Traceable Algorithms

Prasun Dewan
Department of Computer Science
University of North Carolina at Chapel Hill
dewan@cs.unc.edu
Code available at: https://github.com/pdewan/ColabTeaching

Pre-Requisites

- Model-Interactor Separation

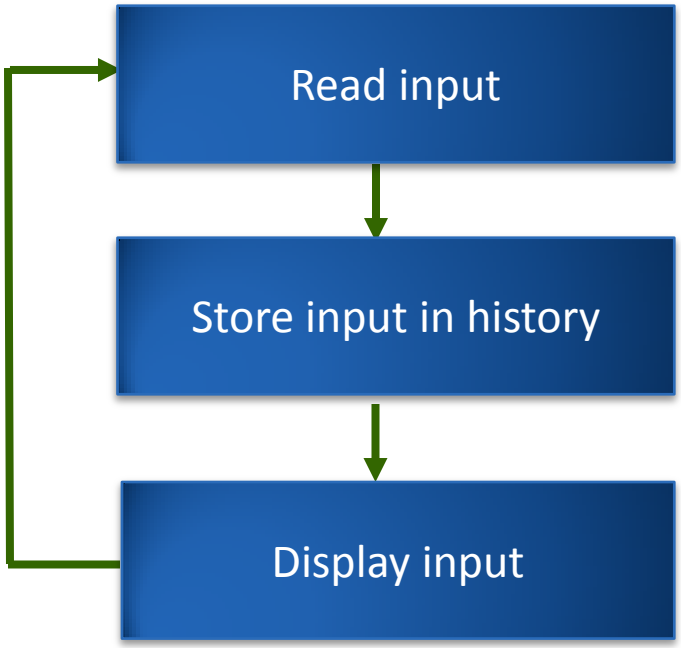
Algorithm vs. Program

- Description of solution to a problem.
- Can be in any "language"
  - graphical
  - natural or programming language
  - natural + programming language (pseudo code)
- Can describe solution to various levels of detail
  - A program is an algorithm
  - An algorithm may not be a program
- Level of detail depends on the task of the reader
  - If debugging or maintaining, then depends on which aspect is faulty or being changed
  - If describing solution depends on what is considered algorithm challenge

Algorithm

```
Please enter an input line or quit or history
The woods are lovely dark and deep
The woods are lovely dark and deep
Please enter an input line or quit or history
But I have promises to keep
But I have promises to keep
Please enter an input line or quit or history
And miles to go before I sleep
And miles to go before I sleep
Please enter an input line or quit or history
history
The woods are lovely dark and deep, But I have promises to keep, And miles to go before I sleep

Please enter an input line or quit or history
quit
Quitting application
```



Algorithm/program separation more useful in monolithic or modular program?

How Useful in Monolithic Program?

```
public class MonolithicEchoer  {
protected static List<String> history = new ArrayList();
public static void main(String[] anArgs) {
  for (;;) {
    System.out.println(PROMPT);
    Scanner scanner = new Scanner(System.in);
    String nextInput = scanner.nextLine();
     if (nextInput.equals(QUIT)) {
         processQuit();
         break;
     } else if (nextInput.equals(HISTORY))
         printHistory();
     else
         processInput(nextInput);
   }
}
```

How Useful in Monolithic Program?

```
protected static void processInput(String anInput) {
  String aFeedback = EchoUtilities.echo(anInput);
  addToHistory(aFeedback);
  displayOutput(aFeedback);
}
protected static void displayOutput(String newValue) {
  System.out.println(newValue);
}
protected  static void addToHistory(String newValue) {
   history.add(history.size(), newValue);
}
}
```

How Useful in More Modular Program?

```
public class ASimpleList<ElementType>
       implements SimpleList<ElementType> {
  List<ElementType> simpleList = new ArrayList();
  List<ListObserver<ElementType>> observers = new ArrayList();
  public void add(ElementType anElement) {
    simpleList.add(simpleList.size(), anElement);
  }
  public void observableAdd(int anIndex, ElementType anElement) {
    add(anIndex, anElement);
    notifyAdd(anIndex, anElement);
  }
  public void notifyAdd(List<ListObserver<ElementType>> observers,
    int index, ElementType newValue) {
    for (ListObserver<ElementType> observer:observers)
      observer.elementAdded(index, newValue);
   }
…
}
```

How Useful in More Modular Program?

```
public class AnEchoInteractor implements EchoerInteractor {
  protected SimpleList<String> history;
  public AnEchoInteractor(SimpleList<String> aHistory) {
    history = aHistory;
  }
  …
  protected void processInput(String anInput) {
    addToHistory(computeFeedback(anInput));
  }
  protected void addToHistory(String newValue) {
     history.observableAdd(newValue);
  }
  public void elementAdded(int anIndex, Object aNewValue) {
     displayOutput(history.get(anIndex));
  }
  protected void displayOutput(String newValue) {
    System.out.println(newValue);
  }
…
```

How Useful in More Modular Program?

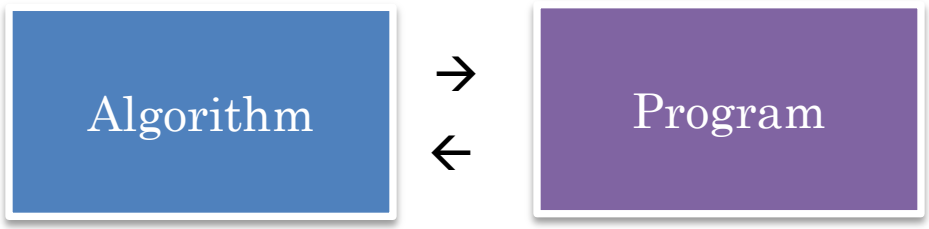
```
public class AnEchoComposerAndLauncher implements
EchoerComposerAndLauncher{
  protected SimpleList<String> history;
  protected EchoerInteractor interactor;
  // factory method
  protected SimpleList<String> createHistory() {
    return new ASimpleList();
  }
  // factory method
  protected EchoerInteractor createInteractor() {
    return new AnEchoInteractor(history);
  }
  protected void connectModelInteractor() {
    interactor = createInteractor();
    history.addObserver(interactor);
  }
…
```

Modularity scatters algorithm among multiple objects

With observer pattern and interfaces sometimes algorithm not known until runtime

Need for higher-level algorithm more in multi-class programs

Which Comes First?



Top-down, bottom-up, middle-out

Separate?

Algorithm

Program

Can get inconsistent

Embellish the program with the algorithm

In-Line Algorithm

```
protected static void processInput(String anInput) {
  // received input
  String aFeedback = EchoUtilities.echo(anInput);
  addToHistory(aFeedback);
  // added input to history
  displayOutput(aFeedback);
 // displayed the input
}
```

Can extract comments from code to view algorithm

Do not get a linear path from scattered objects

Print Statements

```
protected static void processInput(String anInput) {
   System.out.println("received input");
   EchoUtilities.echo(anInput);
   addToHistory(aFeedback);
   System.out.println("added input to history");
   displayOutput(aFeedback);
   System.out.println("displayed the input");
}
```

Can get a linear path

Cannot disable them easily

Cannot separate them from other output

Tracing with Debugger

Debugger makes it difficult to test race conditions

Cannot see the history of actions

Break points do not transfer to another computer

No static documentation

Logging Frameworks

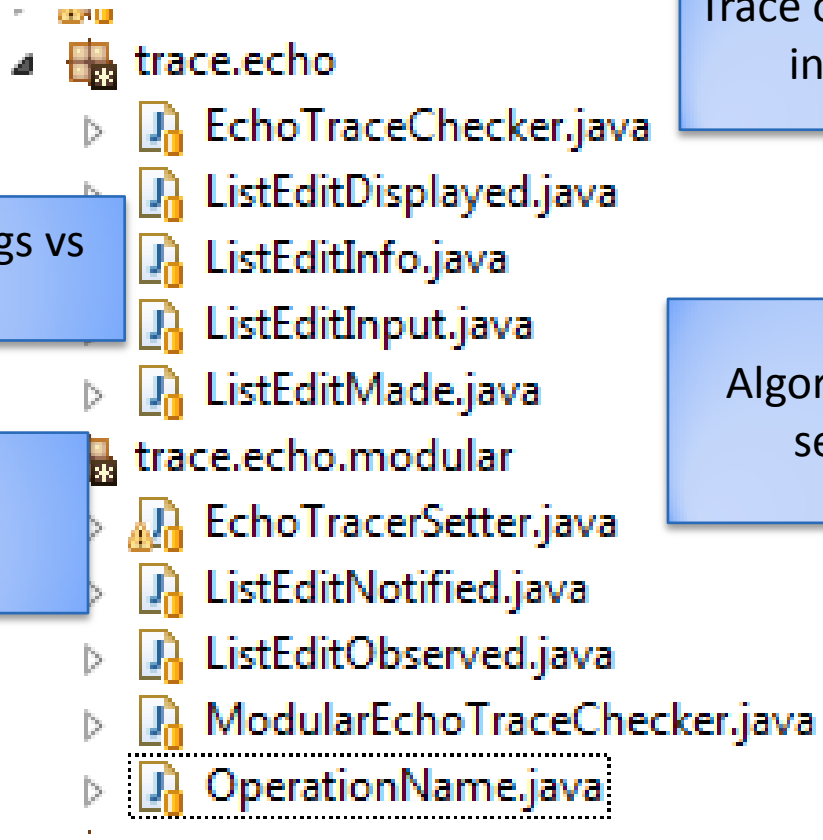
Log rather than print traces

Can display selected portions of the log

Can separate log output from the rest

Will describe log framework developed before Java's

Special Class for Each Algorithm Step/Event



Trace objects and source code in separate packages

Info vs. Warnings vs Error

Algorithm steps can be in separate packages

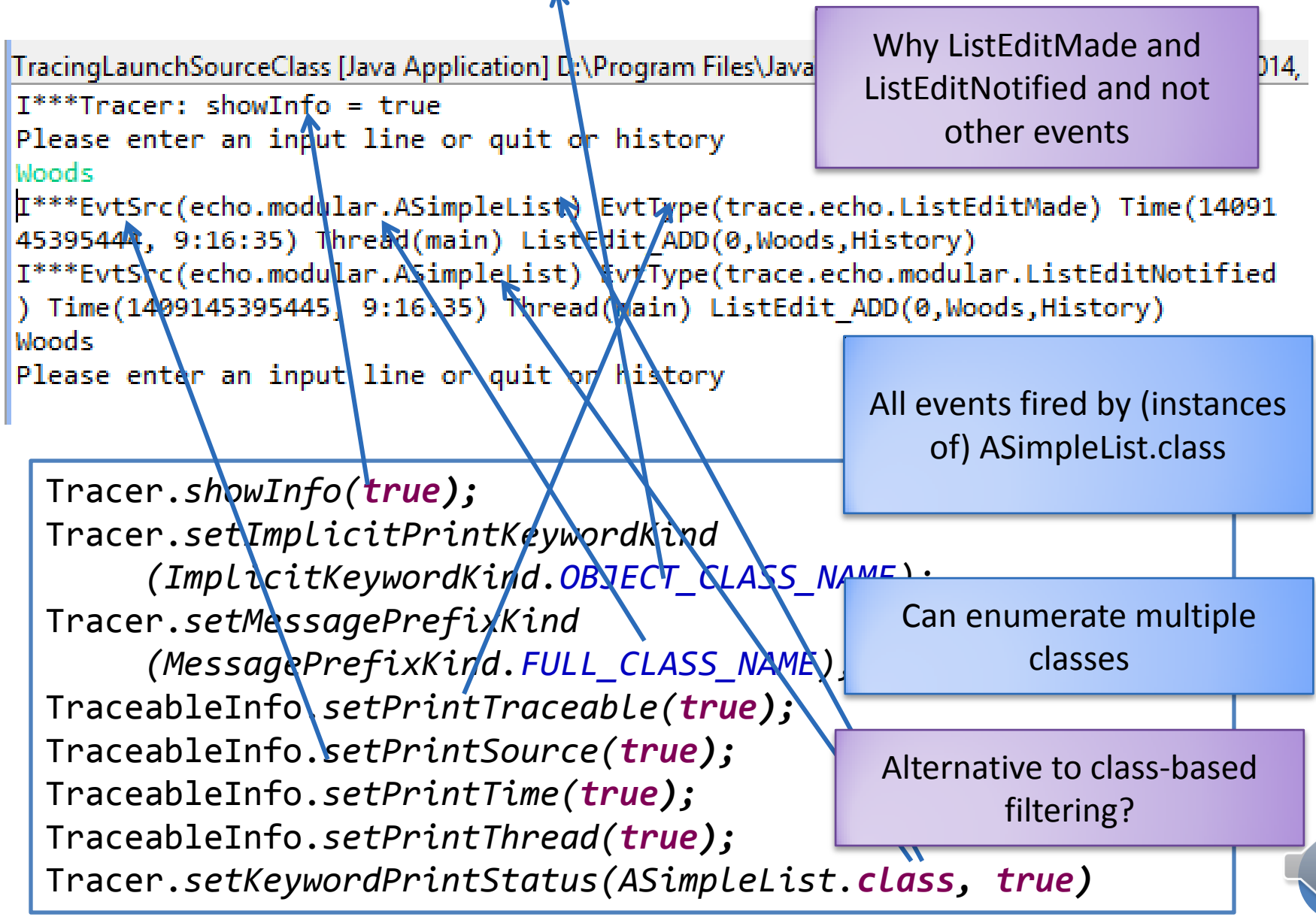
Settings for checkers

Algorithm steps associated with checkers based on event and source

Concrete events vs. Abstract Classes

Each trace event object has a source or announcer

Source Class Filtering



Separate? (Review)

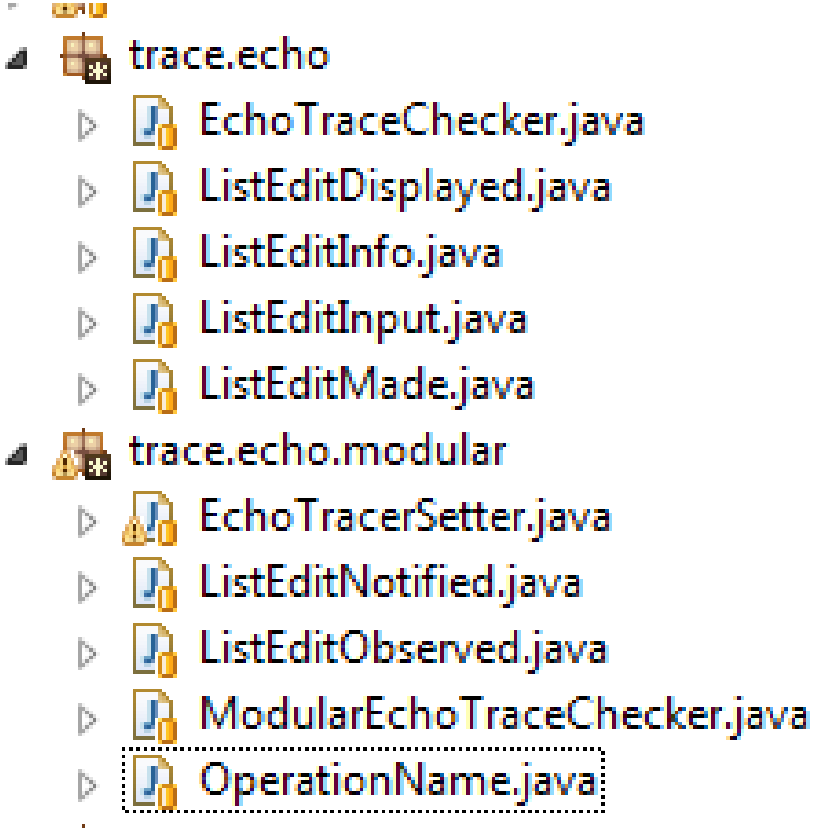
Algorithm

Program

Can get inconsistent

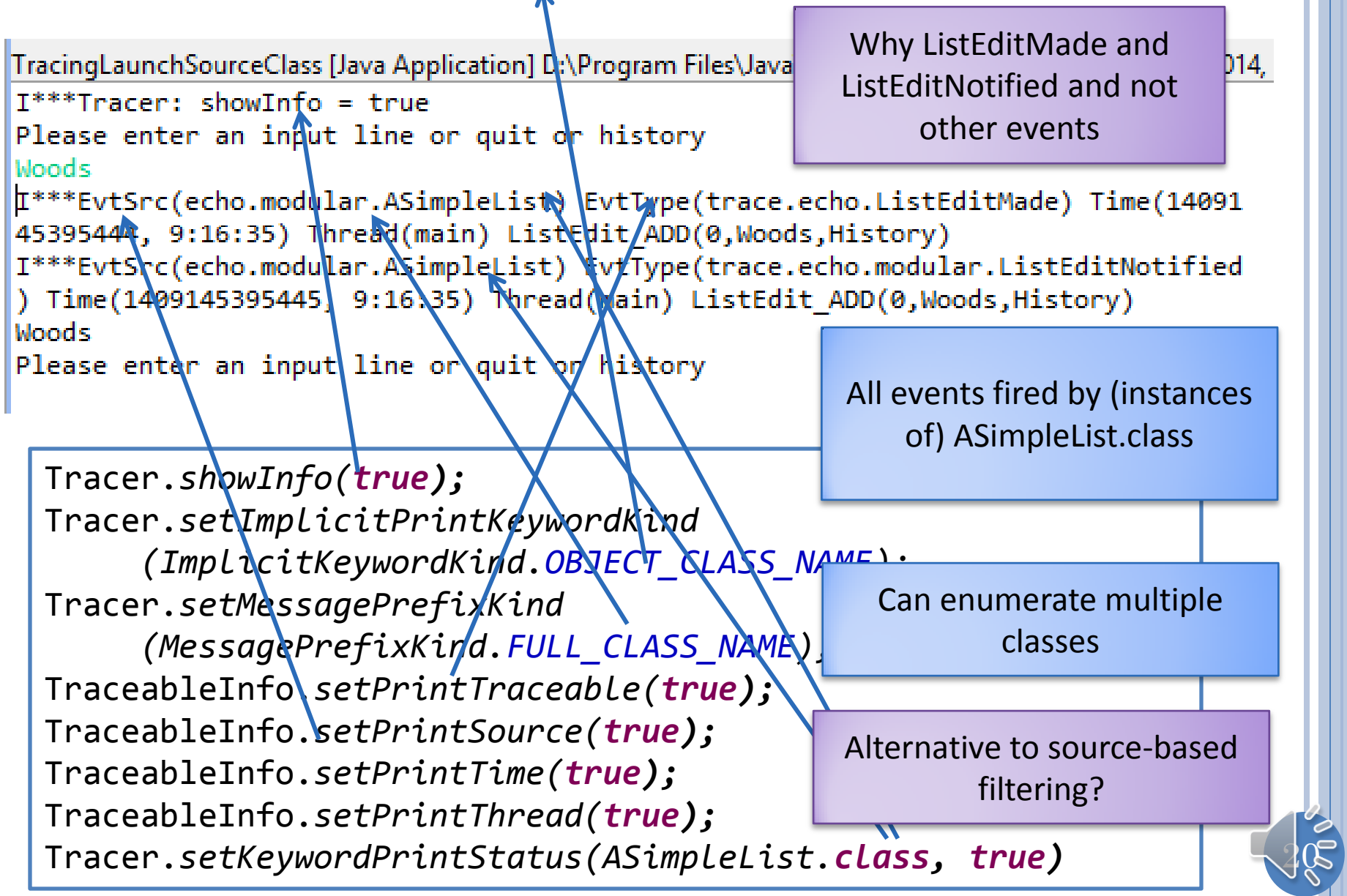
Embellish the program with the algorithm

Special Class for Each Algorithm Step/Event



Each trace event object has a source or announcer

Source Class Filtering



Event Class Filtering

```
TracingLaunchEventClass [Java Application] D:\Program Files\Java\jdk1.7.0_51\bin\javaw.exe (Sep 3, 2014, 9:
I***Tracer: showInfo = true
Please enter an input line or quit or history
Woods
I***EvtType(trace.echo.ListEditMade) EvtSrc(echo.modular.ASimpleList) Time(14097
52299367, 9:51:39) Thread(main) ListEdit_ADD(0,Woods,History)
I***EvtType(trace.echo.modular.ListEditObserved) EvtSrc(echo.modular.AnEchoInter
actor) Time(1409752299368, 9:51:39) Thread(main) ListEdit_ADD(0,Woods,History)
Woods
Please enter an input line or quit or history
```

```
Tracer.showInfo(true);
Tracer.setImplicitPrintKeywordKind
     (ImplicitKeywordKind.OBJECT_CLASS_NAM
Tracer.setMessagePrefixKind
     (MessagePrefixKind.FULL_CLASS_NAME);
TraceableInfo.setPrintSource(true);
TraceableInfo.setPrintTime(true);
TraceableInfo.setPrintThread(true);
Tracer.setKeywordPrintStatus(ListEditMade
Tracer.setKeywordPrintStatus(ListEditObserved.class, true);
```

All events of type ListEditMade or ListEditObserved

Can be announced by different sources

Alternative (source/event) class-based filtering?

Package-based Filtering

```
TracingLaunchEventPackage [Java Application] D:\Program Files\Java\jdk1.7.0_51\bin\javaw.exe (Sep 3, 2014, 9:57:32 AM)
I***Tracer: showInfo = true
Please enter an input line or quit or history
Woods
I***EvtType(trace.echo.ListEditInput) EvtSrc(echo.modular.AnEchoInteractor) Time
(1409752669097, 9:57:49) Thread(main) ListEdit_ADD(0,Woods,History)
I***EvtType(trace.echo.ListEditMade) EvtSrc(echo.modular.ASimpleList) Time(14097
52669194, 9:57:49) Thread(main) ListEdit_ADD(0,Woods,History)
Woods
I***EvtType(trace.echo.ListEditDisplayed) EvtSrc(echo.modular.AnEchoInteractor)
Time(1409752669197, 9:57:49) Thread(main) ListEdit_ADD(0,Woods,History)
                              quit or history
```

trace.echo
- EchoTraceChecker.java
- ListEditDisplayed.java
- ListEditInfo.java
- ListEditInput.java
- ListEditMade.java

trace.echo.modular
- EchoTracerSetter.java
- ListEditNotified.java
- ListEditObserved.java
- ModularEchoTraceChec
- OperationName.java

```
;
ntKeywordKind
Kind.OBJECT_PACKAGE_NAME);
ixKind
nd.FULL_CLASS_NAME);
tSource(true);
tTime(true);
tThread(true);
tStatus(ListEditMade.class, true);
```

All events of types that are in the package of ListEditMade

Filtering by class and package in other contexts?

Assertions

```
public double getBMI() {
    assert weight > 0 && height > 0:"height and weight
should be >0";
    return weight/(height*height);
}
```

Assertion error is like exception, but it can be disabled

java –ea assignment9.MainClass –da bus.uigen

Can enable/disable assertions for specific classes and packages

Enable assertions for MainClass

Disable assertions for bus.uigen package

Similarity between trace objects and assertions is not a coincidence as both support disablable testing

State vs events

Trace Object vs. Events

```
TracingLaunchEventPackage [Java Application] D:\Program Files\Java\jdk1.7.0_21\bin\javaw.exe (Aug 27, 20
Woods
I***EvtType(trace.echo.ListEditInput) EvtSrc(echo.modular.AnEchoInteractor) Time
(1409156327499, 12:18:47) Thread(main) ListEdit_ADD(0,Woods,History)
I***EvtType(trace.echo.ListEditMade) EvtSrc(echo.modular.ASimpleList) Time(14091
56327742, 12:18:47) Thread(main) ListEdit_ADD(0,Woods,History)
I***EvtType(trace.echo.modular.ListEditNotified) EvtSrc(echo.modular.ASimpleList
) Time(1409156327745, 12:18:47) Thread(main) ListEdit_ADD(0,Woods,History)
I***EvtType(trace.echo.modular.ListEditObserved) EvtSrc(echo.modular.AnEchoInter
actor) Time(1409156327747, 12:18:47) Thread(main) ListEdit_ADD(0,Woods,History)
Woods
I***EvtType(trace.echo.ListEditDisplayed) EvtSrc(echo.modular.AnEchoInteractor)
Time(1409156327749, 12:18:47) Thread(main) ListEdit_ADD(0,Woods,History)
Please enter an input line or quit or history
```

a la event type, class of object

a la event firing, source object

computed automatically

a la event parameters

Announcing a trace object is “asserting” an algorithm event

Example Traceable Event Class

Message to be printed

```
public class ListEditInput extends ListEditInfo{
  public ListEditInput(String aMessage,
        OperationName anOperationName, int anIndex,
        Object anElement, String aList, Object aFinder) {
    super(aMessage, anOperationName, anIndex,
        anElement, aList, aFinder);
 public static ListEditInput newCase(
        OperationName anOperationName, int anIndex,
        Object anElement, String aList, Object aFinder) {
        String aMessage = toString(anOperationName, anIndex,
           anElement, aList);
   ListEditInput retVal = new ListEditInput(aMessage,
                          , anIndex, anElement, aList, aFinder);
```

Step-specific parameters

Network wide list name

Finder: source object that created that step

Thread and time automatically computed

Static method to construct a message from the other args and to log the object

Traceable Event Creation

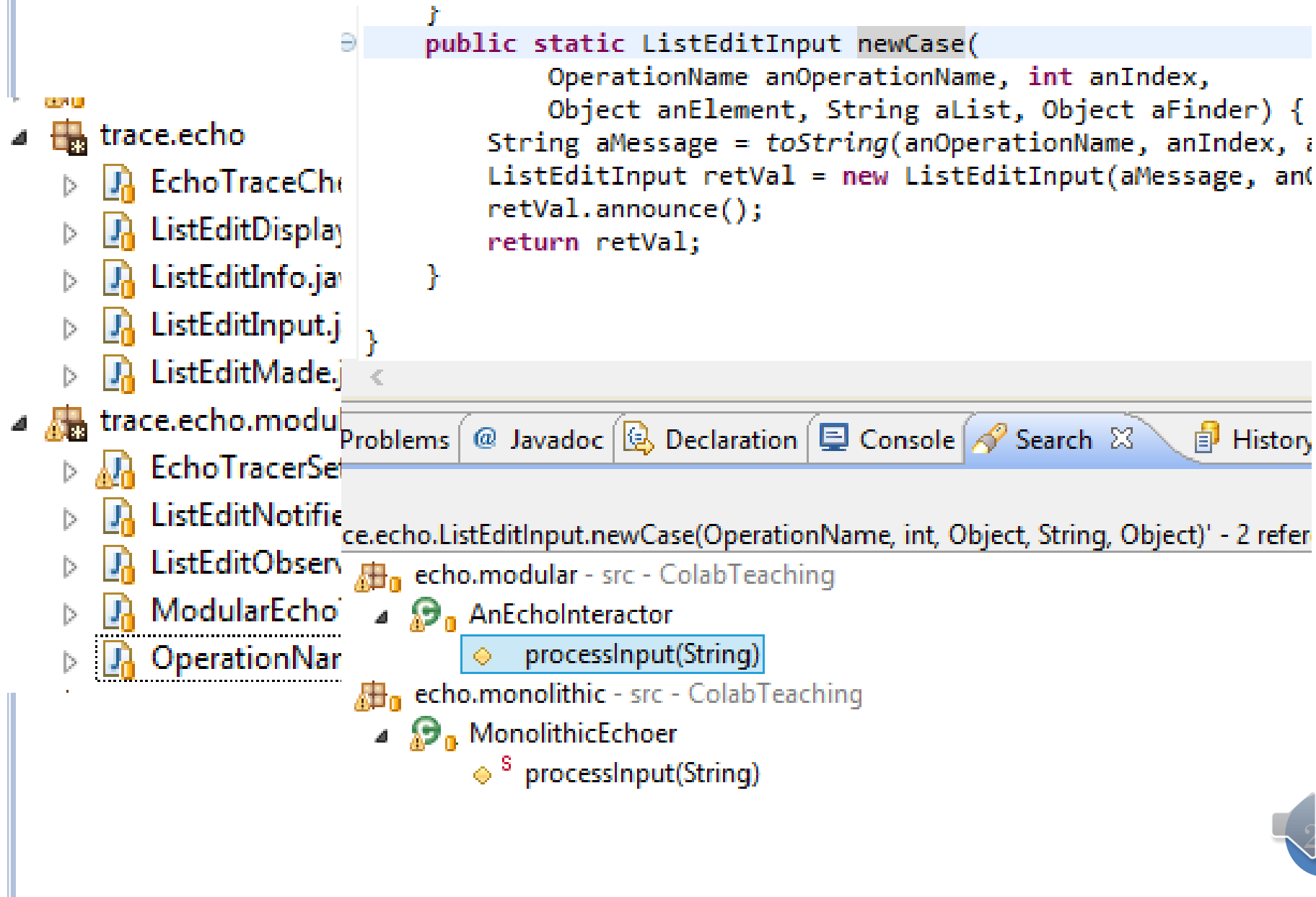
```
protected static void processInput(String anInput) {
  ListEditInput.newCase(OperationName.ADD, history.size(),
      anInput, ApplicationTags.HISTORY,
      MonolithicEchoer.class);
  String aFeedback = EchoUtilities.echo(anInput);
  addToHistory(aFeedback);
  ListEditMade.newCase(OperationName.ADD, history.size(),
      anInput, ApplicationTags.HISTORY,
      MonolithicEchoer.class);
  displayOutput(aFeedback);
  ListEditDisplayed.newCase(OperationName.ADD,
      history.size(), anInput, ApplicationTags.HISTORY,
      MonolithicEchoer.class);
}
```

Source of static (instance) methods is class (instance)

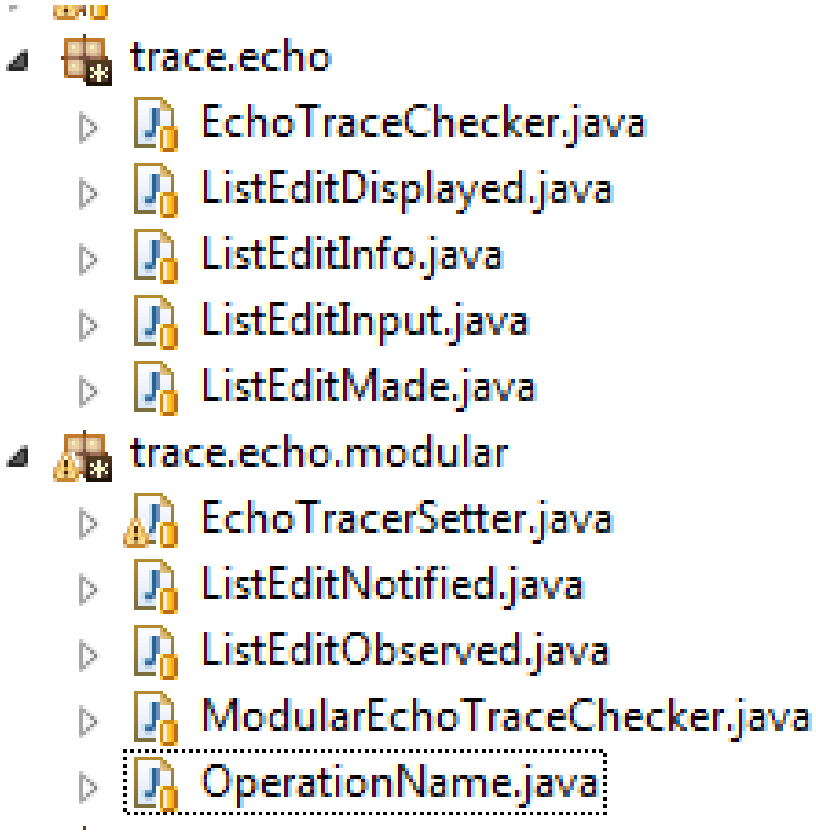
Can go from code to typed algorithm steps

Algorithm step to code?

Find all uses of a Method



Formal Algorithm Description and Testing?



LauncherOfMonolithicEchoTester

LauncherOfModularEchoTester

How is testing done today?

I/O Diff-Based Testing

```
Please enter an input line or quit or history
The woods are lovely dark and deep
The woods are lovely dark and deep
Please enter an input line or quit or history
But I have promises to keep
And miles to go before I sleep
history
The woods are lovely dark and deep, But I have promises to keep, And miles to go before I sleep

Please enter an input line or quit or history
```

Compare “correct” transcript with test transcript

No creativity allowed in implementation defined I/O such as debugging statements and prompts

Cannot distinguish between algorithms that have the same I/O behavior – e.g. monolithic and modular echo

Thread scheduling and other non determinism can effect the trace of a correct algorithm

Trace Diffs

```
TracingLaunchEventPackage [Java Application] D:\Program Files\Java\jdk1.7.0_21\bin\javaw.exe (Aug 27, 20
Woods
I***EvtType(trace.echo.ListEditInput) EvtSrc(echo.modular.AnEchoInteractor) Time
(1409156327499, 12:18:47) Thread(main) ListEdit_ADD(0,Woods,History)
I***EvtType(trace.echo.ListEditMade) EvtSrc(echo.modular.ASimpleList) Time(14091
56327742, 12:18:47) Thread(main) ListEdit_ADD(0,Woods,History)
I***EvtType(trace.echo.modular.ListEditNotified) EvtSrc(echo.modular.ASimpleList
) Time(1409156327745, 12:18:47) Thread(main) ListEdit_ADD(0,Woods,History)
I***EvtType(trace.echo.modular.ListEditObserved) EvtSrc(echo.modular.AnEchoInter
actor) Time(1409156327747, 12:18:47) Thread(main) ListEdit_ADD(0,Woods,History)
Woods
I***EvtType(trace.echo.ListEditDisplayed) EvtSrc(echo.modular.AnEchoInteractor)
Time(1409156327749, 12:18:47) Thread(main) ListEdit_ADD(0,Woods,History)
Please enter an input line or quit or history
```

Compare “correct” trace with test trace

Can use filtering to test algorithms at multiple levels of abstractions and different aspects of algorithms

No formal description of algorithm – who checks the correct implementation

Inter-Trace vs Intra-Trace

```
TracingLaunchEventPackage [Java Application] D:\Program Files\Java\jdk1.7.0_21\bin\javaw.exe (Aug 27, 20
Woods
I***EvtType(trace.echo.ListEditInput) EvtSrc(echo.modular.AnEchoInteractor) Time
(1409156327499, 12:18:47) Thread(main) ListEdit_ADD(0,Woods,History)
I***EvtType(trace.echo.ListEditMade) EvtSrc(echo.modular.ASimpleList) Time(14091
56327742, 12:18:47) Thread(main) ListEdit_ADD(0,Woods,History)
I***EvtType(trace.echo.modular.ListEditNotified) EvtSrc(echo.modular.ASimpleList
) Time(1409156327745, 12:18:47) Thread(main) ListEdit_ADD(0,Woods,History)
I***EvtType(trace.echo.modular.ListEditObserved) EvtSrc(echo.modular.AnEchoInter
actor) Time(1409156327747, 12:18:47) Thread(main) ListEdit_ADD(0,Woods,History)
Woods
I***EvtType(trace.echo.ListEditDisplayed) EvtSrc(echo.modular.AnEchoInteractor)
Time(1409156327749, 12:18:47) Thread(main) ListEdit_ADD(0,Woods,History)
Please enter an input line or quit or history
```

Find relationships among steps within a trace

User input should be followed by a certain sequence of events which are different for different algorithms

The arguments of these events should have certain relationships

Other aspects of the trace such as source may have relationships

Monolithic Specification and testing

For each input I

I should be followed by ListEditInput, ListEditMade, and ListEditDisplayed

The operation name, index, element, and list should be the same in the events above

The element should be echo(I)

Modular Specification and Testing

For each input I

I should be followed by ListEditInput, ListEditMade, ListEditNotified, ListEditObserved, and ListEditDisplayed

The operation name, index, element, and list should be the same in the events above

The element should be echo(I)

The source of ListEditNotified and ListEditObserved should be different

The source of other objects can also be different as a model/interactor may be divided into multiple submodels/interactors

A program that passes the modular tester will pass the monolithic tester

Demoed OT algorithm was tested using traces

Summary

- Algorithm needed for understanding programs, testing, debugging
- Modularity increases need as steps scattered through many classes
- Important to be able to tie program to algorithm and keep them consistent
- Prints cannot be disabled and easily separated from real output,
- Debugging does not support race conditions and does not provide persistent tracing
- Untyped-event log can be turned off and on and filtered based on keyword, class, package
- Typed events allow algorithm steps to be in separate packages, filtering by event type, and ways to find implementations of a step
- Inter-trace diffs allow algorithm rather than I/O comparisons
- Intra-trace processing allows specification of algorithms and testing without correct traces.

Not Covered

- Types events can be listened through a message bus
- A message bus connects observers to observables
- Can block events when certain conditions are met
- Less heavweight untyped traces also possible
- Following slides from previous class cover this
  - They duplicate some of the material here

TRACE

```
Please enter message:
The woods are lovely
I***(inputport.datacomm.simplex.buffer)Forwarding message to send trapper:inputpor
I***(inputport.datacomm.simplex)Forwarding sent message java.nio.HeapByteBuffer[po
I***(inputport.datacomm.simplex.buffer.nio)Sending messsage: java.nio.HeapByteBuff
I***(inputport.datacomm.simplex.buffer.nio)ABufferedWrite with id:0 contents:java.
I***(inputport.datacomm.simplex.buffer.nio)Started storing of buffered write with
I***(inputport.da                                                te op for:java.nio.chann
I***(inputport.da                                                g of buffered write with
Please enter mes
I***(inputport.datacomm.simplex.buffer.nio)Selector select unblocks
I***(inputport.da                                                te op for:java.nio.chann
I***(inputport.da                                                select
I***(inputport.da                                                unblocks
I***(inputport.da                                                heljava.nio.channels.Soc
I***(inputport.da                                                ers about write
I***(inputport.da                                                heljava.nio.channels.Soc
I***(inputport.datacomm.simplex)notifying to send listners message:java.nio.HeapBy
Alice-->Echo Server:(0)java.nio.HeapByteBuffer[pos=0 lim=20 cap=20]
I***(inputport.datacomm.simplex.buffer.nio)notified listeners about write
I***(inputport.datacomm.simplex.buffer.nio)Selector registering read as no pending
I***(inputport.datacomm.simplex.buffer.nio)channel not connected or no pending wri
I***(inputport.datacomm.simplex.buffer.nio)Selector calls select
```

Tracer.*showInfo*(**true**);

May not want to to know about nio

Filtering: Selecting which events to print?

ONLY INPUTPORT.DATACOMM.SIMPLEX.BUFFER

```
AServerSimplexBufferInputPortLauncher [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (Oct 17, 2011 4:22:27
I***(inputport.datacomm.simplex.buffer)Retrieved from global state receive trapp
I***(inputport.datacomm.simplex.buffer)Changing connection status and asking dri
I***(inputport.datacomm.simplex.buffer)Received message:java.nio.HeapByteBuffer[
I***(inputport.datacomm.simplex.buffer)Associating Alice with java.nio.channels.
I***(inputport.datacomm.simplex.buffer)ServerInputPort connected to:java.nio.cha
Echo Server<-->Alice (Opened)
I***(inputport.datacomm.simplex.buffer)Received message:java.nio.HeapByteBuffer[
I***(inputport.datacomm.simplex.buffer)ServerInputPort received message java.nio
```

```
AnAliceSimplexBufferInputPortLauncher [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (Oct 17, 2011 4:22:33
I***(inputport.datacomm.simplex.buffer)Retrieved from global state send trapper:
I***(inputport.datacomm.simplex.buffer)Set my send trapper to:inputport.datacomm
I***(inputport.datacomm.simplex.buffer)Adding send listener:inputport.datacomm.s
I***(inputport.datacomm.simplex.buffer)Asking driver to connect and changing sta
I***(inputport.datacomm.simplex.buffer)Received connected notificaton from drive
I***(inputport.datacomm.simplex.buffer)Sending to server my name: Alice
I***(inputport.datacomm.simplex.buffer)Forwarding message to send trapper:inputp
Alice<-->Echo Server (Opened)
I***(inputport.datacomm.simplex.buffer)Received sent notification from driver
I***(inputport.datacomm.simplex.buffer)Notifying to send listeners message:java.
Alice-->Echo Server:(0)java.nio.HeapByteBuffer[pos=0 lim=5 cap=5]
Please enter message:
The woods are lovely
I***(inputport.datacomm.simplex.buffer)Forwarding message to send trapper:inputp
Please enter message:
I***(inputport.datacomm.simplex.buffer)Received sent notification from driver
I***(inputport.datacomm.simplex.buffer)Notifying to send listeners message:java.
```

Explicit Keywords

```
public static void info(String keyWord, String info);
```

```
Tracer.info("inputport.datacomm.simplex.buffer", "Asking
driver to connect and changing status");
```

```
public static void setKeywordPrintStatus(
    String keyWord,
    Boolean status);
```

```
Tracer.setKeywordPrintStatus("inputport.datacomm.simplex.
buffer", true)
```

```
Tracer.setKeywordPrintStatus(Tracer.ALL_KEYWORDS, false);
```

Have to specify package name each time

Package name can change

Implicit KeyWords

```
public static void info(Object object, String info);
```

```
Tracer.info(this, "Asking driver to connect and changing status");
```

```
public static void setKeywordPrintStatus(Class c, Boolean status);
```

```
Tracer.setKeywordPrintStatus(
            AGenericSimplexBufferClientInputPort.class,
            false);
```

Showing Package Names

AServerSimplexBufferInputPortLauncher [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (Oct 17, 2011 4:22:27

```
I***(inputport.dat                                    tate receive trapp
I***(inputport.dat                                    tus and asking dri
I***(inputport.dat                                    io.HeapByteBuffer[
I***(inputport.datacomm.simplex.buffer)Associating Alice with java.nio.channels.
I***(inputport.dat                                    ed to:java.nio.cha
Echo Server<-->Ali
I***(inputport.dat                                    io.HeapByteBuffer[
I***(inputport.dat                                    d message java.nio
```

Cannot identify classes of the objects printing out messages

Need to not only specify which events to display but what information to display about each event

AnAliceSimplexBufferInputPortLauncher [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (Oct 17, 2011 4:22:33

```
I***(inputport.datacomm.simplex.buffer)Retrieved from global state send trapper:
```

```
Tracer.showInfo(true);
Tracer.setKeywordPrintStatus(Tracer.ALL_KEYWORDS, false);
Tracer.setKeywordPrintStatus(
     AGenericSimplexBufferClientInputPort.class, true);
Tracer.setMessagePrefixKind(MessagePrefixKind.PACKAGE_NAME);
```

```
I***(inputport.datacomm.simplex.buffer)Received sent notification from driver
I***(inputport.datacomm.simplex.buffer)Notifying to send listeners message:java.
Alice-->Echo Server:(0)java.nio.HeapByteBuffer[pos=0 lim=5 cap=5]
Please enter message:
The woods are lovely
I***(inputport.datacomm.simplex.buffer)Forwarding message to send trapper:inputp
Please enter message:
I***(inputport.datacomm.simplex.buffer)Received sent notification from driver
I***(inputport.datacomm.simplex.buffer)Notifying to send listeners message:java.
```

Showing Short Class Names

```
AServerSimplexBufferInputPortLauncher [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (Oct 17, 2011 12:52:32 PM)
I***(AGenericSimplexBufferServerInputPort)Retrieved from global state receive trapper:inputport.da
I***(AGenericSimplexBufferServerInputPort)Changing connection status and asking driver to connect
I***(AGenericSimplexBufferServerInputPort)Received message:java.nio.HeapByteBuffer[pos=4 lim=9 cap
I***(AGenericSimplexBufferServerInputPort)Associating Alice with java.nio.channels.SocketChannel[
I***(AGenericSimplexBufferServerInputPort)ServerInputPort connected to:java.nio.channels.SocketCha
Echo Server<-->Alice (Opened)
I***(AGenericSimplexBufferServerInputPort)Received message:java.nio.HeapByteBuffer[pos=4 lim=24 ca
I***(AGenericSimplexBufferServerInputPort)ServerInputPort received message java.nio.HeapByteBuffe
Echo Server<--Alice:The woods are lovely
```

```
Tracer.showInfo(true);
Tracer.setKeywordPrintStatus(Tracer.ALL_KEYWORDS, false);
Tracer.setKeywordPrintStatus(
    AGenericSimplexBufferClientInputPort.class, true);
Tracer.setMessagePrefixKind(MessagePrefixKind.SHORT_CLASS_NAME)
```

```
I***(AGenericSimplexBufferClientInputPort)Received sent notification from driver
I***(ASendRegistrarAndNotifier)Notifying to send listeners message:java.nio.HeapByte
Alice-->Echo Server:(0)java.nio.HeapByteBuffer[pos=0 lim=5 cap=5]
Please enter message:
The woods are lovely
I***(AGenericSimplexBufferClientInputPort)Forwarding message to send trapper:inputpo
Please enter message:
I***(AGenericSimplexBufferClientInputPort)Received sent notification from driver
I***(ASendRegistrarAndNotifier)Notifying to send listeners message:java.nio.HeapByte
Alice-->Echo Server:(1)java.nio.HeapByteBuffer[pos=0 lim=20 cap=20]
```

MessagePrefix

```
public static void setMessagePrefixKind(
                        MessagePrefixKind newValue)
```

```
public enum MessagePrefixKind {
   NONE,
   OBJECT_TO_STRING,
   SHORT_CLASS_NAME,
   FULL_CLASS_NAME,
   PACKAGE_NAME
}
```

Displaying All Classes in Package

```
AServerSimplexBufferInputPortLauncher [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (Oct 17, 2011 12:52:32 PM)
I***(AGenericSimplexBufferServerInputPort)Retrieved from global state receive trapper:inputport.d
I***(AGenericSimplexBufferServerInputPort)Changing connection status and asking driver to connect
I***(AGenericSimplexBufferServerInputPort)Received message:java.nio.HeapByteBuffer[pos=4 lim=9 ca
I***(AGenericSimplexBufferServerInputPort)Associating Alice with java.nio.channels.SocketChannel[
I***(AGenericSimplexBufferServerInputPort)ServerInputPort connected to:java.nio.channels.SocketCh
Echo Server<-->Alice (Opened)
```

```
Tracer.showInfo(true);
Tracer.setKeywordPrintStatus(Tracer.ALL_KEYWORDS, false);
Tracer.setKeywordPrintStatus(
    AGenericSimplexBufferClientInputPort.class, true);
Tracer.setMessagePrefixKind(MessagePrefixKind.SHORT_CLASS_NAME)
```

```
I***(AGenericSimplexBufferClientInputPort)Sending to server my name: Alice
I***(AGenericSimplexBufferClientInputPort)Forwarding message to send trapper:inputpo
Alice<-->Echo Server (Opened)
I***(AGenericSimplexBufferClientInputPort)Received sent notification from driver
I***(ASendRegistrarAndNotifier)Notifying to send listeners message:java.nio.HeapByte
Alice-->Echo Server:(0)java.nio.HeapByteBuffer[pos=0 lim=5 cap=5]
Please enter message:
The woods are lo
I***(AGenericSim                                              d trapper:inputpo
Please enter mes
I***(AGenericSim                                              n from driver
I***(ASendRegist                                              java.nio.HeapByte
Alice-->Echo Ser
```

What if we want to focus on one class?

Narrow down which events

Controlling Implicit Keyword

```
AServerSimplexBufferInputPortLauncher (1) [Java Application] D:\Program Files\Java\jre1.6.0_04\bin\javaw.exe
Echo Server<-->Alice (Opened)
Echo Server<--Alice:The woods are lovely
```

```
I***(AGenericSimplexBufferClientInputPort)Retrieved from global state send trapper:inputport.datacomm.s
I***(AGenericSimplexBufferClientInputPort)Set my send trapper to:inputport.datacomm.simplex.ASendMessag
I***(AGenericSimplexBufferClientInputPort)Asking driver to connect and changing status
```

```
Tracer.showInfo(true);
Tracer.setKeywordPrintStatus(Tracer.ALL_KEYWORDS, false);
Tracer.setImplicitKeywordKind(ImplicitKeywordKind.OBJECT_CLASS_NAME);
Tracer.setKeywordPrintStatus(
    AGenericSimplexBufferClientInputPort.class, true);
Tracer.setMessagePrefixKind(MessagePrefixKind.SHORT_CLASS_NAME);
```

Implicit Keyword

```
public static void setImplicitKeywordKind(
        ImplicitKeywordKind newValue)
```

```
public enum ImplicitKeywordKind {
   OBJECT_TO_STRING,
   OBJECT_CLASS_NAME,
   OBJECT_PACKAGE_NAME
}
```

Tracer Static Methods Summary

```
setKeywordPrintStatus(String keyWord Boolean status);
```

```
info(String keyWord, String info);
```

```
setImplicitKeywordKind(ImplicitKeywordKind newValue)
```

```
info(Object obj, String info);
```

```
setKeywordPrintStatus(Class cls, Boolean st
```

```
public enum ImplicitKeywordKind {
   OBJECT_TO_STRING,
   OBJECT_CLASS_NAME,
   OBJECT_PACKAGE_NAME
}
```

```
setMessagePrefixKind(MessagePrefixKind n
```

```
public enum MessagePrefixKind {
   NONE,
   OBJECT_TO_STRING,
   SHORT_CLASS_NAME,
   FULL_CLASS_NAME,
   PACKAGE_NAME
}
```

Debugging Capabilities in Tracer?

- Blocking but separate windows for different processes
- See state at traced actions (with and without blocking)
- Separate state for different threads (with and without blocking)
- Have application-specific code learn about traced calls (perhaps in different processes)

Tracing Strings→Objects

```
setKeywordPrintStatus(String keyWord Boolean status);
```

```
info(String keyWord, String info);
```

```
setImplicitKeywordKind(ImplicitKeywordKind newValue)
```

```
info(Object obj, String info);
```

```
setKeywordPrintStatus(Class cls, Boolean st
```

```
public enum ImplicitKeywordKind {
   OBJECT_TO_STRING,
   OBJECT_CLASS_NAME,
   OBJECT_PACKAGE_NAME
}
```

```
setMessagePrefixKind(MessagePrefixKind n
```

```
public enum MessagePrefixKind {
   NONE,
   OBJECT_TO_STRING,
   SHORT_CLASS_NAME,
   FULL_CLASS_NAME,
   PACKAGE_NAME
}
```

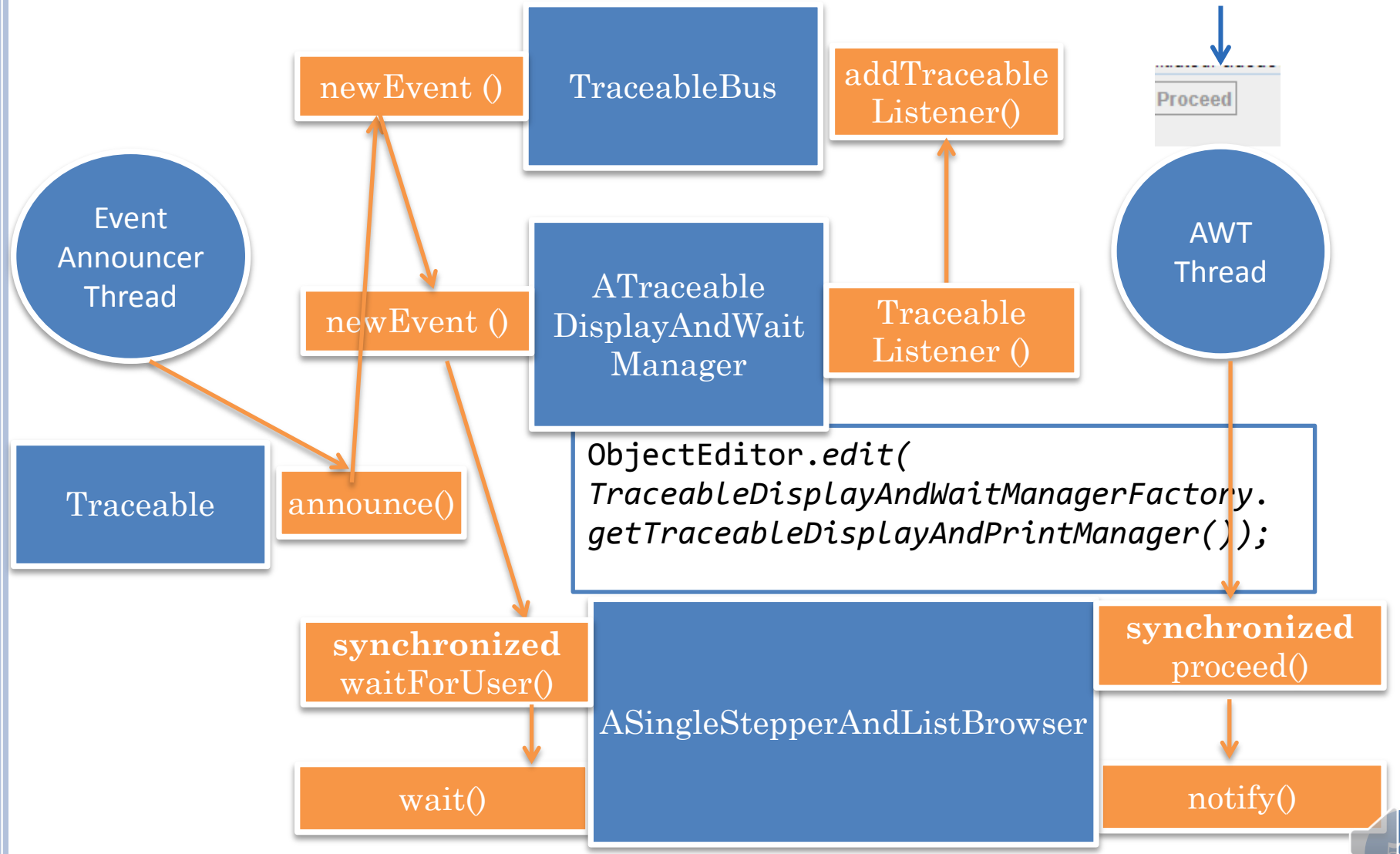
String→Object

```
@DisplayToString(true)
@ComponentWidth(1000)
public class MVCTraceableInfo extends TraceableInfo{
  public MVCTraceableInfo(String aMessage, Object anAnnouncer) {
    super(aMessage, anAnnouncer);
 }
  public static MVCTraceableInfo newInfo(String aMessage, Object aFinder) {
    MVCTraceableInfo retVal = new MVCTraceableInfo(aMessage, aFinder);
    retVal.announce();
    return retVal;
  }
}
```

```
Tracer.info(this, "MVC structure built")
```

```
MVCTraceableInfo( "MVC structure built", this);
```

Tracing Objects



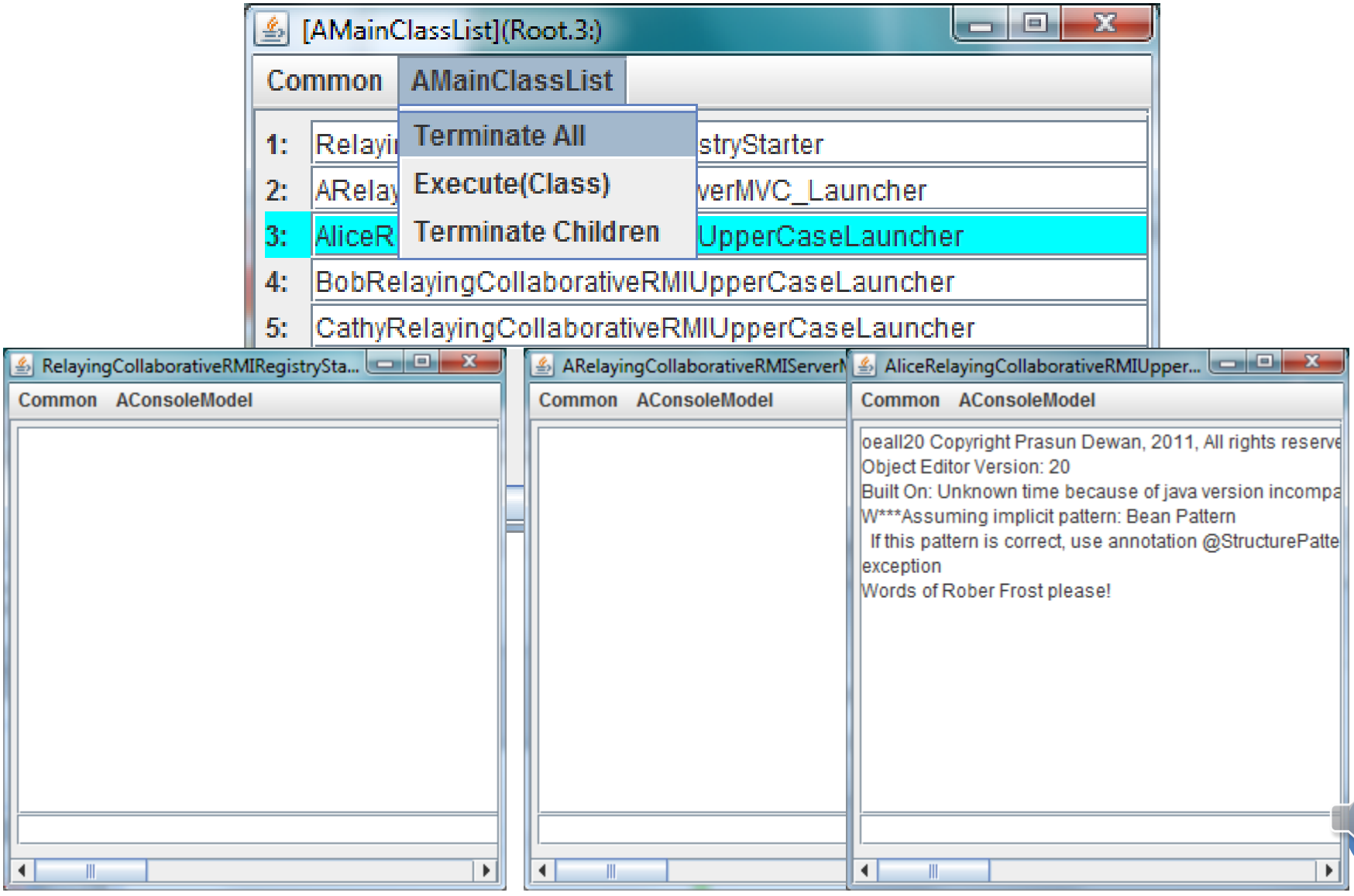
Message Bus

Like an observable it has registration method

Has an announce method

Does not generate events – simply communicates them to observers

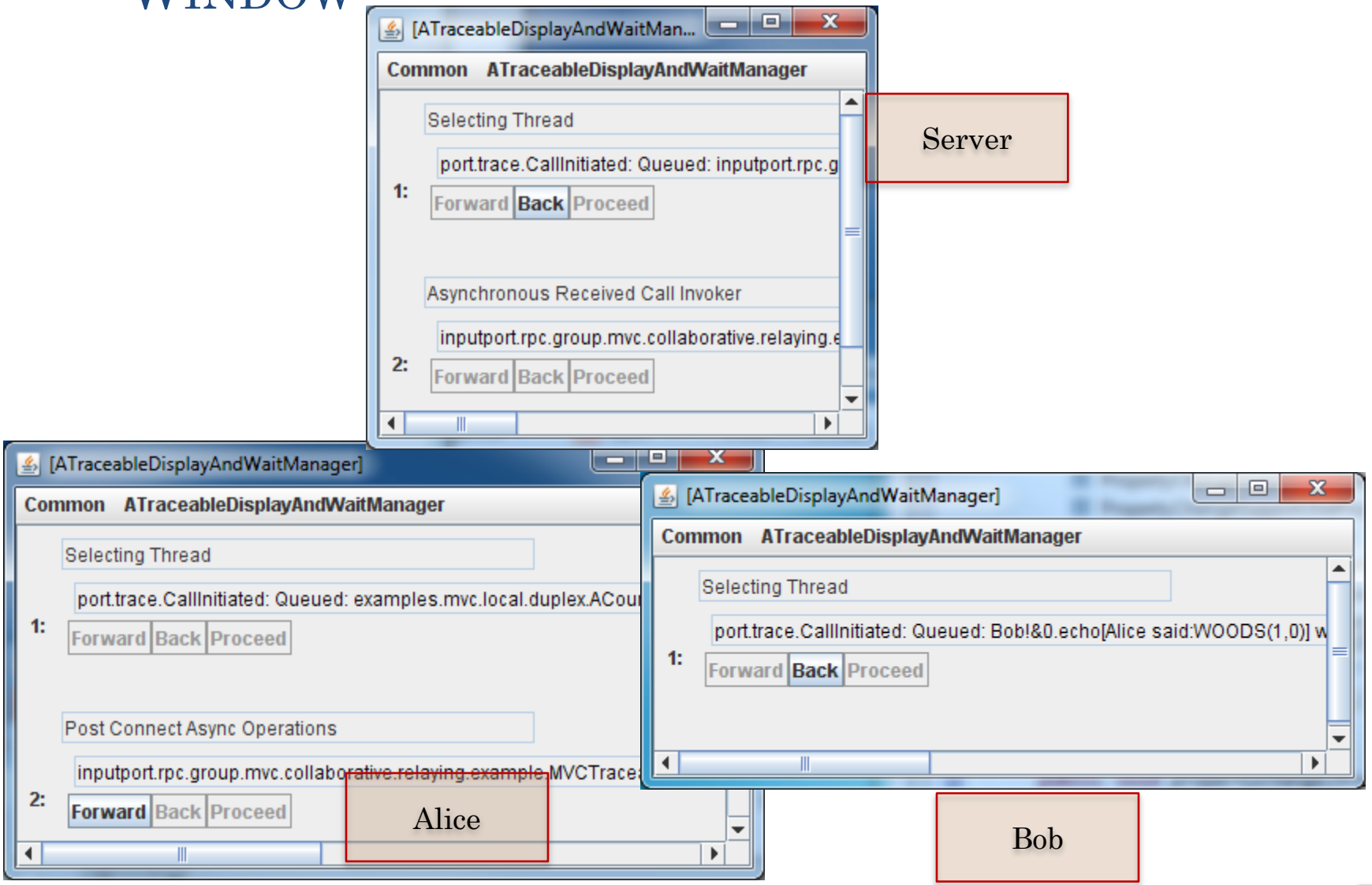
Run Functionality



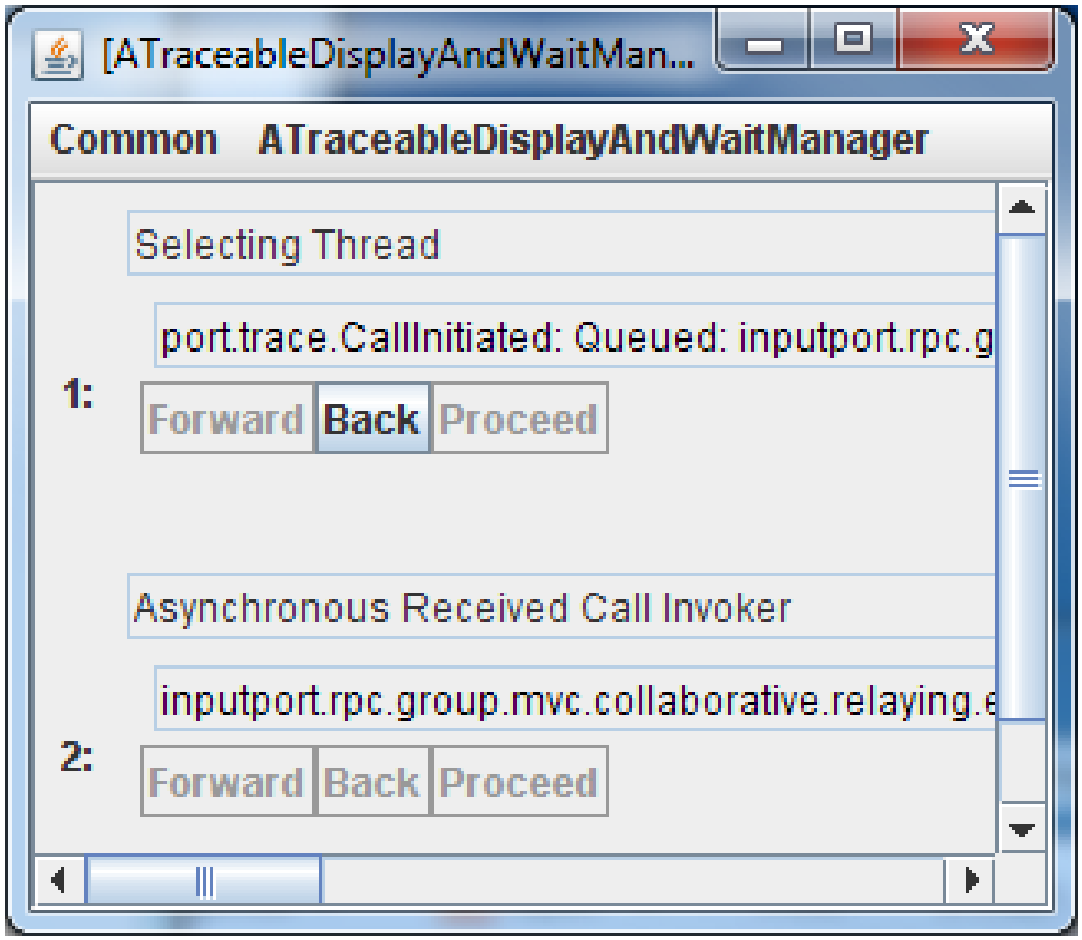
Debug Functionality

```
public ADuplexRPCClientRelayingCollaborativeMVCLauncher(
        String aClientName, String aServerHost,
        String aServerId, String aServerName) {
  super(aClientName, aServerHost, aServerId, aServerName);
  ObjectEditor.edit
      (TraceableDisplayAndWaitManagerFactory.
          getTraceableDisplayAndPrintManager());
  Tracer.setKeywordDisplayStatus(this, true);
}
```

Each Process Can Have Separate Trace Window



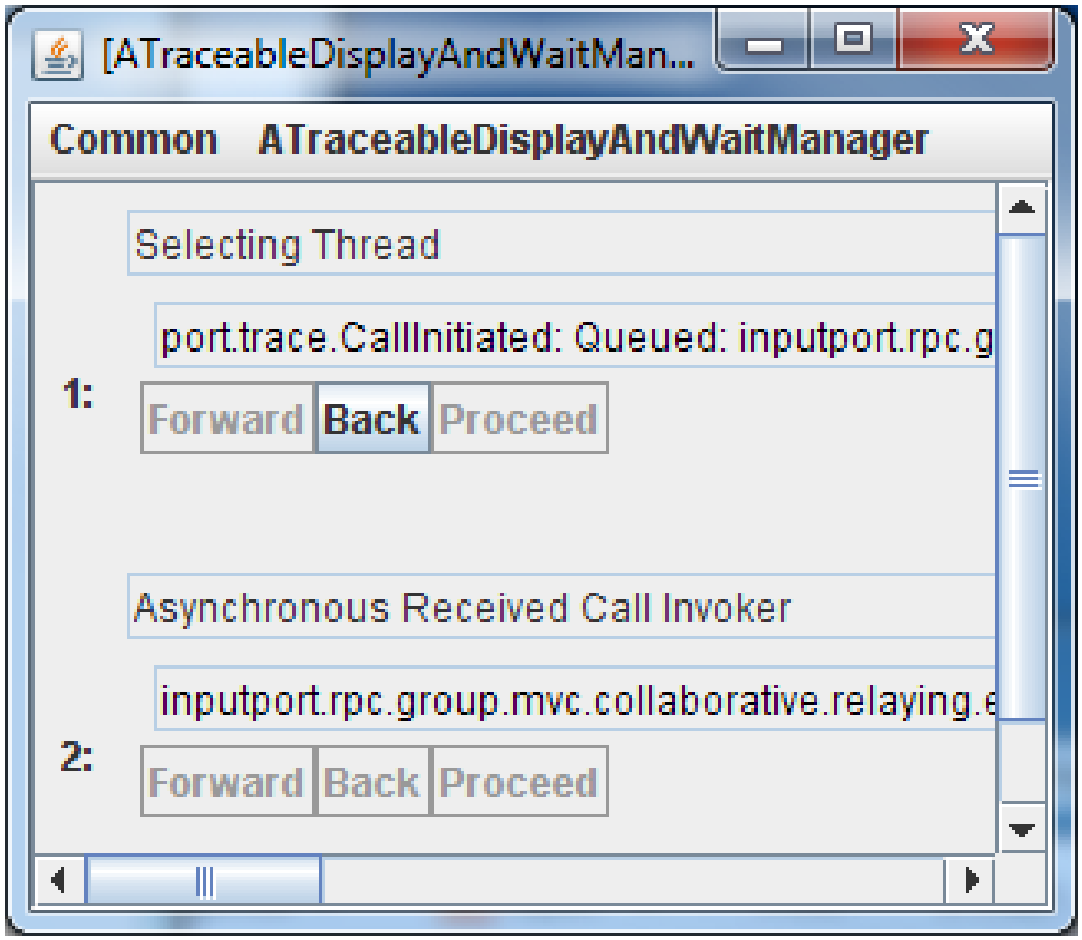
Each Trace Window has Separate Thread Area



Thread interacting with underlying communication channel

Thread invoking remote calls

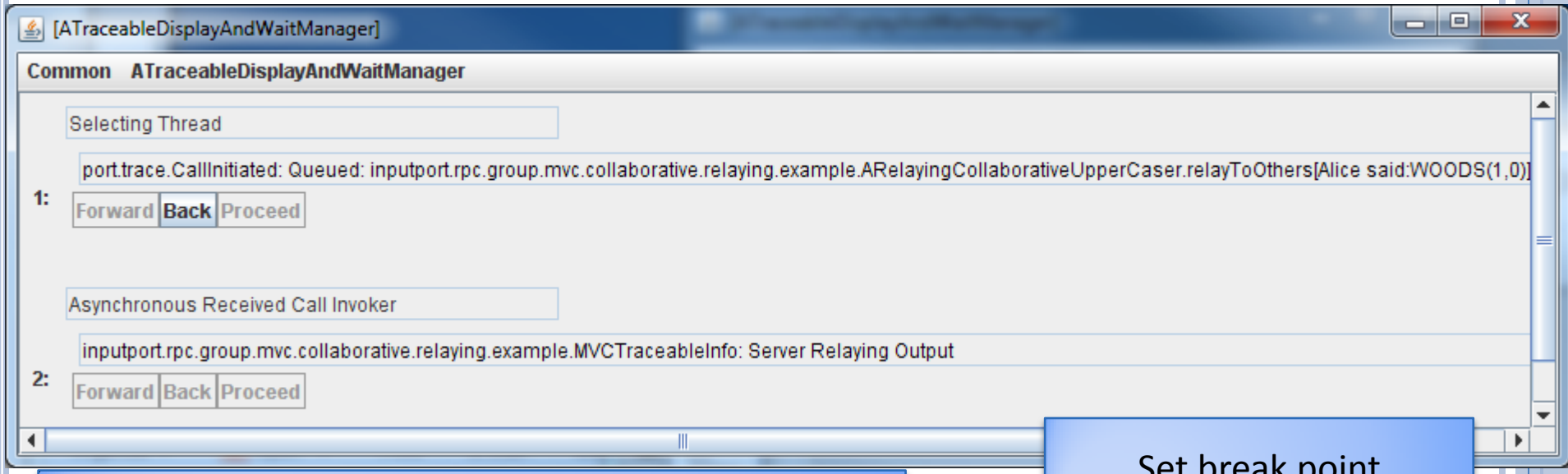
Thread display



Thread name

Other info

Goals



Debugger allows programmer to set breakpoint, resume, and at each breakpoint, pause and see stack trace, variable values, and console message when some line of code is executed

Tracer should provide equivalent

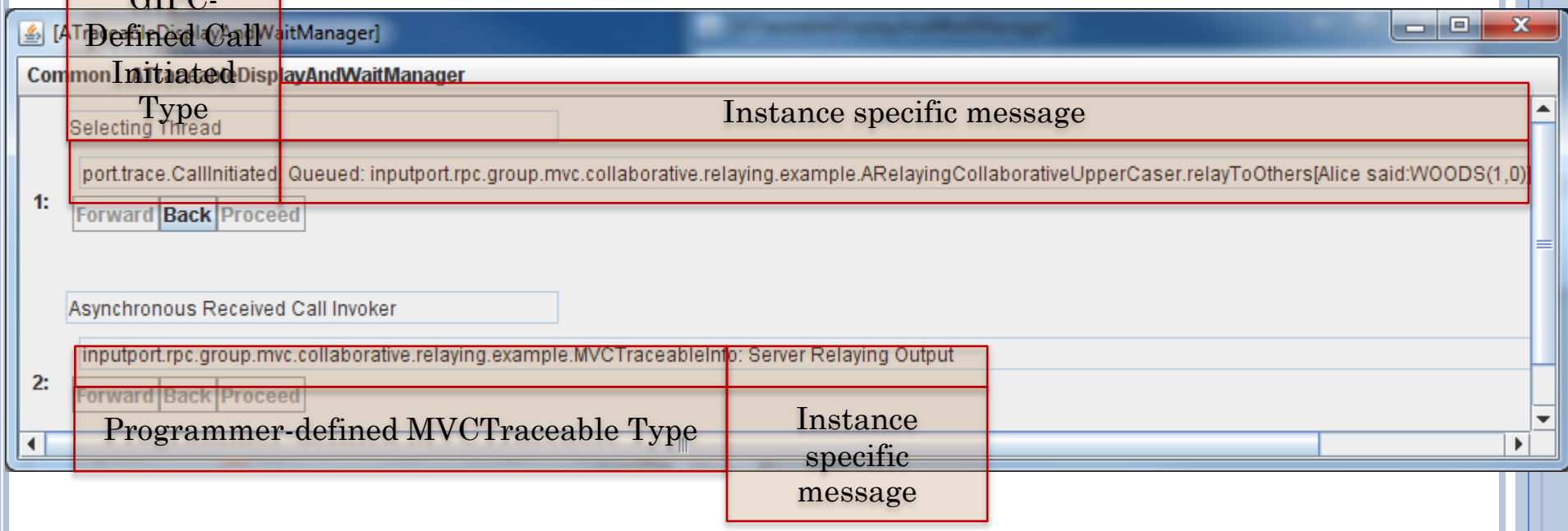
Set break point

BP Resume

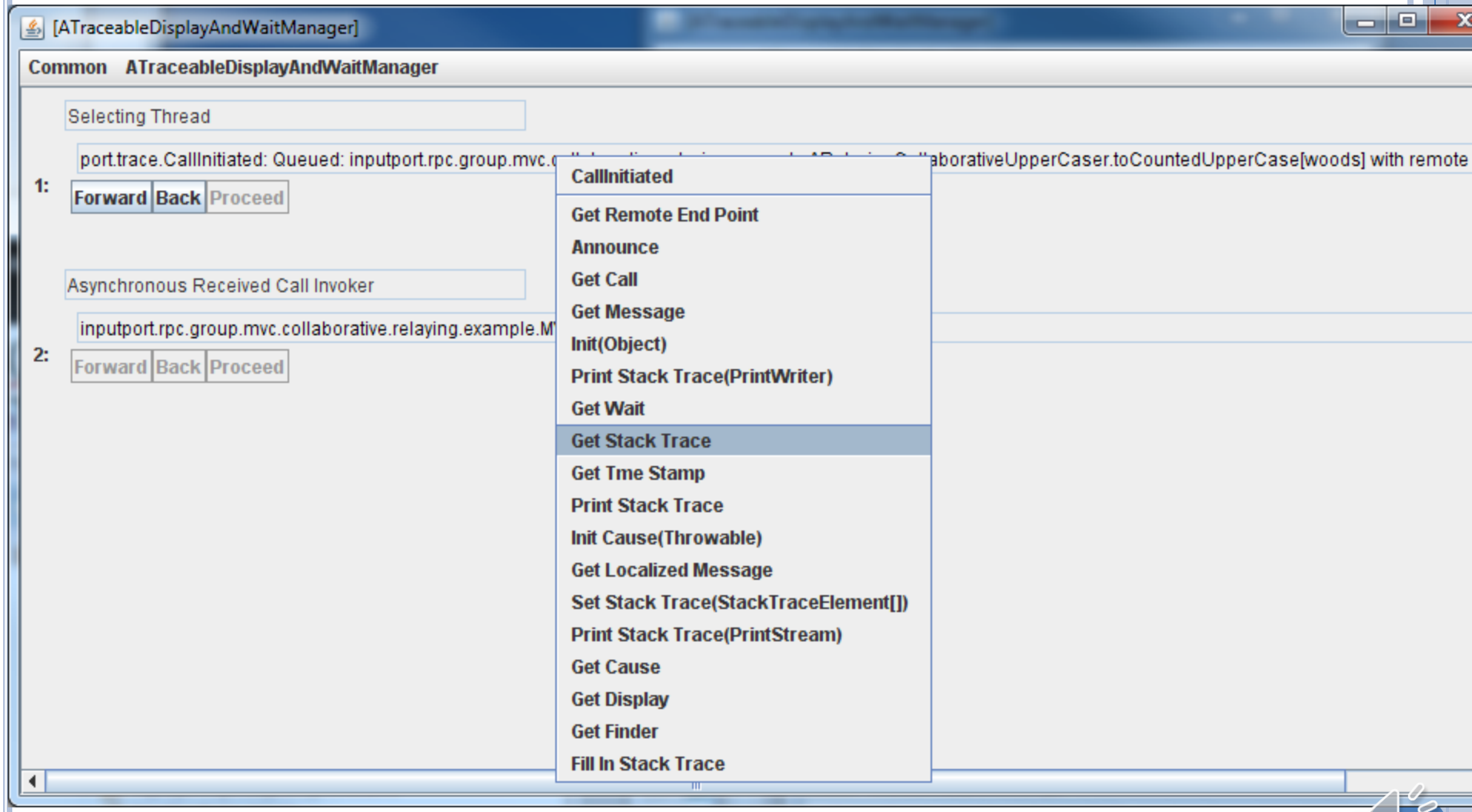
BP Console message

BP Variable Values

Console output Equivalent: Typed Tracing Message

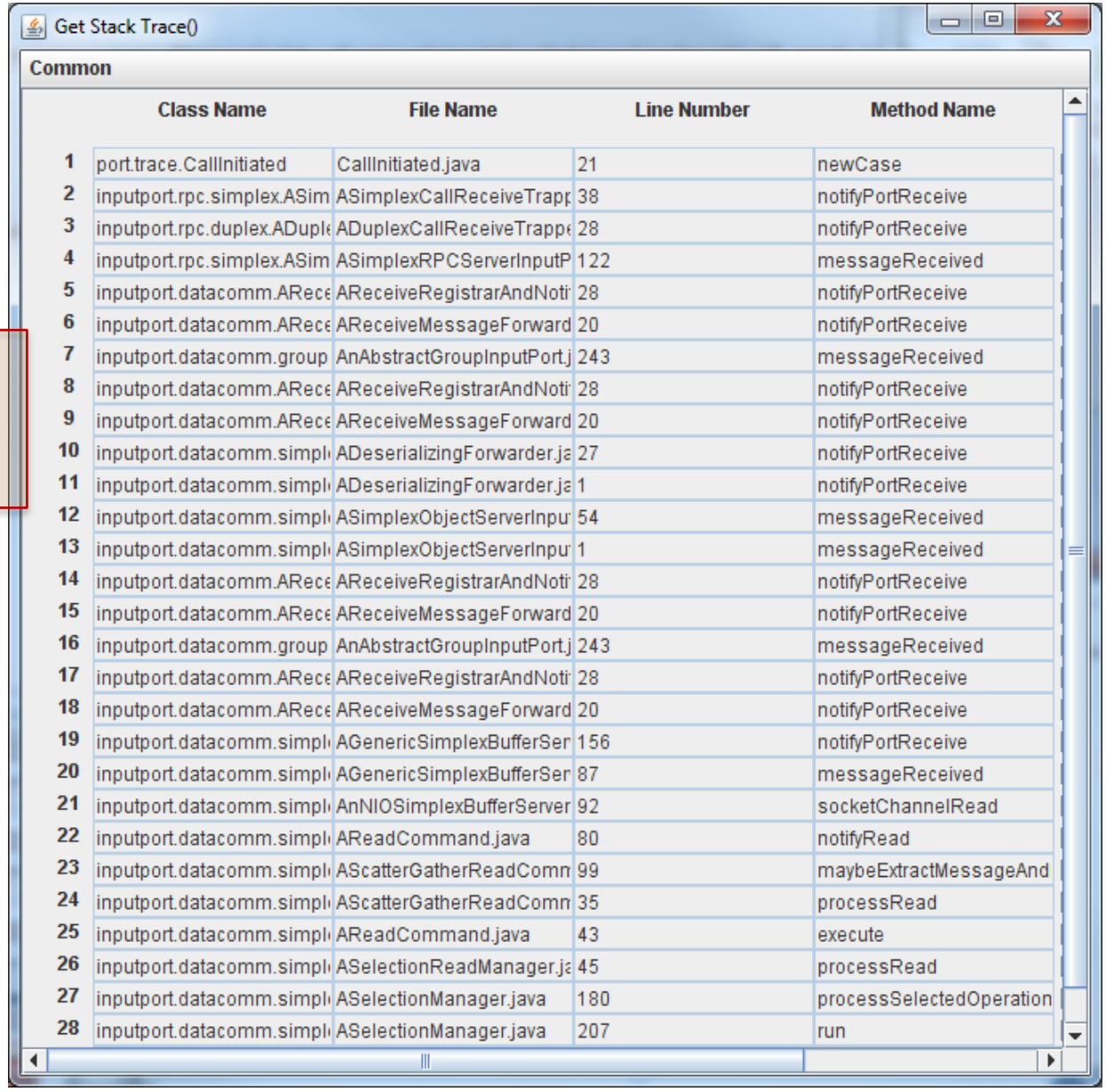


Stack Trace Equivalent: Stack Trace



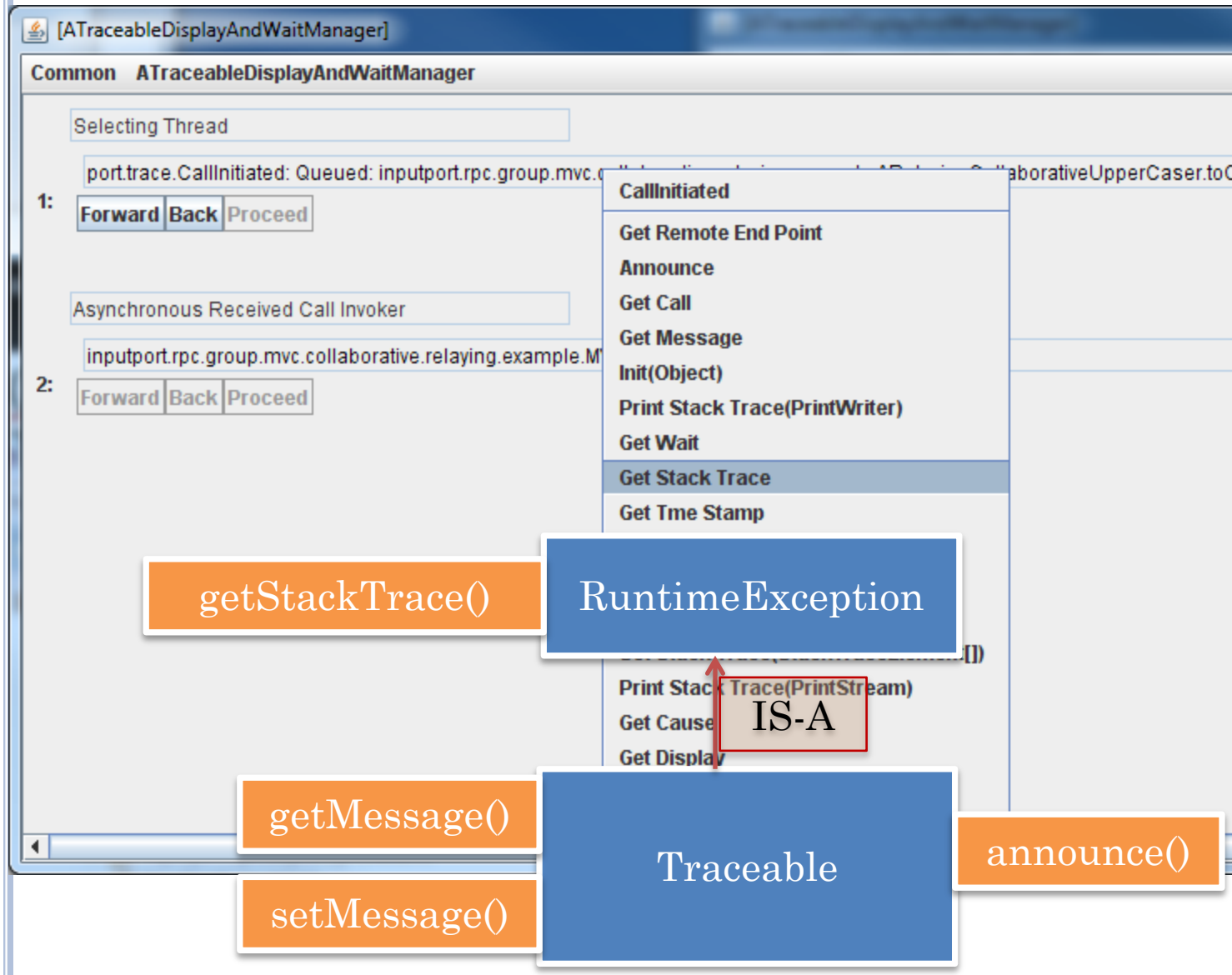
Stack Trace Display

Display of StackTrace Object

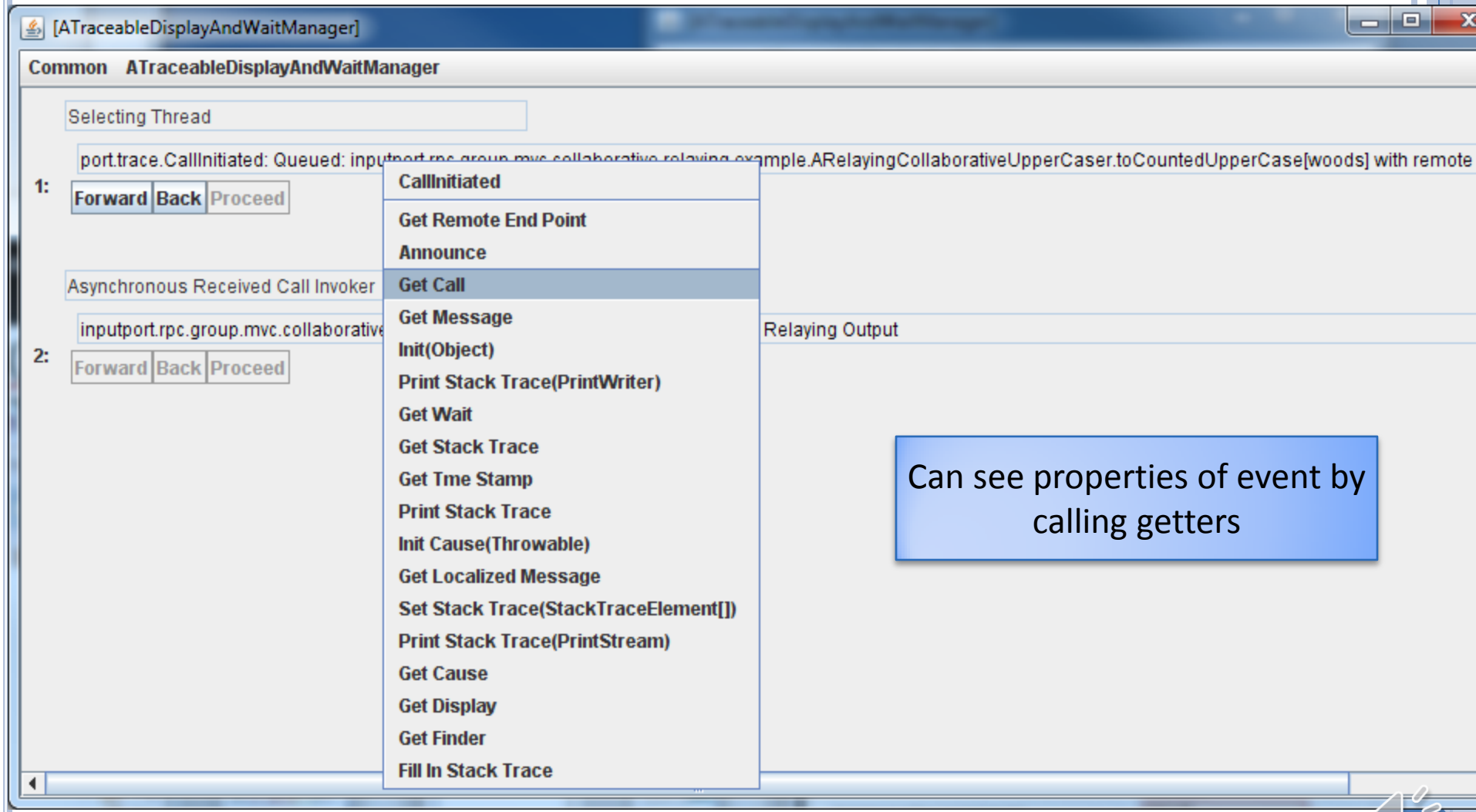


	Class Name	File Name	Line Number	Method Name
1	port.trace.CallInitiated	CallInitiated.java	21	newCase
2	inputport.rpc.simplex.ASim	ASimplexCallReceiveTrapp	38	notifyPortReceive
3	inputport.rpc.duplex.ADuple	ADuplexCallReceiveTrappe	28	notifyPortReceive
4	inputport.rpc.simplex.ASim	ASimplexRPCServerInputP	122	messageReceived
5	inputport.datacomm.ARece	AReceiveRegistrarAndNoti	28	notifyPortReceive
6	inputport.datacomm.ARece	AReceiveMessageForward	20	notifyPortReceive
7	inputport.datacomm.group	AnAbstractGroupInputPort.j	243	messageReceived
8	inputport.datacomm.ARece	AReceiveRegistrarAndNoti	28	notifyPortReceive
9	inputport.datacomm.ARece	AReceiveMessageForward	20	notifyPortReceive
10	inputport.datacomm.simpl	ADeserializingForwarder.ja	27	notifyPortReceive
11	inputport.datacomm.simpl	ADeserializingForwarder.ja	1	notifyPortReceive
12	inputport.datacomm.simpl	ASimplexObjectServerInpu	54	messageReceived
13	inputport.datacomm.simpl	ASimplexObjectServerInpu	1	messageReceived
14	inputport.datacomm.ARece	AReceiveRegistrarAndNoti	28	notifyPortReceive
15	inputport.datacomm.ARece	AReceiveMessageForward	20	notifyPortReceive
16	inputport.datacomm.group	AnAbstractGroupInputPort.j	243	messageReceived
17	inputport.datacomm.ARece	AReceiveRegistrarAndNoti	28	notifyPortReceive
18	inputport.datacomm.ARece	AReceiveMessageForward	20	notifyPortReceive
19	inputport.datacomm.simpl	AGenericSimplexBufferSer	156	notifyPortReceive
20	inputport.datacomm.simpl	AGenericSimplexBufferSer	87	messageReceived
21	inputport.datacomm.simpl	AnNIOSimplexBufferServer	92	socketChannelRead
22	inputport.datacomm.simpl	AReadCommand.java	80	notifyRead
23	inputport.datacomm.simpl	AScatterGatherReadComm	99	maybeExtractMessageAnd
24	inputport.datacomm.simpl	AScatterGatherReadComm	35	processRead
25	inputport.datacomm.simpl	AReadCommand.java	43	execute
26	inputport.datacomm.simpl	ASelectionReadManager.ja	45	processRead
27	inputport.datacomm.simpl	ASelectionManager.java	180	processSelectedOperation
28	inputport.datacomm.simpl	ASelectionManager.java	207	run

Inheritance Hierarchy

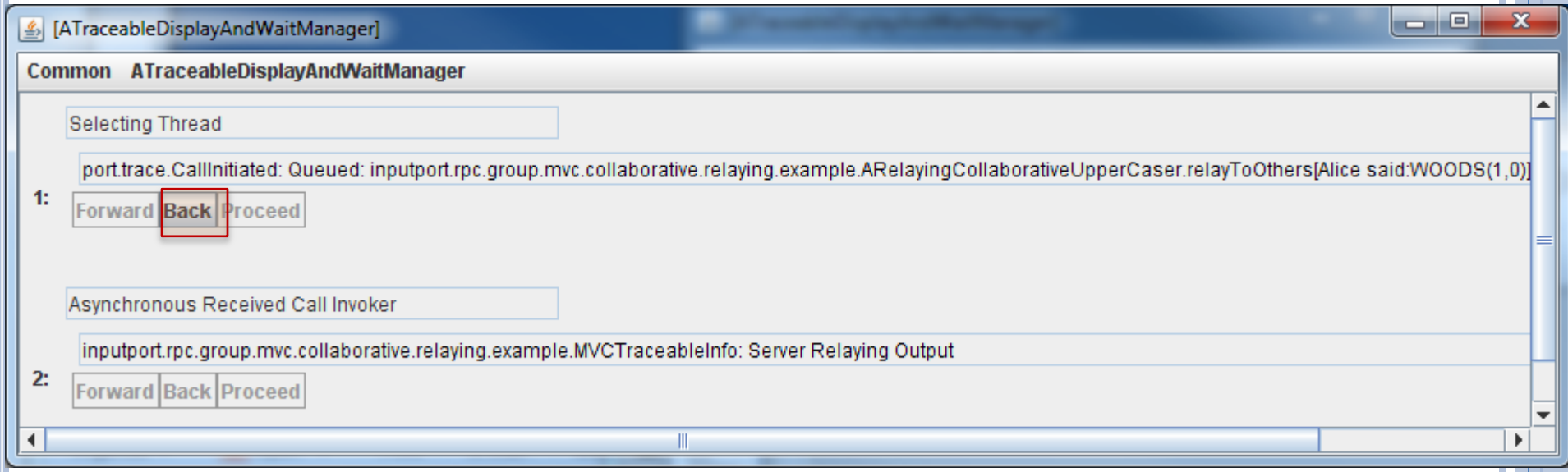


Exploring Variables: Event Properties

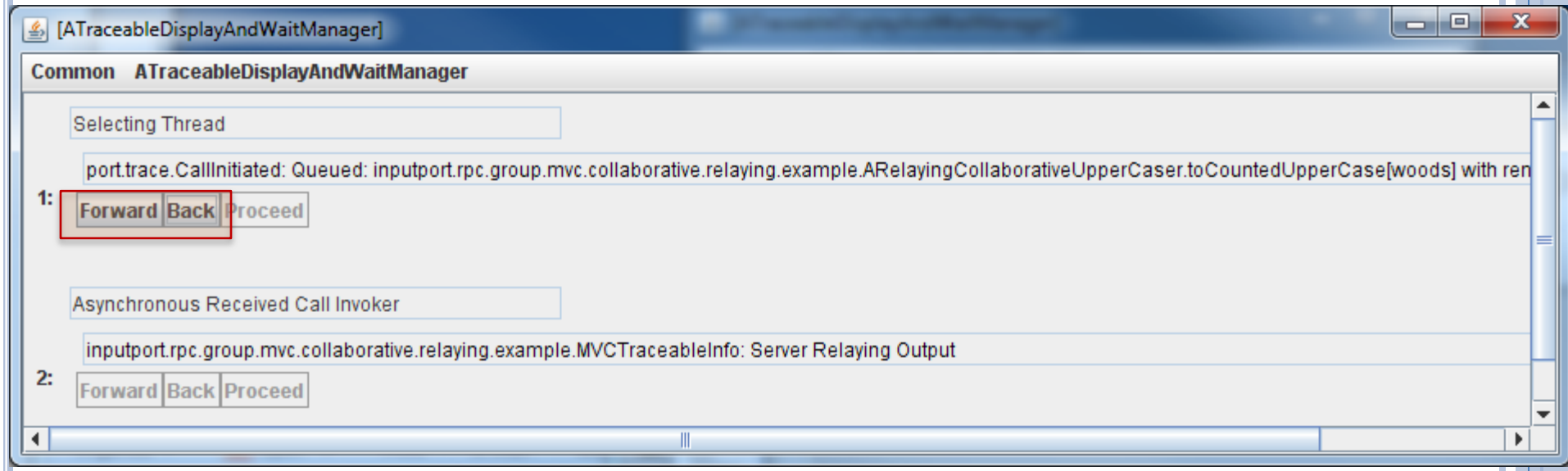


Can see properties of event by calling getters

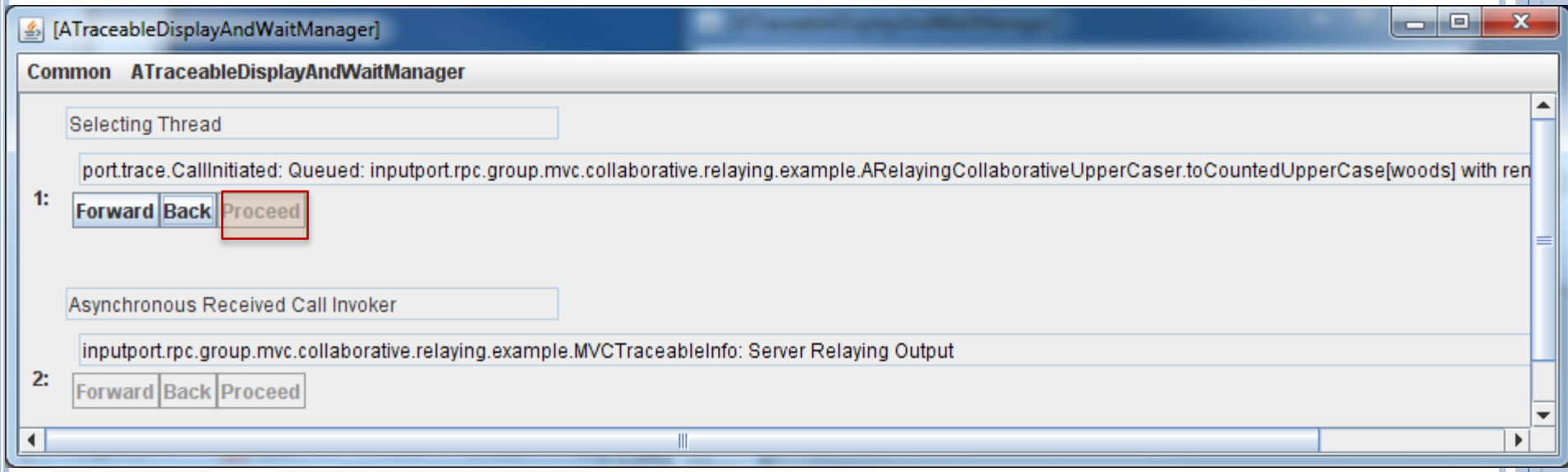
Thread Has History of Typed Trace Info



Browsing: Last to Last Call



Resume: Proceed Button



Set Break Point: Pause/Display all Announced Messages?

May want to pause only some of the announcements

May want to display only some of the announcements

How to specify a set of related announcements that should be displayed or paused?

Using Announcer Object Attributes

static setKeywordDisplayStatus (Object announcer, boolean status)

static setImplicitDisplayKeywordKind (ImplicitKeywordKind val)

static setImplicitWaitKeywordKind (ImplicitKeywordKind val)

Tracer

default

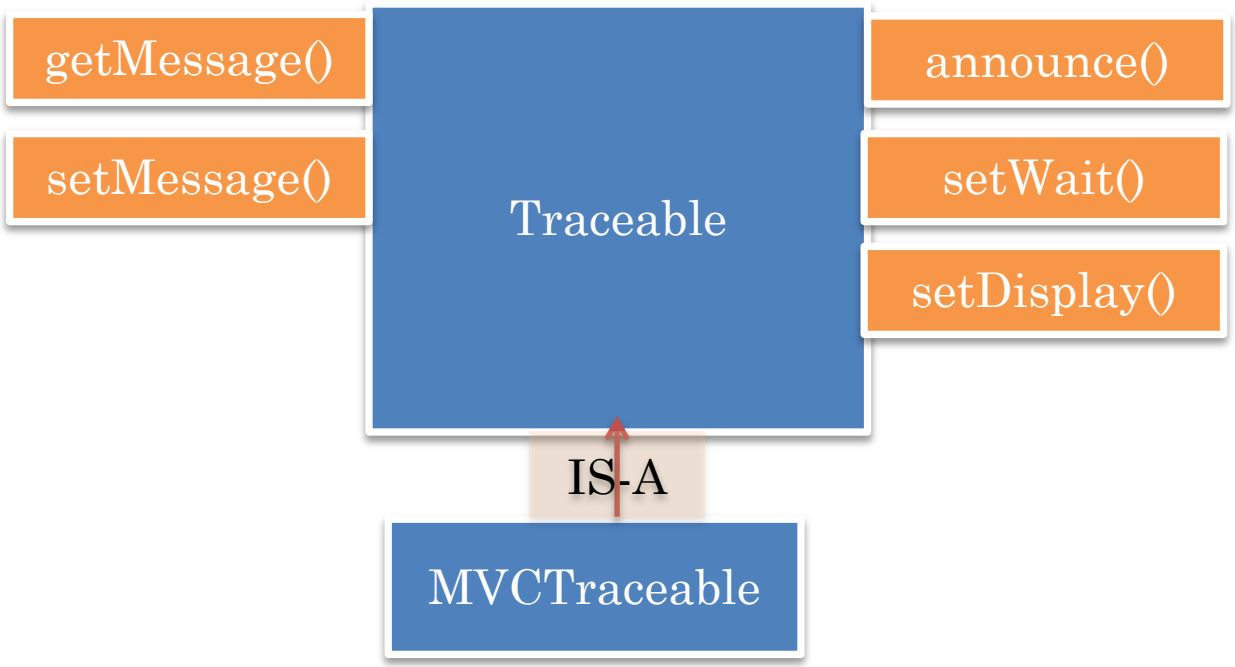
```
public enum ImplicitKeywordKind {
        OBJECT_TO_STRING,
        OBJECT_CLASS_NAME,
        OBJECT_PACKAGE_NAME }
```

setKeywordDisplay(Wait)Status(announcer, true(false)) says that if an event is announced by an object whose toString()/class/package attribute is that of the announcer then it should be displayed(wait).

Print, Display and Wait are three different things you can do with traced information

setImplicitDisplay(Wait)KeywordKind determines if toString(), class or package attribute is used for all events

Using Event Object



Debugger Issues Resolved

- Debugger makes it difficult to test race conditions
- All threads and processes mapped to a single code window
- Cannot see the history of actions taken by a thread
- Break points do not transfer to another computer
- Cannot use a mechanism to set multiple related debug points

Event Class Filtering

```
TracingLaunchEventClass [Java Application] D:\Program Files\Java\jdk1.7.0_51\bin\javaw.exe (Sep 3, 2014, 9:
I***Tracer: showInfo = true
Please enter an input line or quit or history
Woods
I***EvtType(trace.echo.ListEditMade) EvtSrc(echo.modular
52299367, 9:51:39) Thread(main) ListEdit_ADD(0,Woods,His
I***EvtType(trace.echo.modular.ListEditObserved) EvtSrc(
actor) Time(1409752299368, 9:51:39) Thread(main) ListEdi
Woods
Please enter an input line or quit or history
```

Two sources, as the same event fired twice, first by the real source and then by the event object pretending to be the source

All events of type ListEditMade or ListEditObserved

Can be announced by different sources

Alternative (source/event) class-based filtering?

```
Tracer.showInfo(true);
Tracer.setImplicitPrintKeywordKind
     (ImplicitKeywordKind.OBJECT_CLASS_NAM
Tracer.setMessagePrefixKind
     (MessagePrefixKind.SHORT_CLASS_NAME);
TraceableInfo.setPrintSource(true);
TraceableInfo.setPrintTime(true);
TraceableInfo.setPrintThread(true);
Tracer.setKeywordPrintStatus(ListEditMade
Tracer.setKeywordPrintStatus(ListEditObserved.class, true);
```