parameterized state machines & regular expressions in JavaMOP

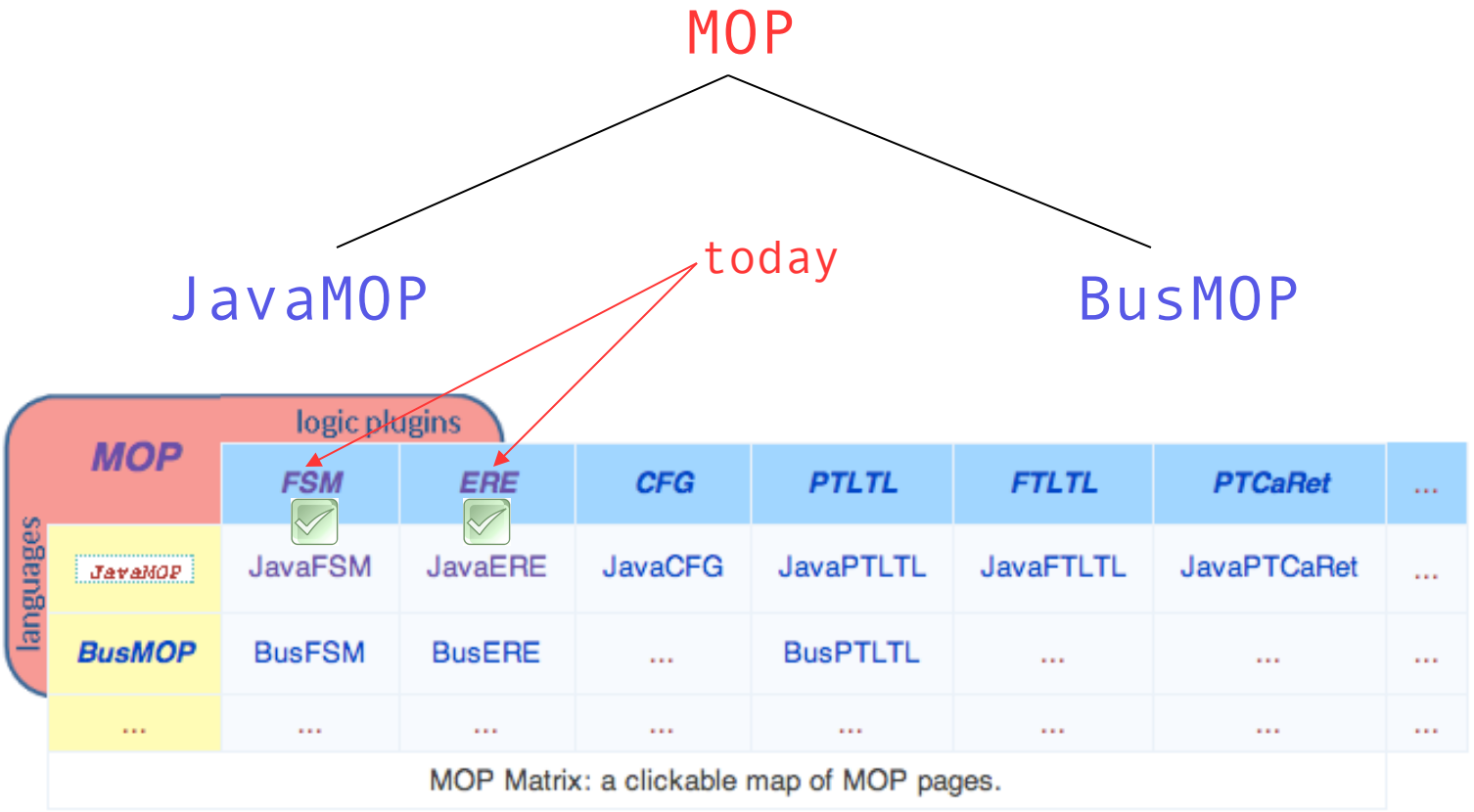
CS 119

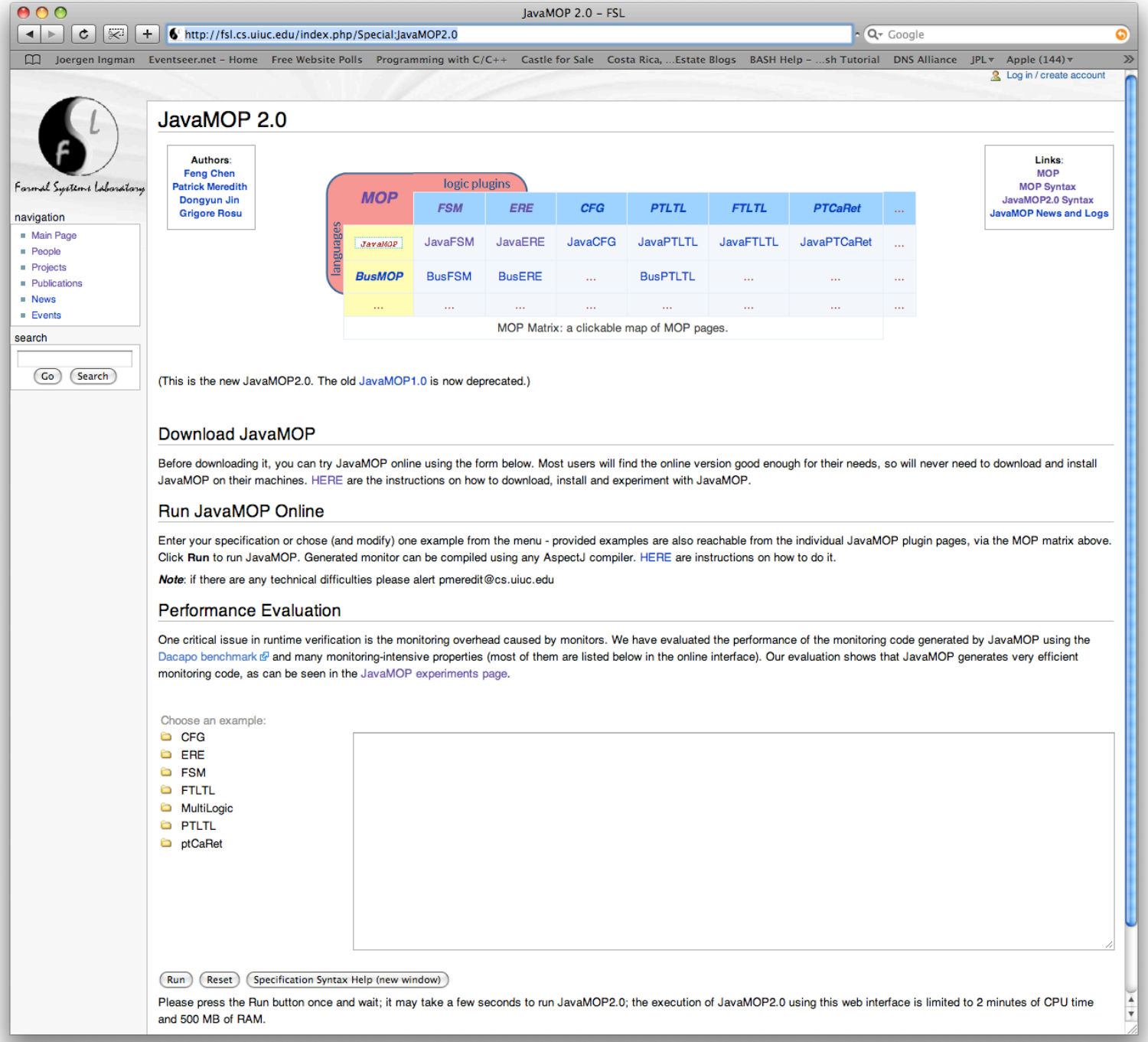
"which file?"

MOP
Monitoring Oriented Programming

- philosophy: no silver-bullet logic for specs
- MOP logic plugins (a subset):
  - FSM (finite state machines)
  - ERE (extended regular expressions)
  - LTL : PTLTL (Past-time LTL) and FTLTL (Future-time LTL)
  - CFG (context free grammars)
- MOP applications:
  - Java
  - Bus
- generic wrt. parameters
  - provide a plugin for a propositional logic, and MOP does the rest wrt. data parameterization.
  - makes designing a new logic extremely easy compared to other frameworks.

instances of MOP




JavaMOP 2.0 – FSL
http://fsl.cs.uiuc.edu/index.php/Special:JavaMOP2.0
Google
Joergen Ingman
Eventseer.net – Home
Free Website Polls
Programming with C/C++
Castle for Sale
Costa Rica, ...Estate Blogs
BASH Help – ...sh Tutorial
DNS Alliance
JPL
Apple (144)
Log in / create account
Formal System Laboratory
navigation
Main Page
People
Projects
Publications
News
Events
search
Go
Search
JavaMOP 2.0
Authors:
Feng Chen
Patrick Meredith
Dongyun Jin
Grigore Rosu
MOP
logic plugins
FSM
ERE
CFG
PTLTL
FTLTL
PTCaRet
...
languages
JavaMOP
JavaFSM
JavaERE
JavaCFG
JavaPTLTL
JavaFTLTL
JavaPTCaRet
BusMOP
BusFSM
BusERE
BusPTLTL
MOP Matrix: a clickable map of MOP pages.
Links:
MOP
MOP Syntax
JavaMOP2.0 Syntax
JavaMOP News and Logs
(This is the new JavaMOP2.0. The old JavaMOP1.0 is now deprecated.)
Download JavaMOP
Before downloading it, you can try JavaMOP online using the form below. Most users will find the online version good enough for their needs, so will never need to download and install JavaMOP on their machines. HERE are the instructions on how to download, install and experiment with JavaMOP.
Run JavaMOP Online
Enter your specification or chose (and modify) one example from the menu - provided examples are also reachable from the individual JavaMOP plugin pages, via the MOP matrix above. Click Run to run JavaMOP. Generated monitor can be compiled using any AspectJ compiler. HERE are instructions on how to do it.
Note: if there are any technical difficulties please alert pmeredit@cs.uiuc.edu
Performance Evaluation
One critical issue in runtime verification is the monitoring overhead caused by monitors. We have evaluated the performance of the monitoring code generated by JavaMOP using the Dacapo benchmark and many monitoring-intensive properties (most of them are listed below in the online interface). Our evaluation shows that JavaMOP generates very efficient monitoring code, as can be seen in the JavaMOP experiments page.
Choose an example:
CFG
ERE
FSM
FTLTL
MultiLogic
PTLTL
ptCaRet
Run
Reset
Specification Syntax Help (new window)
Please press the Run button once and wait; it may take a few seconds to run JavaMOP2.0; the execution of JavaMOP2.0 using this web interface is limited to 2 minutes of CPU time and 500 MB of RAM.

how JavaMOP works

```java
package paramere.hasnext;

import java.util.*;

public class Test1 {
    public static void main(String[] args) {
        Vector<Integer> v = new Vector();
        v.add(1); v.add(2);
        Iterator it = v.iterator();
        while(it.hasNext()) {
            int sum = (Integer)it.next() + (Integer)it.next();
            System.out.println("sum = " + sum);
        }
    }
}
```

```
package paramere.hasnext;

import java.io.*;
import java.util.*;

HasNext(Iterator i) {
    event hasnext after(Iterator i) : call(* Iterator.hasNext()) && target(i) {}
    event next before(Iterator i) : call(* Iterator.next()) && target(i) {}

    ere : (hasnext hasnext* next)*

    @fail {
        System.err.println("! hasNext() not called before calling next()");
        __RESET;
    }
}
```

under the hood

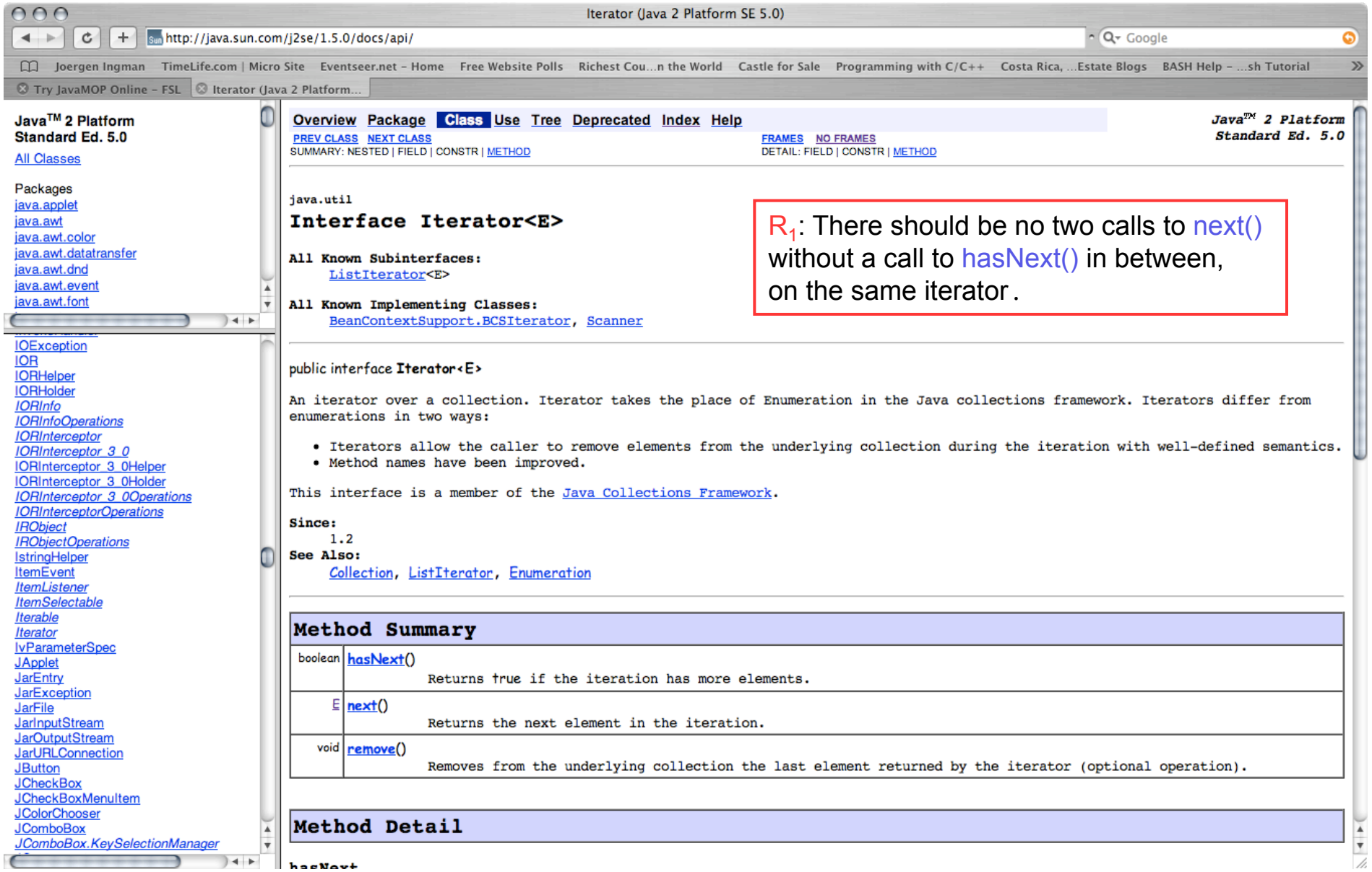
```java
public aspect HasNextMonitorAspect {
    static Map makeMap(Object key){
        if (key instanceof String) {
            return new HashMap();
        } else {
            return new ReferenceIdentityMap(AbstractReferenceMap.W
        }
    }
    static List makeList(){
        return new ArrayList();
    }

    static Map HasNext_i_Map = null;

    pointcut HasNext_hasnext1(Iterator i) : (call(* Iterator.hasNe
    after (Iterator i) : HasNext_hasnext1(i) {
        Object obj = null;
```

AspectJ

JavaMOP

java VM

properties of Java library APIs



java.util

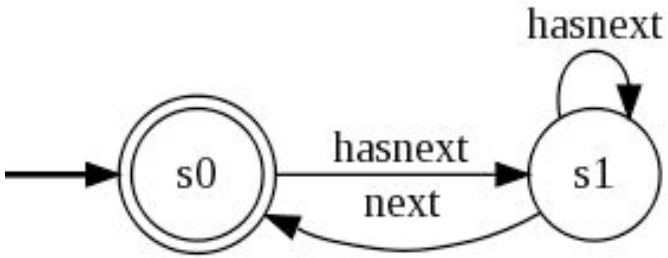
Interface Iterator<E>

All Known Subinterfaces:
ListIterator<E>

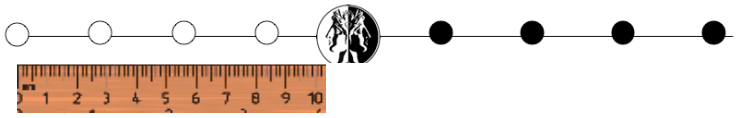
All Known Implementing Classes:
BeanContextSupport.BCSIterator, Scanner

R_1: There should be no two calls to next() without a call to hasNext() in between, on the same iterator.

(hasnext hasnext* next)*



i) total trace semantics
ii) looking for failure



note: match = validation, fail = violation (MOP has changed)

ERE property

PROPOSITIONAL!

```
package paramere.hasnext;

import java.io.*;
import java.util.*;

HasNext {
	event hasnext before() : call(* Iterator.hasNext()) {}
	event next before() : call(* Iterator.next()) {}

	ere : (hasnext hasnext* next)*

	@fail {
		System.err.println("! hasNext() not called before next()");
		__RESET;
	}
}
```

The propositional property does not flag the following error

```
public class Test2 {
    public static void main(String[] args) {
        Vector<Integer> v1 = new Vector();
        Vector<Integer> v2 = new Vector();
        v1.add(1); v1.add(3); v2.add(5); v2.add(7);
        Iterator it1 = v1.iterator();
        Iterator it2 = v2.iterator();
        int sumFirstTwo_v2 = 0;
        if(it2.hasNext())
            sumFirstTwo_v2 += (Integer)it2.next();
        if(it1.hasNext())
            sumFirstTwo_v2 += (Integer)it2.next();
        System.out.println("sumFirstTwo_v2 = " + sumFirstTwo_v2);
    }
}
```

should have been:
`if(it2.hasNext())`

execution: it2.hasNext() it2.next() it1.hasNext() it2.next()
but ignoring iterator ids: hasNext() next() hasNext() next()
matches spec perfectly => no error detected

ERE property

PARAMETERIZED!

```
package paramere.hasnext;

import java.io.*;
import java.util.*;

HasNext(Iterator i) {
	event hasnext before(Iterator i) : call(* Iterator.hasNext()) && target(i) {}
	event next before(Iterator i) : call(* Iterator.next()) && target(i) {}

	ere : (hasnext hasnext* next)*

	@fail {
		System.err.println("! hasNext() not called before next()");
		__RESET;
	}
}
```

execution: it2.hasNext() it2.next() it1.hasNext() it2.next()
violates spec for it2 => error detected

FSM property

PARAMETERIZED!

```
package paramere.hasnext;

import java.io.*;
import java.util.*;

HasNext(Iterator i) {
    event hasnext before(Iterator i) : call(* Iterator.hasNext()) && target(i) {}
    event next before(Iterator i) : call(* Iterator.next()) && target(i) {}

    fsm:

    !s0[
      hasnext -> s1
    ]

   s1[
     hasnext -> s1
     next -> s0
   ]

   @fail {
     System.err.println("! hasNext not called before next");
     __RESET;
   }
}
```

java.util

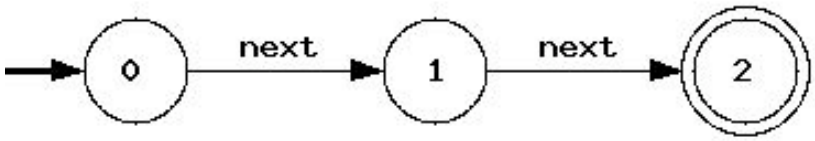
Interface Iterator<E>

All Known Subinterfaces:
ListIterator<E>

All Known Implementing Classes:
BeanContextSupport.BCSIterator, Scanner

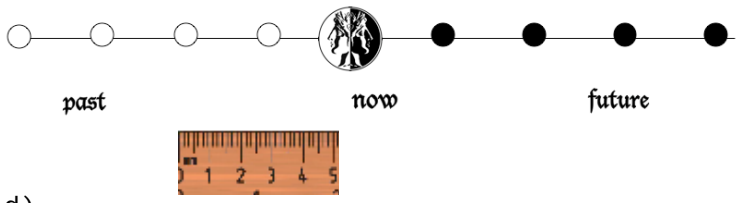
R_1: There should be no two calls to next() without a call to hasNext() in between, on the same iterator.

next next



i) suffix trace semantics

ii) looking for match



note: match = validation, fail = violation (MOP has changed)

suffix/fail property in JavaMOP

PARAMETERIZED!

```
package paramere.hasnext;

import java.io.*;
import java.util.*;

suffix HasNext(Iterator i) {
	event hasnext before(Iterator i) : call(* Iterator.hasNext()) && target(i) {}
	event next before(Iterator i) : call(* Iterator.next()) && target(i) {}

	ere : next next

	@match {
		System.err.println("! hasNext() not called before next()");
		__RESET;
	}
}
```

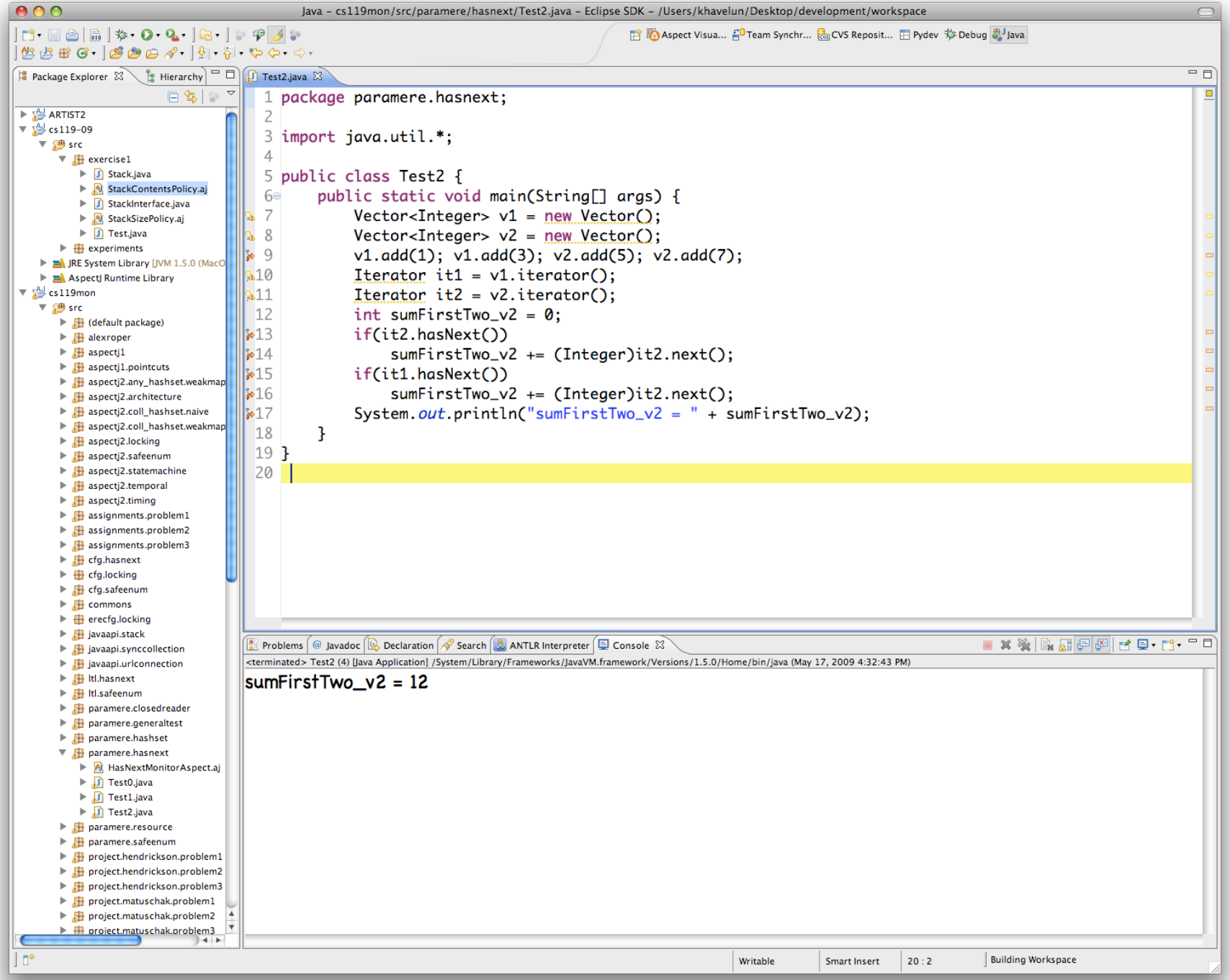
execution: it2.hasNext() it2.next() it1.hasNext() it2.next()
matches spec for it2 => error detected

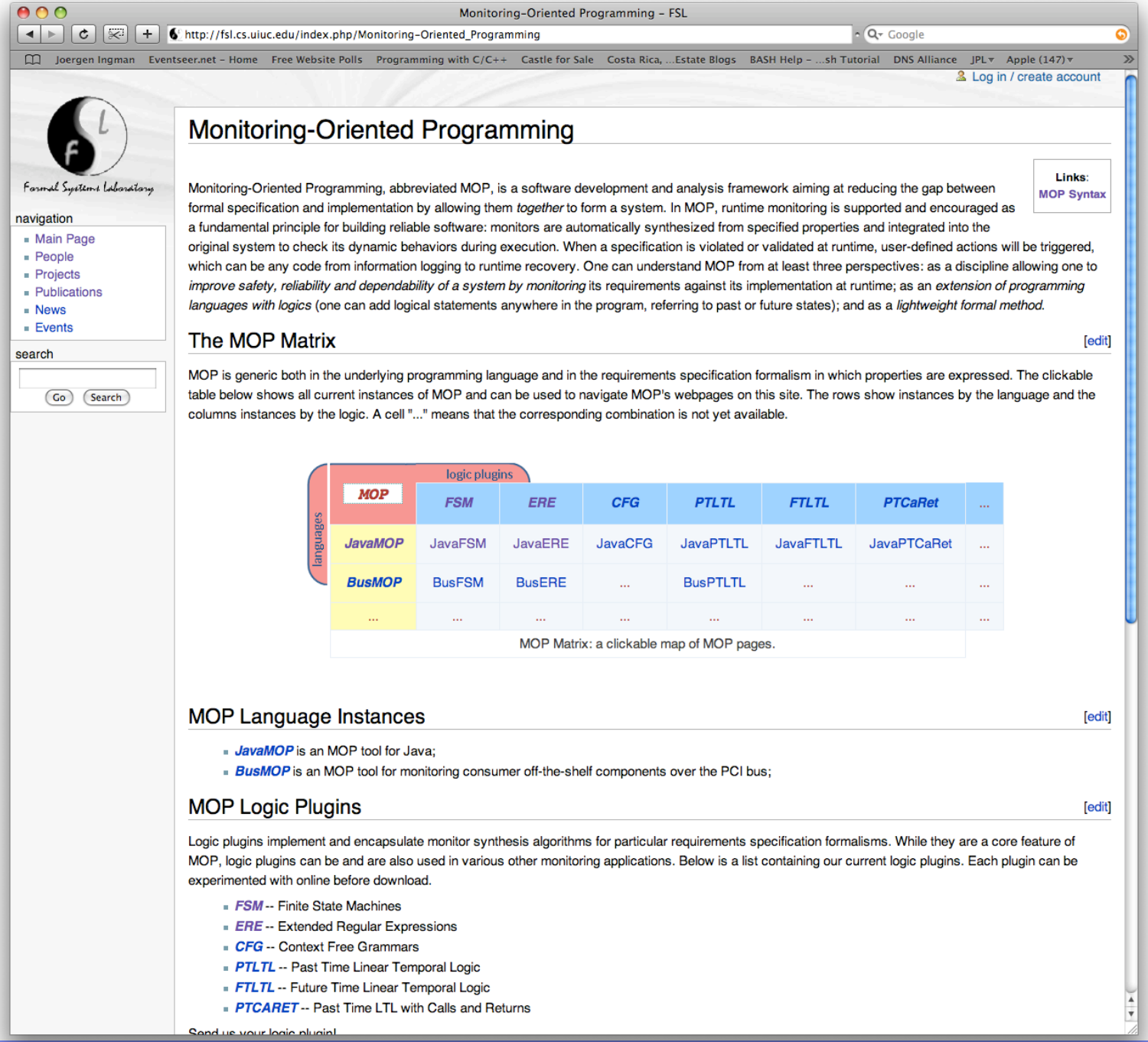
FSM property

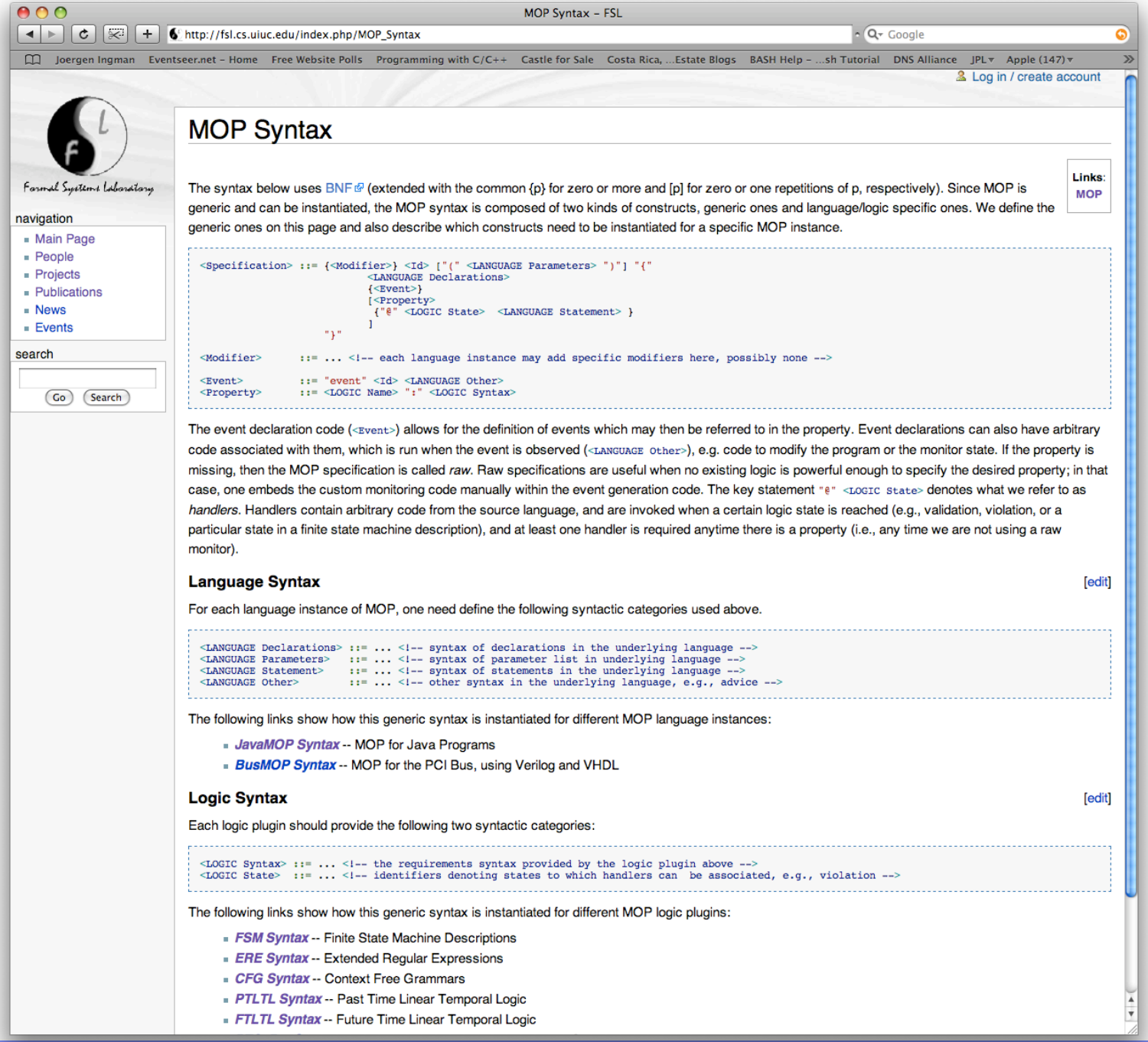
PARAMETERIZED!

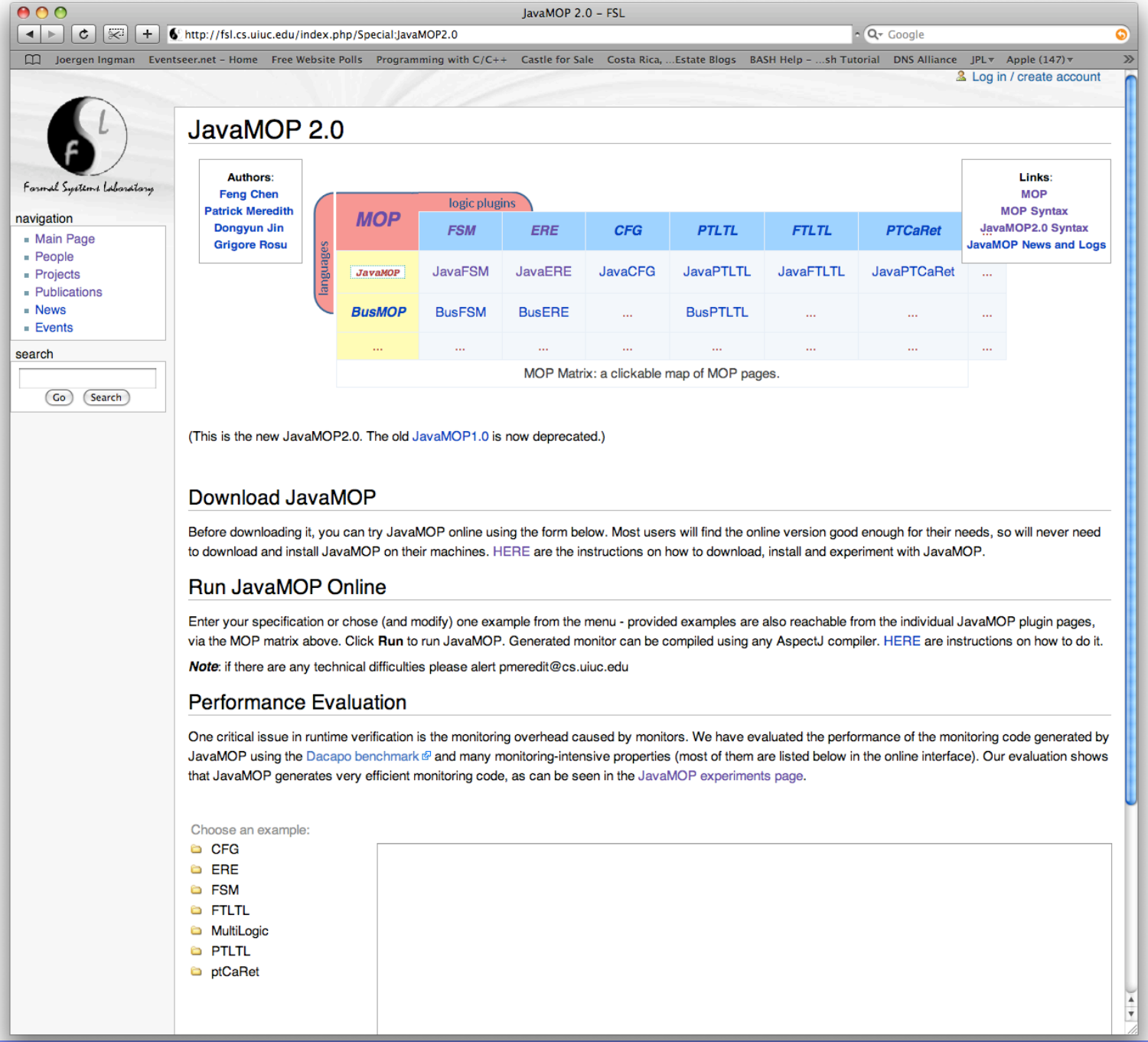
```
package paramere.hasnext;

import java.io.*;
import java.util.*;

suffix HasNext(Iterator i) {
    event hasnext before(Iterator i) : call(* Iterator.hasNext()) && target(i) {}
    event next before(Iterator i) : call(* Iterator.next()) && target(i) {}

    fsm:

    s0[
      next -> s1
    ]

    s1[
      next -> s2
    ]

    !s2[]

    @s2 {
        System.err.println("! hasNext not called before next");
        __RESET;
    }
}
```

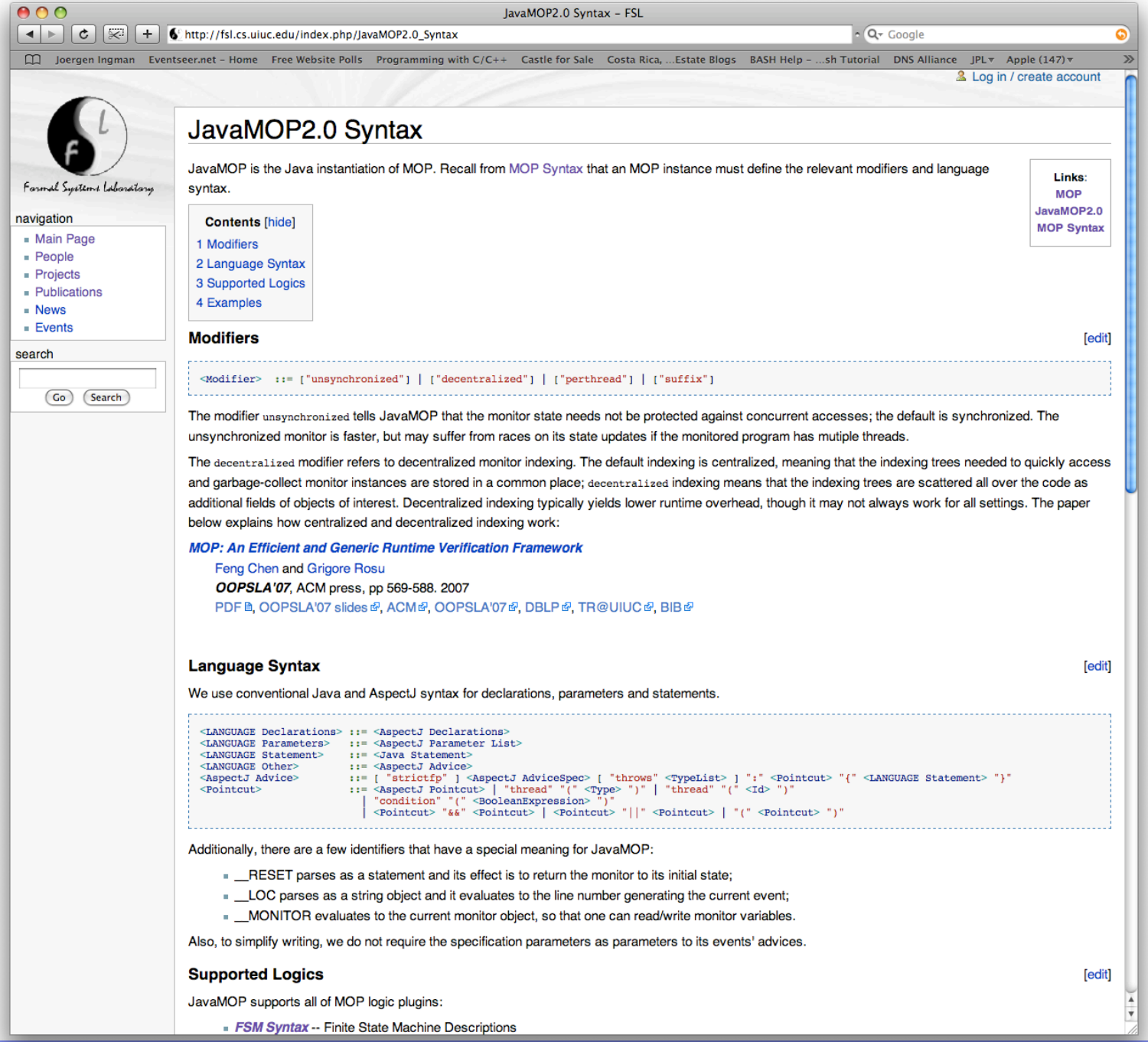
DEMO

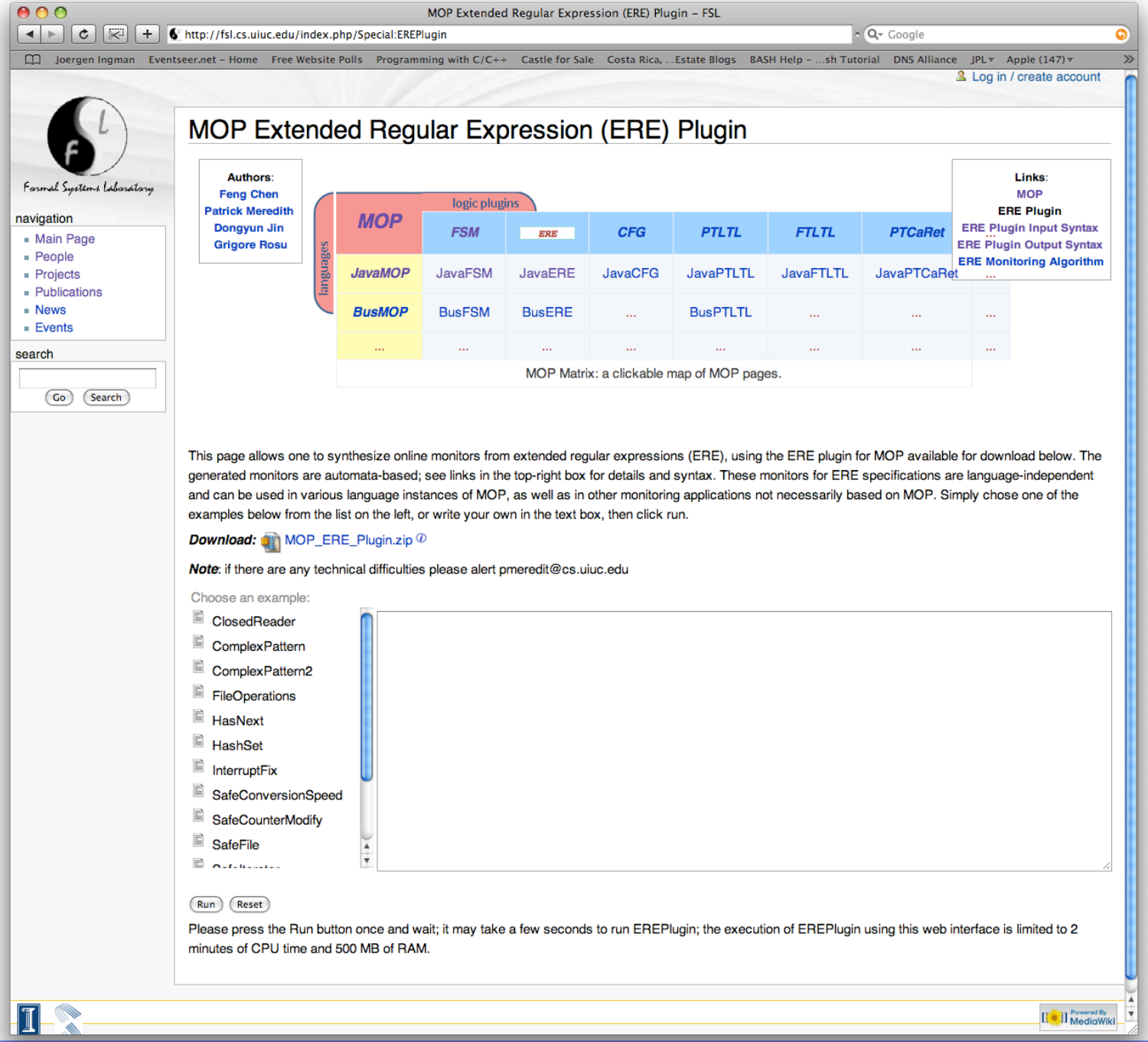

Java – cs119mon/src/paramere/hasnext/Test2.java – Eclipse SDK – /Users/khavelun/Desktop/development/workspace
Package Explorer
Hierarchy
Test2.java
1 package paramere.hasnext;
2
3 import java.util.*;
4
5 public class Test2 {
6 public static void main(String[] args) {
7 Vector<Integer> v1 = new Vector();
8 Vector<Integer> v2 = new Vector();
9 v1.add(1); v1.add(3); v2.add(5); v2.add(7);
10 Iterator it1 = v1.iterator();
11 Iterator it2 = v2.iterator();
12 int sumFirstTwo_v2 = 0;
13 if(it2.hasNext())
14 sumFirstTwo_v2 += (Integer)it2.next();
15 if(it1.hasNext())
16 sumFirstTwo_v2 += (Integer)it2.next();
17 System.out.println("sumFirstTwo_v2 = " + sumFirstTwo_v2);
18 }
19 }
20
ARTIST2
cs119-09
src
exercise1
Stack.java
StackContentsPolicy.aj
StackInterface.java
StackSizePolicy.aj
Test.java
experiments
JRE System Library [JVM 1.5.0 (MacO
AspectJ Runtime Library
cs119mon
src
(default package)
alexroper
aspectj1
aspectj1.pointcuts
aspectj2.any_hashset.weakmap
aspectj2.architecture
aspectj2.coll_hashset.naive
aspectj2.coll_hashset.weakmap
aspectj2.locking
aspectj2.safeenum
aspectj2.statemachine
aspectj2.temporal
aspectj2.timing
assignments.problem1
assignments.problem2
assignments.problem3
cfg.hasnext
cfg.locking
cfg.safeenum
commons
erecfg.locking
javaapi.stack
javaapi.synccollection
javaapi.urlconnection
ltl.hasnext
ltl.safeenum
paramere.closedreader
paramere.generaltest
paramere.hashset
paramere.hasnext
HasNextMonitorAspect.aj
Test0.java
Test1.java
Test2.java
paramere.resource
paramere.safeenum
project.hendrickson.problem1
project.hendrickson.problem2
project.hendrickson.problem3
project.matuschak.problem1
project.matuschak.problem2
project.matuschak.problem3
Problems
Javadoc
Declaration
Search
ANTLR Interpreter
Console
<terminated> Test2 (4) [Java Application] /System/Library/Frameworks/JavaVM.framework/Versions/1.5.0/Home/bin/java (May 17, 2009 4:32:43 PM)
sumFirstTwo_v2 = 12
Writable
Smart Insert
20 : 2
Building Workspace

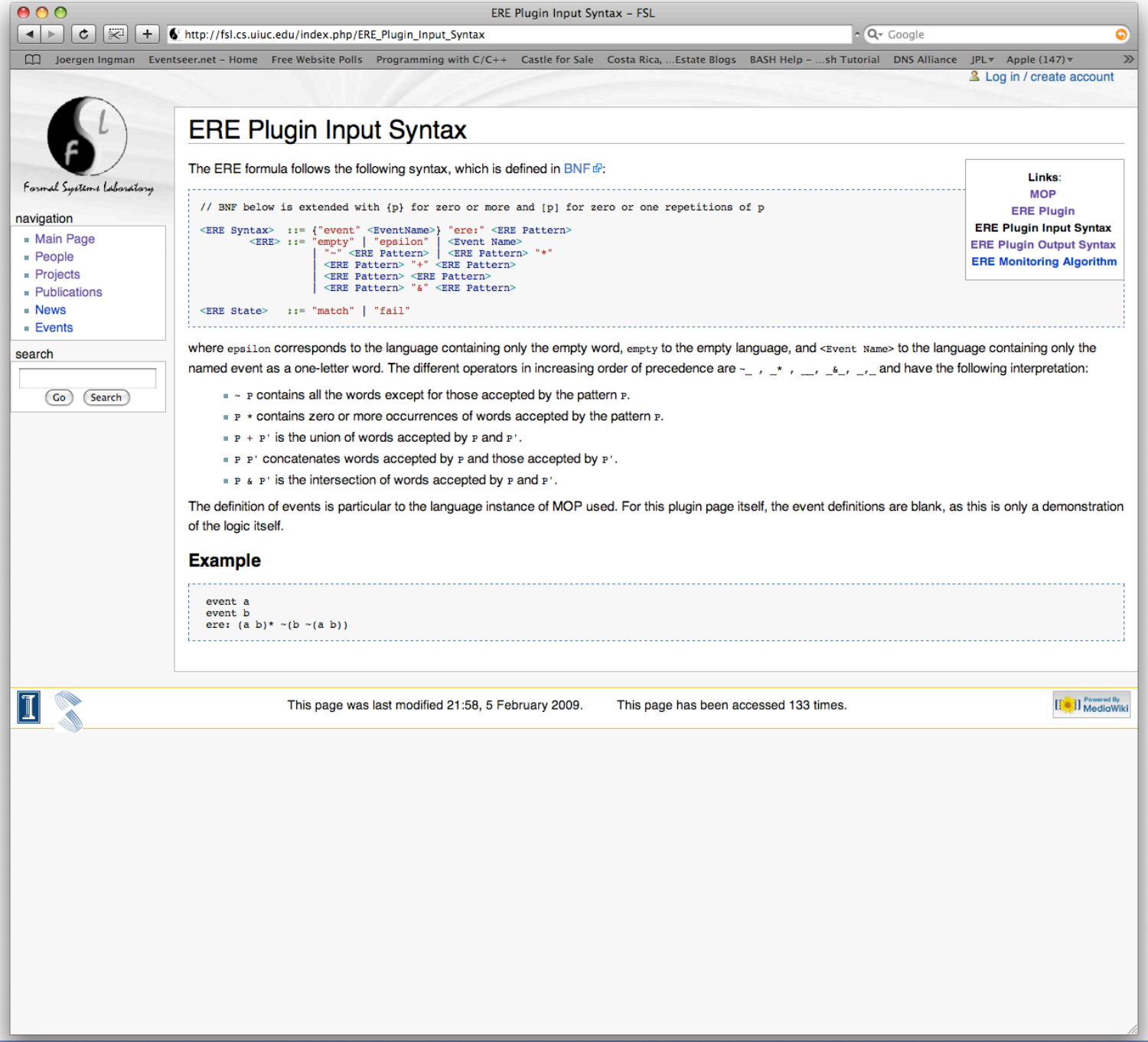

Monitoring-Oriented Programming – FSL
http://fsl.cs.uiuc.edu/index.php/Monitoring-Oriented_Programming
Google
Joergen Ingman
Eventseer.net – Home
Free Website Polls
Programming with C/C++
Castle for Sale
Costa Rica, ...Estate Blogs
BASH Help – ...sh Tutorial
DNS Alliance
JPL
Apple (147)
Log in / create account
Formal Systems Laboratory
navigation
Main Page
People
Projects
Publications
News
Events
search
Go
Search
Monitoring-Oriented Programming
Links:
MOP Syntax
Monitoring-Oriented Programming, abbreviated MOP, is a software development and analysis framework aiming at reducing the gap between formal specification and implementation by allowing them together to form a system. In MOP, runtime monitoring is supported and encouraged as a fundamental principle for building reliable software: monitors are automatically synthesized from specified properties and integrated into the original system to check its dynamic behaviors during execution. When a specification is violated or validated at runtime, user-defined actions will be triggered, which can be any code from information logging to runtime recovery. One can understand MOP from at least three perspectives: as a discipline allowing one to improve safety, reliability and dependability of a system by monitoring its requirements against its implementation at runtime; as an extension of programming languages with logics (one can add logical statements anywhere in the program, referring to past or future states); and as a lightweight formal method.
The MOP Matrix
[edit]
MOP is generic both in the underlying programming language and in the requirements specification formalism in which properties are expressed. The clickable table below shows all current instances of MOP and can be used to navigate MOP's webpages on this site. The rows show instances by the language and the columns instances by the logic. A cell "..." means that the corresponding combination is not yet available.
languages
logic plugins
MOP
FSM
ERE
CFG
PTLTL
FTLTL
PTCaRet
...
JavaMOP
JavaFSM
JavaERE
JavaCFG
JavaPTLTL
JavaFTLTL
JavaPTCaRet
...
BusMOP
BusFSM
BusERE
...
BusPTLTL
...
...
...
...
...
...
...
...
...
...
...
...
...
...
MOP Matrix: a clickable map of MOP pages.
MOP Language Instances
[edit]
JavaMOP is an MOP tool for Java;
BusMOP is an MOP tool for monitoring consumer off-the-shelf components over the PCI bus;
MOP Logic Plugins
[edit]
Logic plugins implement and encapsulate monitor synthesis algorithms for particular requirements specification formalisms. While they are a core feature of MOP, logic plugins can be and are also used in various other monitoring applications. Below is a list containing our current logic plugins. Each plugin can be experimented with online before download.
FSM -- Finite State Machines
ERE -- Extended Regular Expressions
CFG -- Context Free Grammars
PTLTL -- Past Time Linear Temporal Logic
FTLTL -- Future Time Linear Temporal Logic
PTCARET -- Past Time LTL with Calls and Returns
Send us your logic plugin!


MOP Syntax – FSL
http://fsl.cs.uiuc.edu/index.php/MOP_Syntax
Google
Joergen Ingman
Eventseer.net – Home
Free Website Polls
Programming with C/C++
Castle for Sale
Costa Rica, ...Estate Blogs
BASH Help – ...sh Tutorial
DNS Alliance
JPL
Apple (147)
Log in / create account
Formal Systems Laboratory
navigation
Main Page
People
Projects
Publications
News
Events
search
Go
Search
MOP Syntax
Links:
MOP
The syntax below uses BNF (extended with the common {p} for zero or more and [p] for zero or one repetitions of p, respectively). Since MOP is generic and can be instantiated, the MOP syntax is composed of two kinds of constructs, generic ones and language/logic specific ones. We define the generic ones on this page and also describe which constructs need to be instantiated for a specific MOP instance.
<Specification> ::= {<Modifier>} <Id> ["(" <LANGUAGE Parameters> ")"] "{"
<LANGUAGE Declarations>
{<Event>}
[<Property>
{"@" <LOGIC State> <LANGUAGE Statement> }
]
"}"
<Modifier> ::= ... <!-- each language instance may add specific modifiers here, possibly none -->
<Event> ::= "event" <Id> <LANGUAGE Other>
<Property> ::= <LOGIC Name> ":" <LOGIC Syntax>
The event declaration code (<Event>) allows for the definition of events which may then be referred to in the property. Event declarations can also have arbitrary code associated with them, which is run when the event is observed (<LANGUAGE Other>), e.g. code to modify the program or the monitor state. If the property is missing, then the MOP specification is called raw. Raw specifications are useful when no existing logic is powerful enough to specify the desired property; in that case, one embeds the custom monitoring code manually within the event generation code. The key statement "@" <LOGIC State> denotes what we refer to as handlers. Handlers contain arbitrary code from the source language, and are invoked when a certain logic state is reached (e.g., validation, violation, or a particular state in a finite state machine description), and at least one handler is required anytime there is a property (i.e., any time we are not using a raw monitor).
Language Syntax
[edit]
For each language instance of MOP, one need define the following syntactic categories used above.
<LANGUAGE Declarations> ::= ... <!-- syntax of declarations in the underlying language -->
<LANGUAGE Parameters> ::= ... <!-- syntax of parameter list in underlying language -->
<LANGUAGE Statement> ::= ... <!-- syntax of statements in the underlying language -->
<LANGUAGE Other> ::= ... <!-- other syntax in the underlying language, e.g., advice -->
The following links show how this generic syntax is instantiated for different MOP language instances:
JavaMOP Syntax -- MOP for Java Programs
BusMOP Syntax -- MOP for the PCI Bus, using Verilog and VHDL
Logic Syntax
[edit]
Each logic plugin should provide the following two syntactic categories:
<LOGIC Syntax> ::= ... <!-- the requirements syntax provided by the logic plugin above -->
<LOGIC State> ::= ... <!-- identifiers denoting states to which handlers can be associated, e.g., violation -->
The following links show how this generic syntax is instantiated for different MOP logic plugins:
FSM Syntax -- Finite State Machine Descriptions
ERE Syntax -- Extended Regular Expressions
CFG Syntax -- Context Free Grammars
PTLTL Syntax -- Past Time Linear Temporal Logic
FTLTL Syntax -- Future Time Linear Temporal Logic


JavaMOP 2.0 – FSL
http://fsl.cs.uiuc.edu/index.php/Special:JavaMOP2.0
Google
Joergen Ingman
Eventseer.net – Home
Free Website Polls
Programming with C/C++
Castle for Sale
Costa Rica, ...Estate Blogs
BASH Help – ...sh Tutorial
DNS Alliance
JPL
Apple (147)
Log in / create account
Formal Systems Laboratory
navigation
Main Page
People
Projects
Publications
News
Events
search
Go
Search
JavaMOP 2.0
Authors:
Feng Chen
Patrick Meredith
Dongyun Jin
Grigore Rosu
Links:
MOP
MOP Syntax
JavaMOP2.0 Syntax
JavaMOP News and Logs
languages
logic plugins
MOP
FSM
ERE
CFG
PTLTL
FTLTL
PTCaRet
JavaMOP
JavaFSM
JavaERE
JavaCFG
JavaPTLTL
JavaFTLTL
JavaPTCaRet
BusMOP
BusFSM
BusERE
BusPTLTL
MOP Matrix: a clickable map of MOP pages.
(This is the new JavaMOP2.0. The old JavaMOP1.0 is now deprecated.)
Download JavaMOP
Before downloading it, you can try JavaMOP online using the form below. Most users will find the online version good enough for their needs, so will never need to download and install JavaMOP on their machines. HERE are the instructions on how to download, install and experiment with JavaMOP.
Run JavaMOP Online
Enter your specification or chose (and modify) one example from the menu - provided examples are also reachable from the individual JavaMOP plugin pages, via the MOP matrix above. Click Run to run JavaMOP. Generated monitor can be compiled using any AspectJ compiler. HERE are instructions on how to do it.
Note: if there are any technical difficulties please alert pmeredit@cs.uiuc.edu
Performance Evaluation
One critical issue in runtime verification is the monitoring overhead caused by monitors. We have evaluated the performance of the monitoring code generated by JavaMOP using the Dacapo benchmark and many monitoring-intensive properties (most of them are listed below in the online interface). Our evaluation shows that JavaMOP generates very efficient monitoring code, as can be seen in the JavaMOP experiments page.
Choose an example:
CFG
ERE
FSM
FTLTL
MultiLogic
PTLTL
ptCaRet


JavaMOP2.0 Syntax - FSL
http://fsl.cs.uiuc.edu/index.php/JavaMOP2.0_Syntax
Google
Joergen Ingman
Eventseer.net - Home
Free Website Polls
Programming with C/C++
Castle for Sale
Costa Rica, ...Estate Blogs
BASH Help - ...sh Tutorial
DNS Alliance
JPL
Apple (147)
Log in / create account
Formal Systems Laboratory
navigation
Main Page
People
Projects
Publications
News
Events
search
Go
Search
JavaMOP2.0 Syntax
JavaMOP is the Java instantiation of MOP. Recall from MOP Syntax that an MOP instance must define the relevant modifiers and language syntax.
Links:
MOP
JavaMOP2.0
MOP Syntax
Contents [hide]
1 Modifiers
2 Language Syntax
3 Supported Logics
4 Examples
Modifiers
[edit]
<Modifier> ::= ["unsynchronized"] | ["decentralized"] | ["perthread"] | ["suffix"]
The modifier unsynchronized tells JavaMOP that the monitor state needs not be protected against concurrent accesses; the default is synchronized. The unsynchronized monitor is faster, but may suffer from races on its state updates if the monitored program has mutiple threads.
The decentralized modifier refers to decentralized monitor indexing. The default indexing is centralized, meaning that the indexing trees needed to quickly access and garbage-collect monitor instances are stored in a common place; decentralized indexing means that the indexing trees are scattered all over the code as additional fields of objects of interest. Decentralized indexing typically yields lower runtime overhead, though it may not always work for all settings. The paper below explains how centralized and decentralized indexing work:
MOP: An Efficient and Generic Runtime Verification Framework
Feng Chen and Grigore Rosu
OOPSLA'07, ACM press, pp 569-588. 2007
PDF, OOPSLA'07 slides, ACM, OOPSLA'07, DBLP, TR@UIUC, BIB
Language Syntax
[edit]
We use conventional Java and AspectJ syntax for declarations, parameters and statements.
<LANGUAGE Declarations> ::= <AspectJ Declarations>
<LANGUAGE Parameters> ::= <AspectJ Parameter List>
<LANGUAGE Statement> ::= <Java Statement>
<LANGUAGE Other> ::= <AspectJ Advice>
<AspectJ Advice> ::= ["strictfp"] <AspectJ AdviceSpec> ["throws" <TypeList>] ":" <Pointcut> "{" <LANGUAGE Statement> "}"
<Pointcut> ::= <AspectJ Pointcut> | "thread" "(" <Type> ")" | "thread" "(" <Id> ")"
| "condition" "(" <BooleanExpression> ")"
| <Pointcut> "&&" <Pointcut> | <Pointcut> "||" <Pointcut> | "(" <Pointcut> ")"
Additionally, there are a few identifiers that have a special meaning for JavaMOP:
__RESET parses as a statement and its effect is to return the monitor to its initial state;
__LOC parses as a string object and it evaluates to the line number generating the current event;
__MONITOR evaluates to the current monitor object, so that one can read/write monitor variables.
Also, to simplify writing, we do not require the specification parameters as parameters to its events' advices.
Supported Logics
[edit]
JavaMOP supports all of MOP logic plugins:
FSM Syntax -- Finite State Machine Descriptions


MOP Extended Regular Expression (ERE) Plugin – FSL
http://fsl.cs.uiuc.edu/index.php/Special:EREPlugin
Log in / create account
Formal Systems Laboratory
navigation
Main Page
People
Projects
Publications
News
Events
search
Go
Search
MOP Extended Regular Expression (ERE) Plugin
Authors:
Feng Chen
Patrick Meredith
Dongyun Jin
Grigore Rosu
Links:
MOP
ERE Plugin
ERE Plugin Input Syntax
ERE Plugin Output Syntax
ERE Monitoring Algorithm
languages
logic plugins
MOP
FSM
ERE
CFG
PTLTL
FTLTL
PTCaRet
JavaMOP
JavaFSM
JavaERE
JavaCFG
JavaPTLTL
JavaFTLTL
JavaPTCaRet
BusMOP
BusFSM
BusERE
BusPTLTL
MOP Matrix: a clickable map of MOP pages.
This page allows one to synthesize online monitors from extended regular expressions (ERE), using the ERE plugin for MOP available for download below. The generated monitors are automata-based; see links in the top-right box for details and syntax. These monitors for ERE specifications are language-independent and can be used in various language instances of MOP, as well as in other monitoring applications not necessarily based on MOP. Simply chose one of the examples below from the list on the left, or write your own in the text box, then click run.
Download: MOP_ERE_Plugin.zip
Note: if there are any technical difficulties please alert pmeredit@cs.uiuc.edu
Choose an example:
ClosedReader
ComplexPattern
ComplexPattern2
FileOperations
HasNext
HashSet
InterruptFix
SafeConversionSpeed
SafeCounterModify
SafeFile
Run
Reset
Please press the Run button once and wait; it may take a few seconds to run EREPlugin; the execution of EREPlugin using this web interface is limited to 2 minutes of CPU time and 500 MB of RAM.


ERE Plugin Input Syntax – FSL
http://fsl.cs.uiuc.edu/index.php/ERE_Plugin_Input_Syntax
Google
Joergen Ingman
Eventseer.net – Home
Free Website Polls
Programming with C/C++
Castle for Sale
Costa Rica, ...Estate Blogs
BASH Help – ...sh Tutorial
DNS Alliance
JPL
Apple (147)
Log in / create account
Formal Systems Laboratory
navigation
Main Page
People
Projects
Publications
News
Events
search
Go
Search
ERE Plugin Input Syntax
The ERE formula follows the following syntax, which is defined in BNF:
// BNF below is extended with {p} for zero or more and [p] for zero or one repetitions of p
<ERE Syntax> ::= {"event" <EventName>} "ere:" <ERE Pattern>
<ERE> ::= "empty" | "epsilon" | <Event Name>
| "~" <ERE Pattern> | <ERE Pattern> "*"
| <ERE Pattern> "+" <ERE Pattern>
| <ERE Pattern> <ERE Pattern>
| <ERE Pattern> "&" <ERE Pattern>
<ERE State> ::= "match" | "fail"
Links:
MOP
ERE Plugin
ERE Plugin Input Syntax
ERE Plugin Output Syntax
ERE Monitoring Algorithm
where epsilon corresponds to the language containing only the empty word, empty to the empty language, and <Event Name> to the language containing only the named event as a one-letter word. The different operators in increasing order of precedence are ~_ , _* , __, _&_, _,_ and have the following interpretation:
~ P contains all the words except for those accepted by the pattern P.
P * contains zero or more occurrences of words accepted by the pattern P.
P + P' is the union of words accepted by P and P'.
P P' concatenates words accepted by P and those accepted by P'.
P & P' is the intersection of words accepted by P and P'.
The definition of events is particular to the language instance of MOP used. For this plugin page itself, the event definitions are blank, as this is only a demonstration of the logic itself.
Example
event a
event b
ere: (a b)* ~(b ~(a b))
This page was last modified 21:58, 5 February 2009.
This page has been accessed 133 times.
Powered By MediaWiki

MOP Extended Regular Expression (ERE) Plugin - FSL

http://fsl.cs.uiuc.edu/index.php/Special:EREPlugin

Joergen Ingman | Eventseer.net - Home | Free Website Polls | Programming with C/C++ | Castle for Sale | Costa Rica, ...Estate Blogs | BASH Help - ...sh Tutorial | DNS Alliance | JPL | Apple (147)

Note: if there are any technical difficulties please alert pmeredit@cs.uiuc.edu

Choose an example:

- ClosedReader
- ComplexPattern
- ComplexPattern2
- FileOperations
- HasNext
- HashSet
- InterruptFix
- SafeConversionSpeed
- SafeCounterModify
- SafeFile

```
event hasNext
event next

ere : (hasNext hasNext* next)*
```

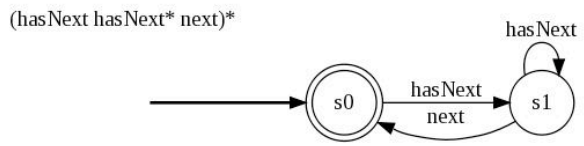
Run | Reset

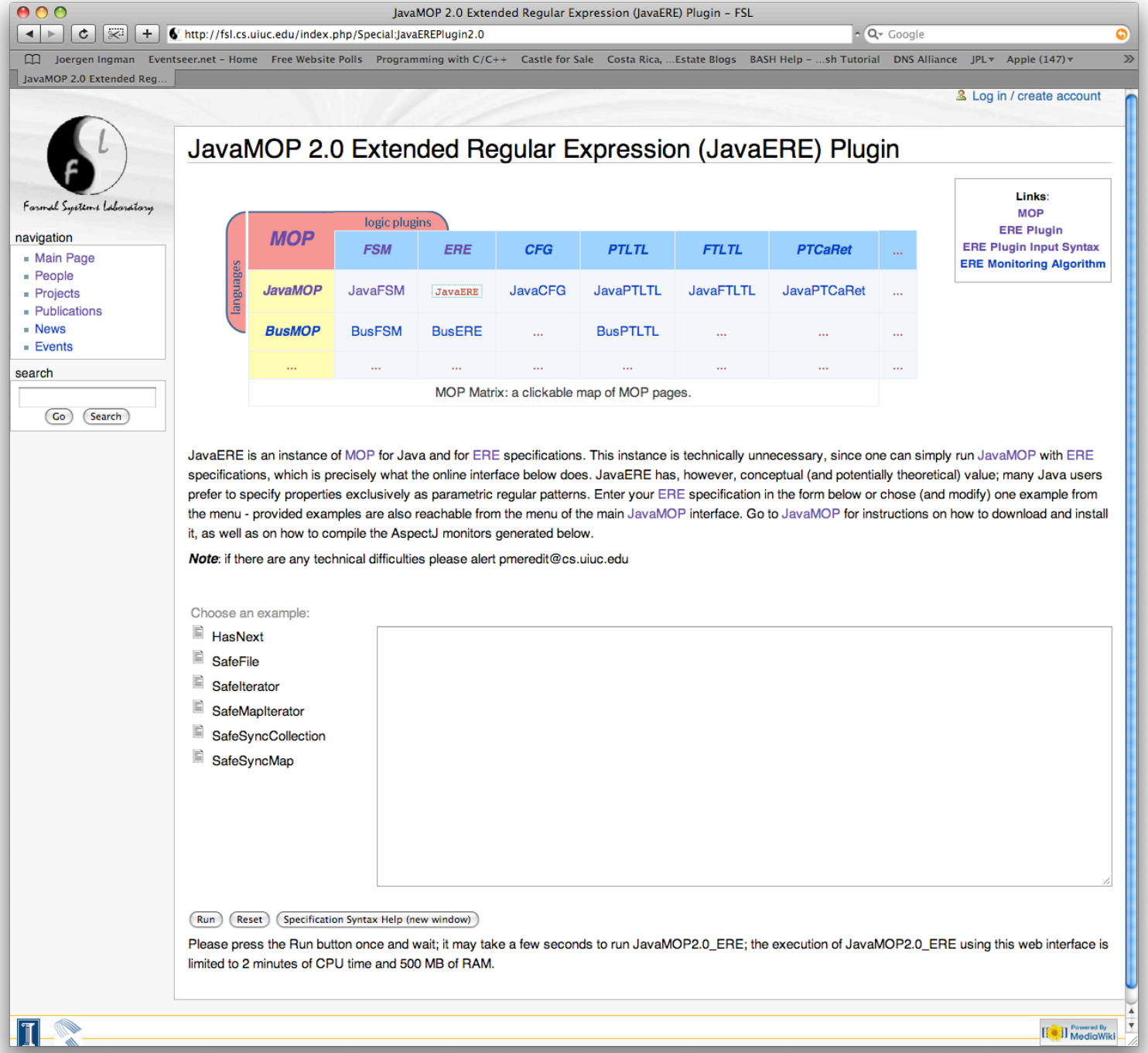
Number of generated monitors (since 14 December 2008): 945 (MOP), 218 (ERE), 83 (Language-neutral ERE)

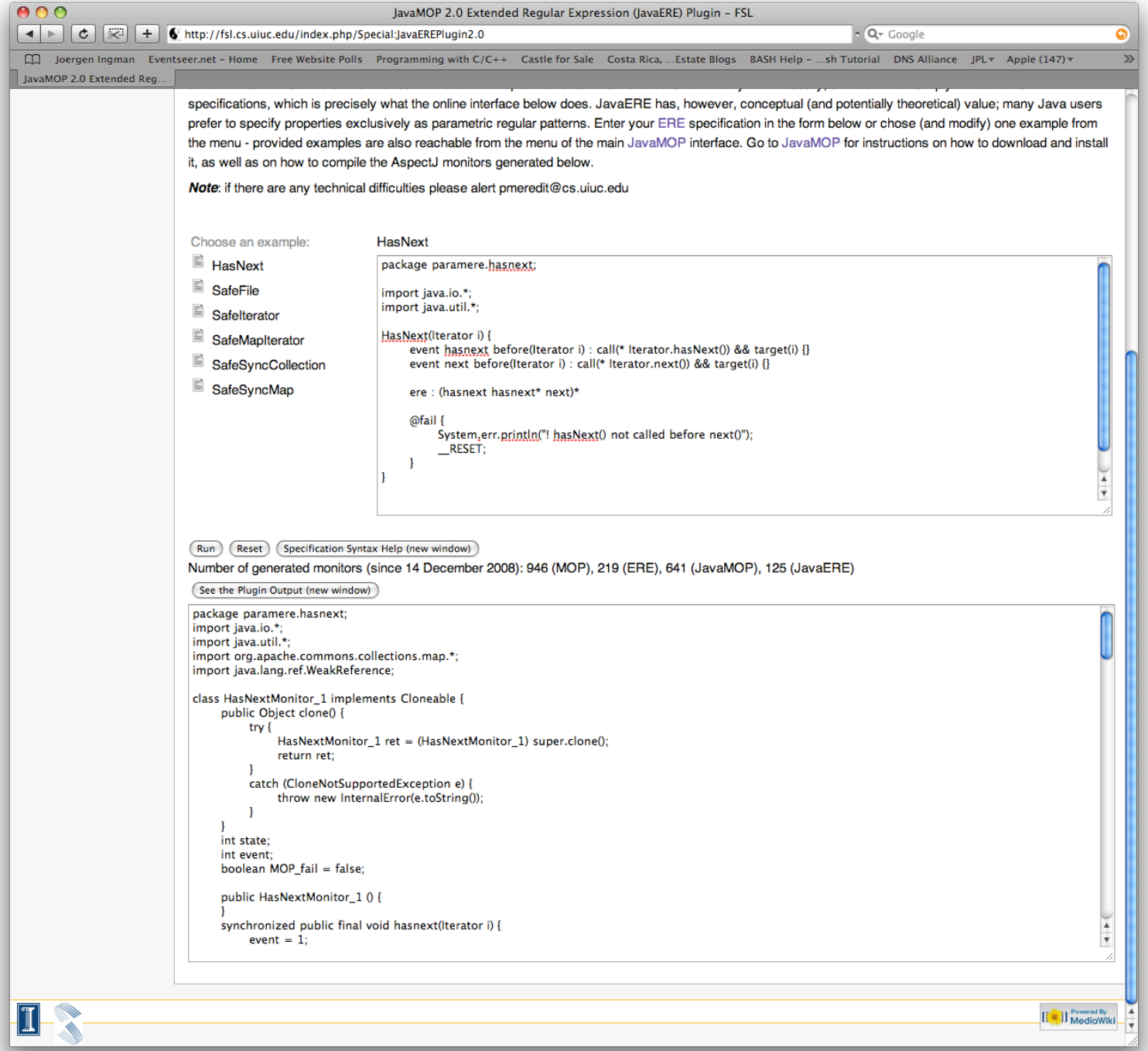
The textbox below shows the generated monitor; see ERE Plugin Output Syntax (opens new window) for its precise syntax and meaning.
Below the textbox is a more easily read graph representation of it.

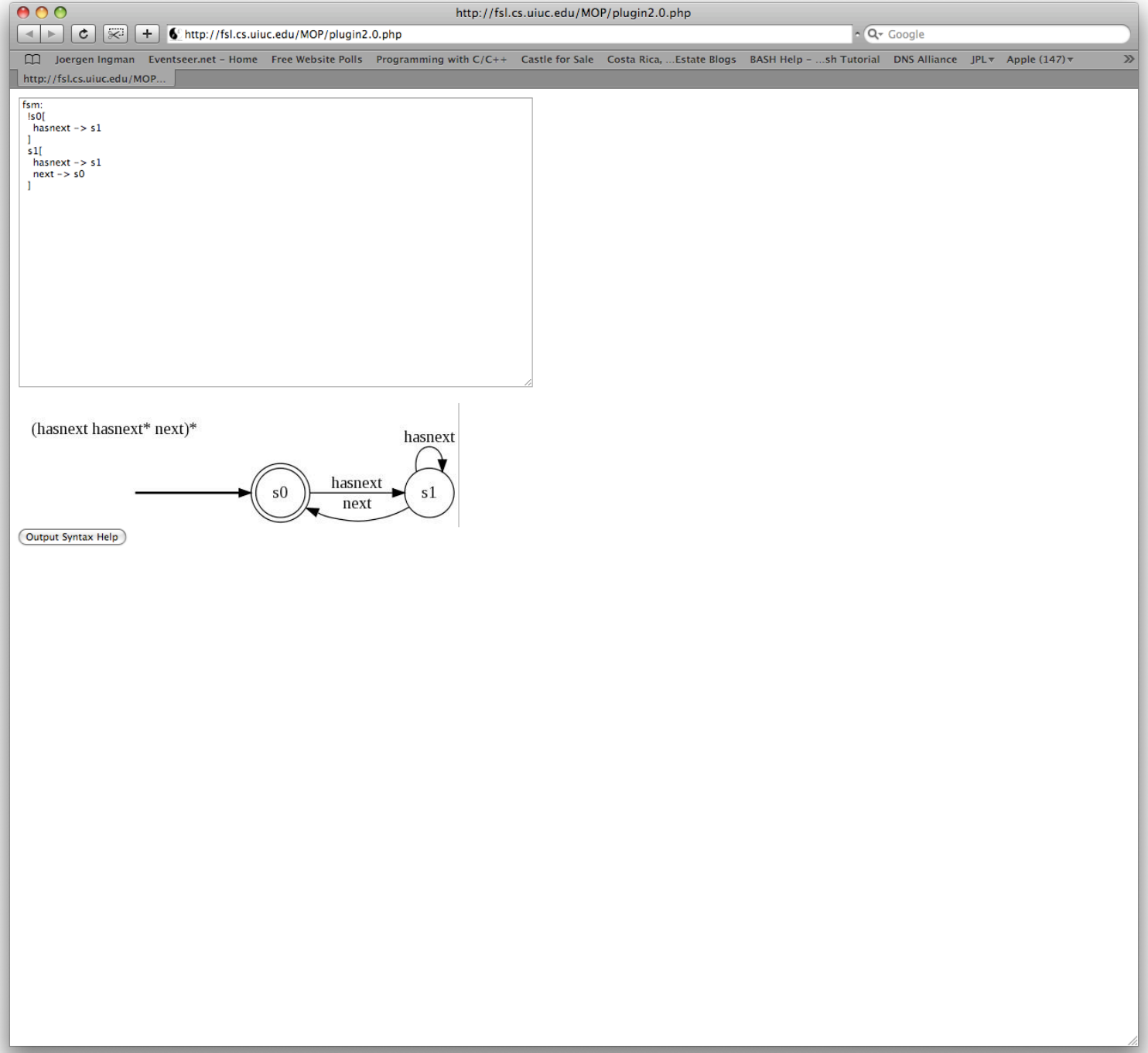
```
fsm:
 !s0[
  hasNext -> s1
 ]
 s1[
  hasNext -> s1
  next -> s0
 ]
```

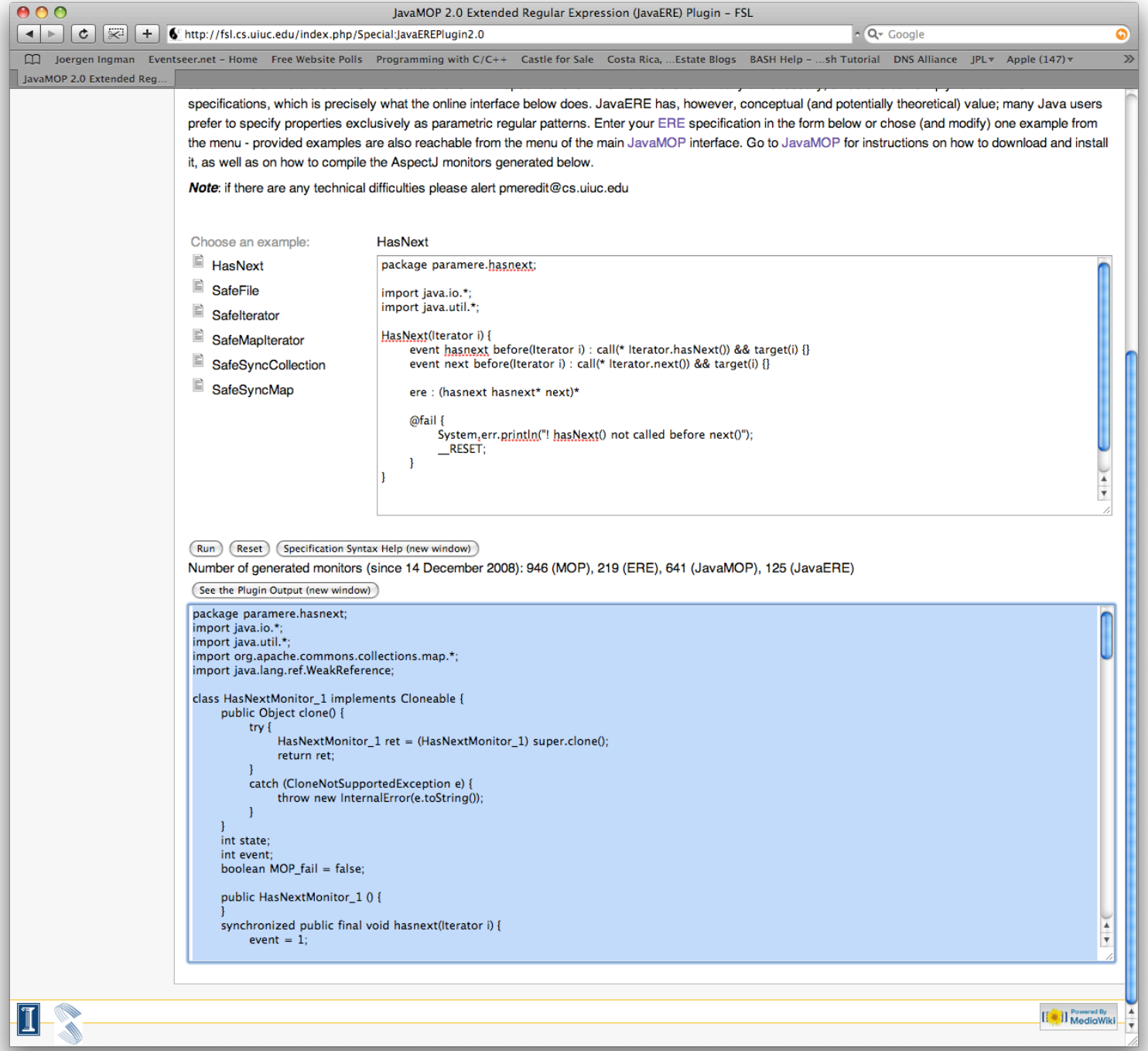
Below is an image representation of the monitor:

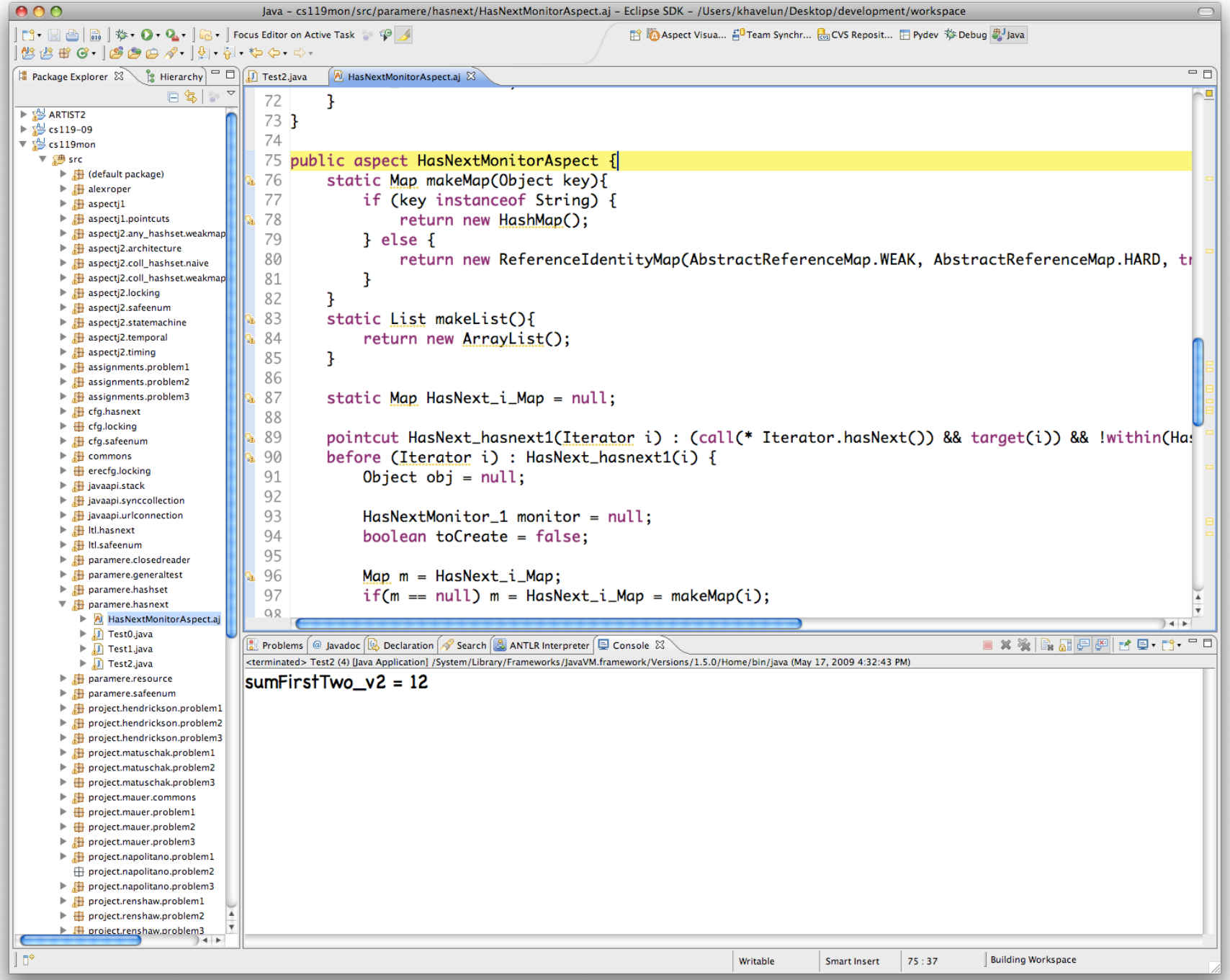


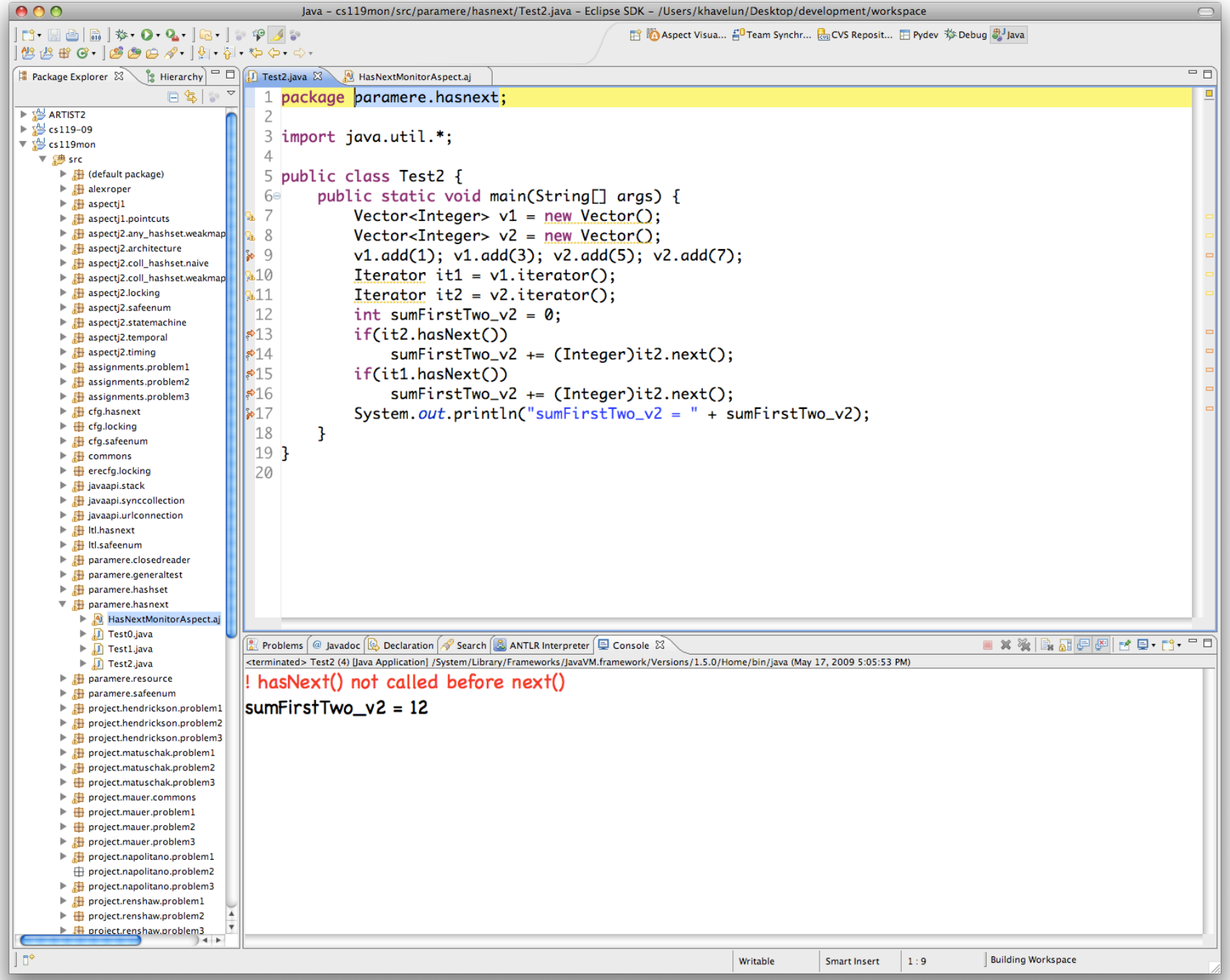

JavaMOP 2.0 Extended Regular Expression (JavaERE) Plugin - FSL
http://fsl.cs.uiuc.edu/index.php/Special:JavaEREPlugin2.0
Google
Joergen Ingman
Eventseer.net - Home
Free Website Polls
Programming with C/C++
Castle for Sale
Costa Rica, ...Estate Blogs
BASH Help - ...sh Tutorial
DNS Alliance
JPL
Apple (147)
JavaMOP 2.0 Extended Reg...
Log in / create account
Formal Systems Laboratory
navigation
Main Page
People
Projects
Publications
News
Events
search
Go
Search
JavaMOP 2.0 Extended Regular Expression (JavaERE) Plugin
Links:
MOP
ERE Plugin
ERE Plugin Input Syntax
ERE Monitoring Algorithm
logic plugins
languages
MOP
FSM
ERE
CFG
PTLTL
FTLTL
PTCaRet
...
JavaMOP
JavaFSM
JavaERE
JavaCFG
JavaPTLTL
JavaFTLTL
JavaPTCaRet
BusMOP
BusFSM
BusERE
BusPTLTL
MOP Matrix: a clickable map of MOP pages.
JavaERE is an instance of MOP for Java and for ERE specifications. This instance is technically unnecessary, since one can simply run JavaMOP with ERE specifications, which is precisely what the online interface below does. JavaERE has, however, conceptual (and potentially theoretical) value; many Java users prefer to specify properties exclusively as parametric regular patterns. Enter your ERE specification in the form below or chose (and modify) one example from the menu - provided examples are also reachable from the menu of the main JavaMOP interface. Go to JavaMOP for instructions on how to download and install it, as well as on how to compile the AspectJ monitors generated below.
Note: if there are any technical difficulties please alert pmeredit@cs.uiuc.edu
Choose an example:
HasNext
SafeFile
SafeIterator
SafeMapIterator
SafeSyncCollection
SafeSyncMap
Run
Reset
Specification Syntax Help (new window)
Please press the Run button once and wait; it may take a few seconds to run JavaMOP2.0_ERE; the execution of JavaMOP2.0_ERE using this web interface is limited to 2 minutes of CPU time and 500 MB of RAM.
Powered By MediaWiki


JavaMOP 2.0 Extended Regular Expression (JavaERE) Plugin - FSL
http://fsl.cs.uiuc.edu/index.php/Special:JavaEREPlugin2.0
Google
Joergen Ingman
Eventseer.net - Home
Free Website Polls
Programming with C/C++
Castle for Sale
Costa Rica, ...Estate Blogs
BASH Help - ...sh Tutorial
DNS Alliance
JPL
Apple (147)
JavaMOP 2.0 Extended Reg...
specifications, which is precisely what the online interface below does. JavaERE has, however, conceptual (and potentially theoretical) value; many Java users prefer to specify properties exclusively as parametric regular patterns. Enter your ERE specification in the form below or chose (and modify) one example from the menu - provided examples are also reachable from the menu of the main JavaMOP interface. Go to JavaMOP for instructions on how to download and install it, as well as on how to compile the AspectJ monitors generated below.
Note: if there are any technical difficulties please alert pmeredit@cs.uiuc.edu
Choose an example:
HasNext
SafeFile
SafeIterator
SafeMapIterator
SafeSyncCollection
SafeSyncMap
HasNext
package paramere.hasnext;
import java.io.*;
import java.util.*;
HasNext(Iterator i) {
event hasnext before(Iterator i) : call(* Iterator.hasNext()) && target(i) {}
event next before(Iterator i) : call(* Iterator.next()) && target(i) {}
ere : (hasnext hasnext* next)*
@fail {
System.err.println("! hasNext() not called before next()");
__RESET;
}
}
Run
Reset
Specification Syntax Help (new window)
Number of generated monitors (since 14 December 2008): 946 (MOP), 219 (ERE), 641 (JavaMOP), 125 (JavaERE)
See the Plugin Output (new window)
package paramere.hasnext;
import java.io.*;
import java.util.*;
import org.apache.commons.collections.map.*;
import java.lang.ref.WeakReference;
class HasNextMonitor_1 implements Cloneable {
public Object clone() {
try {
HasNextMonitor_1 ret = (HasNextMonitor_1) super.clone();
return ret;
}
catch (CloneNotSupportedException e) {
throw new InternalError(e.toString());
}
}
int state;
int event;
boolean MOP_fail = false;
public HasNextMonitor_1 () {
}
synchronized public final void hasnext(Iterator i) {
event = 1;
Powered By MediaWiki


http://fsl.cs.uiuc.edu/MOP/plugin2.0.php
http://fsl.cs.uiuc.edu/MOP/plugin2.0.php
Google
Joergen Ingman
Eventseer.net - Home
Free Website Polls
Programming with C/C++
Castle for Sale
Costa Rica, ...Estate Blogs
BASH Help - ...sh Tutorial
DNS Alliance
JPL
Apple (147)
http://fsl.cs.uiuc.edu/MOP...
fsm:
!s0[
hasnext -> s1
]
s1[
hasnext -> s1
next -> s0
]
(hasnext hasnext* next)*
hasnext
s0
hasnext
next
s1
Output Syntax Help


JavaMOP 2.0 Extended Regular Expression (JavaERE) Plugin - FSL
http://fsl.cs.uiuc.edu/index.php/Special:JavaEREPlugin2.0
specifications, which is precisely what the online interface below does. JavaERE has, however, conceptual (and potentially theoretical) value; many Java users prefer to specify properties exclusively as parametric regular patterns. Enter your ERE specification in the form below or chose (and modify) one example from the menu - provided examples are also reachable from the menu of the main JavaMOP interface. Go to JavaMOP for instructions on how to download and install it, as well as on how to compile the AspectJ monitors generated below.
Note: if there are any technical difficulties please alert pmeredit@cs.uiuc.edu
Choose an example:
HasNext
SafeFile
SafeIterator
SafeMapIterator
SafeSyncCollection
SafeSyncMap
HasNext
package paramere.hasnext;
import java.io.*;
import java.util.*;
HasNext(Iterator i) {
event hasnext before(Iterator i) : call(* Iterator.hasNext()) && target(i) {}
event next before(Iterator i) : call(* Iterator.next()) && target(i) {}
ere : (hasnext hasnext* next)*
@fail {
System.err.println("! hasNext() not called before next()");
__RESET;
}
}
Run
Reset
Specification Syntax Help (new window)
Number of generated monitors (since 14 December 2008): 946 (MOP), 219 (ERE), 641 (JavaMOP), 125 (JavaERE)
See the Plugin Output (new window)
package paramere.hasnext;
import java.io.*;
import java.util.*;
import org.apache.commons.collections.map.*;
import java.lang.ref.WeakReference;
class HasNextMonitor_1 implements Cloneable {
public Object clone() {
try {
HasNextMonitor_1 ret = (HasNextMonitor_1) super.clone();
return ret;
}
catch (CloneNotSupportedException e) {
throw new InternalError(e.toString());
}
}
int state;
int event;
boolean MOP_fail = false;
public HasNextMonitor_1 () {
}
synchronized public final void hasnext(Iterator i) {
event = 1;


Java – cs119mon/src/paramere/hasnext/HasNextMonitorAspect.aj – Eclipse SDK – /Users/khavelun/Desktop/development/workspace
Package Explorer
Hierarchy
Test2.java
HasNextMonitorAspect.aj
ARTIST2
cs119-09
cs119mon
src
(default package)
alexroper
aspectj1
aspectj1.pointcuts
aspectj2.architecture
aspectj2.locking
aspectj2.safeenum
aspectj2.statemachine
aspectj2.temporal
aspectj2.timing
assignments.problem1
assignments.problem2
assignments.problem3
cfg.hasnext
cfg.locking
cfg.safeenum
commons
erecfg.locking
javaapi.stack
javaapi.synccollection
javaapi.urlconnection
ltl.hasnext
ltl.safeenum
paramere.closedreader
paramere.generaltest
paramere.hashset
paramere.hasnext
HasNextMonitorAspect.aj
Test0.java
Test1.java
Test2.java
paramere.resource
paramere.safeenum
project.hendrickson.problem1
project.hendrickson.problem2
project.hendrickson.problem3
project.matuschak.problem1
project.matuschak.problem2
project.matuschak.problem3
project.mauer.commons
project.mauer.problem1
project.mauer.problem2
project.mauer.problem3
project.napolitano.problem1
project.napolitano.problem2
project.napolitano.problem3
project.renshaw.problem1
project.renshaw.problem2
project.renshaw.problem3
72 }
73 }
74
75 public aspect HasNextMonitorAspect {
76 static Map makeMap(Object key){
77 if (key instanceof String) {
78 return new HashMap();
79 } else {
80 return new ReferenceIdentityMap(AbstractReferenceMap.WEAK, AbstractReferenceMap.HARD,
81 }
82 }
83 static List makeList(){
84 return new ArrayList();
85 }
86
87 static Map HasNext_i_Map = null;
88
89 pointcut HasNext_hasnext1(Iterator i) : (call(* Iterator.hasNext()) && target(i)) && !within(Has
90 before (Iterator i) : HasNext_hasnext1(i) {
91 Object obj = null;
92
93 HasNextMonitor_1 monitor = null;
94 boolean toCreate = false;
95
96 Map m = HasNext_i_Map;
97 if(m == null) m = HasNext_i_Map = makeMap(i);
Problems
Javadoc
Declaration
Search
ANTLR Interpreter
Console
<terminated> Test2 (4) [Java Application] /System/Library/Frameworks/JavaVM.framework/Versions/1.5.0/Home/bin/java (May 17, 2009 4:32:43 PM)
sumFirstTwo_v2 = 12
Writable
Smart Insert
75 : 37
Building Workspace


Java – cs119mon/src/paramere/hasnext/Test2.java – Eclipse SDK – /Users/khavelun/Desktop/development/workspace
Package Explorer
Hierarchy
Test2.java
HasNextMonitorAspect.aj
1 package paramere.hasnext;
2
3 import java.util.*;
4
5 public class Test2 {
6 public static void main(String[] args) {
7 Vector<Integer> v1 = new Vector();
8 Vector<Integer> v2 = new Vector();
9 v1.add(1); v1.add(3); v2.add(5); v2.add(7);
10 Iterator it1 = v1.iterator();
11 Iterator it2 = v2.iterator();
12 int sumFirstTwo_v2 = 0;
13 if(it2.hasNext())
14 sumFirstTwo_v2 += (Integer)it2.next();
15 if(it1.hasNext())
16 sumFirstTwo_v2 += (Integer)it2.next();
17 System.out.println("sumFirstTwo_v2 = " + sumFirstTwo_v2);
18 }
19 }
20
Problems
Javadoc
Declaration
Search
ANTLR Interpreter
Console
<terminated> Test2 (4) [Java Application] /System/Library/Frameworks/JavaVM.framework/Versions/1.5.0/Home/bin/java (May 17, 2009 5:05:53 PM)
! hasNext() not called before next()
sumFirstTwo_v2 = 12
Writable
Smart Insert
1 : 9
Building Workspace

END OF DEMO

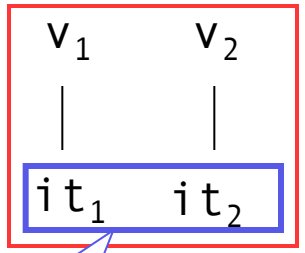
data values

- handling of data values efficiently in monitoring frameworks is one of the current research challenges.
- different frameworks approach it differently, achieving different expressiveness and efficiency.

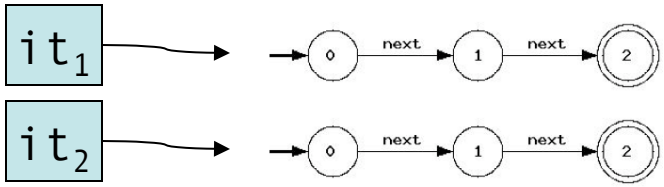
in this simple case: store monitor in iterator

our program again

```
public class Test2 {
    public static void main(String[] args) {
        Vector<Integer> v1 = new Vector();
        Vector<Integer> v2 = new Vector();
        v1.add(1); v1.add(3); v2.add(5); v2.add(7);
        Iterator it1 = v1.iterator();
        Iterator it2 = v2.iterator();
        int sumFirstTwo_v2 = 0;
        if(it2.hasNext())
            sumFirstTwo_v2 += (Integer)it2.next();
        if(it1.hasNext())
            sumFirstTwo_v2 += (Integer)it2.next();
        System.out.println("sumFirstTwo_v2 = " + sumFirstTwo_v2);
    }
}
```



only objects of interest

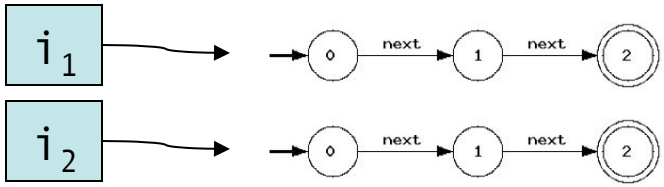


associate a monitor with each object of interest

Piggy-back (transportation), something that is riding on the back of something else

algorithm

- assume a property only mentions one kind of object (an iterator in this case).
- note, there could be many objects monitored (it_1,it_2).
- one can insert a monitor on each object that is monitored.
- alternatively: keep a map from objects to their monitors.
- when an operation of interest occurs (hasNext() or next()) that refers to an object, the monitor of that object is “executed”. If null, a monitor is first created.



associate a monitor with each object of interest

algorithm pseudo-code

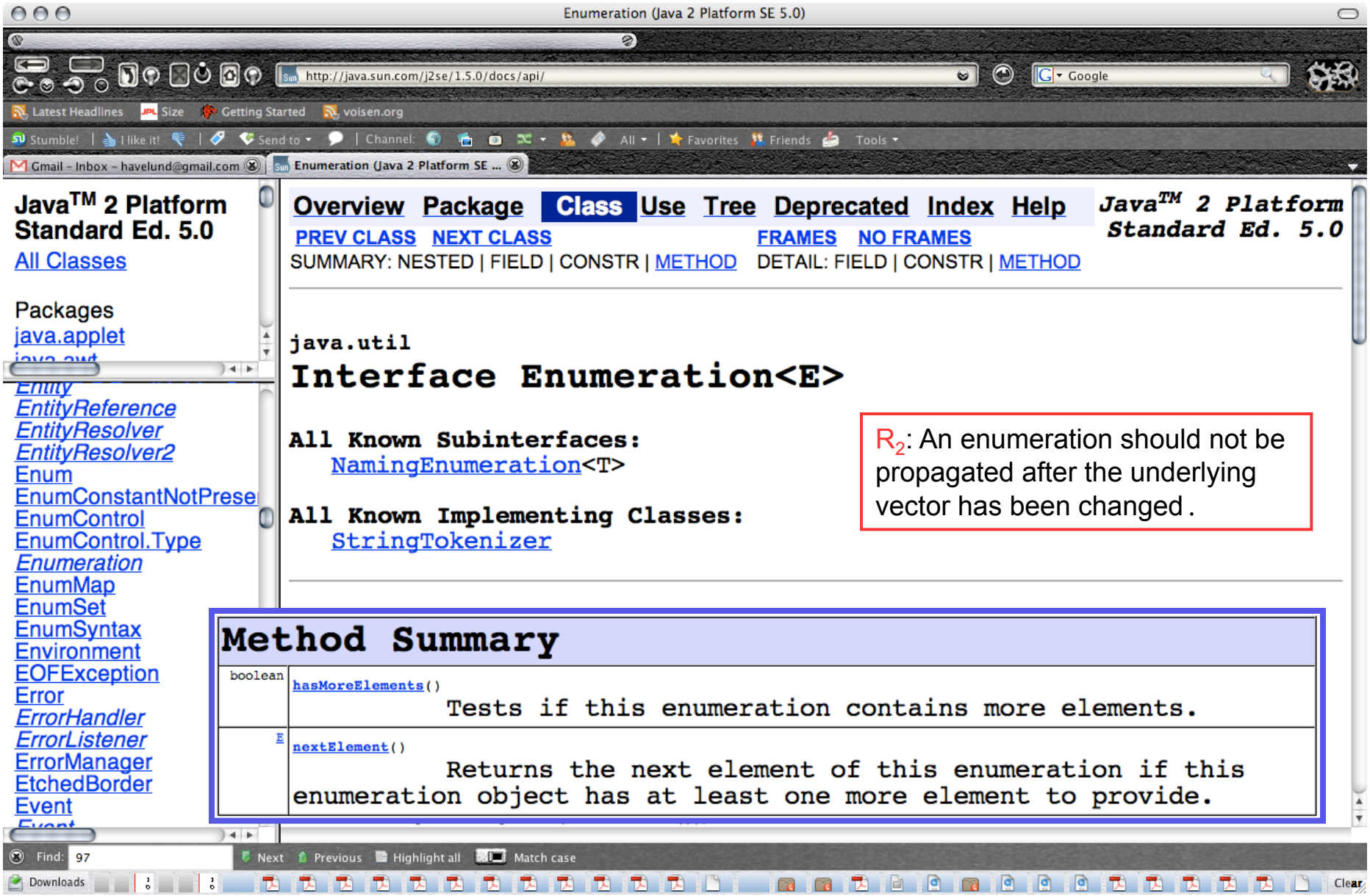
Consider a call of it.next() for example. This could lead to the following monitoring code:

```
if (it.monitor == null)
  it.monitor = new Monitor();
it.monitor.next();
```

however, this does not work when property refers to more than 1 object, where to store the monitor?

not to mention that the java library cannot be instrumented easily.

properties of Java library APIs



R_2: An enumeration should not be propagated after the underlying vector has been changed .

java.util

Interface Enumeration<E>

All Known Subinterfaces:
NamingEnumeration<T>

All Known Implementing Classes:
StringTokenizer

R_2: An enumeration should not be propagated after the underlying vector has been changed .

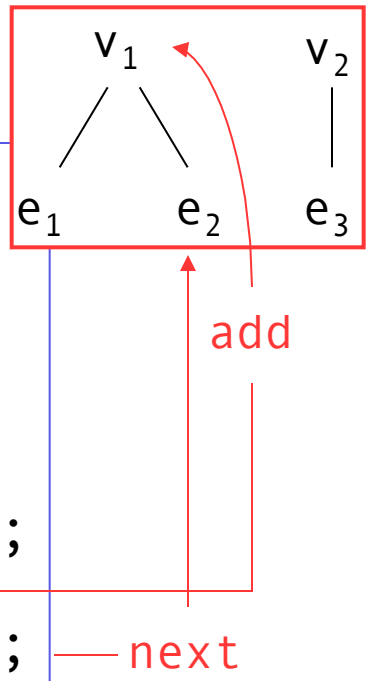
create next* updatesource updatesource* next

now the property must refer to more than one object: a vector and an enumerator

One-monitor-per-object no longer works. There is not a 1-1 correspondence between objects and monitors

many vectors and enumerators

```
Vector v1 = new Vector();
Vector v2 = new Vector();
v1.add(1); v1.add(2); v2.add(4); v2.add(5);
Enumeration e1 = v1.elements();
Enumeration e2 = v1.elements();
Enumeration e3 = v2.elements();
while(e1.hasMoreElements())print(e1.nextElement());
v1.add(99);
while(e2.hasMoreElements())print(e2.nextElement());
while(e3.hasMoreElements())print(e3.nextElement());
```



events:

```
create(v,e)
next(e)
updatesource(v)
```

$\forall$ (v,e) $\in$ { (v_1,e_1),(v_1,e_2),(v_2,e_3) }
create(v,e)
next(e)*
updatesource(v)+
next(e)

catch this pattern

```
package paramere.safeenum;

import java.io.*;
import java.util.*;

suffix SafeEnumeration(Vector v, Enumeration e) {
  event create after(Vector v) returning(Enumeration e) :
    call(Enumeration Vector.elements()) && target(v) {}

  event updatesource before(Vector v) :
    (call(* Vector.remove*(..)) || call(* Vector.add*(..)) ) && target(v) {}

  event next before(Enumeration e) :
    call(* Enumeration.nextElement()) && target(e) {}

  ere : create next* updatesource updatesource* next

  @match {
     System.out.println("improper enumerator usage");
  }
}
```

MOP's distinguished feature: separate handling of values

- *The problem*: how to monitor a universally quantified specification *efficiently*!

```
create<v,e>
udatesource<v>
next<e>
```

$(\forall v,e)\ \varphi$

```
create next* updatesource+ next
```

two approaches to data

- Ruler, PQL, PTQL, Tracematches, … :
  - choose a quantified logic
  - device monitor synthesizer for it:

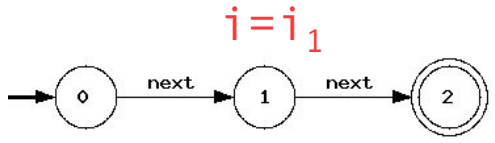
$$(\forall p)\ \varphi \quad \Longrightarrow \quad \mathcal{M}on_{(\forall p)\ \varphi}$$

- MOP:
  - only knows how to generate monitors for φ
  - The $(\forall p)$ is dealt with separately, and once-and-for-all, for all logics

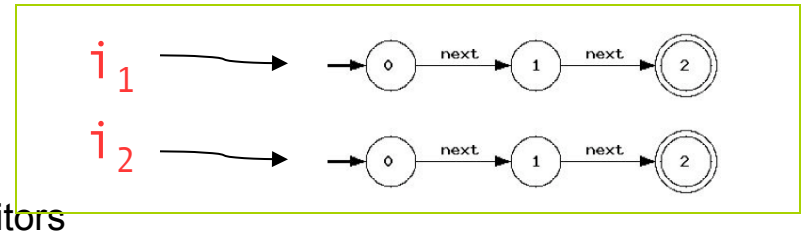
$$(\forall p)\ \varphi \quad \Longrightarrow \quad p \rightarrow \mathcal{M}on_{\varphi}$$

handling of data values

- value embedding scheme : the values become part of the monitors (state machines). This means modifying the state machine concept.

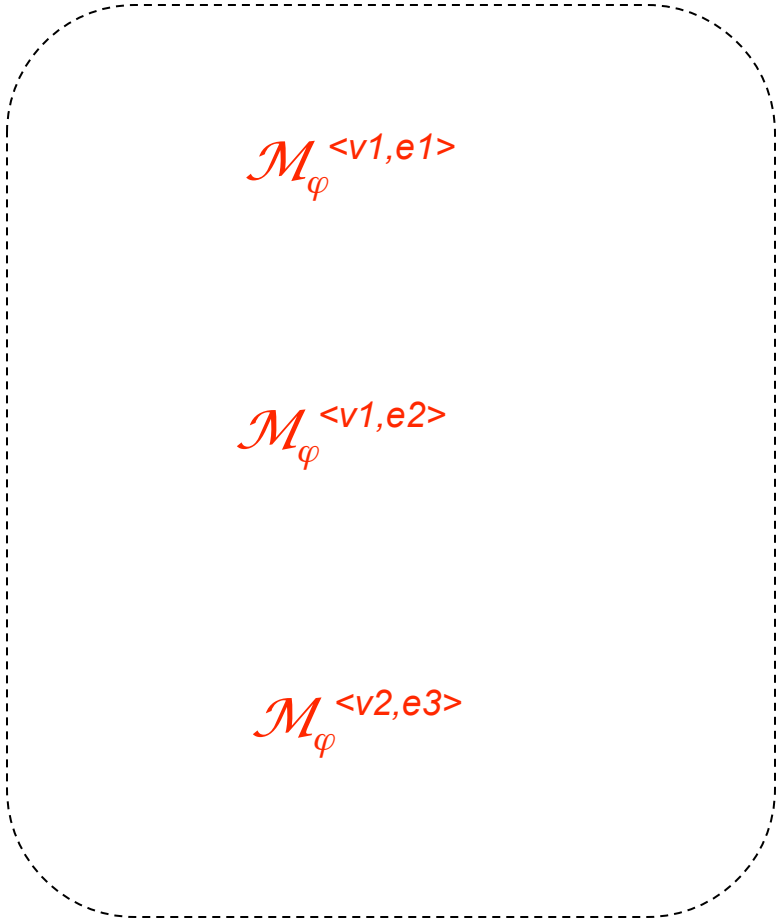


- value indexing scheme : keep automata atomic and index them with data
  - if one object per property max:
    - map objects to monitors
    - or piggyback monitor on object
  - if more than one object per property:
    - centralized index table from data to monitors
    - perhaps combination of piggybacking and centralized index table



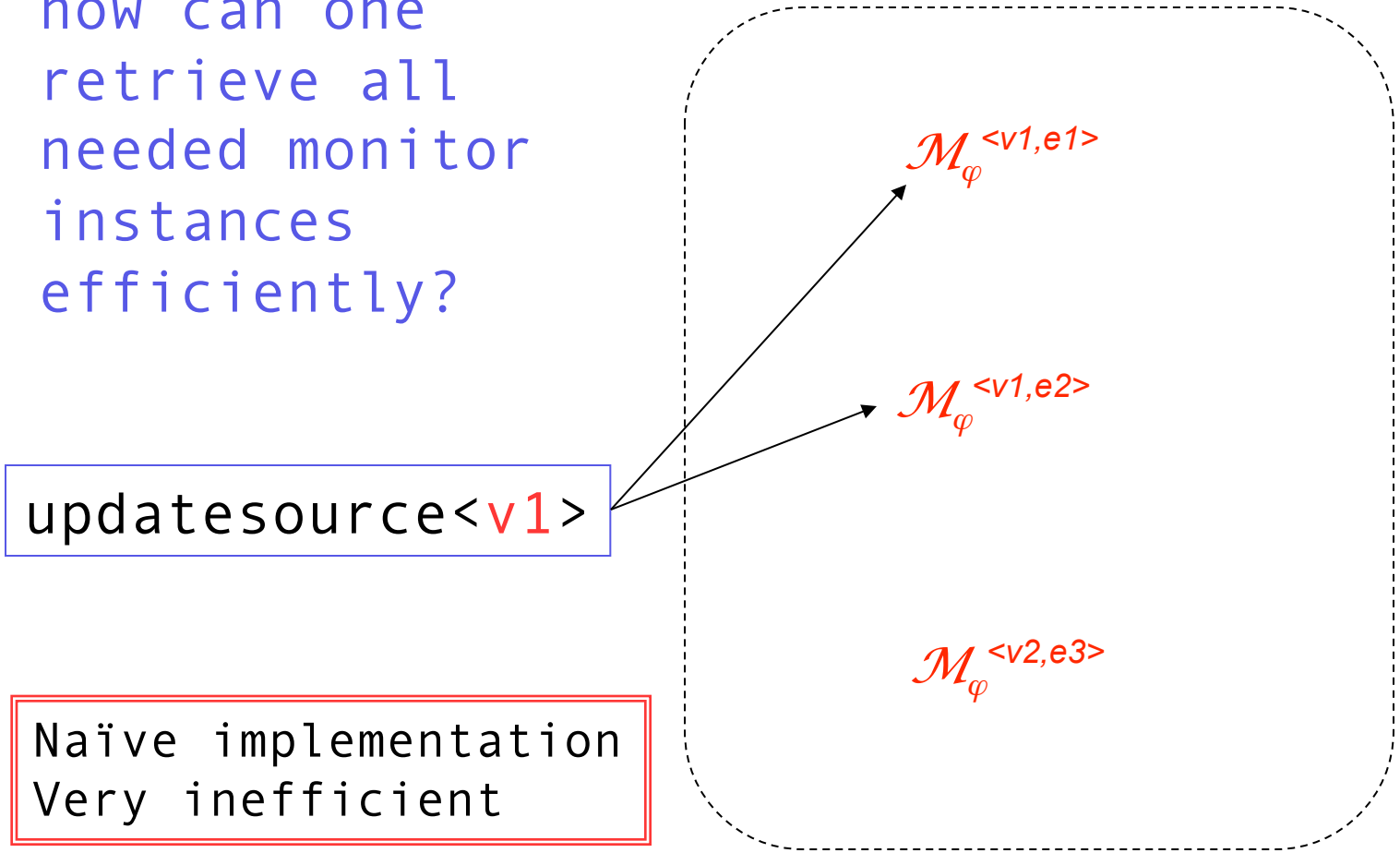
MOP's propositional monitors

one propositional monitor instance per parameter instance



MOP's propositional monitors

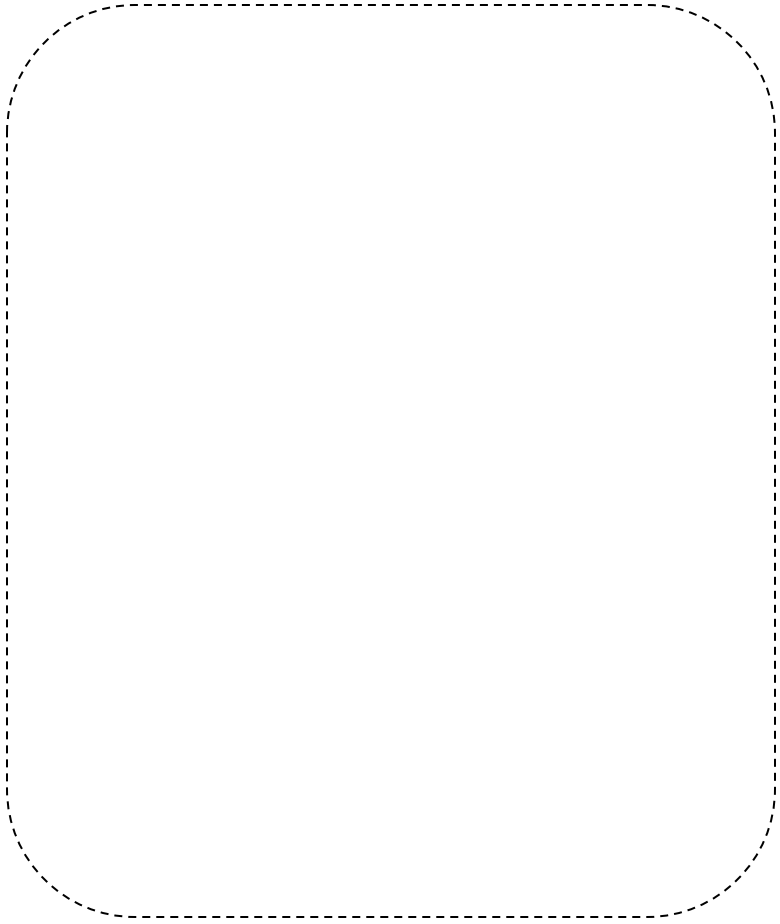
The problem:
how can one retrieve all needed monitor instances efficiently?



Naïve implementation
Very inefficient

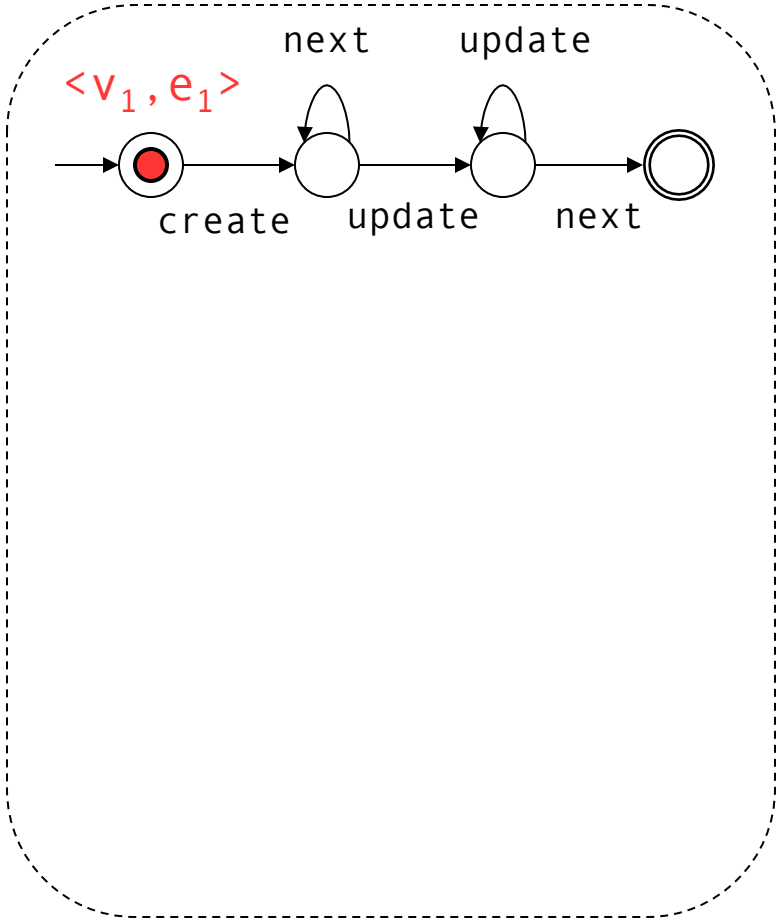
Naïve implementation

```
Vector v1 = new Vector();
Vector v2 = new Vector();
v1.add(1); v1.add(2); v2.add(4); v2.add(5);
Enumeration e1 = v1.elements();
Enumeration e2 = v1.elements();
Enumeration e3 = v2.elements();
while(e1.hasMoreElements())print(e1.nextElement());
v1.add(99);
while(e2.hasMoreElements())print(e2.nextElement());
while(e3.hasMoreElements())print(e3.nextElement());
```



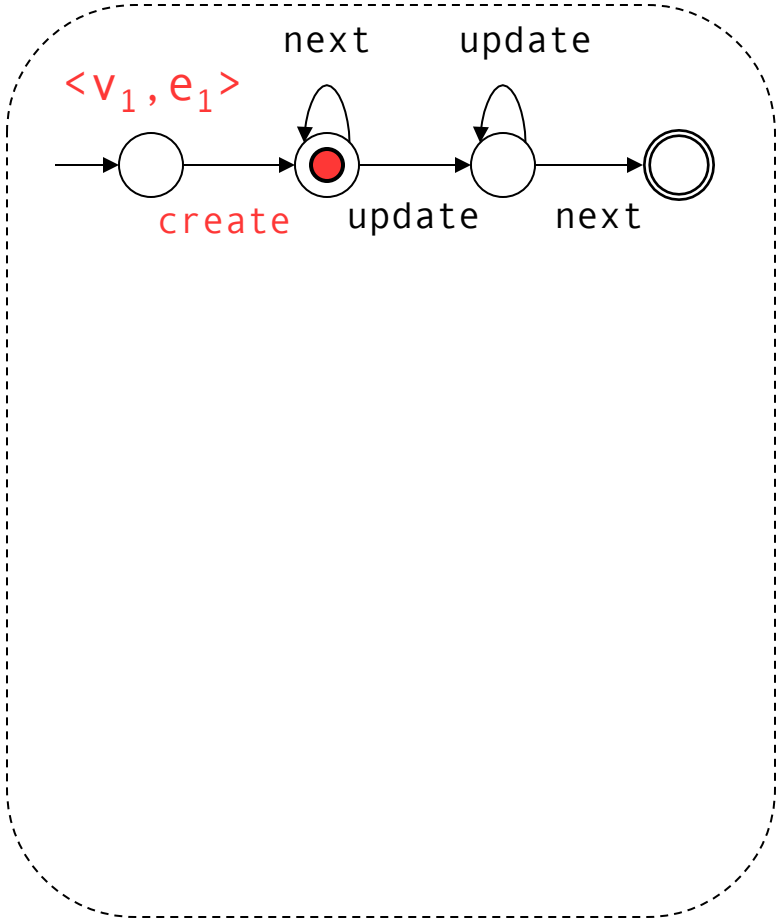
Naïve implementation

```
Vector v1 = new Vector();
Vector v2 = new Vector();
v1.add(1); v1.add(2); v2.add(4); v2.add(5);
Enumeration e1 = v1.elements();
Enumeration e2 = v1.elements();
Enumeration e3 = v2.elements();
while(e1.hasMoreElements())print(e1.nextElement());
v1.add(99);
while(e2.hasMoreElements())print(e2.nextElement());
while(e3.hasMoreElements())print(e3.nextElement());
```



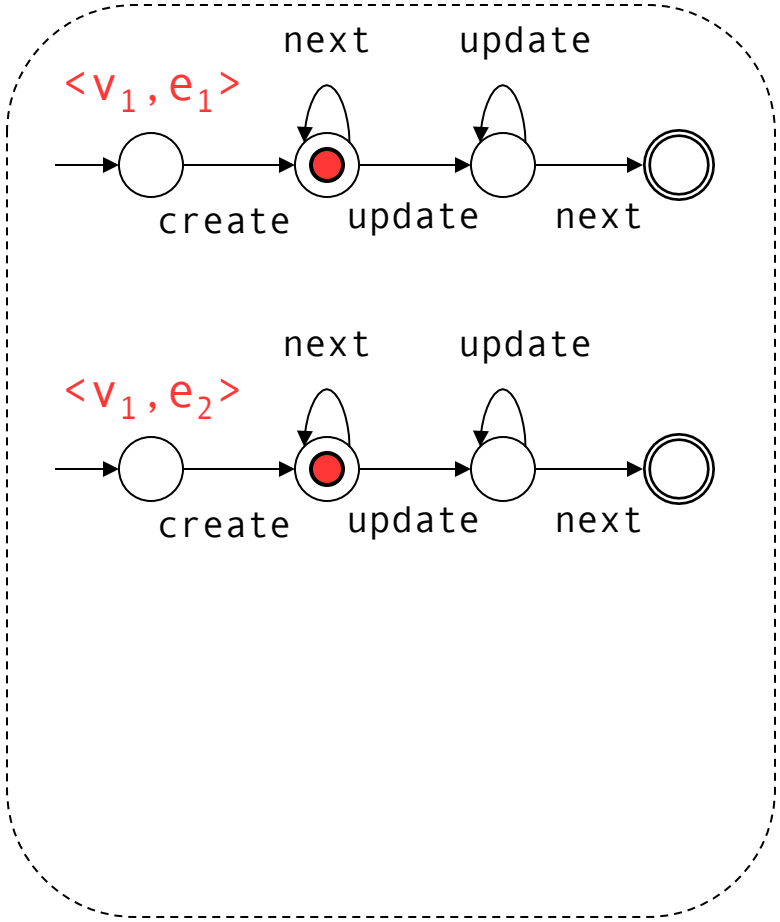
Naïve implementation

```
Vector v1 = new Vector();
Vector v2 = new Vector();
v1.add(1); v1.add(2); v2.add(4); v2.add(5);
Enumeration e1 = v1.elements();
Enumeration e2 = v1.elements();
Enumeration e3 = v2.elements();
while(e1.hasMoreElements())print(e1.nextElement());
v1.add(99);
while(e2.hasMoreElements())print(e2.nextElement());
while(e3.hasMoreElements())print(e3.nextElement());
```



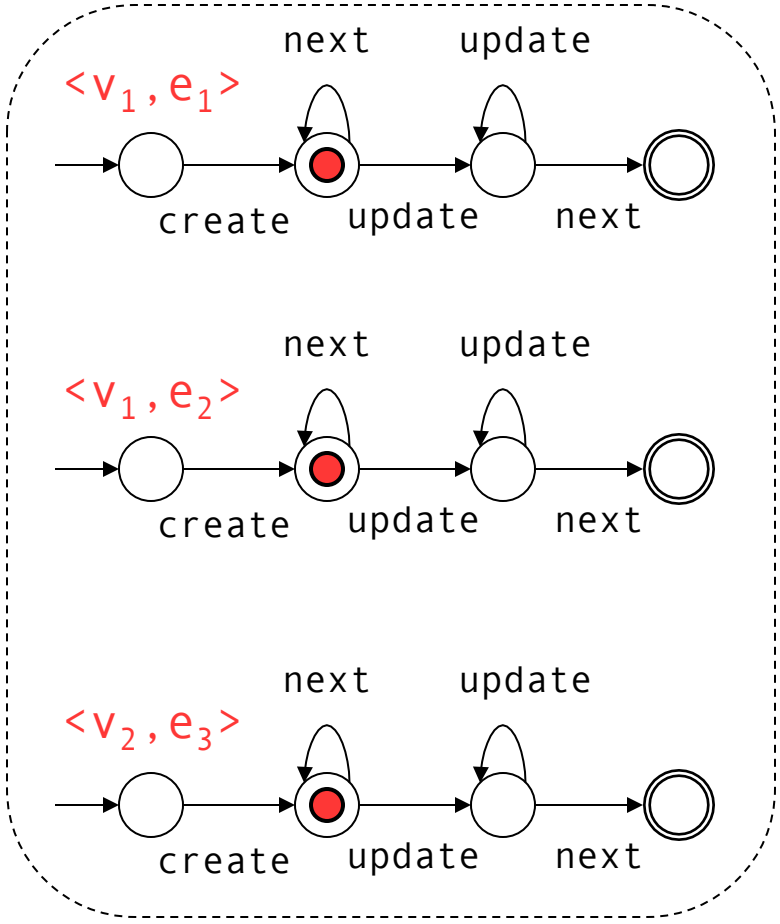
Naïve implementation

```
Vector v1 = new Vector();
Vector v2 = new Vector();
v1.add(1); v1.add(2); v2.add(4); v2.add(5);
Enumeration e1 = v1.elements();
Enumeration e2 = v1.elements();
Enumeration e3 = v2.elements();
while(e1.hasMoreElements())print(e1.nextElement());
v1.add(99);
while(e2.hasMoreElements())print(e2.nextElement());
while(e3.hasMoreElements())print(e3.nextElement());
```



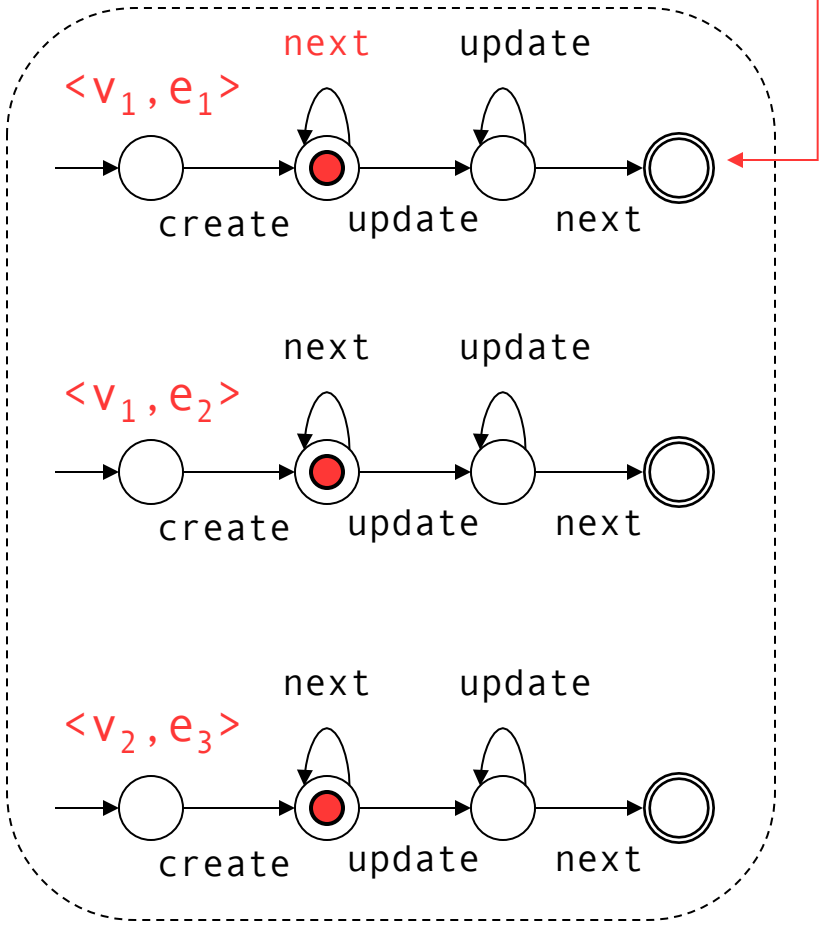
Naïve implementation

```
Vector v1 = new Vector();
Vector v2 = new Vector();
v1.add(1); v1.add(2); v2.add(4); v2.add(5);
Enumeration e1 = v1.elements();
Enumeration e2 = v1.elements();
Enumeration e3 = v2.elements();
while(e1.hasMoreElements())print(e1.nextElement());
v1.add(99);
while(e2.hasMoreElements())print(e2.nextElement());
while(e3.hasMoreElements())print(e3.nextElement());
```



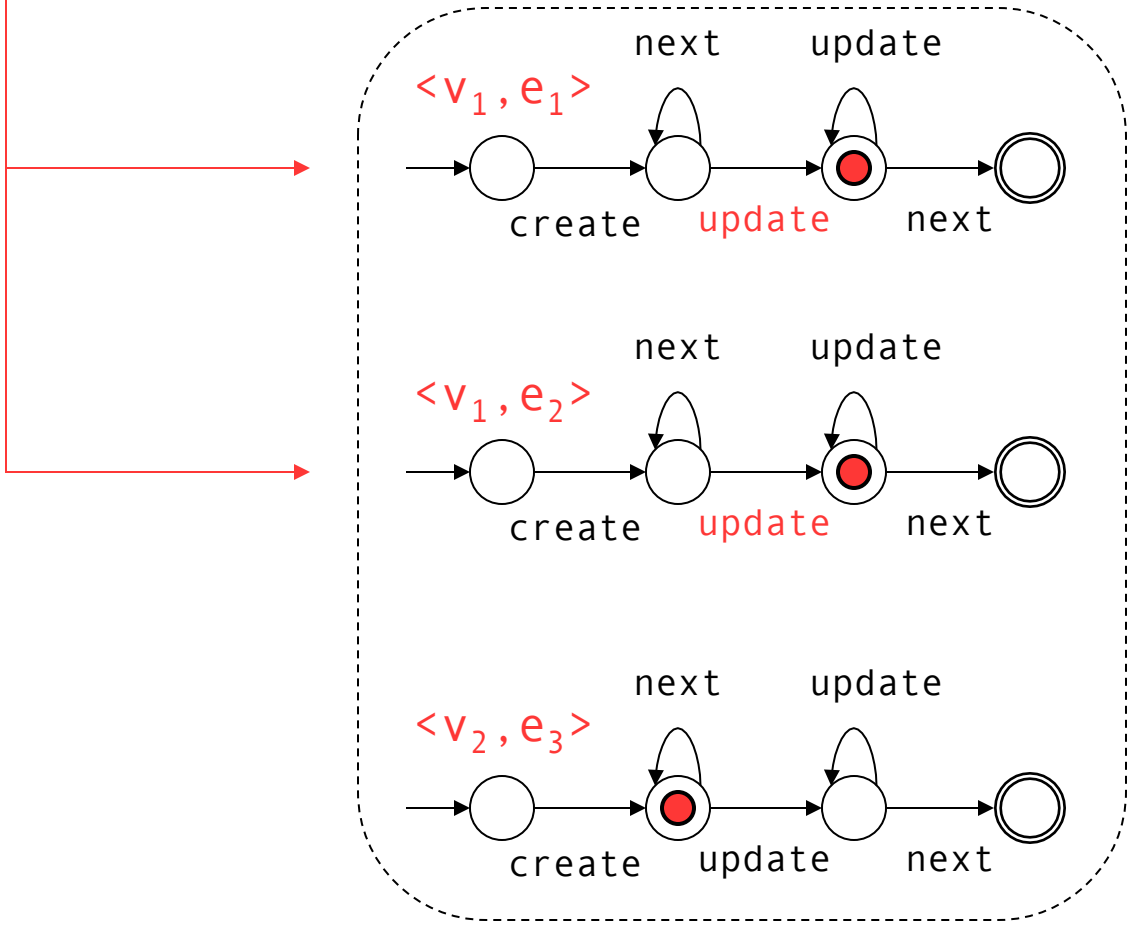
Naïve implementation

```
Vector v1 = new Vector();
Vector v2 = new Vector();
v1.add(1); v1.add(2); v2.add(4); v2.add(5);
Enumeration e1 = v1.elements();
Enumeration e2 = v1.elements();
Enumeration e3 = v2.elements();
while(e1.hasMoreElements())print(e1.nextElement());
v1.add(99);
while(e2.hasMoreElements())print(e2.nextElement());
while(e3.hasMoreElements())print(e3.nextElement());
```



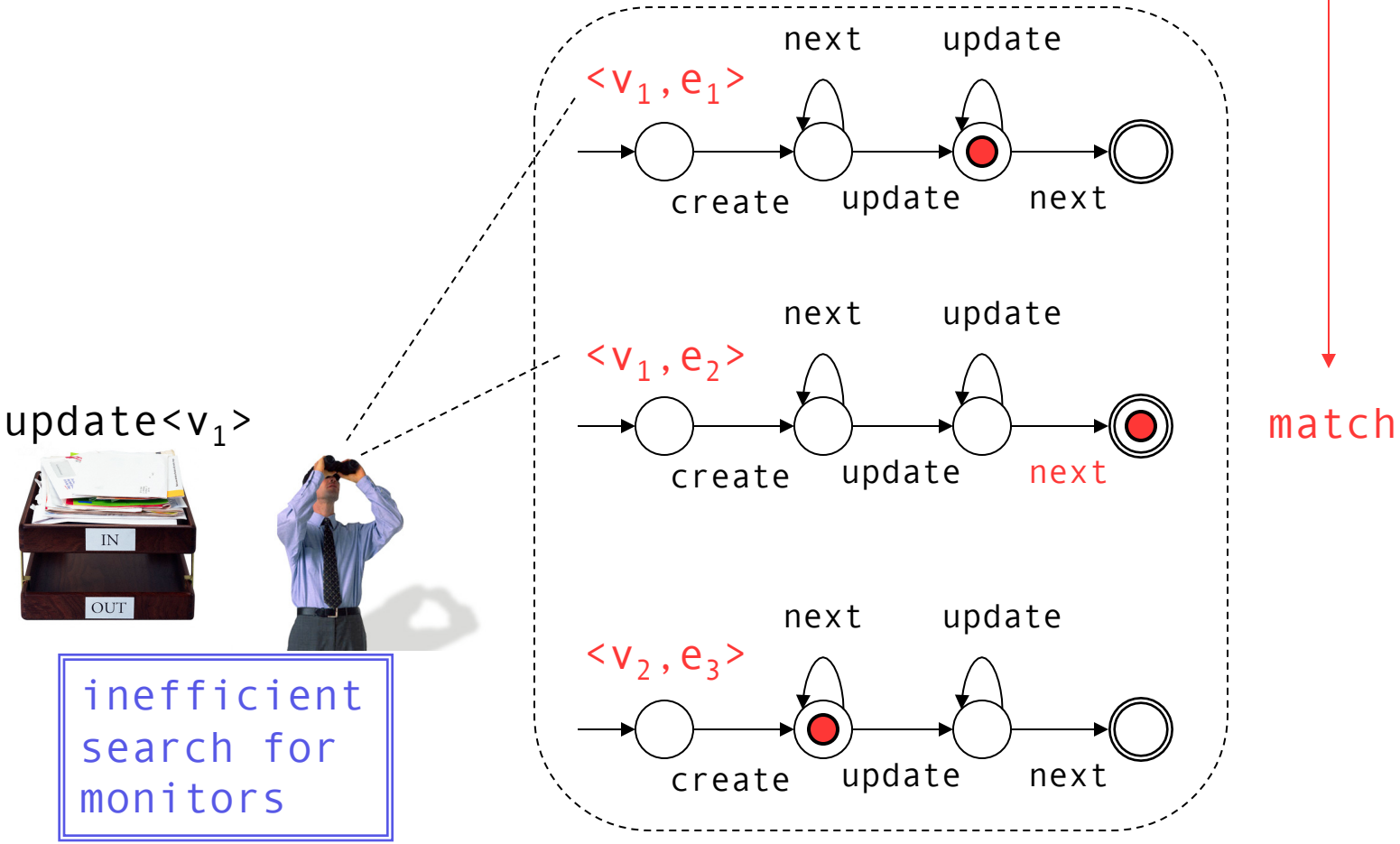
Naïve implementation

```
Vector v1 = new Vector();
Vector v2 = new Vector();
v1.add(1); v1.add(2); v2.add(4); v2.add(5);
Enumeration e1 = v1.elements();
Enumeration e2 = v1.elements();
Enumeration e3 = v2.elements();
while(e1.hasMoreElements())print(e1.nextElement());
v1.add(99);
while(e2.hasMoreElements())print(e2.nextElement());
while(e3.hasMoreElements())print(e3.nextElement());
```



Naïve implementation

```
Vector v1 = new Vector();
Vector v2 = new Vector();
v1.add(1); v1.add(2); v2.add(4); v2.add(5);
Enumeration e1 = v1.elements();
Enumeration e2 = v1.elements();
Enumeration e3 = v2.elements();
while(e1.hasMoreElements())print(e1.nextElement());
v1.add(99);
while(e2.hasMoreElements())print(e2.nextElement());
while(e3.hasMoreElements())print(e3.nextElement());
```



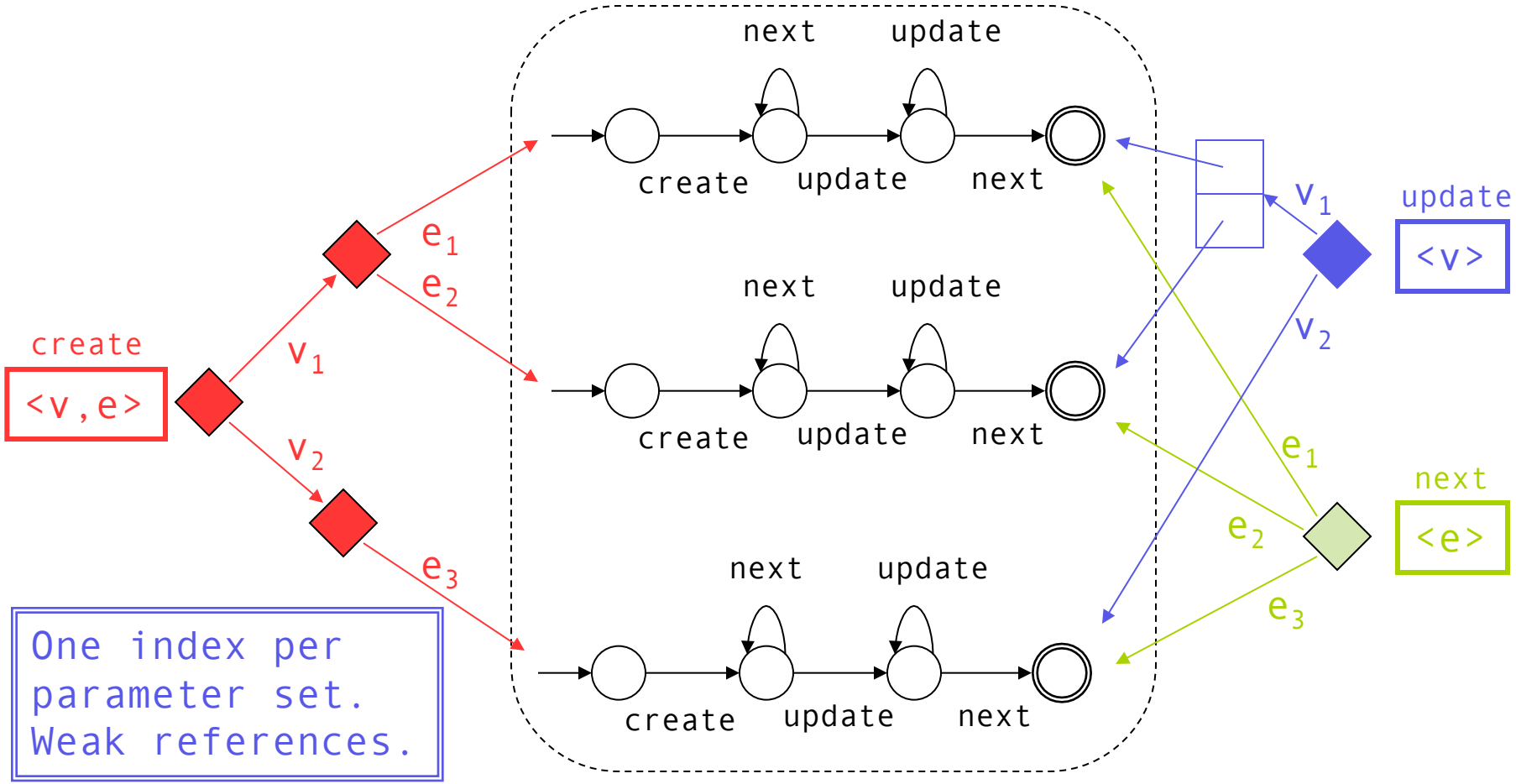
inefficient search for monitors

Indexed implementation

```
Vector v1 = new Vector();
Vector v2 = new Vector();
v1.add(1); v1.add(2); v2.add(4); v2.add(5);
Enumeration e1 = v1.elements();
Enumeration e2 = v1.elements();
Enumeration e3 = v2.elements();
while(e1.hasMoreElements())print(e1.nextElement());
v1.add(99);
while(e2.hasMoreElements())print(e2.nextElement());
while(e3.hasMoreElements())print(e3.nextElement());
```

events:
create<v,e>
update<v>
next<e>

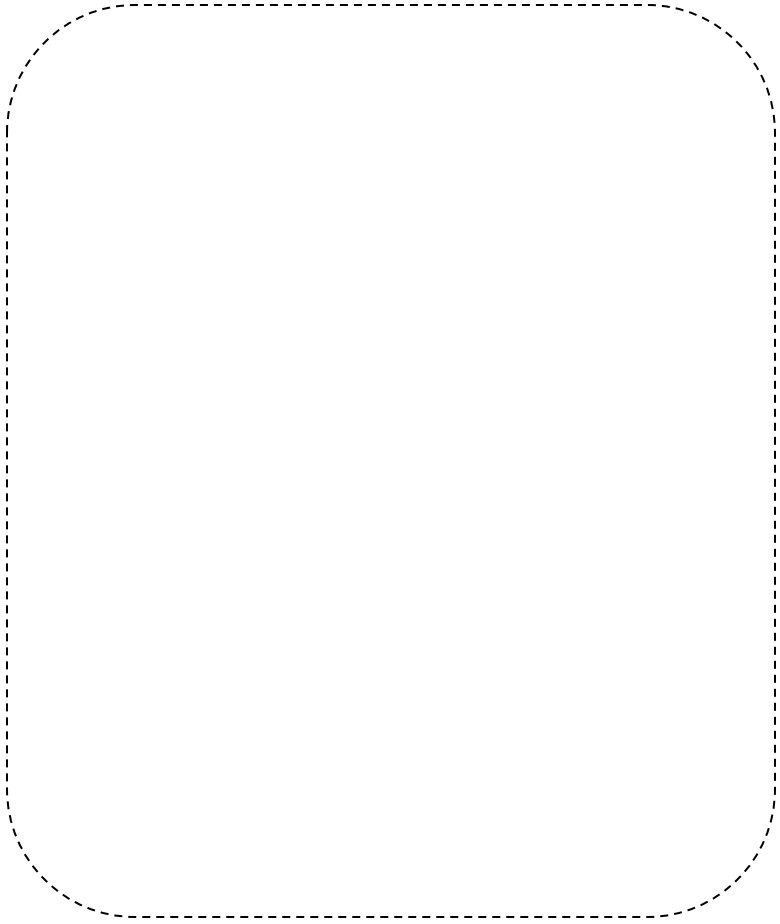
monitor creation event



One index per parameter set. Weak references.

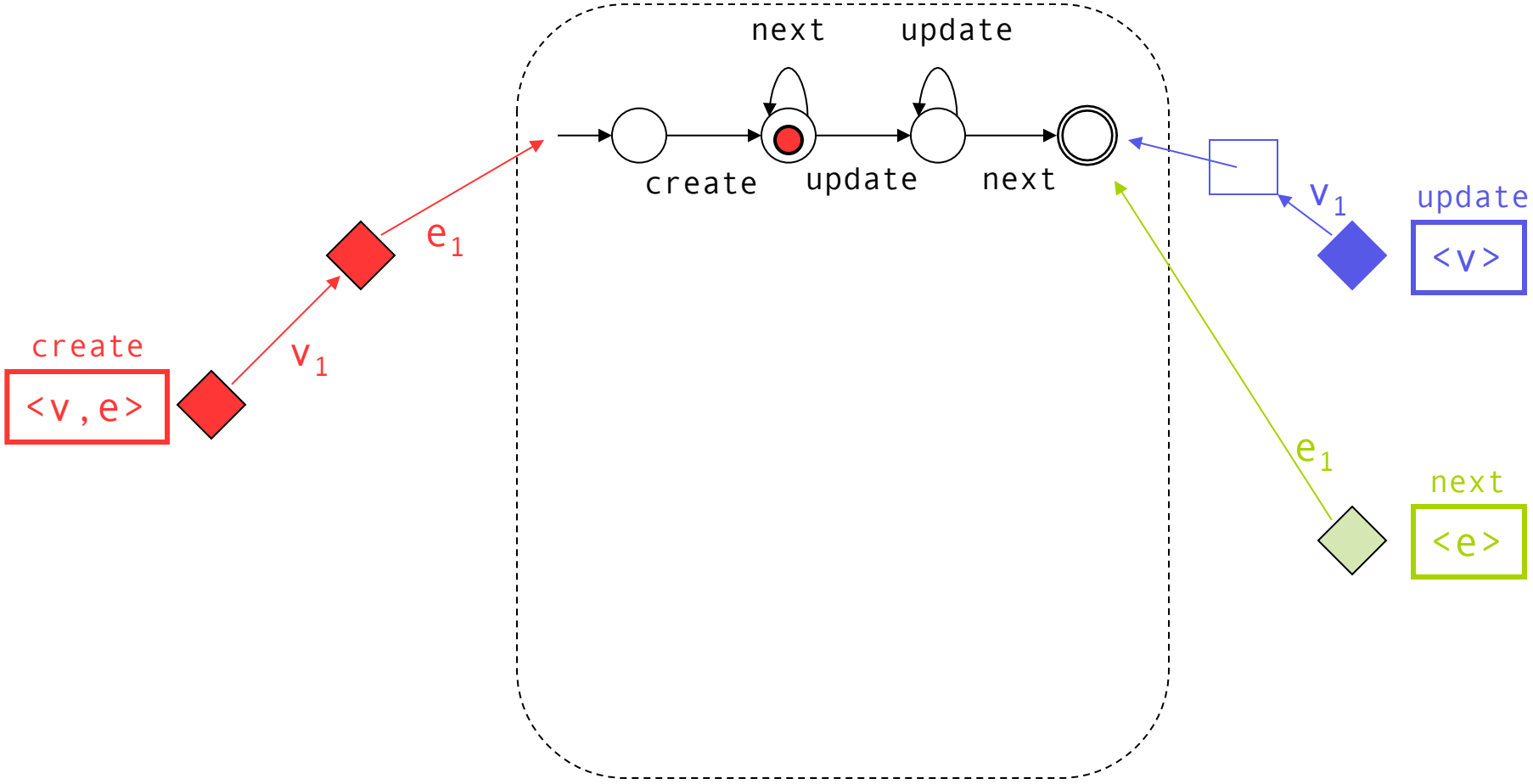
Indexed implementation

```
Vector v1 = new Vector();
Vector v2 = new Vector();
v1.add(1); v1.add(2); v2.add(4); v2.add(5);
Enumeration e1 = v1.elements();
Enumeration e2 = v1.elements();
Enumeration e3 = v2.elements();
while(e1.hasMoreElements())print(e1.nextElement());
v1.add(99);
while(e2.hasMoreElements())print(e2.nextElement());
while(e3.hasMoreElements())print(e3.nextElement());
```



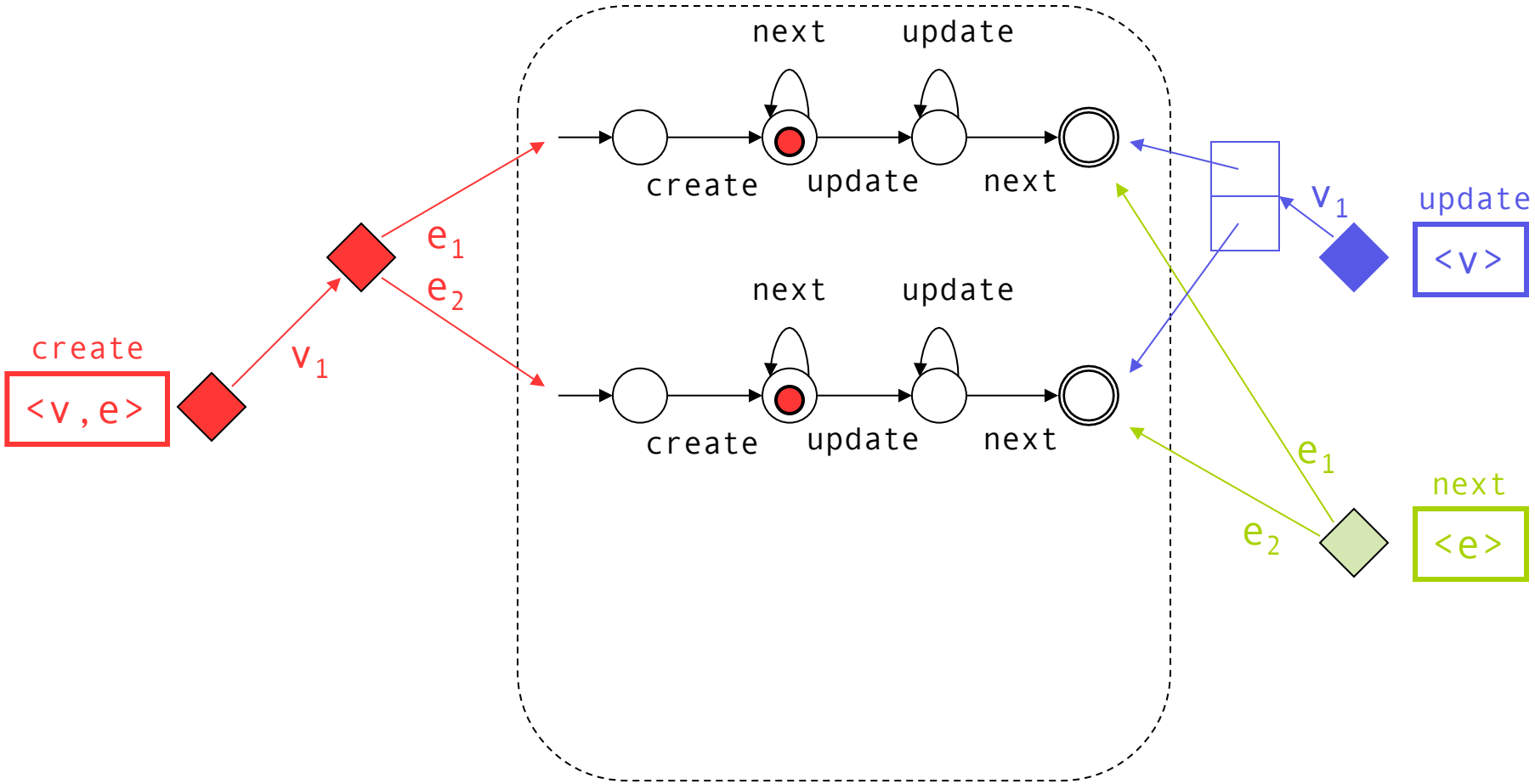
Indexed implementation

```
Vector v1 = new Vector();
Vector v2 = new Vector();
v1.add(1); v1.add(2); v2.add(4); v2.add(5);
Enumeration e1 = v1.elements();
Enumeration e2 = v1.elements();
Enumeration e3 = v2.elements();
while(e1.hasMoreElements())print(e1.nextElement());
v1.add(99);
while(e2.hasMoreElements())print(e2.nextElement());
while(e3.hasMoreElements())print(e3.nextElement());
```



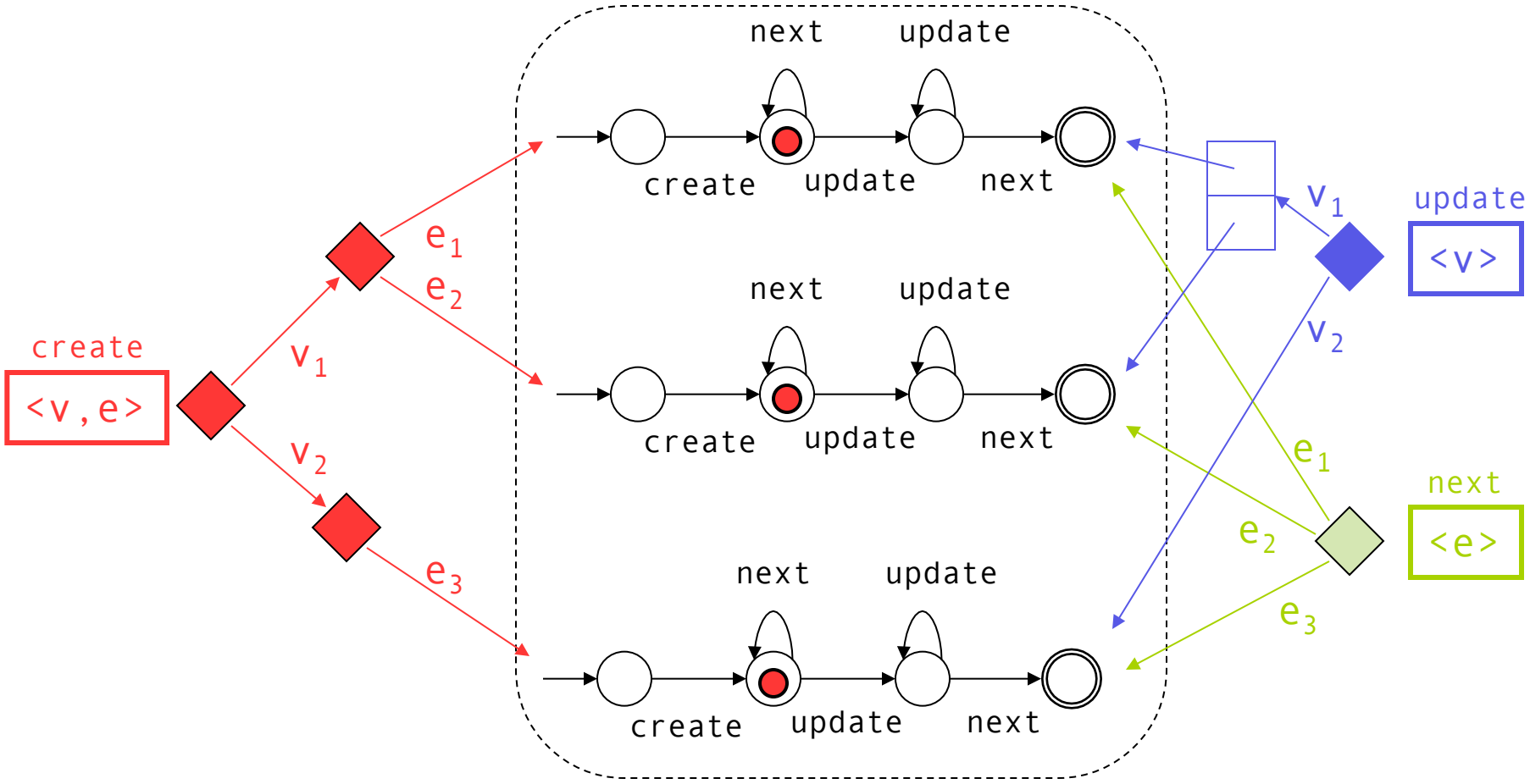
Indexed implementation

```
Vector v1 = new Vector();
Vector v2 = new Vector();
v1.add(1); v1.add(2); v2.add(4); v2.add(5);
Enumeration e1 = v1.elements();
Enumeration e2 = v1.elements();
Enumeration e3 = v2.elements();
while(e1.hasMoreElements())print(e1.nextElement());
v1.add(99);
while(e2.hasMoreElements())print(e2.nextElement());
while(e3.hasMoreElements())print(e3.nextElement());
```



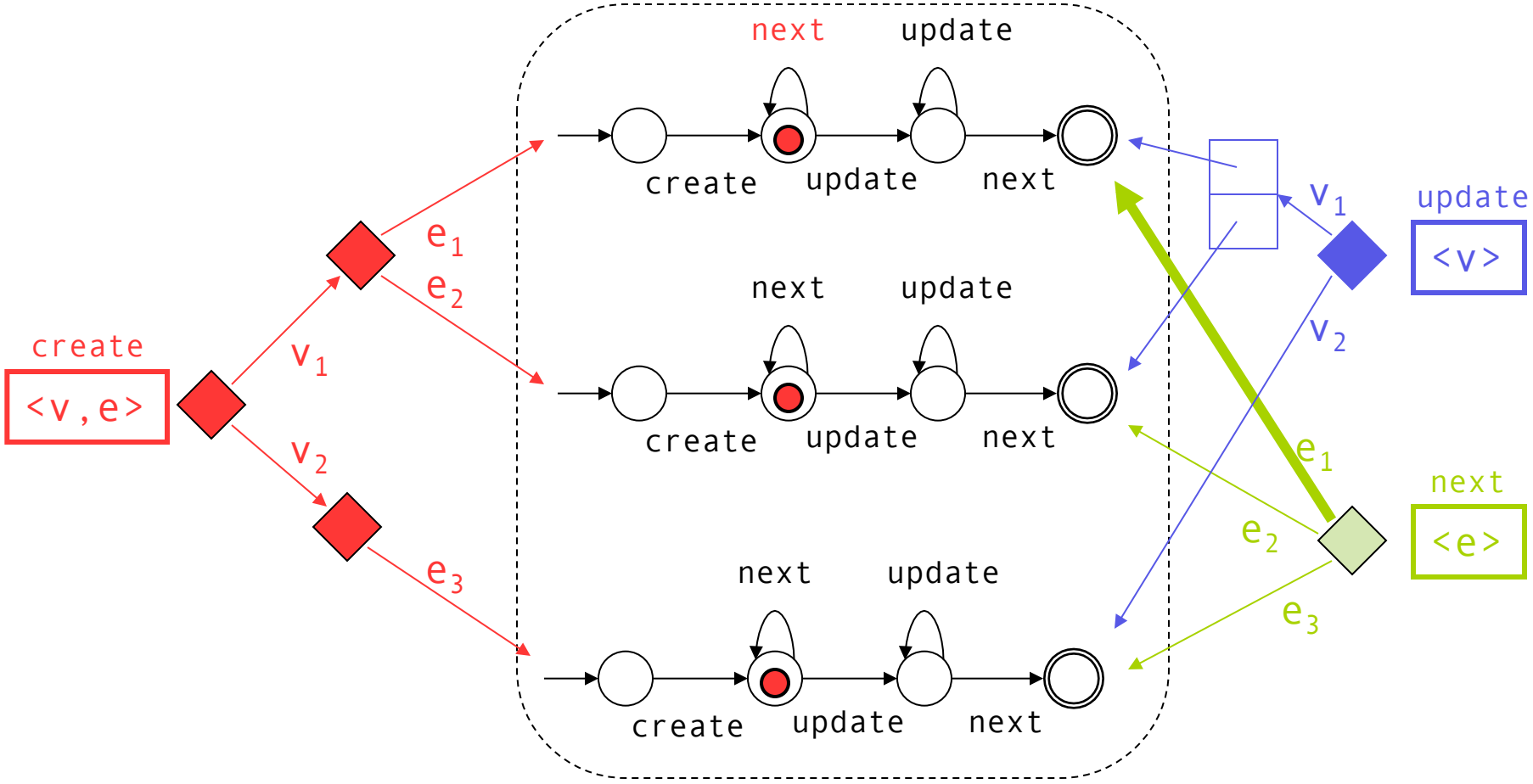
Indexed implementation

```
Vector v1 = new Vector();
Vector v2 = new Vector();
v1.add(1); v1.add(2); v2.add(4); v2.add(5);
Enumeration e1 = v1.elements();
Enumeration e2 = v1.elements();
Enumeration e3 = v2.elements();
while(e1.hasMoreElements())print(e1.nextElement());
v1.add(99);
while(e2.hasMoreElements())print(e2.nextElement());
while(e3.hasMoreElements())print(e3.nextElement());
```



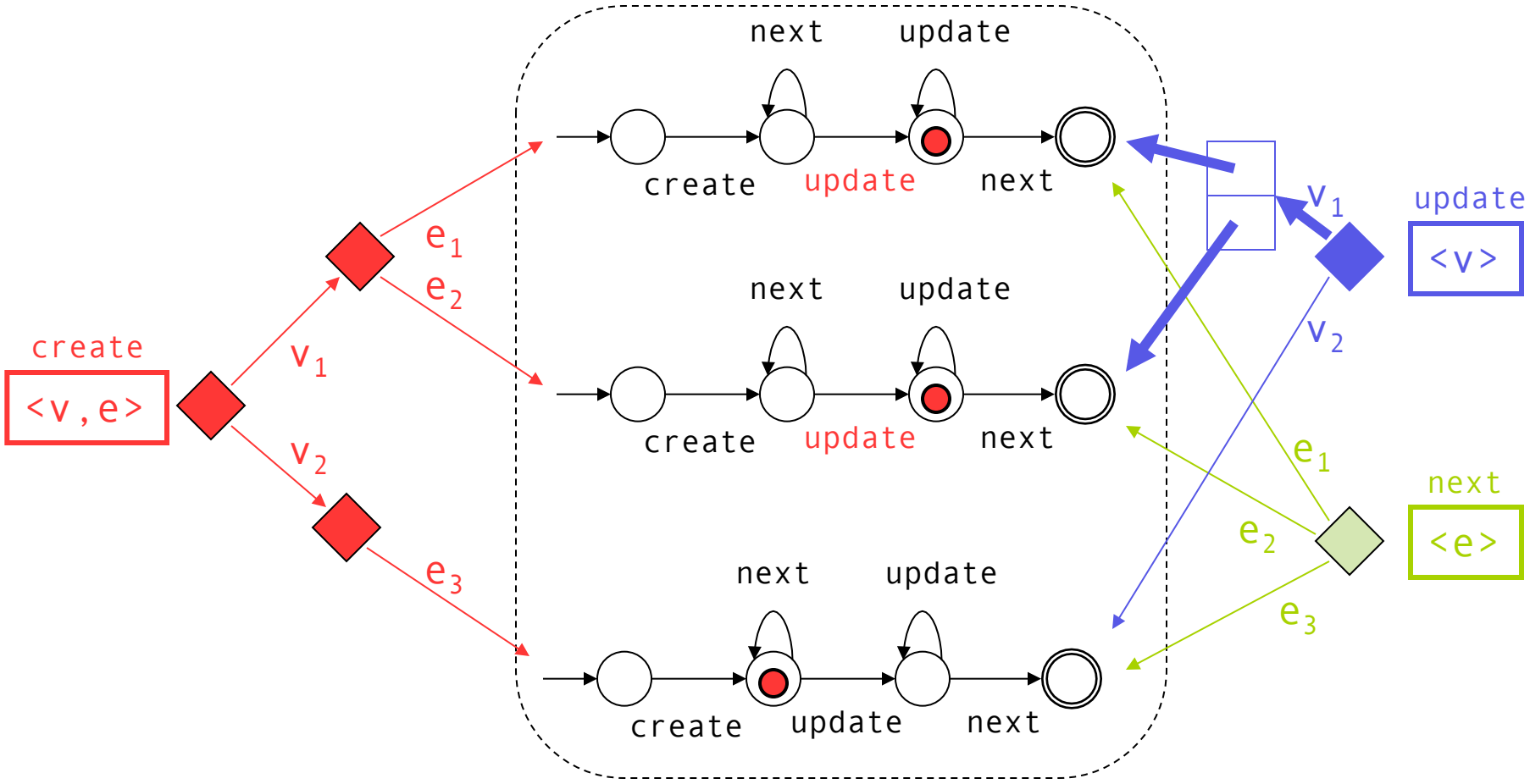
Indexed implementation

```
Vector v1 = new Vector();
Vector v2 = new Vector();
v1.add(1); v1.add(2); v2.add(4); v2.add(5);
Enumeration e1 = v1.elements();
Enumeration e2 = v1.elements();
Enumeration e3 = v2.elements();
while(e1.hasMoreElements())print(e1.nextElement());
v1.add(99);
while(e2.hasMoreElements())print(e2.nextElement());
while(e3.hasMoreElements())print(e3.nextElement());
```



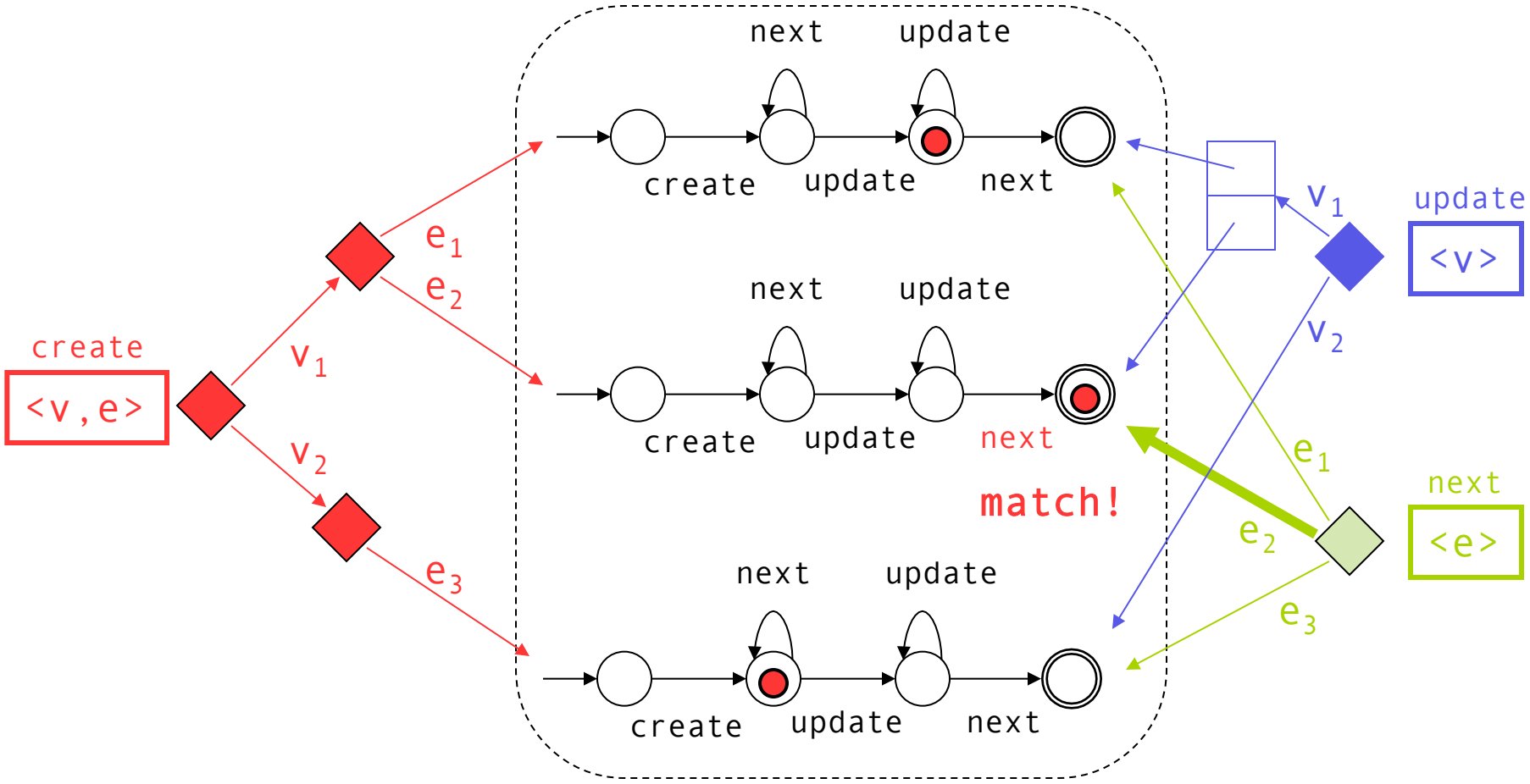
Indexed implementation

```
Vector v1 = new Vector();
Vector v2 = new Vector();
v1.add(1); v1.add(2); v2.add(4); v2.add(5);
Enumeration e1 = v1.elements();
Enumeration e2 = v1.elements();
Enumeration e3 = v2.elements();
while(e1.hasMoreElements())print(e1.nextElement());
v1.add(99);
while(e2.hasMoreElements())print(e2.nextElement());
while(e3.hasMoreElements())print(e3.nextElement());
```



Indexed implementation

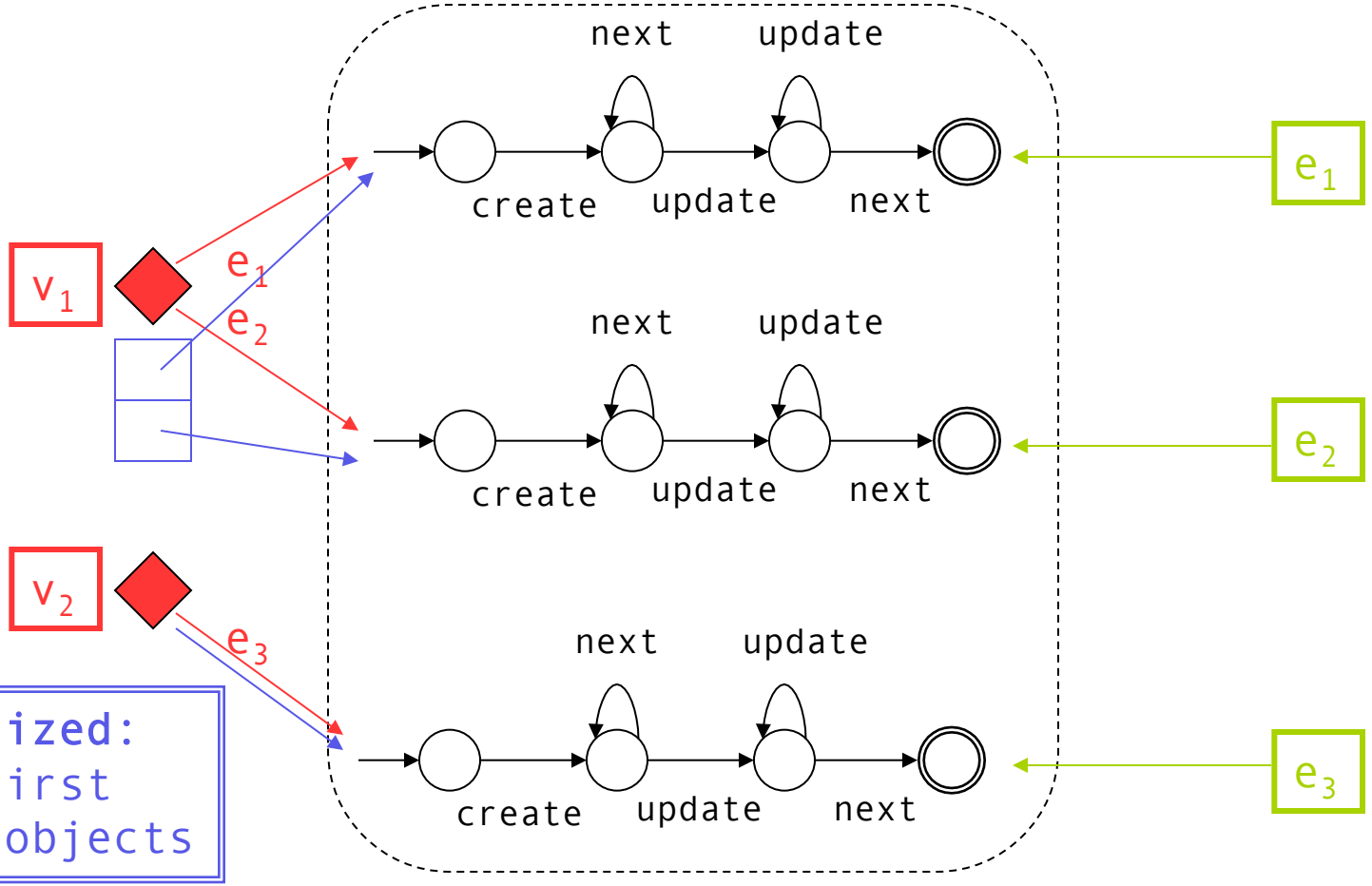
```
Vector v1 = new Vector();
Vector v2 = new Vector();
v1.add(1); v1.add(2); v2.add(4); v2.add(5);
Enumeration e1 = v1.elements();
Enumeration e2 = v1.elements();
Enumeration e3 = v2.elements();
while(e1.hasMoreElements())print(e1.nextElement());
v1.add(99);
while(e2.hasMoreElements())print(e2.nextElement());
while(e3.hasMoreElements())print(e3.nextElement());
```



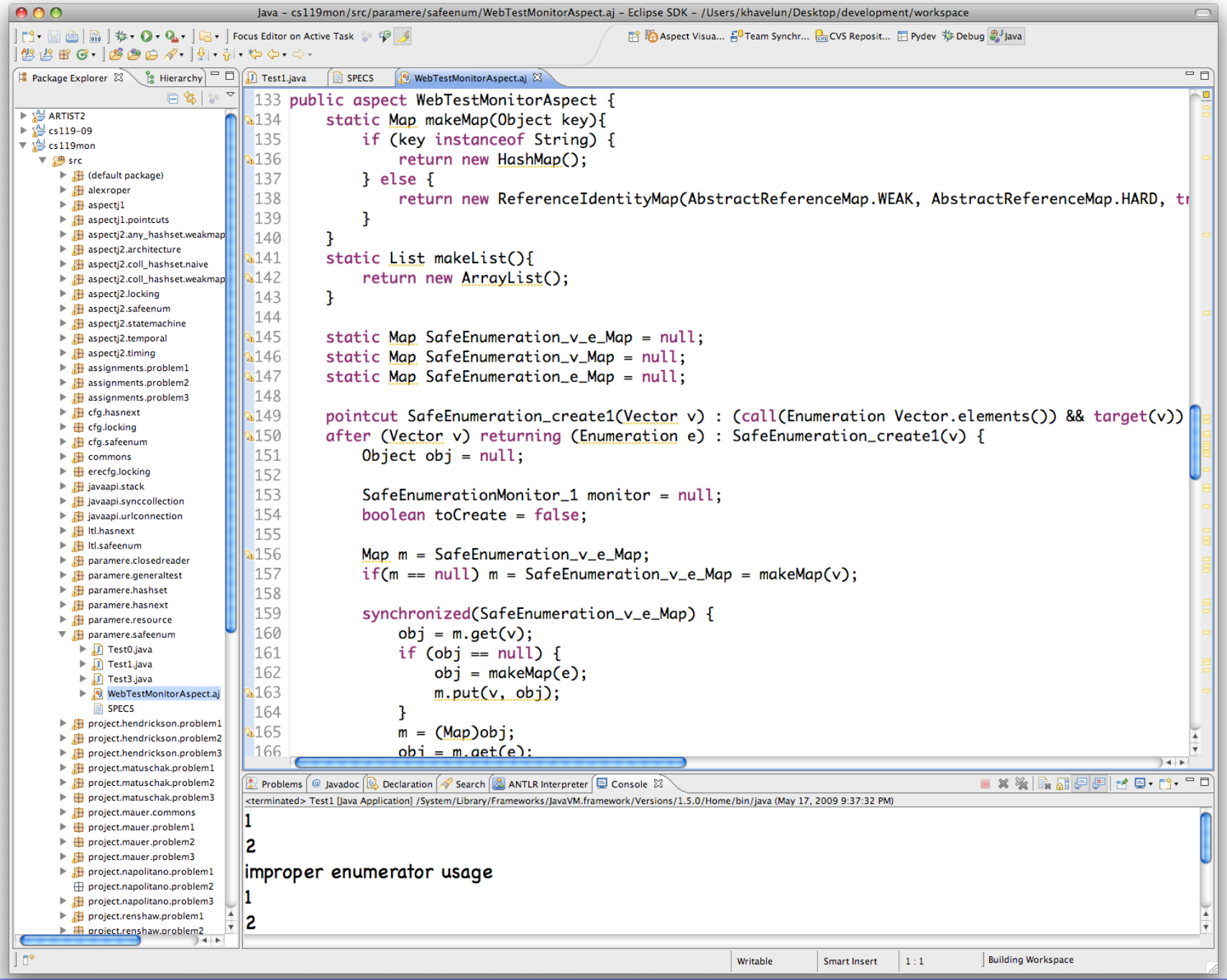
```
Vector v1 = new Vector();
Vector v2 = new Vector();
v1.add(1); v1.add(2); v2.add(4); v2.add(5);
Enumeration e1 = v1.elements();
Enumeration e2 = v1.elements();
Enumeration e3 = v2.elements();
while(e1.hasMoreElements())print(e1.nextElement());
v1.add(99);
while(e2.hasMoreElements())print(e2.nextElement());
while(e3.hasMoreElements())print(e3.nextElement());
```

events:
create<v,e>
update<v>
next<e>

monitor creation event



decentralized:
storing first index in objects


Java – cs119mon/src/paramere/safeenum/WebTestMonitorAspect.aj – Eclipse SDK – /Users/khavelun/Desktop/development/workspace
Package Explorer
Hierarchy
Test1.java
SPECS
WebTestMonitorAspect.aj
public aspect WebTestMonitorAspect {
static Map makeMap(Object key){
if (key instanceof String) {
return new HashMap();
} else {
return new ReferenceIdentityMap(AbstractReferenceMap.WEAK, AbstractReferenceMap.HARD,
}
}
static List makeList(){
return new ArrayList();
}
static Map SafeEnumeration_v_e_Map = null;
static Map SafeEnumeration_v_Map = null;
static Map SafeEnumeration_e_Map = null;
pointcut SafeEnumeration_create1(Vector v) : (call(Enumeration Vector.elements()) && target(v))
after (Vector v) returning (Enumeration e) : SafeEnumeration_create1(v) {
Object obj = null;
SafeEnumerationMonitor_1 monitor = null;
boolean toCreate = false;
Map m = SafeEnumeration_v_e_Map;
if(m == null) m = SafeEnumeration_v_e_Map = makeMap(v);
synchronized(SafeEnumeration_v_e_Map) {
obj = m.get(v);
if (obj == null) {
obj = makeMap(e);
m.put(v, obj);
}
m = (Map)obj;
Problems
Javadoc
Declaration
Search
ANTLR Interpreter
Console
<terminated> Test1 [Java Application] /System/Library/Frameworks/JavaVM.framework/Versions/1.5.0/Home/bin/java (May 17, 2009 9:37:32 PM)
1
2
improper enumerator usage
1
2
Writable
Smart Insert
1 : 1
Building Workspace

properties of Java library APIs

R_3: An collection should not be modified while it is a member of a hashset (don't change the hashcode).

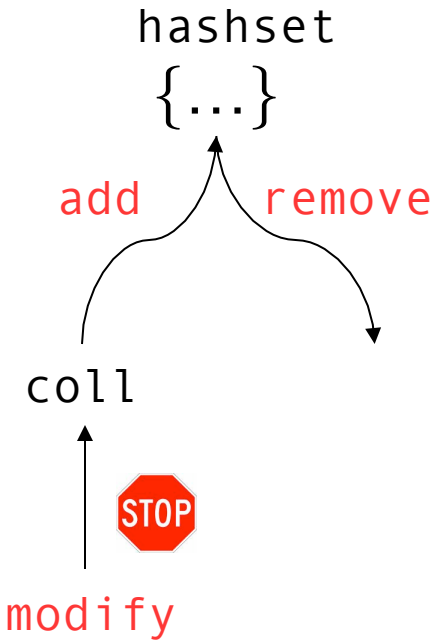
java.util
Class HashSet<E>

of

java.util
Interface Collection<E>

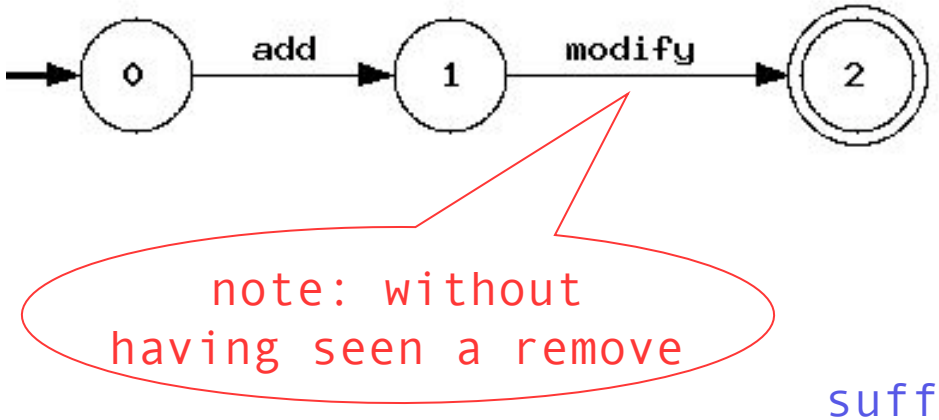
java.lang.Object
└ java.util.AbstractCollection<E>
└ java.util.AbstractSet<E>
└ java.util.HashSet<E>

R_3: An collection should not be modified while it is a member of a hashset (don't change the hashcode).



add	collection to hashset
remove	collection from hashset
modify	collection

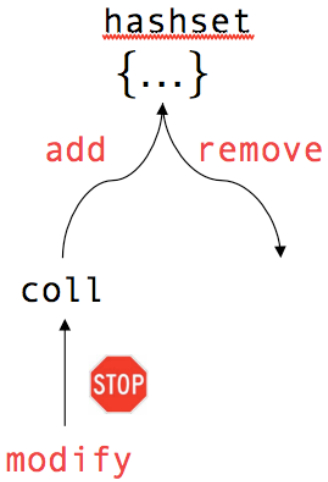
add modify



suffix validation

adding collection to hash set and then updating it

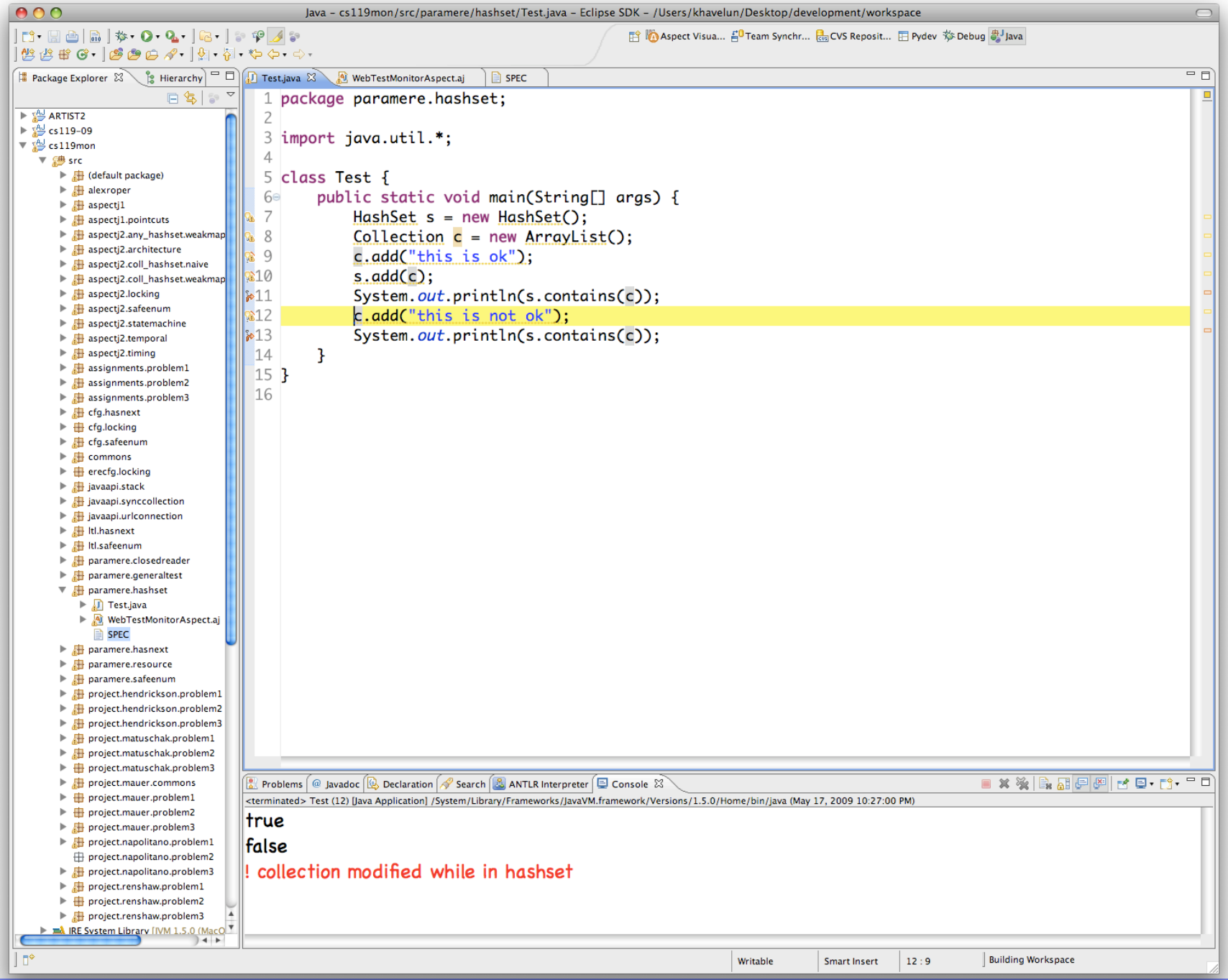
```
HashSet s = new HashSet();
Collection c = new ArrayList();
c.add("this is ok");
s.add(c);
System.out.println(s.contains(c));
c.add("this is not ok");
System.out.println(s.contains(c));
```

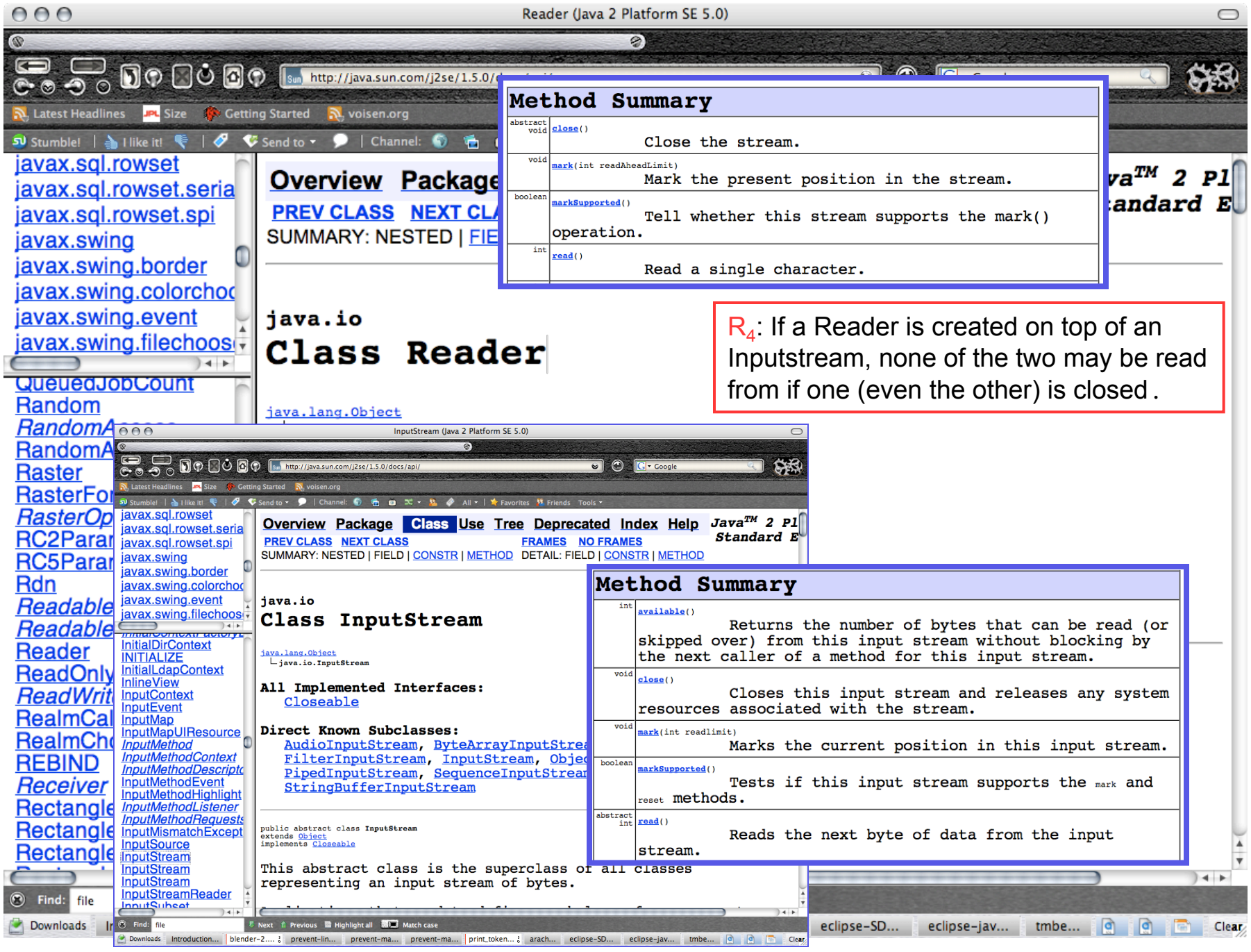


bad update changing hashcode of collection c

subsequently s.contains(c) yields false

```
package paramere.hashset;

import java.io.*;
import java.util.*;

suffix HashSet(HashSet s, Collection c) {
    event put before(HashSet s, Collection c) : call(* HashSet.add(..)) && target(s) && args(c) {}
        event remove before(HashSet s, Collection c): call(* HashSet.remove(..)) && target(s) && args(c) {}
        event modify before(Collection c) : (call(* Collection.add*(..)) || call(* Collection.remove*(..))) &&
                                target(c) {}

    ere : put modify

    @match {
        System.err.println("! collection modified while in hashset");
        __RESET;
    }
}
```


Java - cs119mon/src/paramere/hashset/Test.java - Eclipse SDK - /Users/khavelun/Desktop/development/workspace
Package Explorer
Hierarchy
Test.java
WebTestMonitorAspect.aj
SPEC
1 package paramere.hashset;
3 import java.util.*;
5 class Test {
6 public static void main(String[] args) {
7 HashSet s = new HashSet();
8 Collection c = new ArrayList();
9 c.add("this is ok");
10 s.add(c);
11 System.out.println(s.contains(c));
12 c.add("this is not ok");
13 System.out.println(s.contains(c));
14 }
15 }
Problems
Javadoc
Declaration
Search
ANTLR Interpreter
Console
<terminated> Test (12) [Java Application] /System/Library/Frameworks/JavaVM.framework/Versions/1.5.0/Home/bin/java (May 17, 2009 10:27:00 PM)
true
false
! collection modified while in hashset
Writable
Smart Insert
12 : 9
Building Workspace


Reader (Java 2 Platform SE 5.0)
Method Summary
close()
Close the stream.
mark(int readAheadLimit)
Mark the present position in the stream.
markSupported()
Tell whether this stream supports the mark() operation.
read()
Read a single character.
java.io
Class Reader
java.lang.Object
R4: If a Reader is created on top of an Inputstream, none of the two may be read from if one (even the other) is closed .
InputStream (Java 2 Platform SE 5.0)
Overview Package Class Use Tree Deprecated Index Help
PREV CLASS NEXT CLASS
FRAMES NO FRAMES
SUMMARY: NESTED | FIELD | CONSTR | METHOD
DETAIL: FIELD | CONSTR | METHOD
java.io
Class InputStream
java.lang.Object
java.io.InputStream
All Implemented Interfaces:
Closeable
Direct Known Subclasses:
AudioInputStream, ByteArrayInputStream, FilterInputStream, InputStream, ObjectInputStream, PipedInputStream, SequenceInputStream, StringBufferInputStream
public abstract class InputStream
extends Object
implements Closeable
This abstract class is the superclass of all classes representing an input stream of bytes.
Method Summary
available()
Returns the number of bytes that can be read (or skipped over) from this input stream without blocking by the next caller of a method for this input stream.
close()
Closes this input stream and releases any system resources associated with the stream.
mark(int readlimit)
Marks the current position in this input stream.
markSupported()
Tests if this input stream supports the mark and reset methods.
read()
Reads the next byte of data from the input stream.

something one should not do:
- create reader from input stream,
- close down reader,
- continue reading from inputstream

```
try{
  InputStream i = new StringBufferInputStream("AB");
  InputStreamReader r = new InputStreamReader(i);
  System.out.println((char)r.read());
  r.close();
  System.out.println((char)i.read());
} catch(IOException e) {
  System.out.println("*** io error");
}
```

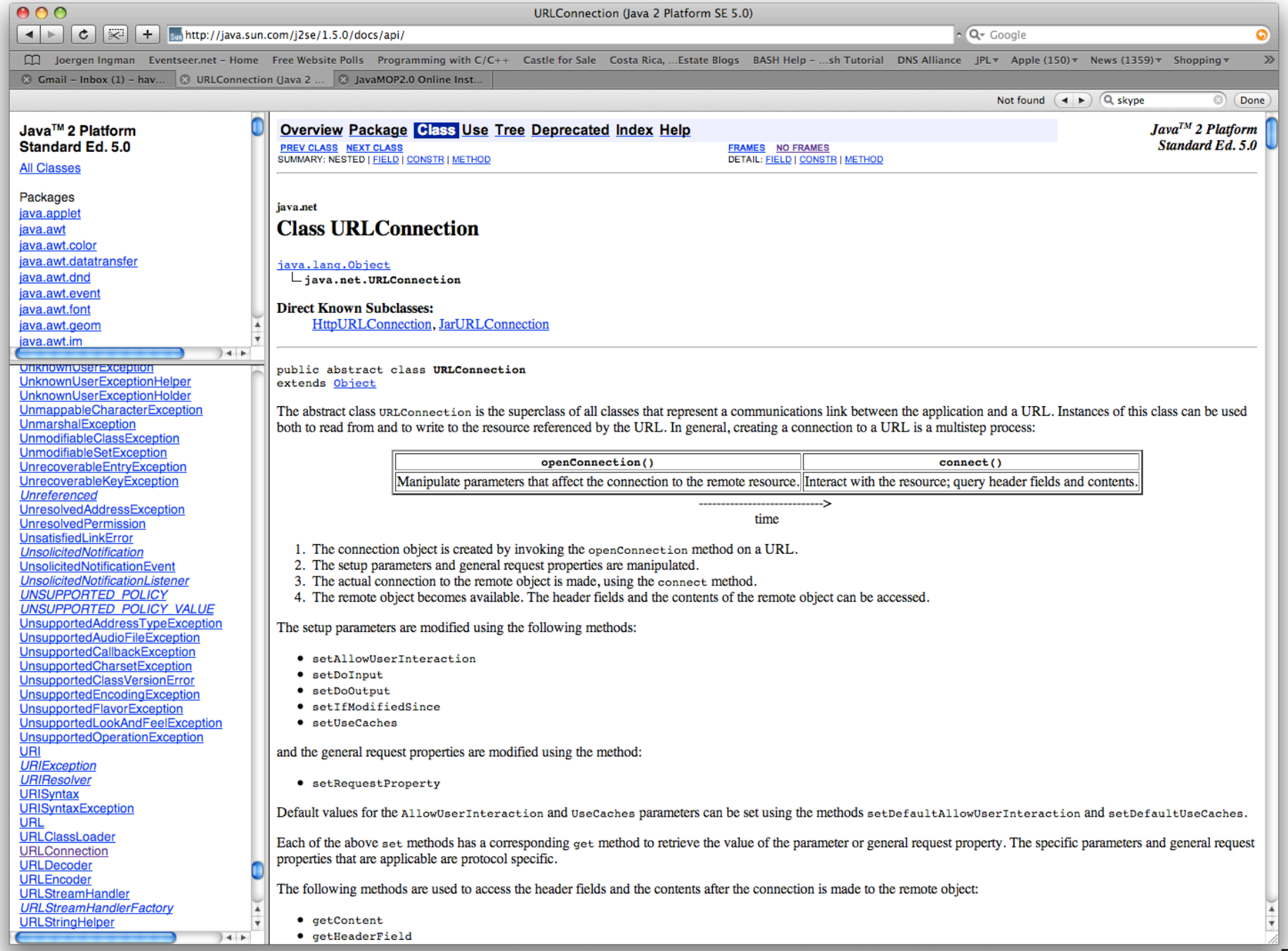
```
closing down reader
reading inputstream
```

output:
A
?

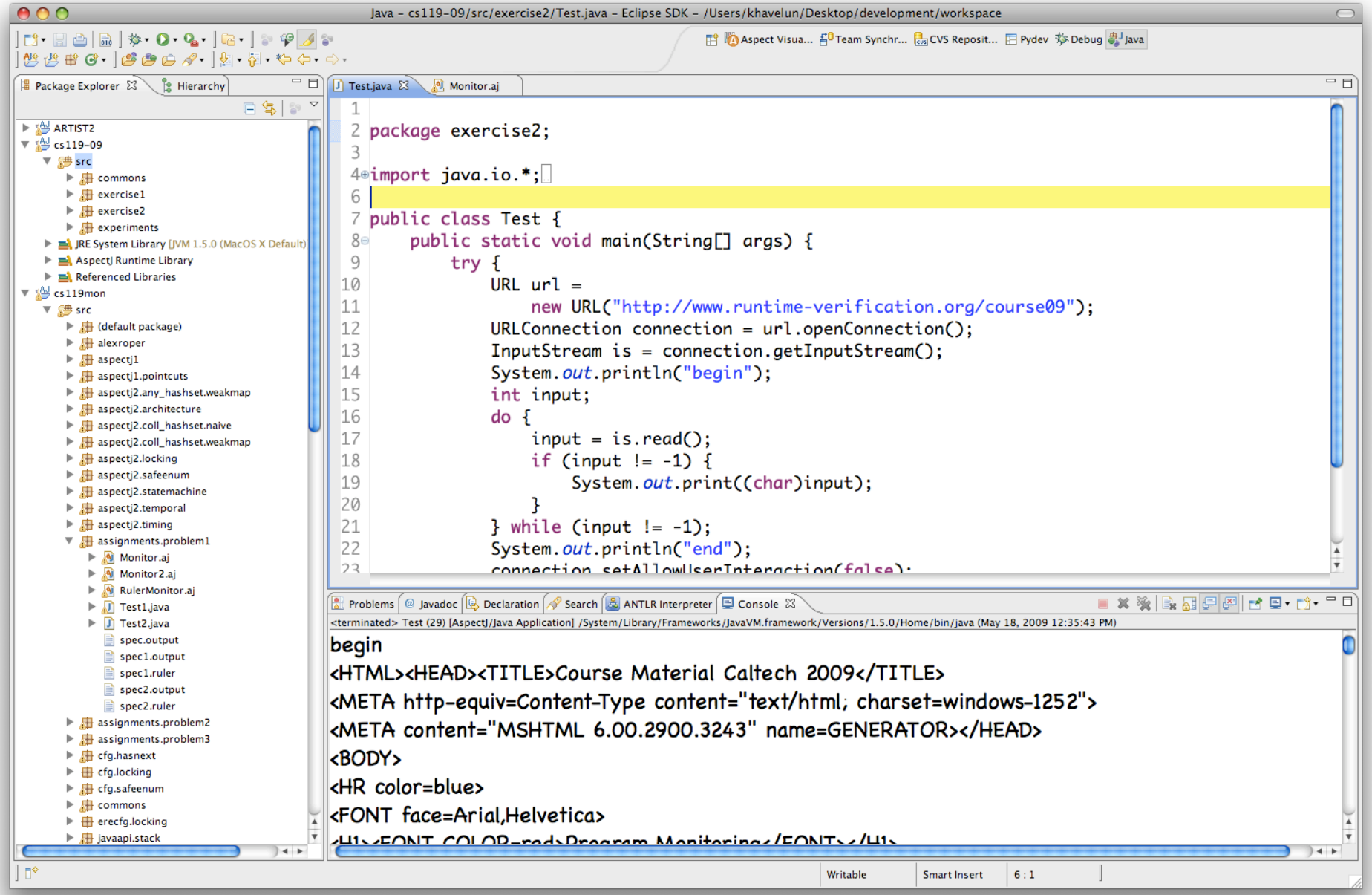
```
package paramere.closedreader;

import java.io.*;
import java.util.*;

suffix ClosedReader(Reader r, InputStream s) {
        event create after(InputStream s) returning (Reader r):
          call(Reader+.new(..)) && args(s) {}
        event closeS before(InputStream s) : call(* InputStream.close()) && target(s) {}
        event closeR before(Reader r) : call(* Reader.close()) && target(r) {}
        event readS before(InputStream s) : call(* InputStream.read*(..)) && target(s) {}
        event readR before(Reader r) : call(* Reader.read*(..)) && target(r) {}

    ere : create (readS + readR)* (closeS + closeR) (readS + readR)

    @match {
        System.out.println("! one input source has been closed");
        __RESET;
    }
}
```

Introduction to

ASSIGNMENT 2


URLConnection (Java 2 Platform SE 5.0)
http://java.sun.com/j2se/1.5.0/docs/api/
Google
Joergen Ingman
Eventseer.net - Home
Free Website Polls
Programming with C/C++
Castle for Sale
Costa Rica, ...Estate Blogs
BASH Help - ...sh Tutorial
DNS Alliance
JPL
Apple (150)
News (1359)
Shopping
Gmail - Inbox (1) - hav...
URLConnection (Java 2 ...
JavaMOP2.0 Online Inst...
Not found
skype
Done
Java™ 2 Platform Standard Ed. 5.0
All Classes
Packages
java.applet
java.awt
java.awt.color
java.awt.datatransfer
java.awt.dnd
java.awt.event
java.awt.font
java.awt.geom
java.awt.im
UnknownUserException
UnknownUserExceptionHelper
UnknownUserExceptionHolder
UnmappableCharacterException
UnmarshalException
UnmodifiableClassException
UnmodifiableSetException
UnrecoverableEntryException
UnrecoverableKeyException
Unreferenced
UnresolvedAddressException
UnresolvedPermission
UnsatisfiedLinkError
UnsolicitedNotification
UnsolicitedNotificationEvent
UnsolicitedNotificationListener
UNSUPPORTED_POLICY
UNSUPPORTED_POLICY_VALUE
UnsupportedAddressTypeException
UnsupportedAudioFileException
UnsupportedCallbackException
UnsupportedCharsetException
UnsupportedClassVersionError
UnsupportedEncodingException
UnsupportedFlavorException
UnsupportedLookAndFeelException
UnsupportedOperationException
URI
URIException
URIResolver
URISyntax
URISyntaxException
URL
URLClassLoader
URLConnection
URLDecoder
URLEncoder
URLStreamHandler
URLStreamHandlerFactory
URLStringHelper
Overview Package Class Use Tree Deprecated Index Help
PREV CLASS NEXT CLASS
FRAMES NO FRAMES
SUMMARY: NESTED | FIELD | CONSTR | METHOD
DETAIL: FIELD | CONSTR | METHOD
Java™ 2 Platform Standard Ed. 5.0
java.net
Class URLConnection
java.lang.Object
java.net.URLConnection
Direct Known Subclasses:
HttpURLConnection, JarURLConnection
public abstract class URLConnection
extends Object
The abstract class URLConnection is the superclass of all classes that represent a communications link between the application and a URL. Instances of this class can be used both to read from and to write to the resource referenced by the URL. In general, creating a connection to a URL is a multistep process:
openConnection()
connect()
Manipulate parameters that affect the connection to the remote resource.
Interact with the resource; query header fields and contents.
time
1. The connection object is created by invoking the openConnection method on a URL.
2. The setup parameters and general request properties are manipulated.
3. The actual connection to the remote object is made, using the connect method.
4. The remote object becomes available. The header fields and the contents of the remote object can be accessed.
The setup parameters are modified using the following methods:
setAllowUserInteraction
setDoInput
setDoOutput
setIfModifiedSince
setUseCaches
and the general request properties are modified using the method:
setRequestProperty
Default values for the AllowUserInteraction and UseCaches parameters can be set using the methods setDefaultAllowUserInteraction and setDefaultUseCaches.
Each of the above set methods has a corresponding get method to retrieve the value of the parameter or general request property. The specific parameters and general request properties that are applicable are protocol specific.
The following methods are used to access the header fields and the contents after the connection is made to the remote object:
getContent
getHeaderField

```java
package exercise2;

import java.io.*;
import java.net.*;

public class Test {
    public static void main(String[] args) {
        try {
            URL url =
                new URL("http://www.runtime-verification.org/course09");
            URLConnection connection = url.openConnection();
            InputStream is = connection.getInputStream();
            System.out.println("begin");
            int input;
            do {
                input = is.read();
                if (input != -1) {
                    System.out.print((char)input);
                }
            } while (input != -1);
            System.out.println("end");
            connection.setAllowUserInteraction(false);
        }catch(Exception e) {}
    }
}
```


Java - cs119-09/src/exercise2/Test.java - Eclipse SDK - /Users/khavelun/Desktop/development/workspace
Aspect Visua...
Team Synchr...
CVS Reposit...
Pydev
Debug
Java
Package Explorer
Hierarchy
ARTIST2
cs119-09
src
commons
exercise1
exercise2
experiments
JRE System Library [JVM 1.5.0 (MacOS X Default)]
AspectJ Runtime Library
Referenced Libraries
cs119mon
src
(default package)
alexroper
aspectj1
aspectj1.pointcuts
aspectj2.any_hashset.weakmap
aspectj2.architecture
aspectj2.coll_hashset.naive
aspectj2.coll_hashset.weakmap
aspectj2.locking
aspectj2.safeenum
aspectj2.statemachine
aspectj2.temporal
aspectj2.timing
assignments.problem1
Monitor.aj
Monitor2.aj
RulerMonitor.aj
Test1.java
Test2.java
spec.output
spec1.output
spec1.ruler
spec2.output
spec2.ruler
assignments.problem2
assignments.problem3
cfg.hasnext
cfg.locking
cfg.safeenum
commons
erecfg.locking
javaapi.stack
Test.java
Monitor.aj
1 package exercise2;
2
3 import java.io.*;
4
5 public class Test {
6 public static void main(String[] args) {
7 try {
8 URL url =
9 new URL("http://www.runtime-verification.org/course09");
10 URLConnection connection = url.openConnection();
11 InputStream is = connection.getInputStream();
12 System.out.println("begin");
13 int input;
14 do {
15 input = is.read();
16 if (input != -1) {
17 System.out.print((char)input);
18 }
19 } while (input != -1);
20 System.out.println("end");
Problems
Javadoc
Declaration
Search
ANTLR Interpreter
Console
<terminated> Test (29) [AspectJ/Java Application] /System/Library/Frameworks/JavaVM.framework/Versions/1.5.0/Home/bin/java (May 18, 2009 12:35:43 PM)
begin
<HTML><HEAD><TITLE>Course Material Caltech 2009</TITLE>
<META http-equiv=Content-Type content="text/html; charset=windows-1252">
<META content="MSHTML 6.00.2900.3243" name=GENERATOR></HEAD>
<BODY>
<HR color=blue>
<FONT face=Arial,Helvetica>
Writable
Smart Insert
6 : 1

policy

1. *it is not allowed to call a* `set`*-method on a URLConnection object after a* `get`*-method has been called. In this case the* `get` *method is* `getInputStream()`.

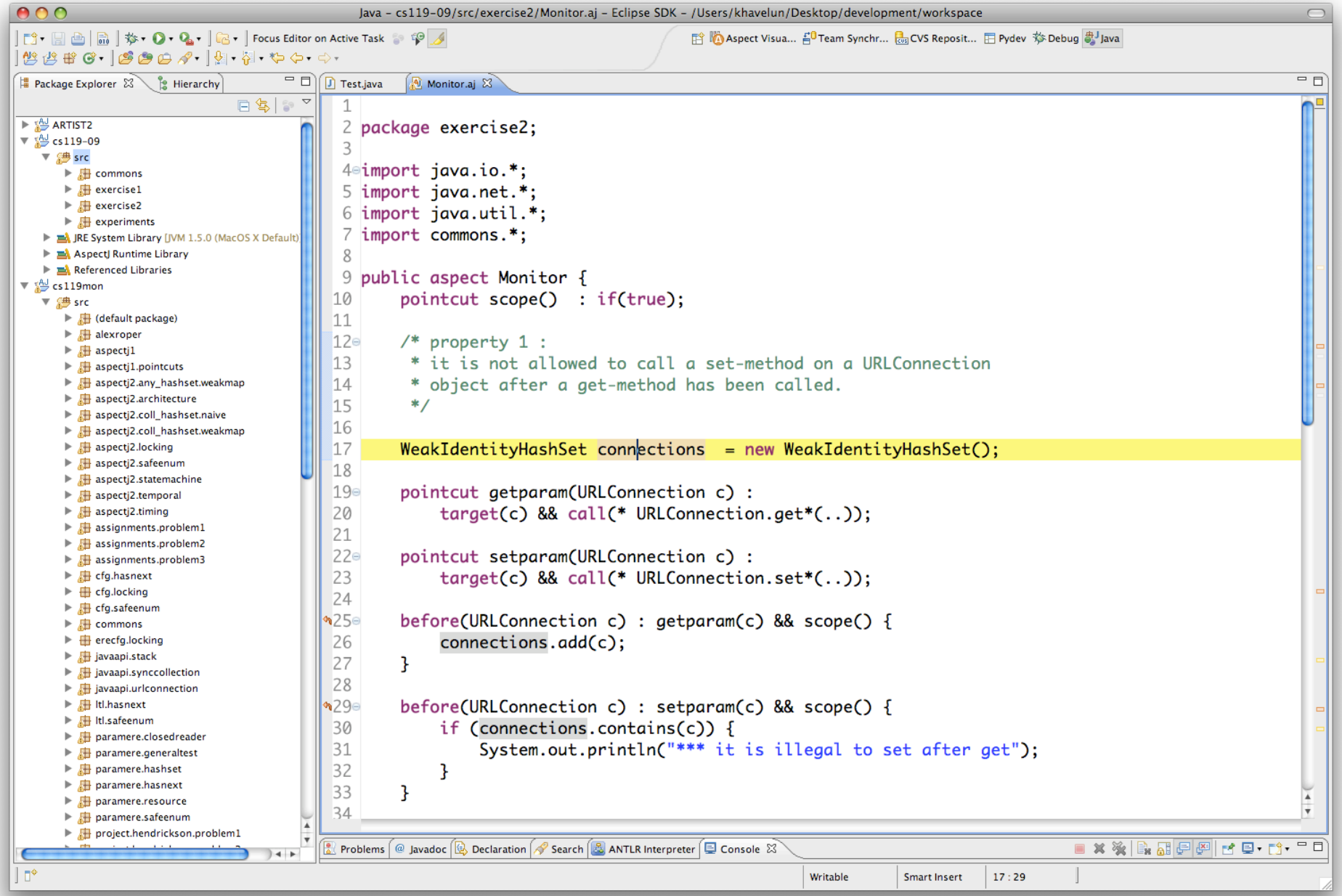
2. *an input stream that has been retrieved from a URLConnection object using the* `getInputStream()` *method should be closed before the main program terminates.*

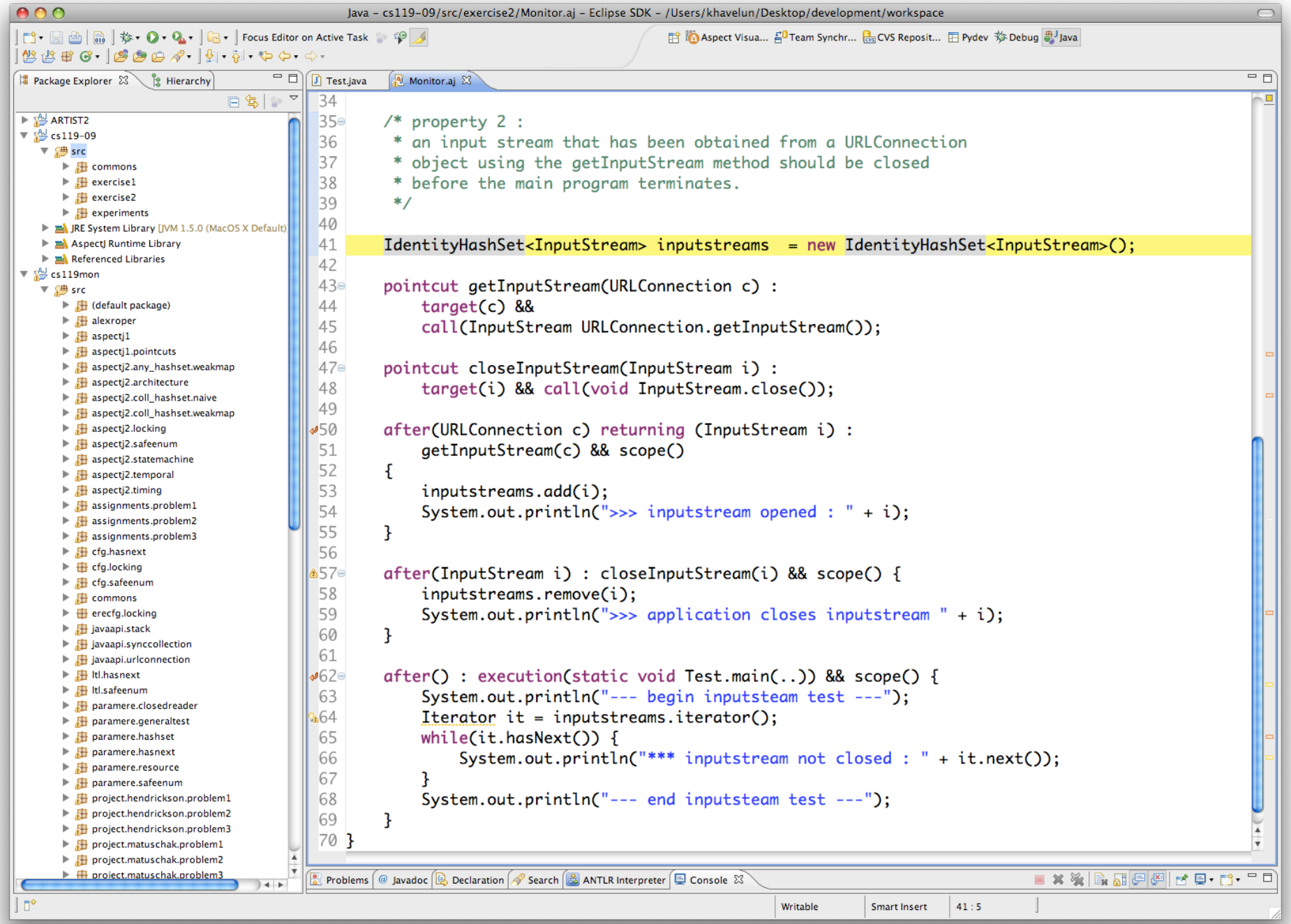
the assignment

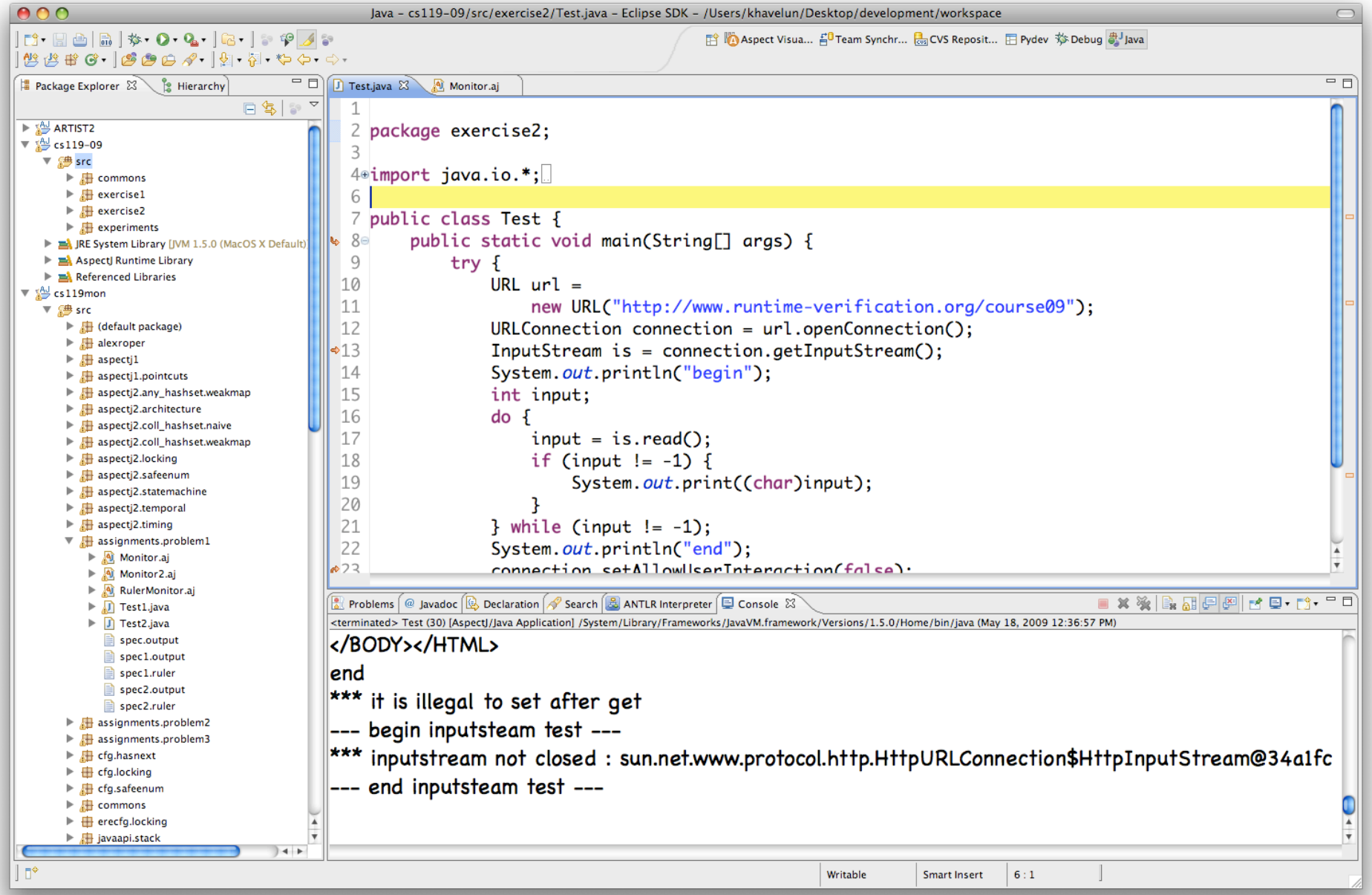
- see note 5 for precise details.

- using JavaMOP, write two parameterized specifications, using either regular expressions (JavaERE) or state machines, that check the policy.

The aspectJ Solution


Java - cs119-09/src/exercise2/Monitor.aj - Eclipse SDK - /Users/khavelun/Desktop/development/workspace
Package Explorer
Hierarchy
Test.java
Monitor.aj
ARTIST2
cs119-09
src
commons
exercise1
exercise2
experiments
JRE System Library [JVM 1.5.0 (MacOS X Default)]
AspectJ Runtime Library
Referenced Libraries
cs119mon
src
(default package)
alexroper
aspectj1
aspectj1.pointcuts
aspectj2.any_hashset.weakmap
aspectj2.architecture
aspectj2.coll_hashset.naive
aspectj2.coll_hashset.weakmap
aspectj2.locking
aspectj2.safeenum
aspectj2.statemachine
aspectj2.temporal
aspectj2.timing
assignments.problem1
assignments.problem2
assignments.problem3
cfg.hasnext
cfg.locking
cfg.safeenum
commons
erecfg.locking
javaapi.stack
javaapi.synccollection
javaapi.urlconnection
ltl.hasnext
ltl.safeenum
paramere.closedreader
paramere.generaltest
paramere.hashset
paramere.hasnext
paramere.resource
paramere.safeenum
project.hendrickson.problem1
package exercise2;
import java.io.*;
import java.net.*;
import java.util.*;
import commons.*;
public aspect Monitor {
pointcut scope() : if(true);
/* property 1 :
* it is not allowed to call a set-method on a URLConnection
* object after a get-method has been called.
*/
WeakIdentityHashSet connections = new WeakIdentityHashSet();
pointcut getparam(URLConnection c) :
target(c) && call(* URLConnection.get*(..));
pointcut setparam(URLConnection c) :
target(c) && call(* URLConnection.set*(..));
before(URLConnection c) : getparam(c) && scope() {
connections.add(c);
}
before(URLConnection c) : setparam(c) && scope() {
if (connections.contains(c)) {
System.out.println("*** it is illegal to set after get");
}
}
Problems
Javadoc
Declaration
Search
ANTLR Interpreter
Console
Writable
Smart Insert
17 : 29


Java - cs119-09/src/exercise2/Monitor.aj - Eclipse SDK - /Users/khavelun/Desktop/development/workspace
Package Explorer
Hierarchy
Test.java
Monitor.aj
ARTIST2
cs119-09
src
commons
exercise1
exercise2
experiments
JRE System Library [JVM 1.5.0 (MacOS X Default)]
AspectJ Runtime Library
Referenced Libraries
cs119mon
src
(default package)
alexroper
aspectj1
aspectj1.pointcuts
aspectj2.any_hashset.weakmap
aspectj2.architecture
aspectj2.coll_hashset.naive
aspectj2.coll_hashset.weakmap
aspectj2.locking
aspectj2.safeenum
aspectj2.statemachine
aspectj2.temporal
aspectj2.timing
assignments.problem1
assignments.problem2
assignments.problem3
cfg.hasnext
cfg.locking
cfg.safeenum
commons
erecfg.locking
javaapi.stack
javaapi.synccollection
javaapi.urlconnection
ltl.hasnext
ltl.safeenum
paramere.closedreader
paramere.generaltest
paramere.hashset
paramere.hasnext
paramere.resource
paramere.safeenum
project.hendrickson.problem1
project.hendrickson.problem2
project.hendrickson.problem3
project.matuschak.problem1
project.matuschak.problem2
project.matuschak.problem3
34
35 /* property 2 :
36 * an input stream that has been obtained from a URLConnection
37 * object using the getInputStream method should be closed
38 * before the main program terminates.
39 */
40
41 IdentityHashSet<InputStream> inputstreams = new IdentityHashSet<InputStream>();
42
43 pointcut getInputStream(URLConnection c) :
44 target(c) &&
45 call(InputStream URLConnection.getInputStream());
46
47 pointcut closeInputStream(InputStream i) :
48 target(i) && call(void InputStream.close());
49
50 after(URLConnection c) returning (InputStream i) :
51 getInputStream(c) && scope()
52 {
53 inputstreams.add(i);
54 System.out.println(">>> inputstream opened : " + i);
55 }
56
57 after(InputStream i) : closeInputStream(i) && scope() {
58 inputstreams.remove(i);
59 System.out.println(">>> application closes inputstream " + i);
60 }
61
62 after() : execution(static void Test.main(..)) && scope() {
63 System.out.println("--- begin inputsteam test ---");
64 Iterator it = inputstreams.iterator();
65 while(it.hasNext()) {
66 System.out.println("*** inputstream not closed : " + it.next());
67 }
68 System.out.println("--- end inputsteam test ---");
69 }
70 }
Problems
Javadoc
Declaration
Search
ANTLR Interpreter
Console
Writable
Smart Insert
41 : 5


Java - cs119-09/src/exercise2/Test.java - Eclipse SDK - /Users/khavelun/Desktop/development/workspace
Package Explorer
Hierarchy
ARTIST2
cs119-09
src
commons
exercise1
exercise2
experiments
JRE System Library [JVM 1.5.0 (MacOS X Default)]
AspectJ Runtime Library
Referenced Libraries
cs119mon
src
(default package)
alexroper
aspectj1
aspectj1.pointcuts
aspectj2.any_hashset.weakmap
aspectj2.architecture
aspectj2.coll_hashset.naive
aspectj2.coll_hashset.weakmap
aspectj2.locking
aspectj2.safeenum
aspectj2.statemachine
aspectj2.temporal
aspectj2.timing
assignments.problem1
Monitor.aj
Monitor2.aj
RulerMonitor.aj
Test1.java
Test2.java
spec.output
spec1.output
spec1.ruler
spec2.output
spec2.ruler
assignments.problem2
assignments.problem3
cfg.hasnext
cfg.locking
cfg.safeenum
commons
erecfg.locking
javaapi.stack
Test.java
Monitor.aj
1 package exercise2;
3 import java.io.*;
6
7 public class Test {
8 public static void main(String[] args) {
9 try {
10 URL url =
11 new URL("http://www.runtime-verification.org/course09");
12 URLConnection connection = url.openConnection();
13 InputStream is = connection.getInputStream();
14 System.out.println("begin");
15 int input;
16 do {
17 input = is.read();
18 if (input != -1) {
19 System.out.print((char)input);
20 }
21 } while (input != -1);
22 System.out.println("end");
Problems
Javadoc
Declaration
Search
ANTLR Interpreter
Console
<terminated> Test (30) [AspectJ/Java Application] /System/Library/Frameworks/JavaVM.framework/Versions/1.5.0/Home/bin/java (May 18, 2009 12:36:57 PM)
</BODY></HTML>
end
*** it is illegal to set after get
--- begin inputsteam test ---
*** inputstream not closed : sun.net.www.protocol.http.HttpURLConnection$HttpInputStream@34a1fc
--- end inputsteam test ---
Writable
Smart Insert
6 : 1

end