Environment Modeling
with
JavaPathfinder (JPF)
Part 3 - Lab

Oksana Tkachuk
SGT / NASA Ames Research Center
<oksana.tkachuk@nasa.gov>

Peter C. Mehlitz
SGT / NASA Ames Research Center
<peter.c.mehlitz@nasa.gov>

Roadmap

Ames Research Center

- Working with jpf-core
  - install, build and run

- Working with jpf-awt
  - install, build and run extensions
  - basic understanding of jpf-awt environment modeling (user input scripts)
  - working with application properties
  - use of listeners for trace analysis

Install - Steps

- verify System requirements
- start with `jpf-core`
  - download
  - build
  - test
  - configure
- repeat for JPF extension projects (`jpf-symbc, jpf-awt, ...`)
  - for available projects, see “Projects” navigation bar on wiki
    same procedure for JPF extensions (e.g. jpf-awt)

Install - Prerequisites

- JDK6 and above (Windows: make sure JDK, *not* JRE is used)
  - Windows, Linux: http://www.oracle.com/technetwork/java/javase/downloads
  - OS X: via “System Preferences” > “Software Update”
- Mercurial (Version Control System, uses Python):

  http://mercurial.selenic.com/wiki/Download
- optional IDEs:
  - Eclipse: http://www.eclipse.org
  - NetBeans: http://www.netbeans.org

Install - Getting JPF

- preferred: get jpf-core sources from Mercurial repository (from shell)

```
hg clone http://babelfish.arc.nasa.gov/hg/jpf/jpf-core
```

- alternatively get *.zip snapshot attachment from `http://babelfish.arc.nasa.gov/trac/jpf/wiki/projects/jpf-core`
- (optionally) get JPF extension project sources (e.g. `jpf-awt`, `jpf-aprop`)
- (optionally) create `${user.home}/.jpf/site.properties` file
  - Windows: `%userprofile%` or `System.getProperty("user.home")`
  - Unix, Linux, OS X: `~/`

Install - Building JPF

- jpf-core comes with all the required build tools (except javac)
- build from within downloaded jpf-core directory

```
bin/ant build
```

```
Buildfile: /Users/pcmehlitz/projects/jpf/jpf-core/build.xml
...
-init:
    [mkdir] Created dir: /Users/pcmehlitz/projects/jpf/jpf-core/build
...
build:
      [jar] Building jar: /Users/pcmehlitz/projects/jpf/jpf-core/build/jpf.jar
      [jar] Building jar: /Users/pcmehlitz/projects/jpf/jpf-core/build/jpf-classes.jar
      [jar] Building jar: /Users/pcmehlitz/projects/jpf/jpf-core/build/jpf-annotations.jar
      [jar] Building jar: /Users/pcmehlitz/projects/jpf/jpf-core/build/RunJPF.jar
      [jar] Building jar: /Users/pcmehlitz/projects/jpf/jpf-core/build/RunTest.jar
      [jar] Building jar: /Users/pcmehlitz/projects/jpf/jpf-core/build/RunAnt.jar
BUILD SUCCESSFUL
```

- same procedure for JPF extension projects

Install - Testing JPF build

- JPF comes with regression test suite
- test from within downloaded jpf-core directory

```
bin/ant test
```

```
Buildfile: /Users/pcmehlitz/projects/jpf/jpf-core/build.xml
...
test:
    [junit] Running TypeNameTest
    [junit] Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0.385 sec
    ...
    [junit] Running gov.nasa.jpf.util.script.ScriptEnvironmentTest
    [junit] Tests run: 3, Failures: 0, Errors: 0, Time elapsed: 0.028 sec

BUILD SUCCESSFUL
Total time: 1 minute 44 seconds
```

- same procedure for extension projects (if they have tests)

Install - Creating site.properties

- run from within jpf-core directory

```
bin/jpf -addproject
```

this will create a `${user.home}/.jpf/site.properties` file

- or just edit `~/.jpf/site.properties` manually:

project name

project path

```
jpf.home = ${user.home}/projects/jpf

jpf-core = ${jpf.home}/jpf-core
jpf-numeric = ${jpf.home}/jpf-numeric
...
extensions=${jpf-core},...
```

Running JPF

- for purists (tedious, do only if you have to)
  - setting up classpaths `>export CLASSPATH=...jpf-core/build/jpf.jar...`
  - invoking JVM `>java gov.nasa.jpf.JPF +listener=... x.y.MySUT`

- using site config and starter jars (much easier and portable)
  - explicitly `>java -jar tools/RunJPF.jar MySUT-verify.jpf`
  - using ./bin scripts:

```
bin/jpf MySUT-verify.jpf
```

- running JPF from within JUnit
- running JPF from your program (tools using JPF)

- using NetBeans or Eclipse plugins
  - "Verify.." context menu item for selected *.jpf application property file
  - using provided launch configs (Eclipse) or run targets (NetBeans)

JPF Tutorial: Running from Eclipse

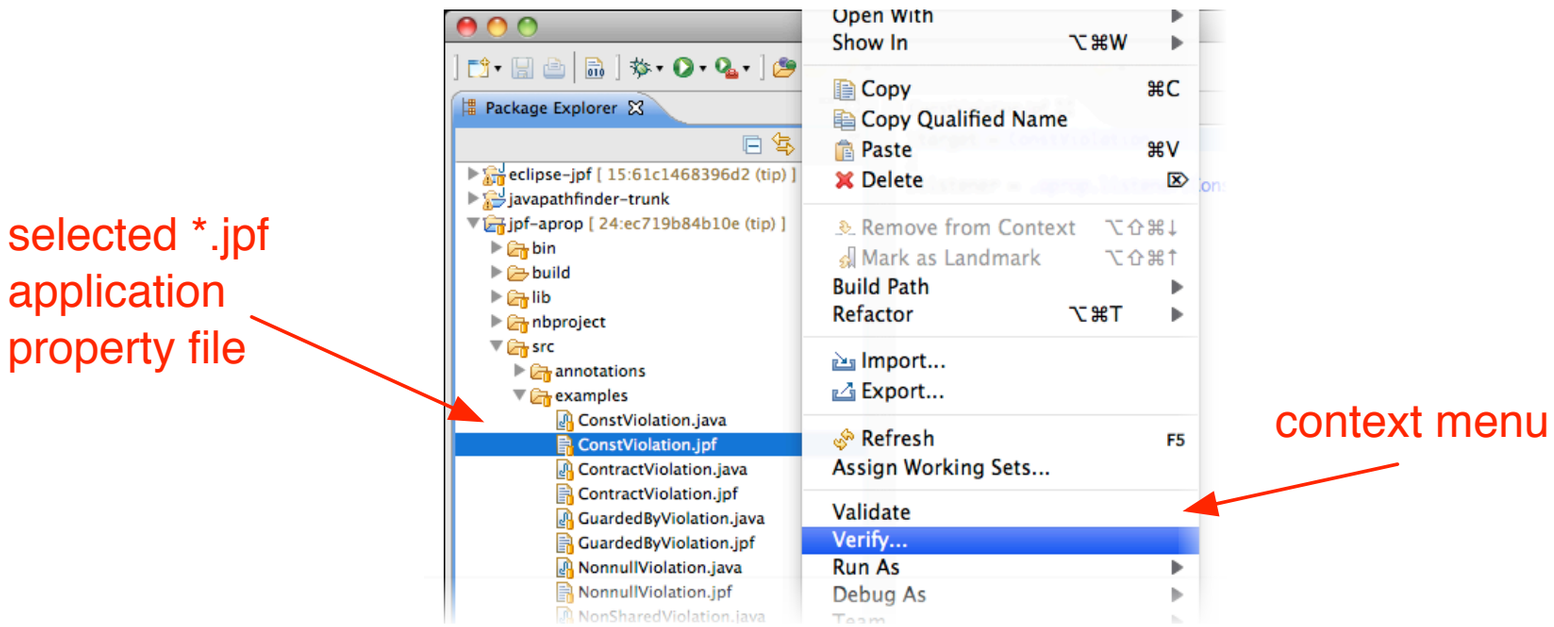
- use project provided launch configuration (requires eclipse/run-JPF.launch in project)
  - select *.jpf file in projects view
  - invoke Run As→Run Configurations→run-JPF from context menu
  - results in Output view

debugging

- use Eclipse JPF plugin

  from http://babelfish.arc.nasa.gov/trac/jpf/wiki/projects/eclipse-jpf
  - install from update site if you don't want to rebuild

    http://babelfish.arc.nasa.gov/trac/jpf/raw-attachment/wiki/install/eclipse-plugin/update/
  - optionally install jpf-shell extension if you want JPF to run in own window
  - launch JPF by selecting *.jpf file and invoking "Verify.." context menu item



JPF Tutorial: Running from NetBeans

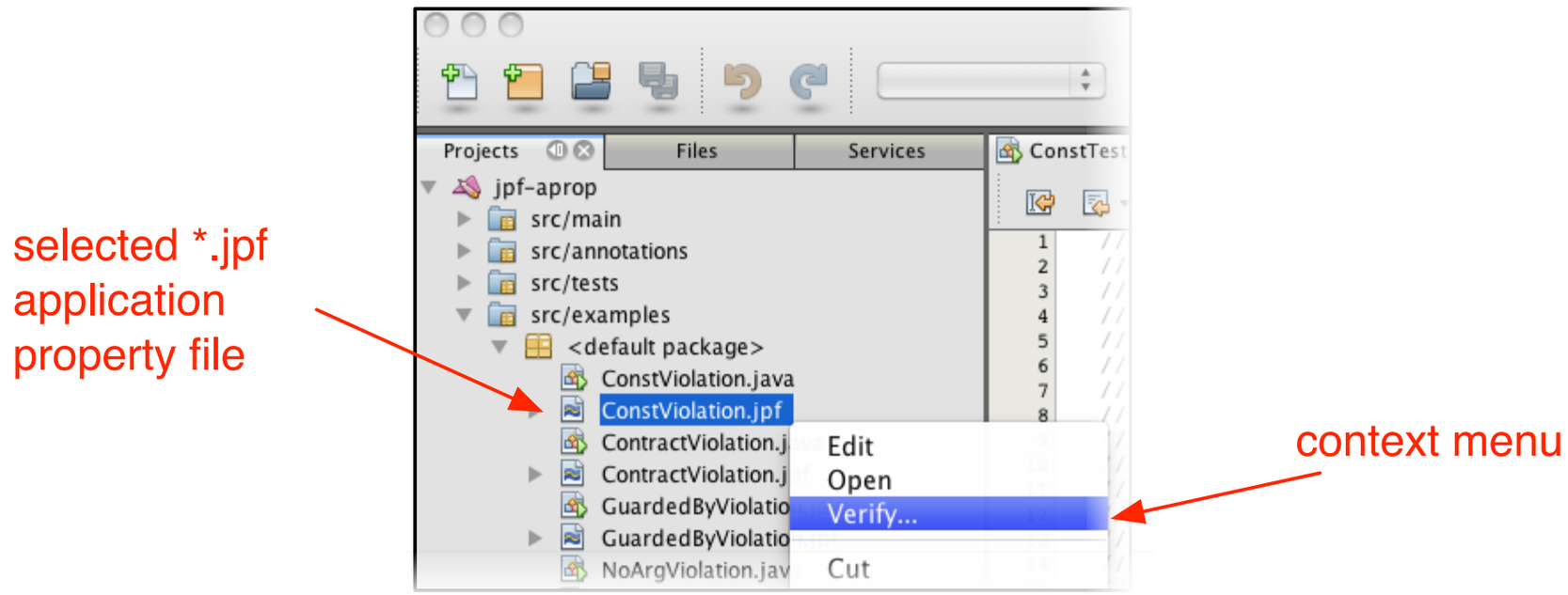
Ames Research Center

- use project provided run/debug tasks (requires nbproject/ide-file-targets.xml in project)
  - select *.jpf file in projects view
  - invoke Run→Run File from menubar (not in context menu)
  - results in Output view

debugging

- use NetBeans JPF plugin

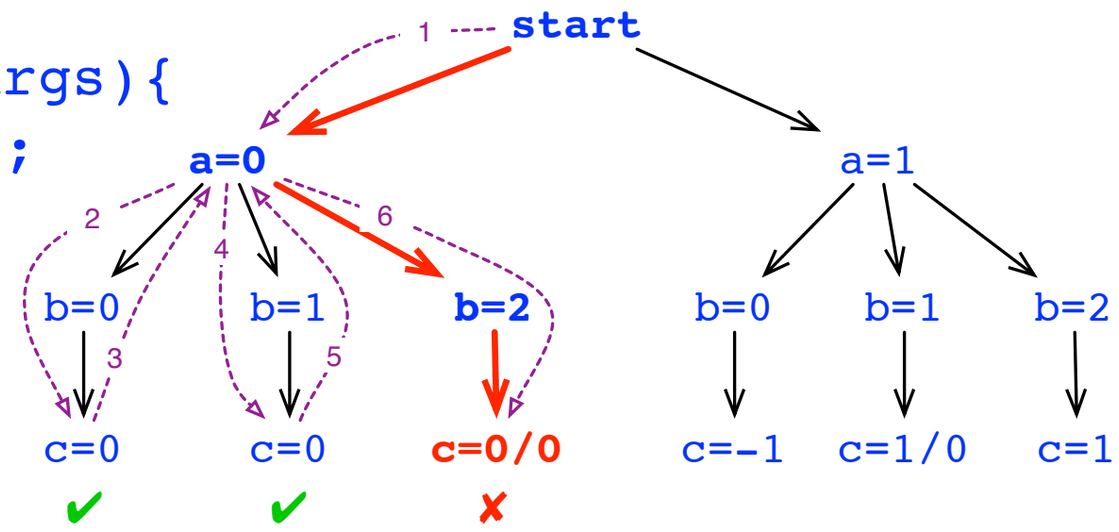
  from http://babelfish.arc.nasa.gov/trac/jpf/wiki/projects/netbeans-jpf
  - download & install attached *.nbm if you don't want to build
  - optionally install jpf-shell extension if you want JPF to run in own window
  - launch JPF by selecting *.jpf file and invoking "Verify.." context menu item



Running JPF - Example

- src/examples/Rand.java

```
public static void main(String[] args){
    Random random = new Random(42);
    int a = random.nextInt(2);
    int b = random.nextInt(3);
    int c = a/(b+a -2);
}
```



- certain combination of random values can cause division by zero

- src/examples/Rand.jpf

```
target = Rand
cg.enumerate_random = true
```

- execute:

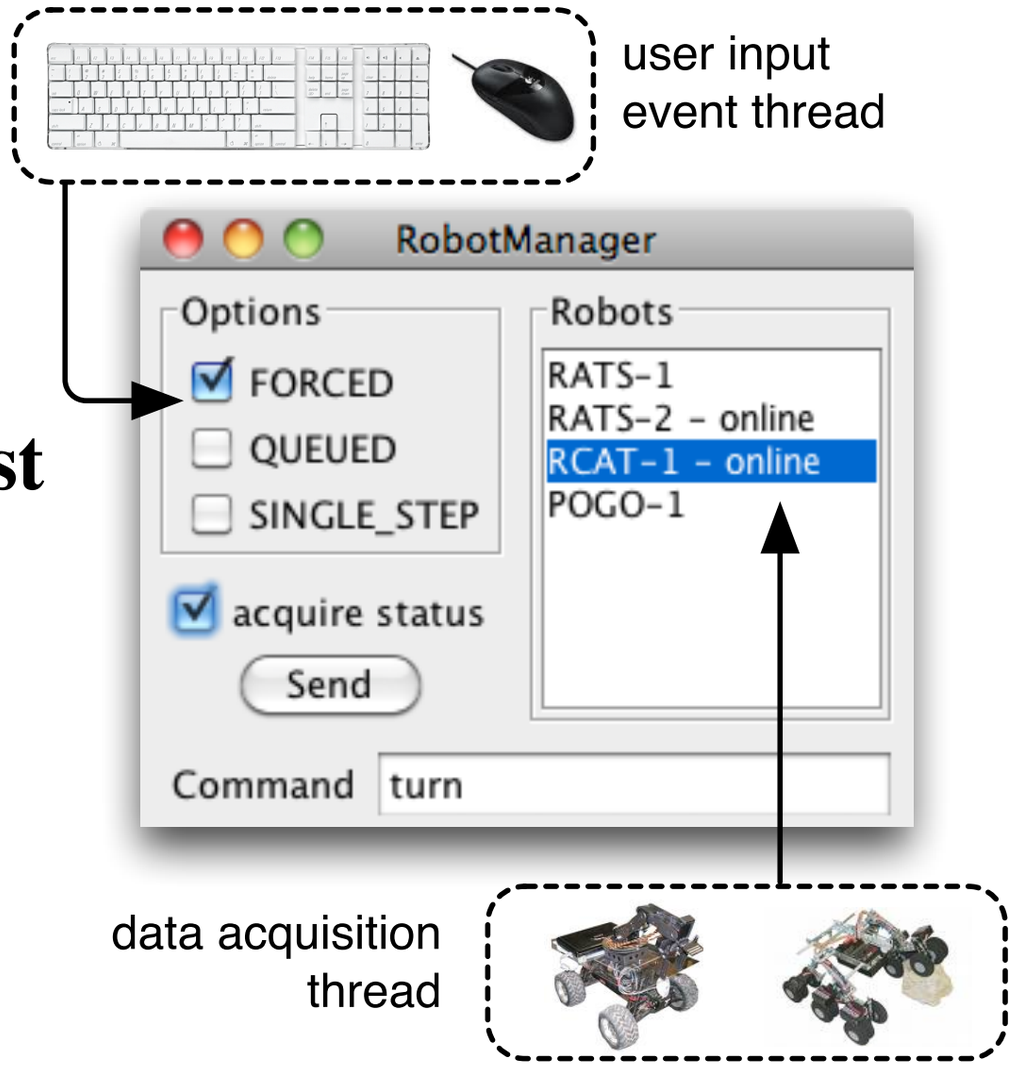
bin/jpf src/examples/Rand.jpf

```
JavaPathfinder v6.0 - (C) RIACS/NASA Ames Research Center
================================== system under test
application: Rand.java
...
================================== error #1
gov.nasa.jpf.jvm.NoUncaughtExceptionsProperty
java.lang.ArithmeticException: division by zero
        at Rand.main(Rand.java:16)
```

```
a=0
  b=0          ,a=0
=>  c=0        ,a=0,b=0
  b=1          ,a=0
=>  c=0        ,a=0,b=1
  b=2          ,a=0
```

JPF-AWT - Example

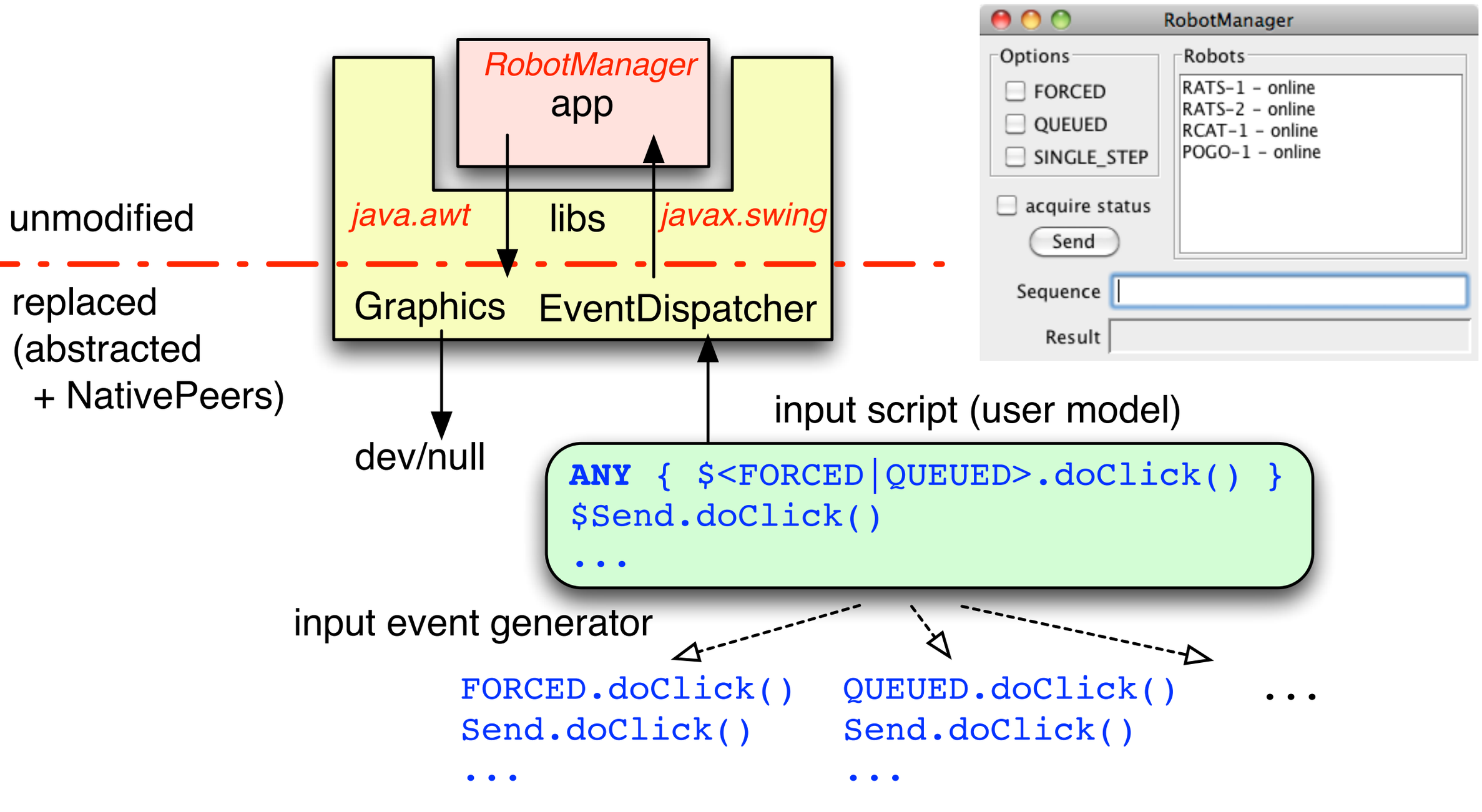
- graphical user interface app
  ⇒ uses huge framework (Swing,AWT)
  400+ classes
- concurrent data acquisition
- user input and concurrency ⇒ **hard to test**
- needs tool that automatically finds errors, supports environment modeling and faithful execution




```
jpf src/examples/RobotManager-thread.jpf
...
========================== error #1
NoUncaughtExceptionsProperty
java.lang.NullPointerException: Calling 'processSequence(String)' on null object
        at RobotManager.sendSequence(RobotManager.java:265)
        at RobotManagerView.sendSequence(RobotManager.java:565)
...
```

JPF-AWT - How does it work?

- modeled libraries + NativePeers
- user simulated with input script supporting alternatives



JPF-AWT - Get/Build/Run

- get jpf-awt extension

  `hg clone http://babelfish.arc.nasa.gov/hg/jpf/jpf-awt`

- from within `./jpf-awt`
  - build it: `bin/ant`
  - run example

```
bin/jpf src/examples/RobotManager-thread.jpf
```

```
...
====================================== error #1
gov.nasa.jpf.jvm.NoUncaughtExceptionsProperty
java.lang.NullPointerException: Calling 'processSequence(Ljava/lang/
String;)Ljava/lang/String;' on null object
        at RobotManager.sendSequence(RobotManager.java:265)
        at RobotManagerView.sendSequence(RobotManager.java:565)
        at RobotManagerView$3.actionPerformed(RobotManager.java:337)
...
elapsed time:       00:00:06
states:             new=3558, visited=4013, backtracked=7498, end=0
search:             maxDepth=526, constraints hit=0
choice generators:  thread=3289 (signal=0, lock=8, shared ref=2980), data=269
heap:               new=13464, released=10200, max live=3221, gc-cycles=4885
instructions:       615756
max memory:         81MB
loaded code:        classes=406, methods=5003
```

JPF-AWT - Application Properties File

- application property file (e.g. `RobotManager-thread.jpf`) specifies:
  - system under test main class (target)
  - user input event script (*.es) to explore
  - [optionally] JPF extensions to use
  - [optionally] listeners to use for property verification, trace analysis etc. plus related listener config

```
@using  jpf-aprop

target = RobotManager

awt.script=${config_path}/RobotManager-thread.es

listener=.listener.ChoiceTracker
choice.class=gov.nasa.jpf.awt.UIActionGenerator
```

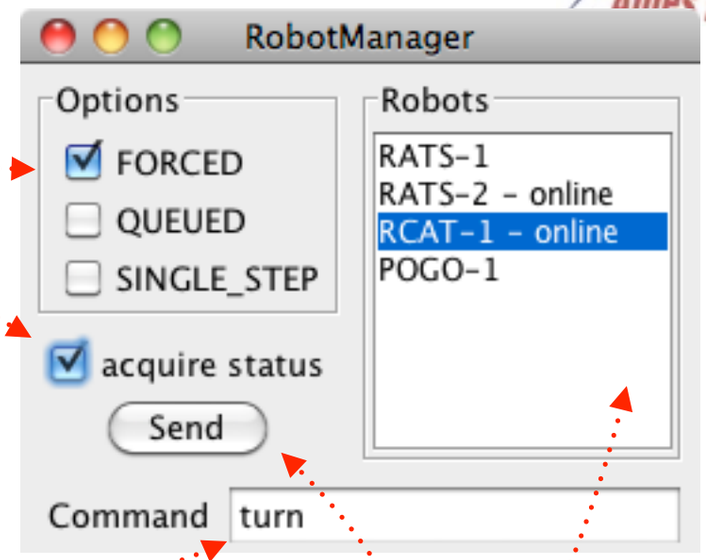
JPF-AWT - Input Event Script

- Specifies input event sequences to explore
- event specifications: `$component.method(arg,...)`
- lexical expansion: strings `<alt1|alt2>`, chars `[0-9]`
- loops: `REPEAT [count] {event ..}`
- choices: `ANY {event ..}`

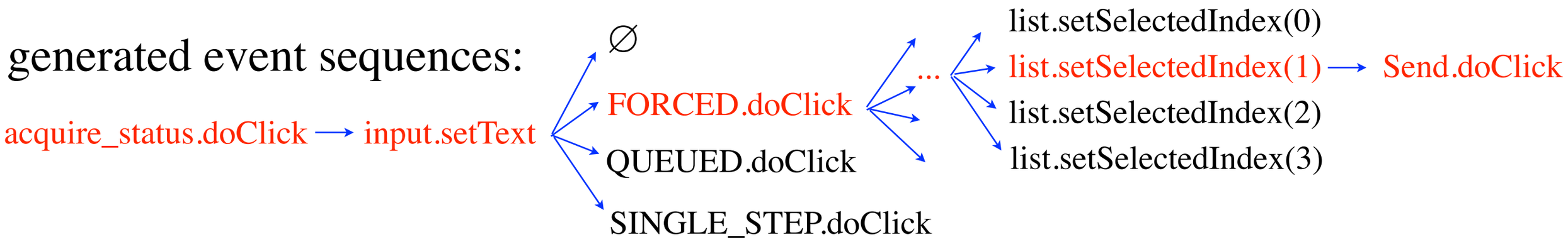
```
// start the acquisition thread
$acquire_status.doClick()

// enter command sequence (sequence)
!$Command:input.setText("turn")

// try all option combinations
REPEAT 3 { ANY { NONE,$<FORCED|QUEUED|SINGLE_STEP>.doClick() } }

// select all robots (list selection)
ANY { $Robots:list.setSelectedIndex([0-3]) }

// send sequence (send button)
$Send.doClick()
```

JPF-AWT - Input Script


```
// start the acquisition thread
$acquire_status.doClick()

// enter command sequence (sequence)
!$Command:input.setText("turn")

// try all option combinations
REPEAT 3 {
  ANY { NONE,$<FORCED|QUEUED|SINGLE_STEP>.doClick() }
}

// select all robots (list selection)
ANY { $Robots:list.setSelectedIndex([0-3]) }

// send sequence (send button)
$Send.doClick()
```



generated event sequences:



JPF-AWT - non-concurrent example (1)

Ames Research Center

```
// systematically try all options for all robots
// try to find illegal option/robot combinations
!$Command:input.setText("turn")
REPEAT 3 {
  ANY { NONE,$<FORCED|QUEUED|SINGLE_STEP>.doClick() }
}
ANY { $Robots:list.setSelectedIndex([0-3]) }
$Send.doClick()
```

src/examples/RobotManager-nothread.es

src/examples/RobotManager-nothread.jpf

```
target = RobotManager
awt.script=${config_path}/RobotManager-nothread.es

listener=.listener.ChoiceTracker
choice.class=gov.nasa.jpf.awt.UIActionGenerator

report.console.property_violation=error
classpath=${jpf-awt}/build/examples;${classpath}
sourcepath=${jpf-awt}/src/examples;${sourcepath}
```

```
bin/jpf src/examples/RobotManager-nothread.jpf
```

```
...
====================================================== error #1
gov.nasa.jpf.jvm.NoUncaughtExceptionsProperty
java.lang.AssertionError: POGOs cannot force queued sequences
   at POGO.processSequence(RobotManager.java:92)
   ...
```

JPF-AWT - non-concurrent example (2)

Ames Research Center

```
...
============================== error #1
NoUncaughtExceptionsProperty
java.lang.AssertionError: POGOs cannot force queued sequences
    at POGO.processSequence(RobotManager.java:92)
    ...
============================== trace #1
------------------------------ transition #0 thread: 0
ThreadChoiceFromSet {id:"<root>" ...
  RobotManager.java:270   : if (args.length > 0) {
  RobotManager.java:276   : RobotManager me = new RobotManager();
...
------------------------------ transition #50 thread: 2
UIActionSingleChoice[id="processScriptAction",>$Send.doClick()]
...
  RobotManager.java:92    : assert !(Option.FORCED.isSet() &&
    Option.QUEUED.isSet()) : "POGOs cannot force...";
```

trace too long

needs analysis

src/examples/RobotManager-nothread.jpf

```
listener=.listener.ChoiceTracker
choice.class=gov.nasa.jpf.awt.UIActionGenerator
```

```
============================== choice trace #1
  0: UIActionSingleChoice[id="processScriptAction",>!$Command:input.setText("turn")]
  1: UIActionFromSet[…, >NONE,$FORCED.doClick,$QUEUED.doClick,$SINGLE_STEP.doClick]
  2: UIActionFromSet[…,>$FORCED.doClick,…]
  3: UIActionFromSet[…,>$QUEUED.doClick,…]
  4: UIActionFromSet[…,>$Robots:list.setSelectedIndex(3)]
  5: UIActionSingleChoice[…,>$Send.doClick]
```

JPF-AWT - concurrent example

Ames Research Center

src/examples/RobotManager-thread.es

```
...
// start background data acquisition thread
$acquire_status.doClick()
...
```

src/examples/RobotManager-thread.jpf

```
target = RobotManager
awt.script=${config_path}/RobotManager-thread.es
...
# the modeled data acquisition thread needs it
cg.enumerate_random=true

# we are not interested in the known POGO assertion
vm.disable assertions=POGO
```

```
bin/jpf src/examples/RobotManager-thread.jpf
```

```
...
====================================================== error #1
NoUncaughtExceptionsProperty
java.lang.NullPointerException: Calling 'processSequence(…)' on null object
        at RobotManager.sendSequence(RobotManager.java:266)
        at RobotManagerView.sendSequence(RobotManager.java:566)
        at RobotManagerView$3.actionPerformed(RobotManager.java:338)
...
```

JPF-AWT - Trace Analysis (1)

Ames Research Center

- JPF finds deep NullPointerException that depends on scheduling sequences **and** user input

```
...
------------------------------------------------- transition #71 thread: 3
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet {id:"sharedField" ,2/2,isCascaded:false}
  RobotManager.java:42              : return id;
  RobotManager.java:242             : onlineRobots.remove(robot.getId());
      [98 insn w/o sources]
  RobotManager.java:242             : onlineRobots.remove(robot.getId());
  RobotManager.java:244             : listModel.changed(robot);
------------------------------------------------- transition #72 thread: 2
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet {id:"sharedField" ,1/2,isCascaded:false}
  RobotManager.java:256             : return onlineRobots.get(robotName);
      [47 insn w/o sources]
  RobotManager.java:256             : return onlineRobots.get(robotName);
  RobotManager.java:564             : Robot robot = model.getOnlineRobot(selectedRobotName);
  RobotManager.java:565             : String result = model.sendSequence(robot, sequence);
  RobotManager.java:265             : return robot.processSequence(sequence);
...
==================================================== statistics
elapsed time:       00:00:06
states:             new=3558, visited=4013, backtracked=7498, end=0
search:             maxDepth=526, constraints hit=0
choice generators:  thread=3289 (signal=0, lock=8, shared ref=2980), data=269
heap:               new=13464, released=10200, max live=3221, gc-cycles=4885
instructions:       615756
max memory:         81MB
loaded code:        classes=406, methods=5003
```

trace way too long

JPF-AWT - Trace Analysis (2)

- error is too general

```
java.lang.NullPointerException:
   Calling 'processSequence(Ljava/lang/String;)Ljava/lang/String;' on null object
         at RobotManager.sendSequence(RobotManager.java:269)
         at RobotManagerView.sendSequence(RobotManager.java:567)
         at RobotManagerView$3.actionPerformed(RobotManager.java:341)
         at javax.swing.AbstractButton.fireActionPerformed(AbstractButton.java:120)
         ...
```

- needs trace analysis

src/examples/RobotManager-thread-oma.jpf

```
...
listener=.listener.OverlappingMethodAnalyzer
method.include=*RobotManager*
```

call
execute
return

```
…
2:   < RobotManager@13d.isRobotOnline(Ljava/lang/String;)Z        thread 2
2: >-  RobotManager@13d.getOnlineRobot(Ljava/lang/String;)LRobot;
---------------------------------------- #13
3: >-  RobotManager@13d.isRobotOnline(Ljava/lang/String;)Z
---------------------------------------- #14                     thread 3
3:   < RobotManager@13d.isRobotOnline(Ljava/lang/String;)Z
3: >-  RobotManager@13d.setRobotOnline(LRobot;Z)V
---------------------------------------- #15
2:   < RobotManager@13d.getOnlineRobot(Ljava/lang/String;)LRobot;
2: >-  RobotManager@13d.sendSequence(LRobot;Ljava/lang/String;)Ljava/lang/String
```

JPF-AWT - Property Annotations

- better approach - get specific properties from SUT annotations
- get and install http://babelfish.arc.nasa.gov/hg/jpf/jpf-aprop
- more efficient to check, **self explaining** - no need for further analysis

src/examples/RobotManager.java

```
// RobotManager instances are not thread-safe
@NonShared
public class RobotManager {
  ...
```

src/examples/RobotManager-thread-aprop.jpf

```
@using jpf-aprop
listener=.listener.NonSharedChecker
nonshared.throw_on_cycle=true
```

```
...
========================================== error #1
NoUncaughtExceptionsProperty
AssertionError: NonShared object: RobotManager@140 accessed in live thread cycle:
                                  AWT-EventQueue-0,Thread-3,AWT-EventQueue-0,main
    at RobotManagerView.sendSequence(RobotManager.java:564)
    at RobotManagerView$3.actionPerformed(RobotManager.java:338)
    at javax.swing.AbstractButton.fireActionPerformed(AbstractButton.java:120)
    ....
========================================== statistics
elapsed time:       00:00:03
states:             new=1809, visited=2009, backtracked=3758, end=0
...
```

Conclusions

- we (hopefully) learned how to
  - get, build and install jpf-core
  - get, build and install extensions (jpf-awt, jpf-aprop)
  - run JPF
  - use domain specific extensions (jpf-awt) for environment modeling
  - use listeners for trace analysis

- .. if not
  - check out http://babelfish.arc.nasa.gov/trac/jpf
  - all answers will be there (eventually)
  - .. if not - try http://groups.google.com/group/java-pathfinder
    - we are here to help

Conclusions

- we (hopefully) learned how to
  - get, build and install jpf-core
  - get, build and install extensions (jpf-awt, jpf-aprop)
  - run JPF
  - use domain specific extensions (jpf-awt) for environment modeling
  - use listeners for trace analysis

- .. if not
  - check out http://babelfish.arc.nasa.gov/trac/jpf
  - all answers will be there (eventually)
  - .. if not - try http://groups.google.com/group/java-pathfinder
    - we are here to help

Thank You!