

March 3rd, 2004

Object Oriented Design Course

38

JDepend (3)

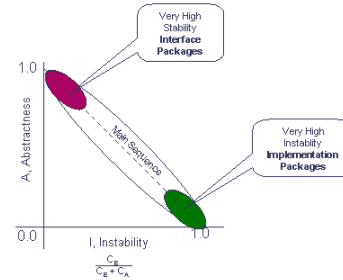
- **A** - Abstractness (0-1)
 - Ratio (0.0-1.0) of Abstract Classes (and interfaces) in this package.
 - $AC/(CC+AC)$
- **I** - Instability (0-1)
 - Ratio (0.0-1.0) of Efferent Coupling to Total Coupling.
 - $Ce/(Ce+Ca)$
- **D** - Distance from the Main Sequence (0-1)
- **Cyclic** - If the package contains a dependency cycle

March 3rd, 2004

Object Oriented Design Course

39

The main sequence



March 3rd, 2004

Object Oriented Design Course

40

Examples - Pet Store

Used By - Affferent Dependencies (58 Packages)
com.sun.j2ee.blueprints.catalog.model (CC: 4 AC: 0 Ca: 4 Ce: 0 A: 0 I: 0 D: 1)
com.sun.j2ee.blueprints.catalog.util (CC: 2 AC: 0 Ca: 0 Ce: 0 A: 0 I: 0 D: 1)
com.sun.j2ee.blueprints.contactinfo.ejb (CC: 2 AC: 3 Ca: 6 Ce: 5 A: 0.6 I: 0.45 D: 0.05)
com.sun.j2ee.blueprints.creditcard.ejb (CC: 1 AC: 3 Ca: 6 Ce: 2 A: 0.75 I: 0.25 D: 0)
com.sun.j2ee.blueprints.customer.account.ejb (CC: 0 AC: 3 Ca: 3 Ce: 2 A: 1 I: 0.4 D: 0.4)
com.sun.j2ee.blueprints.customer.ejb (CC: 0 AC: 3 Ca: 4 Ce: 2 A: 1 I: 0.33 D: 0.33)
com.sun.j2ee.blueprints.customer.profile.ejb (CC: 1 AC: 3 Ca: 6 Ce: 0 A: 0.75 I: 0 D: 0.25)
com.sun.j2ee.blueprints.encodingfilter.web (CC: 1 AC: 0 Ca: 0 Ce: 0 A: 0 I: 0 D: 1)
com.sun.j2ee.blueprints.lineitem.ejb (CC: 1 AC: 2 Ca: 2 Ce: 2 A: 0.67 I: 0.5 D: 0.17)
com.sun.j2ee.blueprints.petstore.controller.ejb (CC: 3 AC: 8 Ca: 2 Ce: 6 A: 0.73 I: 0.75 D: 0.48 Cyclic)
com.sun.j2ee.blueprints.petstore.controller.ejb.actions
com.sun.j2ee.blueprints.petstore.controller.web
com.sun.j2ee.blueprints.petstore.controller.web.actions
com.sun.j2ee.blueprints.petstore.controller.ejb.actions (CC: 6 AC: 0 Ca: 0 Ce: 24 A: 0 I: 1 D: 0 Cyclic)
com.sun.j2ee.blueprints.petstore.controller.events (CC: 12 AC: 0 Ca: 3 Ce: 4 A: 0 I: 0.57 D: 0.43)
com.sun.j2ee.blueprints.petstore.controller.ejb (CC: 3 AC: 8 Ca: 2 Ce: 6 A: 0.73 I: 0.75 D: 0.48 Cyclic)

March 3rd, 2004

Object Oriented Design Course

41

Examples - Pet Store (2)

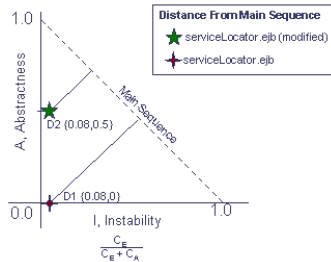
Used By - Efferent Dependencies (51 Packages)
com.sun.j2ee.blueprints.petstore.controller.ejb (CC: 3 AC: 8 Ca: 2 Ce: 6 A: 0.73 I: 0.75 D: 0.48 Cyclic)
com.sun.j2ee.blueprints.cart.ejb
com.sun.j2ee.blueprints.customer.ejb
com.sun.j2ee.blueprints.servicelocator
com.sun.j2ee.blueprints.servicelocator.ejb
com.sun.j2ee.blueprints.waf.controller.ejb
com.sun.j2ee.blueprints.waf.controller.ejb.action
com.sun.j2ee.blueprints.waf.controller.ejb
com.sun.j2ee.blueprints.waf.event
com.sun.j2ee.blueprints.waf.event
com.sun.j2ee.blueprints.waf.event
com.sun.j2ee.blueprints.waf.exceptions
com.sun.j2ee.blueprints.petstore.controller.ejb.actions (CC: 6 AC: 0 Ca: 0 Ce: 24 A: 0 I: 1 D: 0 Cyclic)

March 3rd, 2004

Object Oriented Design Course

42

How to improve the rating?



March 3rd, 2004

Object Oriented Design Course

43

Profiling

- A profiler is a program that can track the performance of another program
- Used to solve performance problems
 - "How come a simple file viewer take 30 seconds to start, and over 2 minutes to find text in a medium text file?"
- Used to solve memory problems
 - "Why does my text editor take 50MB on startup, and 300MB after a hour of work?"

March 3rd, 2004

Object Oriented Design Course

44

Performance Tuning

- How can I make my program faster?
- The 80 / 20 Principle
 - 80% of the time is spent in 20% of the code
 - Key Issue: Find the bottlenecks
- Classic Mistake: Assume the bottlenecks
 - You can't know where they'll be
- Classic Mistake II: Optimize in Advance
 - Start with the right design, then optimize

March 3rd, 2004

Object Oriented Design Course

45

Performance Tuning Process

- Step 1: Identify the bottlenecks
 - Use a profiler!
 - Find & measure the bottlenecks
- Step 2: Decide how to solve bottlenecks
 - Make them faster (new algorithm, data str.)
 - Call them less often (caching, lazy execution)
- Step 3: Measure again
 - Only way to make sure improvement happened

March 3rd, 2004

Object Oriented Design Course

46

Eclipse Profiler Plugin

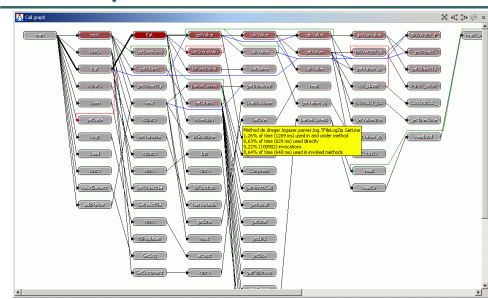
- We'll demonstrate on the (free!) Eclipse Profiler Plugin
- What is tracked
 - CPU
 - Memory usage
 - Number of objects
 - Object graph
 - Call graph

March 3rd, 2004

Object Oriented Design Course

47

Call Graph

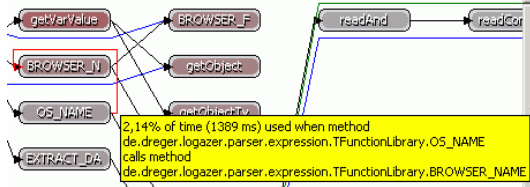


March 3rd, 2004

Object Oriented Design Course

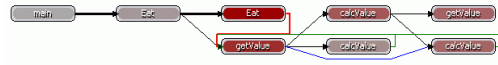
48

Call hint



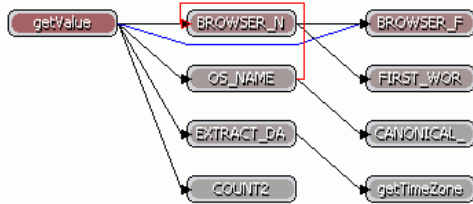
March 3rd, 2004 Object Oriented Design Course 49

Callers



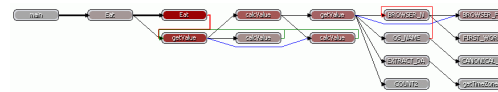
March 3rd, 2004 Object Oriented Design Course 50

Calles



March 3rd, 2004 Object Oriented Design Course 51

Callers and calles



March 3rd, 2004 Object Oriented Design Course 52

CPU Profiling

- How many invocations were?
- How much time have we spent in a package / class / method?
- Finds the bottlenecks
 - Just sort by time or number of invocations

March 3rd, 2004 Object Oriented Design Course 53

Packages

Package	Inv.	Time	%	Time/Inv.	Total time	Inv. time
org.apache.commons.framework	9862	13.14	22%	0.001335	208	41%
org.apache.commons.lang	1407	44%	10.6%	0.031280	183	9%
org.apache.commons.collections	4754	0.27	0%	0.000057	112	0%
org.apache.commons.io	4276	0.00	0%	0.000000	296	0%
org.apache.commons.math	2571	0.34	0%	0.000132	100	0%
org.apache.commons.logging	31	0.00	0%	0.000000	20	0%
org.apache.commons.lang2	52	0.01	0%	0.000192	50	0%
org.apache.commons.math2	4070	0.00	0%	0.000000	100	0%
org.apache.commons.lang3	2571	0.34	0%	0.000132	110	0%
org.apache.commons.math3	8592	14.07	14%	0.016307	200	7%
org.apache.commons.math4	2	0.00	0%	0.000000	2003	40%
org.apache.commons.math5	2	0.00	0%	0.000000	110	0%
org.apache.commons.math6	2	0.00	0%	0.000000	821	10%
org.apache.commons.math7	1	0.00	0%	0.000000	2033	7%
org.apache.commons.math8	34640	48.72	17%	0.001406	3809	40%
org.apache.commons.math9	111423	14.70	13%	0.000132	3944	41%
org.apache.commons.math10	40	0.01	0%	0.000250	1341	16%
org.apache.commons.math11	51	0.00	0%	0.000000	2113	40%
org.apache.commons.math12	14652	1.93	7%	0.000132	1605	7%
org.apache.commons.math13	177032	23.38	20%	0.000132	890	7%
org.apache.commons.math14	2576	0.34	0%	0.000132	1392	25%
org.apache.commons.math15	2	0.00	0%	0.000000	110	0%
org.apache.commons.math16	2	0.00	0%	0.000000	100	0%

March 3rd, 2004 Object Oriented Design Course 54

Classes

Method	Count	%	Count	%
getOwner	1862042	20.7%	13359	14.0%
getRole	1662047	5.2%	37199	4.0%
getFunction	1575512	7.8%	21462	4.9%
getRole	1489997	4.0%	23203	4.1%
getFunction	904170	2.8%	20199	3.6%
getRole	862960	3.3%	18744	3.7%
getFunction	374594	1.3%	2660	0.1%
getRole	105814	0.3%	1548	0.2%
getFunction	520807	0.1%	886	0.1%
getRole	1368192	0.4%	882	0.1%
getFunction	520807	0.1%	5208	0.1%
getRole	520807	1.8%	62736	11.2%
getFunction	5195194	7.4%	5208	0.1%
getRole	2560094	8.5%	42316	7.0%
getFunction	4282707	2.2%	13224	2.8%
getRole	2720139	8.1%	15718	3.0%
getFunction	5000100	1.9%	5659	1.0%
getRole	402070	1.4%	4516	0.8%
getFunction	1205710	0.9%	4982	0.7%
getRole	791136	0.2%	1141	0.1%
getFunction	1294992	0.4%	1123	0.1%
getRole	197282	0.0%	859	0.1%
getFunction	5586888	5.9%	4044	2.0%
getRole	3886258	12.4%	35214	6.3%
getFunction	1205710	4.1%	29265	5.0%
getRole	1789910	6.1%	20861	4.0%
getFunction	1829790	5.3%	20663	3.7%
getRole	1890219	6.3%	13662	2.4%
getFunction	4752596	1.6%	12351	2.3%

March 3rd, 2004 Object Oriented Design Course 55

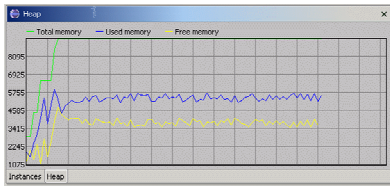
Methods

Method	Count	%	Count	%
getOwner	1862042	20.7%	13359	14.0%
getRole	1662047	5.2%	37199	4.0%
getFunction	1575512	7.8%	21462	4.9%
getRole	1489997	4.0%	23203	4.1%
getFunction	904170	2.8%	20199	3.6%
getRole	862960	3.3%	18744	3.7%
getFunction	374594	1.3%	2660	0.1%
getRole	105814	0.3%	1548	0.2%
getFunction	520807	0.1%	886	0.1%
getRole	1368192	0.4%	882	0.1%
getFunction	520807	0.1%	5208	0.1%
getRole	520807	1.8%	62736	11.2%
getFunction	5195194	7.4%	5208	0.1%
getRole	2560094	8.5%	42316	7.0%
getFunction	4282707	2.2%	13224	2.8%
getRole	2720139	8.1%	15718	3.0%
getFunction	5000100	1.9%	5659	1.0%
getRole	402070	1.4%	4516	0.8%
getFunction	1205710	0.9%	4982	0.7%
getRole	791136	0.2%	1141	0.1%
getFunction	1294992	0.4%	1123	0.1%
getRole	197282	0.0%	859	0.1%
getFunction	5586888	5.9%	4044	2.0%
getRole	3886258	12.4%	35214	6.3%
getFunction	1205710	4.1%	29265	5.0%
getRole	1789910	6.1%	20861	4.0%
getFunction	1829790	5.3%	20663	3.7%
getRole	1890219	6.3%	13662	2.4%
getFunction	4752596	1.6%	12351	2.3%

March 3rd, 2004 Object Oriented Design Course 56

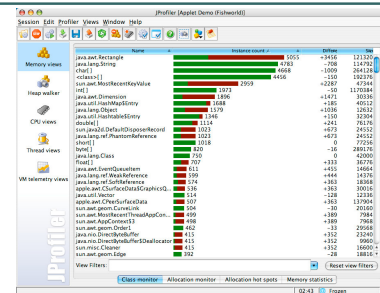
Memory

- How much memory does the program take?
- Are there memory leaks?



March 3rd, 2004 Object Oriented Design Course 57

Memory Monitor



March 3rd, 2004 Object Oriented Design Course 58

Profiling - summary

- How does my application behave?
- What are the critical paths?
- Where are the bottlenecks?
- Do I have memory leaks?
 - Java users - you are not exempted!

March 3rd, 2004 Object Oriented Design Course 59

Summary

- Personal Productivity Tools
 - Refactoring
 - Static Analysis & Metrics
 - Profilers
- Use them!
- There's more - see [Eclipse Plugins](#)

March 3rd, 2004 Object Oriented Design Course 60