



Cava := Eclipse asSmalltalkPlugin.

Johan Brichau (Université de Louvain-la-Neuve, Belgium)

Coen De Roover (Vrije Universiteit Brussel, Belgium)

- **Why use Eclipse in Smalltalk tools ?**
- **Cava**
 - **JavaConnect**
 - **Eclipse interface**
- **Example demos:**
 - **SOUL**
 - **IntensiVE**
 - **Template Queries**

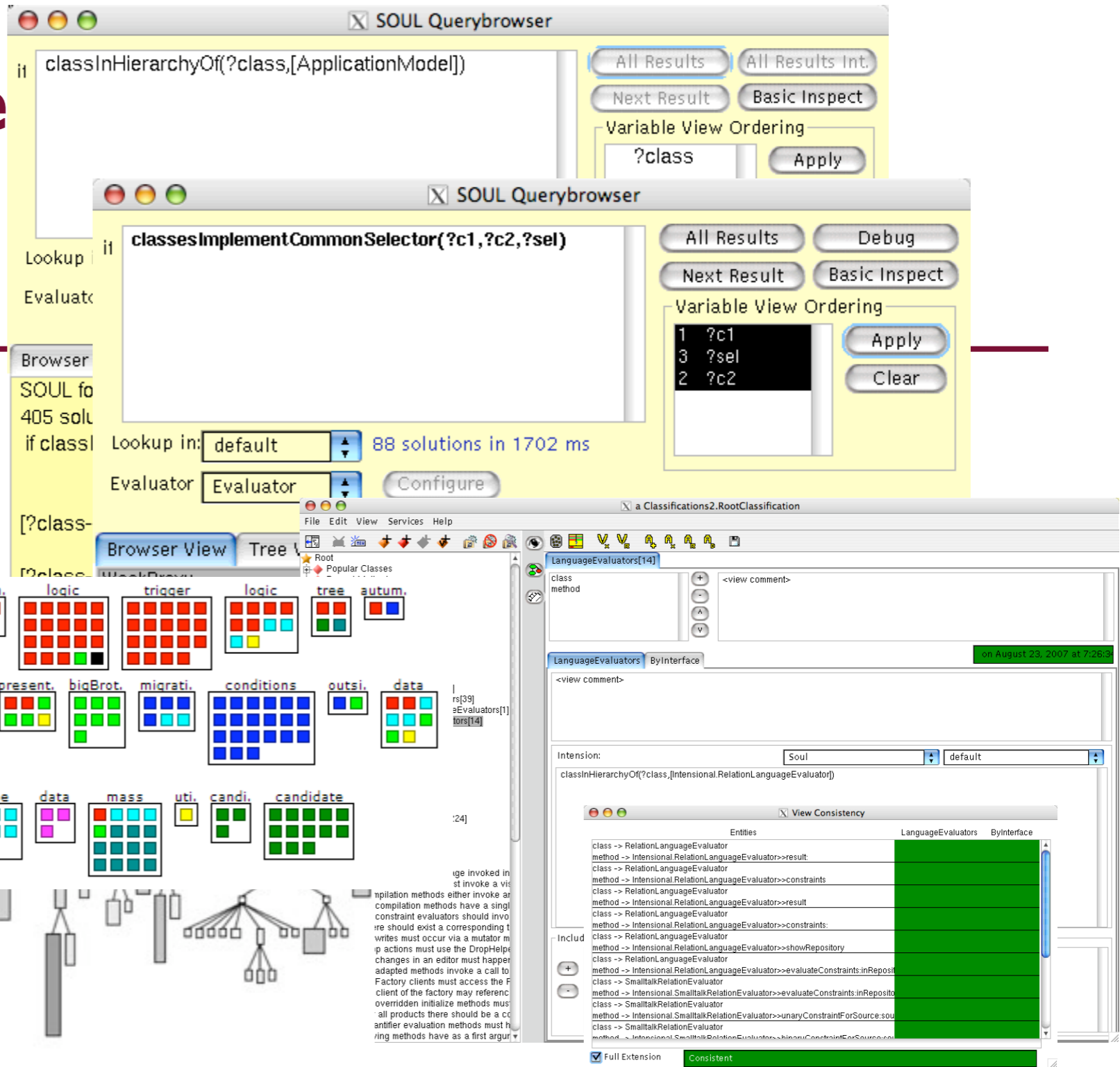
Smalltalk-based program-analysis tools



Smalltalk-based program-analysis tools



Smalltalk-base



The screenshot displays the SOUL Querybrowser interface, which is used for querying Smalltalk objects. It shows two windows with queries and their results. The top window has the query `classInHierarchyOf(?class,[ApplicationModel])` and buttons for 'All Results', 'All Results Int.', 'Next Result', and 'Basic Inspect'. The bottom window has the query `classesImplementCommonSelector(?c1,?c2,?sel)` and buttons for 'All Results', 'Debug', 'Next Result', and 'Basic Inspect'. A 'Variable View Ordering' panel is visible, showing a list of variables: 1 ?c1, 3 ?sel, 2 ?c2, with 'Apply' and 'Clear' buttons.

Below the query windows, a class browser is visible, showing a tree of classes. The classes are represented by icons with colored squares. The browser is titled 'Browser View' and 'Tree View'. The classes shown include: securit., kickout, view, clean., logic, trigger, logic, tree, autum., miqra., matchin., util, present., bigBrot., miqrati., conditions, outsi., data, core, dataB., test, database, data, mass, uti., candi., candidate.

On the right side, a 'View Consistency' window is open, showing a table of consistency results. The table has columns for 'Entities', 'LanguageEvaluators', and 'ByInterface'. The table contains several rows of class and method names, with a green bar indicating 'Consistent' status. The 'Full Extension' checkbox is checked.

Working with Java (source) code in Smalltalk-based program-analysis tools



Working with Java (source) code in Smalltalk-based program-analysis tools



Working with Java (source) code in Smalltalk-based program-analysis tools



Working with Java (source) code in Smalltalk-based program-analysis tools



SourceForge.net
Create, Participate, Evaluate

Parsing Java Code 
Written by Johan Fabry
Website design Copyright © 2005 Sverri Olsen.



Parsing Java Code 
Written by Johan Fabry
Website design Copyright © 2005 Sverri Olsen.



.....

- **Implement symbolic resolution**
- **Implement call-graph, control-flow, ... analysis**
- **Causal link is lost!**
 - **Need mutable Java source code representation**
 - **Need Java exporter**
- **...**
- **Keep up with java language changes !!**

But we can reuse:



JDT :

<u>IBinding</u>	<u>resolveBinding()</u>
---------------------------------	---

Resolves and returns the binding for the entity referred to by this name.

<u>IMethodBinding</u>	<u>resolveMethodBinding()</u>
---------------------------------------	---

Resolves and returns the binding for the method invoked by this expression.

<u>IType[]</u>	<u>getSubclasses(IType type)</u>
--------------------------------	--

Returns the direct resolved subclasses of the given class or interface.

<u>IType[]</u>	<u>getSubtypes(IType type)</u>
--------------------------------	--

Returns the direct resolved subtypes of the given type.

<u>IType</u>	<u>getSuperclass(IType type)</u>
------------------------------	--

Returns the resolved superclass of the given class, or interface, or if the given type is an interface.

<u>IType[]</u>	<u>getSuperInterfaces(IType type)</u>
--------------------------------	---

Returns the direct resolved interfaces that the given type implements or inherits from.

<u>IType[]</u>	<u>getSupertypes(IType type)</u>
--------------------------------	--

Returns the resolved supertypes of the given type, in the type hierarchy.



Plugins :

<u>CallGraph</u>	<u>getCallGraph()</u>
----------------------------------	---------------------------------------



Cava

JavaConnect

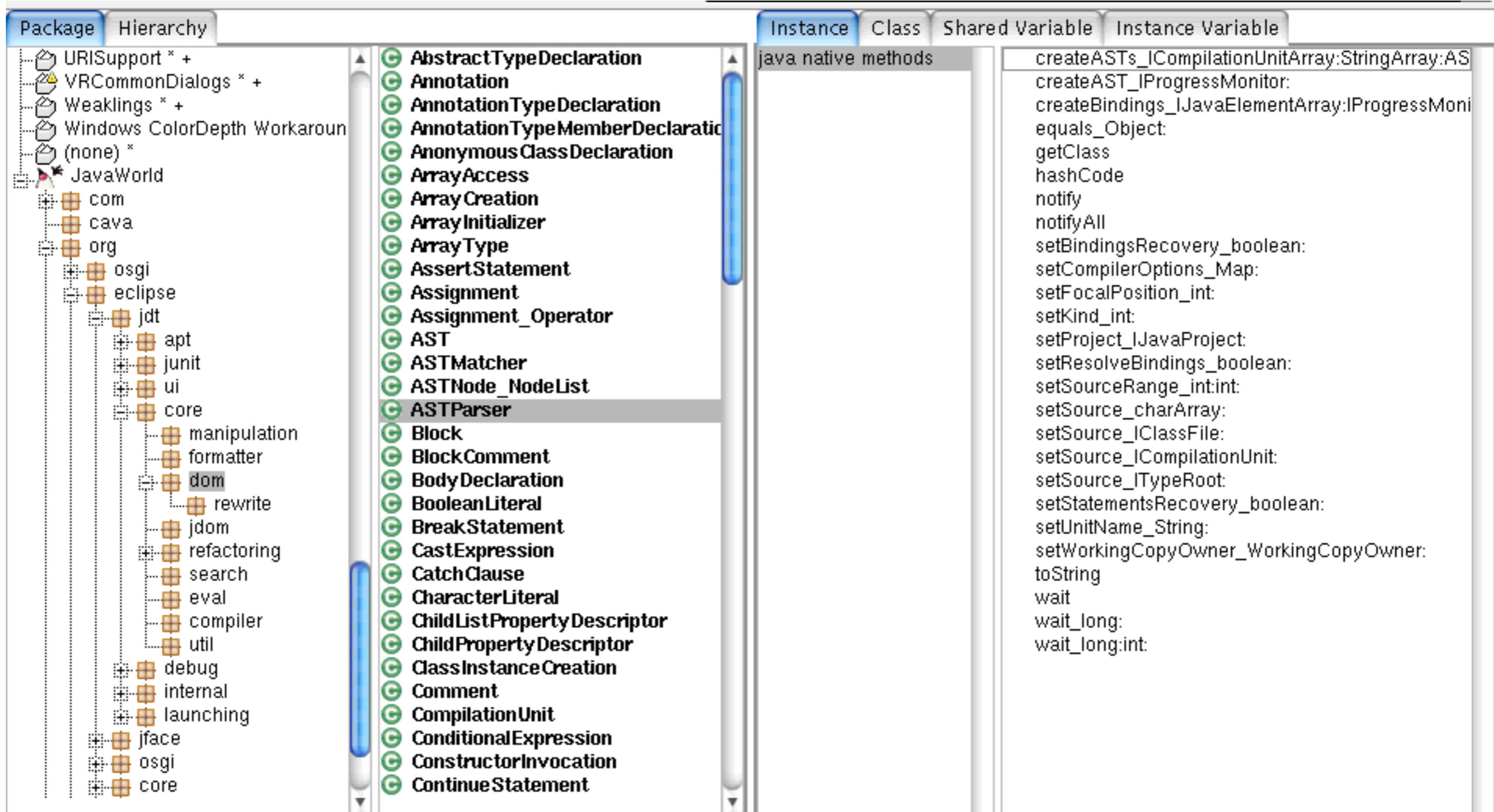
**Launch Java applications
inside Smalltalk**

Eclipse interface

**Java parsetrees as Smalltalk
object-trees**

Library of logic SOUL rules

Reason about Java code



The screenshot displays the Eclipse IDE interface with the JavaConnect plugin. The left pane shows a package hierarchy for 'JavaWorld', with 'org.eclipse.jdt.core' expanded. The middle pane lists Java AST nodes, with 'ASTParser' selected. The right pane shows instance variables for 'ASTParser', including methods like 'createASTs_ICompilationUnitArray:StringArray:AS' and 'wait_long:int'.

Package	Hierarchy	Instance	Class	Shared Variable	Instance Variable
URISupport * +		java native methods			createASTs_ICompilationUnitArray:StringArray:AS
VRCommonDialogs * +					createAST_IProgressMonitor:
Weaklings * +					createBindings_IJavaElementArray:IProgressMoni
Windows ColorDepth Workaroun					equals_Object:
(none) *					getClass
JavaWorld					hashCode
com					notify
cava					notifyAll
org					setBindingsRecovery_boolean:
osgi					setCompilerOptions_Map:
eclipse					setFocalPosition_int:
jdt					setKind_int:
apt					setProject_IJavaProject:
junit					setResolveBindings_boolean:
ui					setSourceRange_int:int:
core					setSource_charArray:
manipulation					setSource_IClassFile:
formatter					setSource_ICompilationUnit:
dom					setSource_ITypeRoot:
rewrite					setStatementsRecovery_boolean:
jdom					setUnitName_String:
refactoring					setWorkingCopyOwner_WorkingCopyOwner:
search					toString
eval					wait
compiler					wait_long:
util					wait_long:int:
debug					
internal					
launching					
jface					
osgi					
core					

Cava's SOUL library



Cava Console

Stop Eclipse Start Soot

Eclipse workspace location:
 Choose

Eclipse project:
 Cava

Eclipse project main class:



Eclipse workspace

```
package za.org.meraka.dictionarymaker.algorithms;
import java.util.*;

/**
 * DefaultAndRefineRules
 * This is an implementation of the DefaultAndRefine algorithms for both rule
 * extraction and pronunciation prediction. It is based on the DMRule
 * implementation of the algorithms.
 * @author Thoms Fogwill <tfogwill@users.sourceforge.net>
 * @date Feb 28, 2006
 */
public class DefaultAndRefineRules implements PronunciationPredictor,
RuleExtractor {
    // ...
}

```

Cava's SOUL library



Cava Console

Stop Eclipse Start Soot

SOUL Querybrowser

?project isJavaProject

All Results Debug

Next Result Basic Inspect

Variable View Ordering

?project Apply Clear

Lookup in: JavaEclipse

Evaluator: Evaluator Configure

Browser View Tree View Text View

Java - DictionaryMaker/src/za/org/meraka/dictionarymaker/algorithms/DefaultAndRefineRules.java - Eclipse SDK - /Users/jbrichau/Research/EclipseWorkspace

```
package za.org.meraka.dictionarymaker.algorithms;
import java.util.*;

@import java.util.*;

/**
 * DefaultAndRefineRules
 * This is an implementation of the DefaultAndRefine algorithms for both rule
 * extraction and pronunciation prediction. It is based on the DMRule
 * implementation of the algorithms.
 * @author Thoms Fogwill <tfogwill@users.sourceforge.net>
 * @date Feb 28, 2006
 */
```

Cava's SOUL library



Cava Console

Stop Eclipse Start Soot

Eclipse wo
/Volumes
Eclipse pro
All proje
Eclipse pro

SOUL Querybrowser

?project isJavaProject

All Results Debug
Next Result Basic Inspect

Variable View Ordering

?project Apply Clear

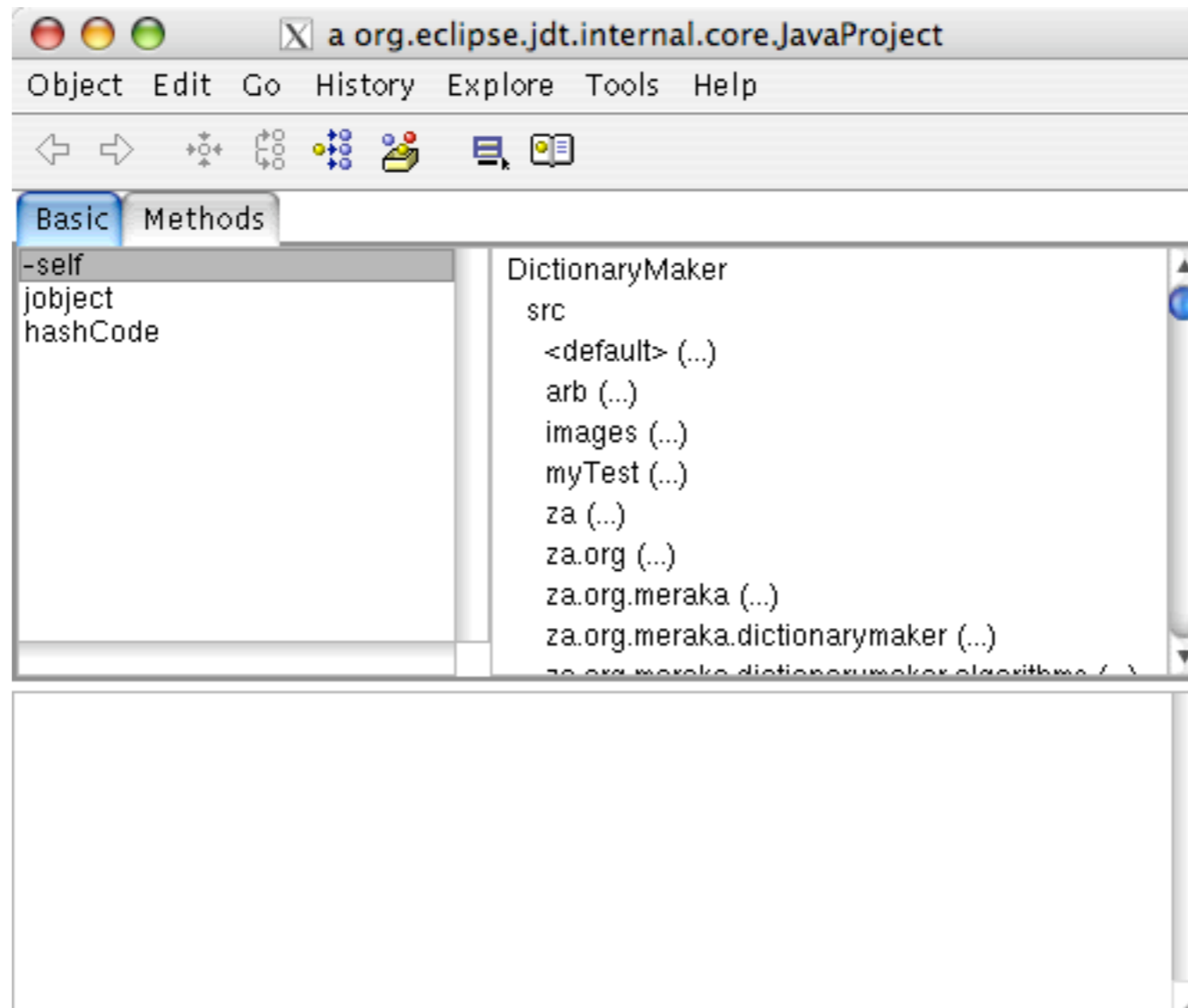
Lookup in: JavaEclipse 10 solutions in 1 ms

Evaluator Evaluator Configure

Browser View Tree View Text View

```
JavaConnect-Tests <project root> <default> (...) javaconnect (...) javaconnect.tests (...) javaconnect.tests.subtests (...) /System/Library/Frameworks/JavaVM.framework/Versions/1.5.0/Classes/classes.jar <default> (...)  
Conduits <project root> <default> (...) Conduits (...) /System/Library/Frameworks/JavaVM.framework/Versions/1.5.0/Classes/classes.jar <default> (...)  
DictionaryMaker src <default> (...) arb (...) images (...) myTest (...) za (...) za.org (...) za.org.meraka (...) za.org.meraka.dictionarymaker.algorithms <default> (...)  
Conduits in AspectJ <project root> <default> (...) Conduits (...) /System/Library/Frameworks/JavaVM.framework/Versions/1.5.0/Classes/classes.jar <default> (...)  
CavaPlugin src <default> (...) cava (...) /System/Library/Frameworks/JavaVM.framework/Versions/1.5.0/Classes/classes.jar <default> (...)  
EclipseJavaConnect <project root> <default> (...) workspace (...) /System/Library/Frameworks/JavaVM.framework/Versions/1.5.0/Classes/classes.jar <default> (...)  
Cava-Tests src <default> (...) /System/Library/Frameworks/JavaVM.framework/Versions/1.5.0/Classes/classes.jar <default> (...)  
Alcyone dev/bds/src/main/java <default> (...) com (...) com.iba (...) com.iba.tcs (...) com.iba.tcs.bds (...) com.iba.tcs.bds.proxy (...) dev <default> (...)  
EclipseParserTest <project root> <default> (...) edu (...) edu.vub (...) edu.vub.soul (...) edu.vub.soul.java (...) edu.vub.soul.java.parser <default> (...)  
IntroAOSDExample src <default> (...) /System/Library/Frameworks/JavaVM.framework/Versions/1.5.0/Classes/classes.jar <default> (...)
```


	the given instance:
String	toString() Returns a string representation of this node suitable for debugging purposes only.



Customizing JDT



The screenshot shows the Eclipse IDE interface for a Java project named 'JavaProject' with the file 'displayString'. The top menu bar includes 'Browser', 'Edit', 'Find', 'View', 'Package', 'Class', 'Protocol', 'Method', 'Tools', and 'Help'. Below the menu is a toolbar with various icons and a 'Find:' search box. The main workspace is divided into several panes:

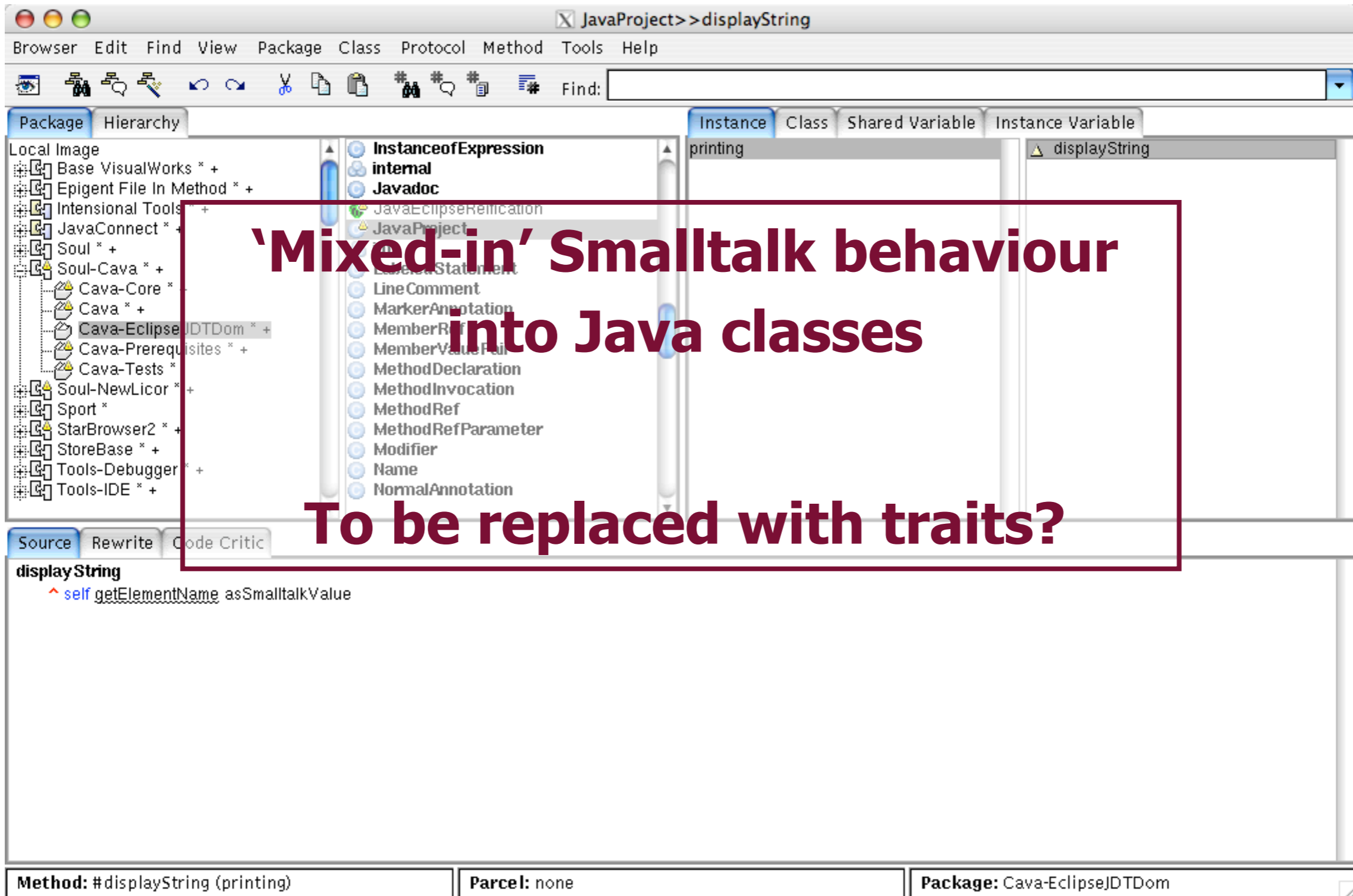
- Package Hierarchy:** Shows a tree view of packages. The 'Cava-EclipseJDTDom' package is selected.
- Instance of Expression:** A list of Java classes and annotations, including 'InstanceofExpression', 'internal', 'Javadoc', 'JavaEclipseReification', 'JavaProject', 'jdt', 'LabeledStatement', 'LineComment', 'MarkerAnnotation', 'MemberRef', 'MemberValuePair', 'MethodDeclaration', 'MethodInvocation', 'MethodRef', 'MethodRefParameter', 'Modifier', 'Name', and 'NormalAnnotation'. 'JavaProject' is selected.
- Instance:** A pane showing the 'printing' instance.
- Instance Variable:** A pane showing the 'displayString' instance variable.
- Source:** A code editor showing the source code for the 'displayString' method:

```
displayString  
^ self getElementName asSmalltalkValue
```

At the bottom of the IDE, there are three status bars:

- Method:** #displayString (printing)
- Parcel:** none
- Package:** Cava-EclipseJDTDom

Customizing JDT



The screenshot shows the Eclipse IDE interface with the JavaProject displayString instance selected. The Package Hierarchy on the left shows the project structure, including Cava-EclipseJDTDom. The Instance Variable view on the right shows the displayString instance. The Source view at the bottom shows the code for displayString, which includes a call to getElementName asSmalltalkValue. A red-bordered text box is overlaid on the center of the screenshot, containing the text: **'Mixed-in' Smalltalk behaviour into Java classes** and **To be replaced with traits?**

'Mixed-in' Smalltalk behaviour into Java classes

To be replaced with traits?

Method: #displayString (printing) Parcel: none Package: Cava-EclipseJDTDom

Customized behaviour



The screenshot shows the SOUL Querybrowser application window. The title bar reads "SOUL Querybrowser". The main query area contains the text "it ?project isJavaProject". To the right of the query area are several control buttons: "All Results", "Debug", "Next Result", and "Basic Inspect". Below these is a "Variable View Ordering" section with a list containing "?project" and buttons for "Apply" and "Clear".

Below the query area, there are two dropdown menus: "Lookup in:" set to "JavaEclipse" and "Evaluator" set to "Evaluator". To the right of the "Lookup in:" dropdown is the text "10 solutions in 1 ms". To the right of the "Evaluator" dropdown is a "Configure" button.

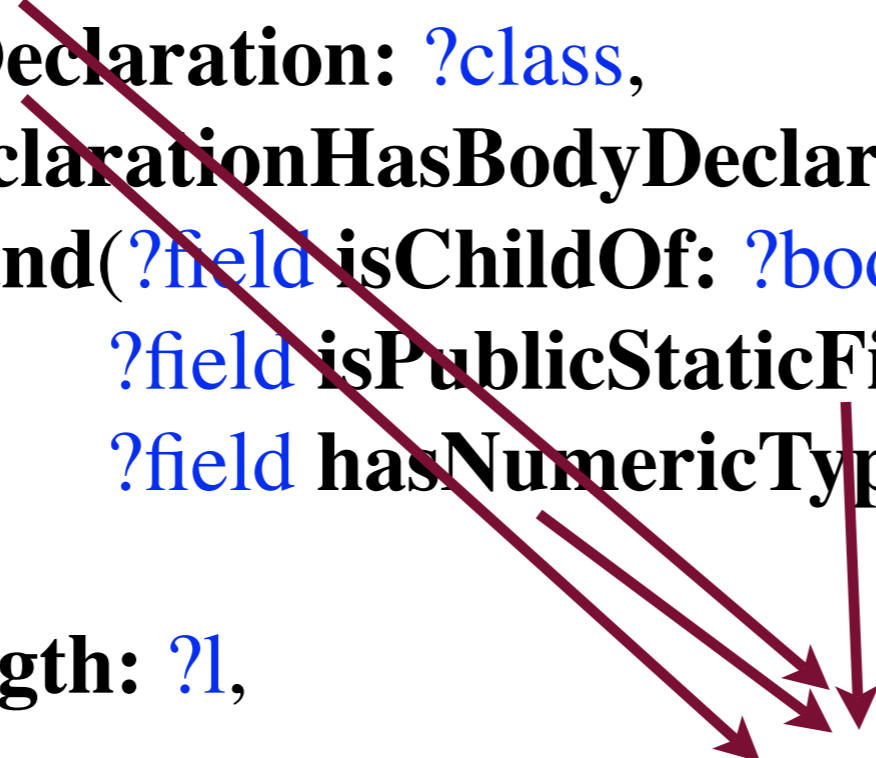
At the bottom of the window, there are three tabs: "Browser View" (which is selected), "Tree View", and "Text View". The "Browser View" tab displays a list of project names:

- JavaConnect-Tests
- Conduits
- DictionaryMaker
- Conduits in AspectJ
- CavaPlugin
- EclipseJavaConnect
- Cava-Tests
- Alcyone
- EclipseParserTest
- IntroAOSDExample

- **“I want to refactor all for-loops that can use the enhanced Java 5 style”**
- **“I want to detect all classes that should be enums”**
- **“Events published on the bus should not be modified anymore”**
- **“Custom events should be subclasses of EventX or EventY”**
- **...**

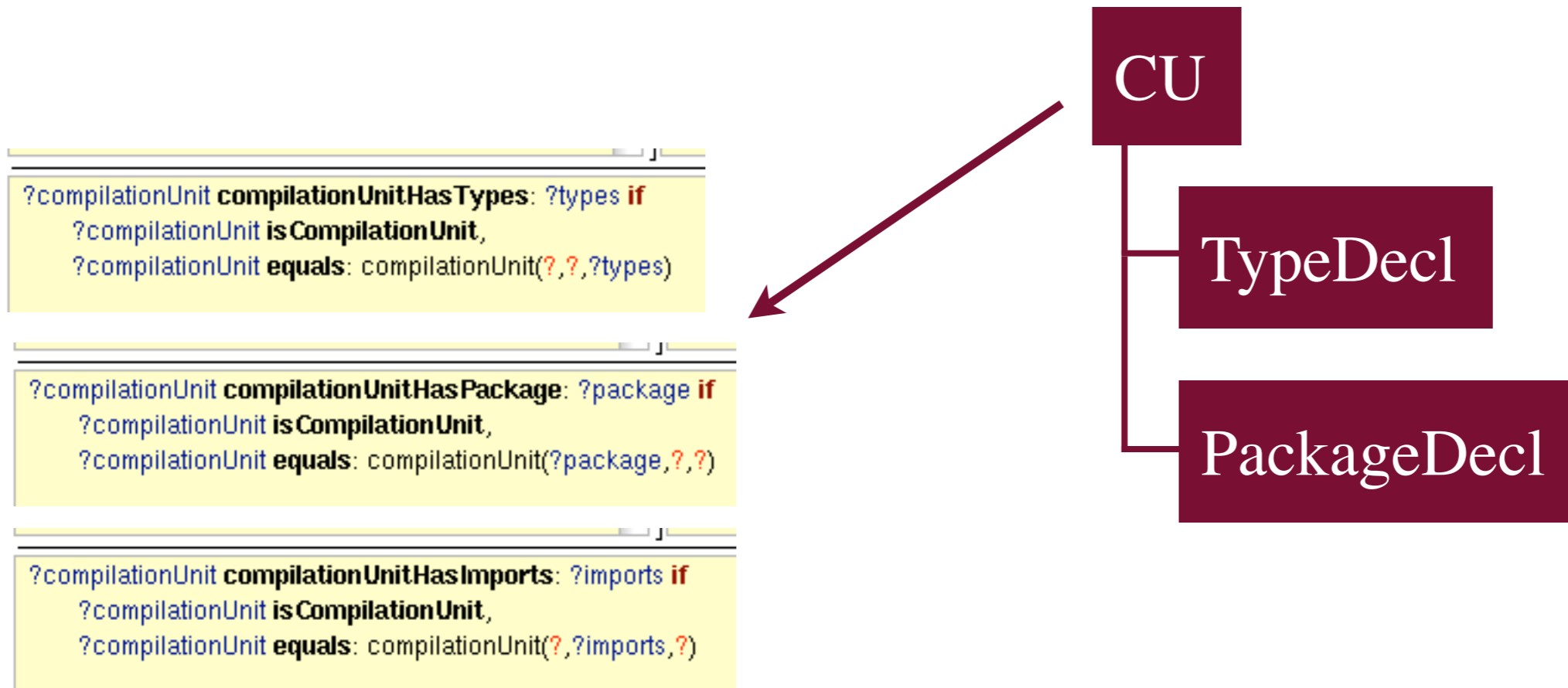
```
?cu isCompilationUnit,  
?cu hasClassDeclaration: ?class,  
?class classDeclarationHasBodyDeclarations: ?body,  
findall(?field, and(?field isChildOf: ?body,  
                    ?field isPublicStaticFinalFieldDeclaration,  
                    ?field hasNumericType),  
        ?fields),  
?fields hasLength: ?1,  
[?1 > 2]
```

```
?cu isCompilationUnit,  
?cu hasClassDeclaration: ?class,  
?class classDeclarationHasBodyDeclarations: ?body,  
findall(?field,and(?field isChildOf: ?body,  
                    ?field isPublicStaticFinalFieldDeclaration,  
                    ?field hasNumericType),  
        ?fields),  
?fields hasLength: ?1,  
[?1 > 2]
```

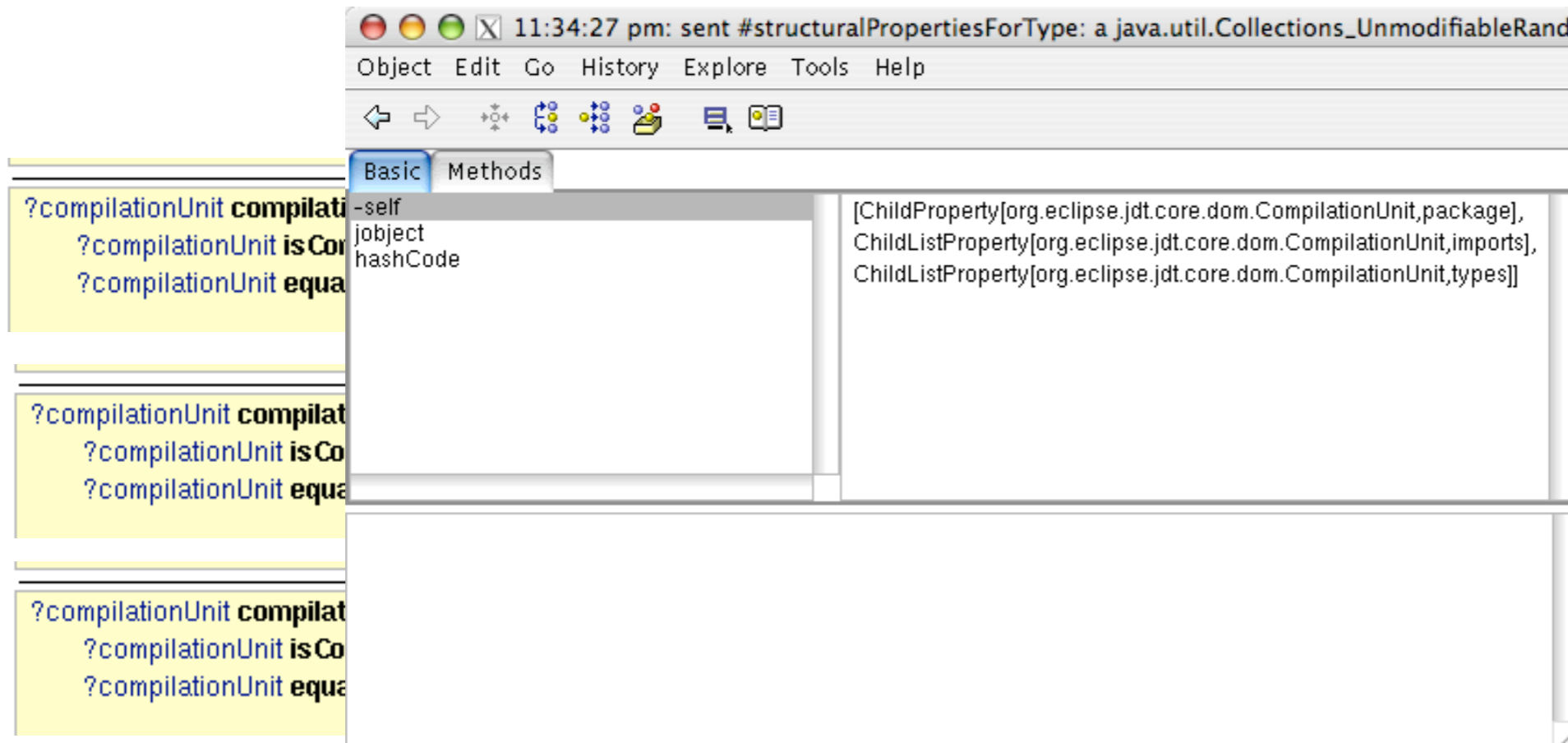


**Automatically derived from
ASTNode's property descriptors**

Automatic generation of AST navigation



Automatic generation of AST navigation



For-loops Example



```
for (Iterator i = c.iterator(); i.hasNext(); ) {  
    String s = (String) i.next();  
    ...  
}
```



```
for (String s : c) {  
    ...  
}
```

For-loops Example



The screenshot shows the Eclipse IDE with a Smalltalk plugin interface. The main window is titled "a Classifications2.RootClassification". The left sidebar shows a project tree with "Enhanced For Loops" selected. The main area displays the "Enhanced For Loops" configuration page, which includes a description: "All For-loops that can use the enhanced Java-5 style should also follow it". Below this, there is a section for "All for loops" with a "PreJava5ForLoop" tab and a "Never" button. A modal dialog titled "Evaluation of All for loops intension" is open in the center, showing a progress bar at 14%. The dialog also displays the intension: "?forStatement isChildC equals(forStatement(?i ?forStatement))". The bottom of the main window has "Includes" and "Excludes" sections with plus and minus buttons.

For-loops example



Entities	All for loops	PreJava5ForLoop
cu -> ContainerImpl.java forStatement -> for (Iterator iter=components.iterator(); iter.hasNext();) {	Green	Red
cu -> ContainerImpl.java forStatement -> for (Iterator iter=resources.iterator(); iter.hasNext();) {	Green	Red
cu -> ContainerImpl.java forStatement -> for (Iterator iter=channels.iterator(); iter.hasNext();) {	Green	Red
cu -> ContainerImpl.java forStatement -> for (Iterator iter=resources.iterator(); iter.hasNext();) {	Green	Red
cu -> ContainerImpl.java forStatement -> for (Iterator iter=resources.iterator(); iter.hasNext();) {	Green	Red
cu -> LollipopsProxy.java forStatement -> for (int i=0; i < len; ++i) {	Green	Red
cu -> LollipopsProxy.java forStatement -> for (int i=0; i < pLollipopCount; ++i) {	Green	Red
cu -> CosNotificationNumericEventConverter.java forStatement -> for (Iterator iter=values.iterator(); iter.hasNext(); i++) {	Green	Red
cu -> LevelMatcherEditor.java forStatement -> for (Iterator<JCheckBox> iter=eventCheckBoxes.values().iterator(); iter.hasNext();) {	Green	Red
cu -> LevelMatcherEditor.java forStatement -> for (Iterator<Map.Entry<SystemEvent.Level, JCheckBox>> iter=eventCheckBoxes.entrySet().iterator(); iter.hasNext();) {	Green	Red
cu -> CosNotificationNumericEventConverter.java forStatement -> for (int i=0; i < data.length; i++) {	Green	Red
cu -> DomainControllerImpl.java forStatement -> for (int i=0; i < CATEGORY_NUM; i++) {	Green	Red
cu -> DomainControllerImpl.java	Green	Red

Full Extension **INCONSISTENT!**

Detect for-loops to be enhanced



```
?cu hasEnhancedForLoopPossibility if
  ?cu isCompilationUnit,
  ?forStatement isChildOf: ?cu,
  equals(forStatement(?initializers,?expression,?updaters,?body),?forStatement),
  not(or(and(variableDeclarationExpression(?primitiveType(int),?fragments) isChildOf: ?initializers,
    variableDeclarationFragment(?varName,?,numberLiteral(['0'])) isChildOf: ?fragments,
    member(?exp,?updaters),
    ?exp isIncrementAssignmentOfVariable: ?varName,
    arrayAccess(?,?varName) isChildOf: ?body),
    and(methodInvocation(?,?,{hasNext},?) isChildOf: ?expression,pr(methodInvocation(?,?,{next},?) isChildOf: ?body,methodInvocation(?,?,{next},?) isChildOf: ?updaters))))))
```

Detecting Accessor Methods ?



```
public Integer gethour() {  
    return this.hour;  
}
```

```
public Integer gethourlazy() {  
    if(hour==null)  
        hour = this.currentHour();  
    return hour;  
}
```

```
public Integer getBuffer() {  
    Integer temp;  
    temp = buffer;  
    buffer = null;  
    return temp;  
}
```

```
public boolean setBuffer(Integer  
i) {  
    if(buffer==null) {  
        buffer = i;  
        return true;  
    }  
    else return false;  
}
```

```
public void sethour(Integer i) {  
    if(i.intValue()<0 || i.intValue()>23) {  
    } else {  
        hour = i;  
        this.notifyDependents();  
    }  
}
```

Detecting Accessor Methods ?



```
public Integer gethour() {  
    return this.hour;  
}
```

```
public Integer gethour1()  
if(hour==null)  
hour  
return  
}
```

```
public Integer  
Integer t  
temp = buf  
buffer = null;  
return temp;  
}
```

Template Queries

```
if jtClassDeclaration(?c){  
    class ?c {  
        private ?type ?field;  
        public ?type ?name() { return ?field; }  
    }  
}
```

```
public void sethour(Integer i) {  
    if(i.intValue()<0 || i.intValue()>23) {  
    } else {  
        hour = i;  
        this.notifyDependents();  
    }  
}
```

Static analysis of Java programs

Points-to analysis

Call-graph analysis

Static analysis of Java programs

Points-to analysis

Call-graph analysis

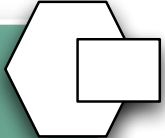
More interesting code analysis

Accessor Method Template Query

```

if jtClassDeclaration(?c){
    class ?c {
        private ?type ?field;
        public ?type ?name() { return ?field; }
    }
}

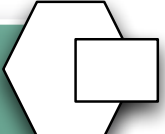
```



```

public Integer gethour() {
    return this.hour;
}
public Integer gethourlazy() {
    if(hour==null)
        hour = this.currentHour();
    return hour;
}
public Integer getBuffer() {
    Integer temp;
    temp = buffer;
    buffer = null;
    return temp;
}

```



```
if jtStatement(?s) {  
  while(?iterator.hasNext()) {  
    ?collection.add(?element);  
  }  
},  
jtExpression(?iterator){?collection.iterator()}
```

```
public List list;  
  
public void insertElement(Object x) {  
  Iterator i = list.iterator();  
  while(i.hasNext()) {  
    Object o = i.next();  
    operation(x, (Collection) this.self().list);  
  }  
}  
  
public void operation(Object o, Collection c) {  
  c.add(o);  
}
```

JavaConnect

- public repository
- <http://www.info.ucl.ac.be/~jbrichau/javaconnect.html>

Cava

- public repository (soon)
- currently focused at SOUL and IntensiVE
- working on integration for MOOSE
- looking for more!



J. Brichau, C. De Roover, K. Mens, **Open Unification for Program Query Languages**,
To be published at SCCC'07, Chile, 2007.

C. De Roover, J. Brichau, C. Noguera, T. D'Hondt, and L. Duchien. **Behavioural similarity matching using concrete source code templates in logic queries**. In *Proceedings of the ACM Sigplan Workshop on Partial Evaluation and Program Manipulation (PEPM)*, 2007.