JPF Lab
Formal Methods Summer School 2011
Menlo College

Neha Rungta
SGT / NASA Ames Research Center
<neha.s.rungta@nasa.gov>

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

JPF Lab: Roadmap

- where to get help
- install, build and test JPF
- running JPF examples
- extending JPF

JPF Lab: where to get help

http://babelfish.arc.nasa.gov/trac/jpf

- public read access
- edit for account holders (also non-NASA)

bug tracking

- Trac ticket system

project blog

- announcements
- important changes

hierarchical navigation menu

- intro
- installation
- user docu
- developer docu
- **extension projects**

Java Path Finder

http://babelfish.arc.nasa.gov/trac/jpf/wiki

JPF .. the swiss army knife of Java™ verification

logged in as pcmehlitz@TI.ARC.NASA.GOV | Logout | Preferences

Search

JPF-Wiki | Timeline | Roadmap | View Tickets | New Ticket | Search | Admin | Blog

Start Page | Index | History | Last Change

JPFWiki - Welcome Page
Introduction...
Installing JPF...
User Guide...
Developer Guide...
Projects...
Change(B)log
About...
Papers
FAQ
Playground
Table of Context

Latest JPF News

02/14/2010	ISSTA 2010 Tutorial on Automated Testing with Java PathFinder announced
02/12/2010	Call for Google Summer of Code 2010 project proposals out on JPF Google group
01/30/2010	JPF Google group replaces old mailing lists
01/12/2010	Fujitsu press announcement released about using and extending Symbolic PathFinder (projects/jpf-symbc) for comprehensive testing of Java web applications
09/02/2009	JPF server on http://babelfish.arc.nasa.gov/trac/jpf goes live, featuring the JPFWiki and separate Mercurial repositories for JPF core and extension projects
07/22/2009	JPF wins the 2009 "Outstanding Technology Development Award" of the Federal Laboratory Consortium (FLC), Far West Division

Welcome to the JPF Wiki

This is the main page for Java™ Pathfinder, or ***"JPF"*** as we call it from here. JPF is a highly customizable execution environment for verification of Java™ bytecode programs. The system was developed at the NASA Ames Research Center, open sourced in 2005, and is freely available from this server under the NOSA 1.3 license.

The JPFWiki is our primary source of documentation. It is divided into the following sections (which you will always see in the TOC menu to the right):

JPF Lab: Install, Build & Test

- prerequisites:
  - JDK6 (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

JPF Lab: Install JPF

- get jpf-core sources
  `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-numeric`, `jpf-awt`, `jpf-aprop)`
- create `${user.home}/.jpf/site.properties` file
  - Windows: `%userprofile%` or `System.getProperty("user.home")`
  - Unix, Linux, OS X: `~/`

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

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

JPF Lab: Build JPF

- JPF 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
```

JPF Lab: Test JPF

- 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
```

JPF Lab: 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 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)

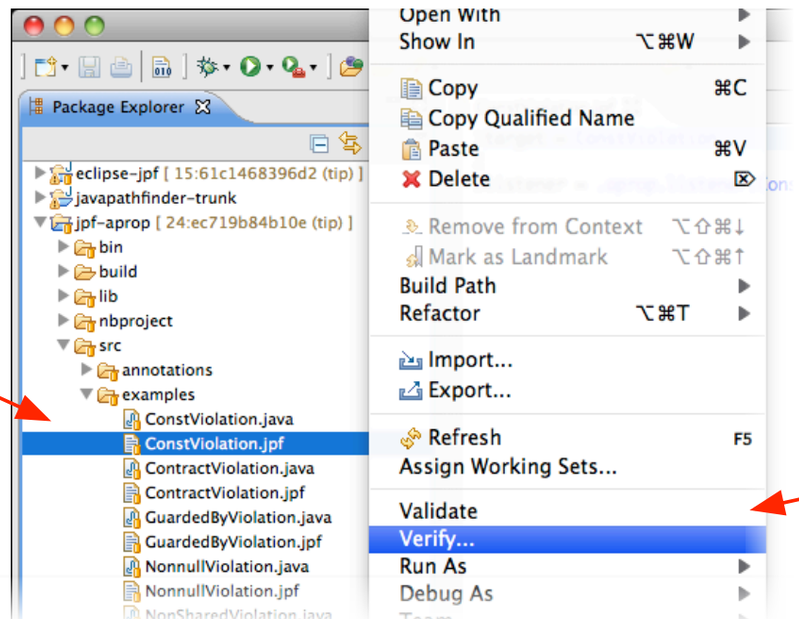
JPF Lab: Running from Eclipse

- 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

selected *.jpf application property file



context menu

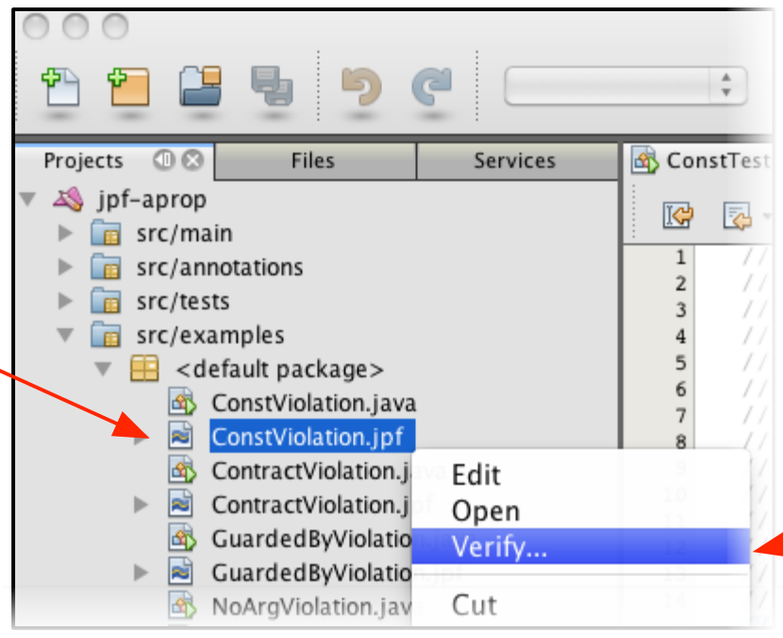
JPF Lab: Running from NetBeans

- 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

selected *.jpf application property file

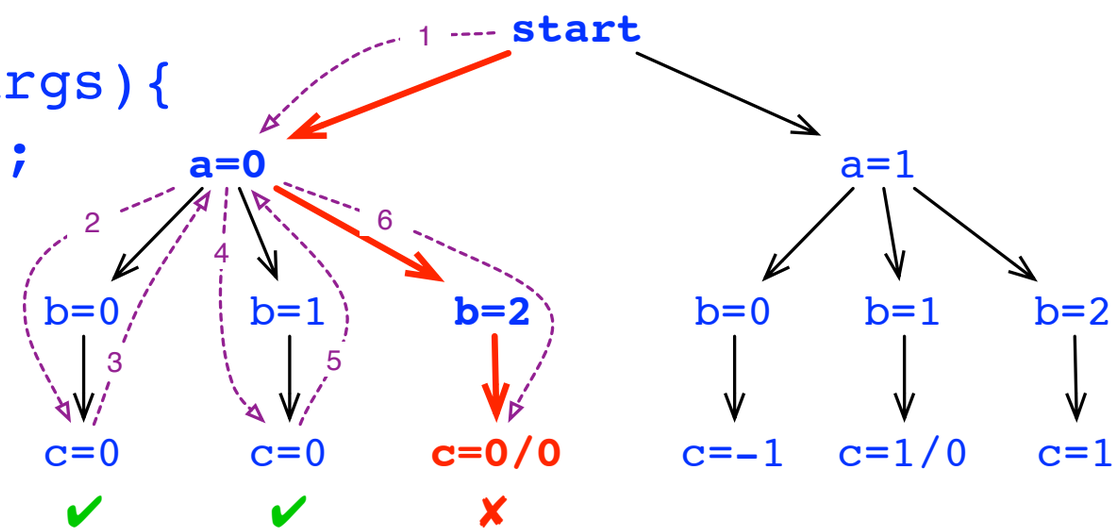


context menu

JPF Lab: Random Data 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
```

```
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
```

- 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)
```

JPF Lab: Data Race Example

- src/examples/Racer.java

```
int d = 42;

public void run () {
     doSomething(1001);                    // (1)
     d = 0;                                // (2)
}

public static void main (String[] args){
     Racer racer = new Racer();
     Thread t = new Thread(racer);
     t.start();

     doSomething(1000);                    // (3)
     int c = 420 / racer.d;                // (4)
     System.out.println(c);
 }
```

- src/examples/Racer.jpf

```
target = Racer
listener=gov.nasa.jpf.listener.PreciseRaceDetector
```

JPF Lab: Data Race Example

- src/examples/Racer.java

```
int d = 42;

public void run () {
     doSomething(1001);                          // (1)
     d = 0;                                      // (2)
}

public static void main (String[] args){
     Racer racer = new Racer();
     Thread t = new Thread(racer);
     t.start();

     doSomething(1000);                          // (3)
     int c = 420 / racer.d;                      // (4)
     System.out.println(c);
}
```

data race

- src/examples/Racer.jpf

```
target = Racer
listener=gov.nasa.jpf.listener.PreciseRaceDetector
```

JPF Lab: Data Race Example

- run >bin/jpf src/examples/Racer.jpf

```
============================ system under test
application: Racer.java
============================ search started: 5/23/11 11:35 AM
...
============================ error #1
gov.nasa.jpf.listener.PreciseRaceDetector
race for field Racer@13d.d
  main at Racer.main(Racer.java:16)
                    "int c = 420 / racer.d;
  Thread-0 at Racer.run(Racer.java:7)
                    "d = 0;

============================ trace #1
---------------------------- transition #0 thread: 0
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet[id="root",isCascaded:false,{>main}]
      [2894 insn w/o sources]
  Racer.java:11                   : Racer racer = new Racer();
  Racer.java:1                    : public class Racer implements Runnable {
      [1 insn w/o sources]
  Racer.java:3                    : int d = 42;
...
---------------------------- transition #5 thread: 0
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet[id="sharedField",isCascaded:false,{>main,Thread-0}]
  Racer.java:16                   : int c = 420 / racer.d;              // (4)
...
```

JPF Lab: Examine Example Execution

- to see what is really going on, run with additional listener:
  >bin/jpf +listener+=,.listener.ExecTracker src/examples/Racer.jpf

```
                # choice: ThreadChoiceFromSet[id="root",isCascaded:false,{>main}]
        Racer.java:11                    : Racer racer = new Racer();
0 : [0] new Racer@317
0 : [1] dup
0 : [2] invokespecial Racer.<init>()V
        Racer.java:1                     : public class Racer implements Runnable {
0 : [0] aload_0
...
------------------------------------ [1] forward: 0 new
                # choice: ThreadChoiceFromSet[id="start",isCascaded:false,{>main,Thread-0}]
0 : [0] executenative JPF_java_lang_Thread.start____V
0 : [1] nativereturn java.lang.Thread.start()V
        Racer.java:15                    : doSomething(1000);                    // (3)
...
------------------------------------ [4] forward: 3 new end
------------------------------------ [3] backtrack: 2
------------------------------------ [3] done: 2
------------------------------------ [2] backtrack: 1
                # choice: ThreadChoiceFromSet[id="sleep",isCascaded:false,{main,>Thread-0}]
        Racer.java:6                     : doSomething(1001);                    // (1)
1 : [-1] runstart
1 : [0] sipush
1 : [1] invokestatic Racer.doSomething(I)V
...
```

JPF Lab: Numeric Inspection Example

- get jpf-numeric extension

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

- build it

  `bin/ant`

- run example

  `bin/jpf src/examples/CatastrophicCanellation.jpf`

- try to find this with testing..

```
[WARNING] cancellation of:
-7.917111340668963E36+7.917111340668962E36=-1.1805916207174113E21
      at CatastrophicCancellation.main(CatastrophicCancellation.java:29)

res=-1.1805916207174113E21 (should be -0.827396...)
```

JPF Lab: MultiThreaded GUI Example

- get jpf-awt extension

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

- build it

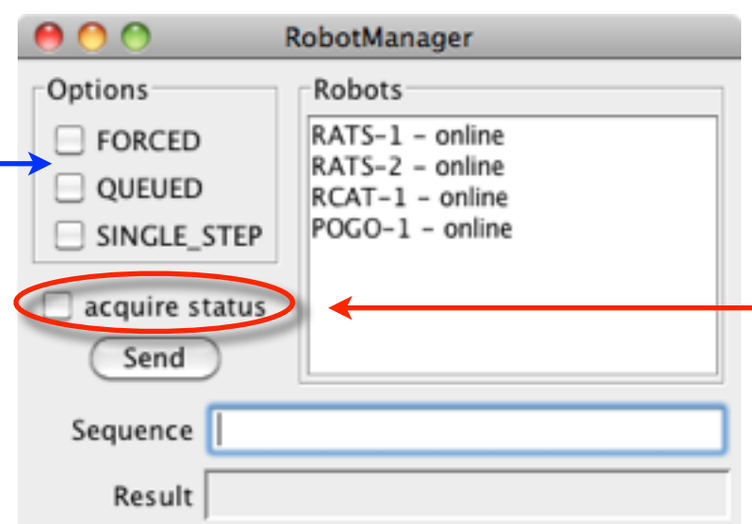
  `bin/ant`

- run example

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

- try to find this with testing..

Thread 1:
user input



Thread 2:
data acquisition

JPF Lab: Extending JPF - Goal

- goal: create a listener that detects if numeric field values are outside their specified range
- example program

```
public class SUT {
  int data; // should be within [0..42]

  void setData(int d){
    data = d;
  }

  public static void main(String[] args){
    SUT sut = new SUT();
    sut.setData( 42);  // should not trigger violation
    sut.setData(-42);  // should trigger violation
  }
}
```

JPF Lab: Extending JPF - Approach (1)

- create your sandbox test project
- get jpf-template:
  `hg clone http://babelfish.arc.nasa.gov/hg/jpf/jpf-template`
- create project
  `jpf-template/bin/create_project jpf-core jpf-lab`
  `cd jpf-lab`
- create example/test: src/examples/SUT.java

```
public class SUT {
  int data; // should be within [0..42]

  void setData(int d){
    data = d;
  }

  public static void main(String[] args){
    SUT sut = new SUT();
    sut.setData( 42);  // should not trigger violation
    sut.setData(-42);  // should trigger violation
  }
}
```

JPF Lab: Extending JPF - Approach (2)

- write listener: src/main/lab/RangeChecker.java

```
public class RangeChecker
    extends gov.nasa.jpf.PropertyListenerAdapter
```

- initialize from Config

```
FieldSpec fieldSpec;
int min, max;
```

rc.field=x.y.SUT.data
rc.min = 0
rc.max = 42

```
public RangeChecker (Config conf){
  String spec = conf.getString("rc.field");
  fieldSpec = FieldSpec.createFieldSpec(spec);
  min = conf.getInt("rc.min", Integer.MIN_VALUE);
  max = conf.getInt("rc.max", Integer.MAX_VALUE);
```

JPF Lab: Extending JPF - Approach (3)

- intercept PUTFIELD post execution notification in listener

```
@Override
public void instructionExecuted(JVM vm){

  Instruction insn = vm.getLastInstruction();

  if (insn instanceof PUTFIELD){
    if (isRelevantField(..)){
      if (isValueOutOfRange(..)){
        storeError();
```

JPF Lab: Extending JPF - Approach (4)

- filling in the blanks: checking field and values

```
FieldSpec fieldSpec;
int min, max;

void instructionExecuted(JVM vm){..
  if (insn instanceof PUTFIELD){
    PUTFIELD put = (PUTFIELD)insn;
    if (isRelevantField(put)){
      if (isValueOutOfRange(put)){
        storeError(vm, put); ..

boolean isRelevantField(PUTFIELD insn){
  FieldInfo fi = insn.getFieldInfo();
  return fieldSpec.matches(fi);
}

boolean isValueOutOfRange(PUTFIELD insn){
  int v = (int)insn.getLastValue();
  return (v < min) || (v > max);
}
```

JPF Lab: Extending JPF - Approach (5)

- finishing it up: get execution context and report the error

```
String error;

void storeError (JVM vm, PUTFIELD insn){
  ThreadInfo ti = vm.getLastThreadInfo();
  FieldInfo fi = insn.getFieldInfo();
  error = String.format(
    "field %s=%d out of range in thread %s at %s",
      fi.getFullName(), insn.getLastValue(),
      ti.getName(), insn.getSourceLocation());
}

@Override
public boolean check(Search search, JVM vm) {
  return (error == null);
}

@Override
public String getErrorMessage(){ return error; }
```

JPF Lab: Extending JPF - Approach (6)

- create src/examples/SUT.jpf

```
target = SUT
listener=lab.RangeChecker
rc.field=SUT.data
rc.min=0
rc.max=42
```

- run it: bin/jpf src/examples/SUT.jpf

```
...
============================ error #1
lab.RangeChecker
field SUT.data=-42 out of range in thread main
     at SUT.setData(SUT.java:5)
...
```

- Congratulations - Your First Listener!

JPF Lab: Extending JPF - Sources

```
package lab;

import gov.nasa.jpf.*;import gov.nasa.jpf.search.*;
import gov.nasa.jpf.jvm.*;
import gov.nasa.jpf.jvm.bytecode.*;
import gov.nasa.jpf.util.*;

public class RangeChecker
          extends PropertyListenerAdapter {
  FieldSpec fieldSpec;
  int min, max;
  String error;

  public RangeChecker (Config conf){
    String spec = conf.getString("rc.field");
    fieldSpec = FieldSpec.createFieldSpec(spec);
    min = conf.getInt( "rc.min", Integer.MIN_VALUE);
    max = conf.getInt( "rc.max", Integer.MAX_VALUE);
  }

  protected boolean isRelevantField(PUTFIELD insn){
    return fieldSpec.matches(insn.getFieldInfo());
  }

  protected boolean isValueOutOfRange(PUTFIELD insn){
    int v = (int)insn.getLastValue();
    return (v < min) || (v > max);
  }

  protected void storeError (JVM vm, PUTFIELD insn){
    ThreadInfo ti = vm.getLastThreadInfo();
    FieldInfo fi = insn.getFieldInfo();

    error = String.format(
        "field %s=%d out of range in thread %s at %s",
        fi.getFullName(), insn.getLastValue(),
        ti.getName(), insn.getSourceLocation());
  }
```

```
  @Override
  public void instructionExecuted(JVM vm){
    Instruction insn = vm.getLastInstruction();
    if (insn instanceof PUTFIELD){
      PUTFIELD put = (PUTFIELD)insn;
      if (isRelevantField(put)){
         if (isValueOutOfRange(put)){
           storeError(vm, put);
           vm.breakTransition();
         }
      }
    }
  }

  @Override
  public boolean check(Search search, JVM vm) {
    return (error == null);
  }

  @Override
  public String getErrorMessage(){
    return error;
  }
}
```

JPF Lab: Challenge

- write a listener that detects time based comparisons and creates a ChoiceGenerator executing <u>both</u> branches

```
long t1 = System.currentTimeMillis();
..
long t2 = System.currentTimeMillis();
if (t2 - t1 > MAX_TIME) {
  println("disaster");
} else {
  println("all fine");
}
```

```
0:    invokestatic  #2; // System.currentTimeMillis()
3:    lstore_1
4:    invokestatic  #2; // System.currentTimeMillis()
7:    lstore_3
8:    lload_3
9:    lload_1
10:   lsub
11:   ldc2_w    #3; //long 42l
14:   lcmp
15:   ifle      29
18:   getstatic #5; // System.out
21:   ldc       #6; // "disaster"
23:   invokevirtual  #7; // PrintStream.println()
26:   goto      37
29:   getstatic #5; // System.out
32:   ldc       #8; // "all fine"
34:   invokevirtual  #7; // PrintStream.println()
37:   return
```

JPF Lab: Challenge

- hint - write listener that
  - detects System.currentTimeMillis() calls and uses JPF attributes to tag returned values
  - detects LSUB, LCMP operands with time tag, and tags result value (needs both pre- and post-exec notification)
  - uses pre-execute notification to intercept IF_.. instructions
    - to check if operand has time tag
    - if not re-executed create a BooleanChoiceGenerator and re-execute (!ThreadInfo.isFirstStepInstruction())
    - if re-executed explicitly sets follow-on PC depending on current choice value (ThreadInfo.skipInstruction(nextPC))

- become famous: best submission will make it into JPF distribution!