JPF Tutorial
ASE 2011

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>

Willem Visser
University of Stellenbosch
<wvisser@cs.sun.ac.za>

JPF-Tutorial: Where Are the Slides?

- for now: http://babelfish.arc.nasa.gov/trac/jpf/raw-attachment/wiki/presentations/start/ASE11-Tutorial.pdf

- or go to http://babelfish.arc.nasa.gov/trac/jpf/wiki/presentations/start, scroll down to the "Attachments" section, expand and *click on the download icon trailing the filename*

- might get its own "tutorial" wiki section in the future

JPF Tutorial: Roadmap

- Basics
  - what is JPF
  - core concepts and extension mechanisms
  - where to look for help

  2h

- Install and run JPF

  - core
  - extensions

  2h

- Extending JPF

  - safety property listener
  - control flow variation listener using data value attributes

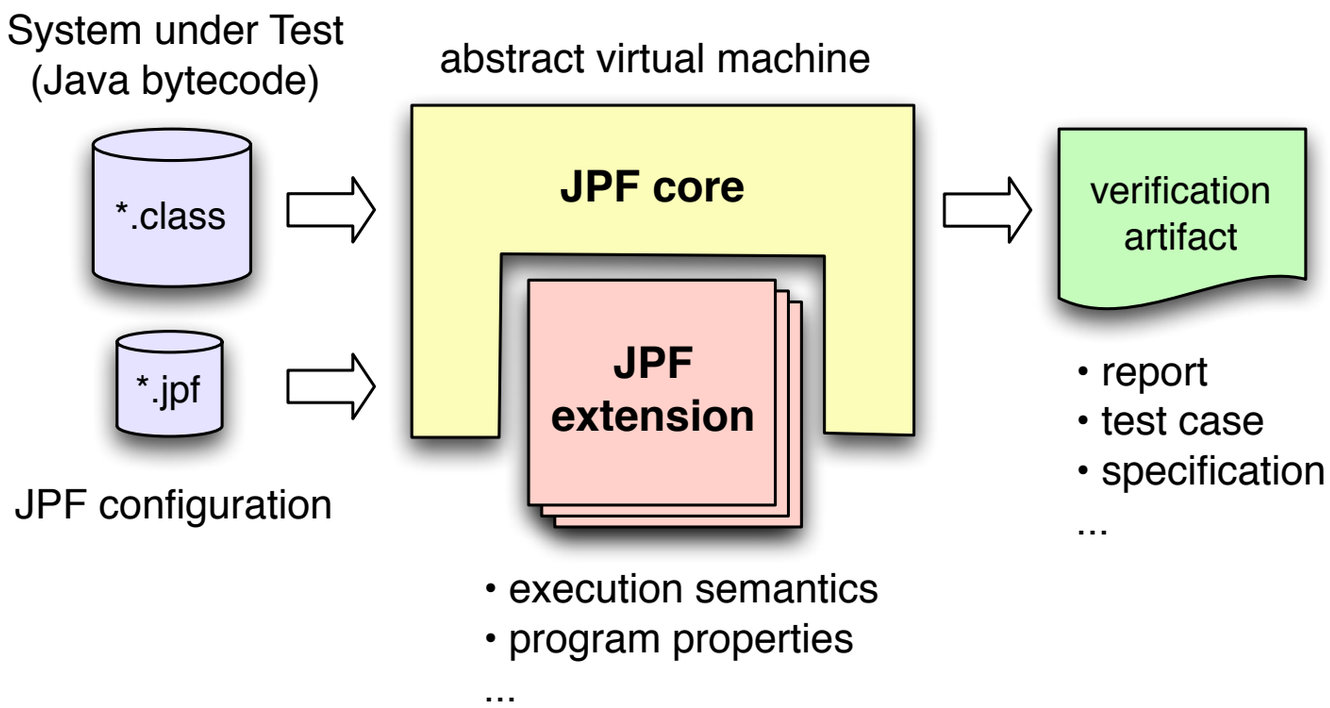
  3h

Tutorial Part 1: Basics

- the inevitable "What is JPF"
- mechanisms that are used in hands-on examples:
  - ChoiceGenerators
  - Listeners
  - Attributes
- JPF configuration
- JPF project layout
- Where to get help

Basics: JPF - What is it?

- a model checker? a virtual machine? ..

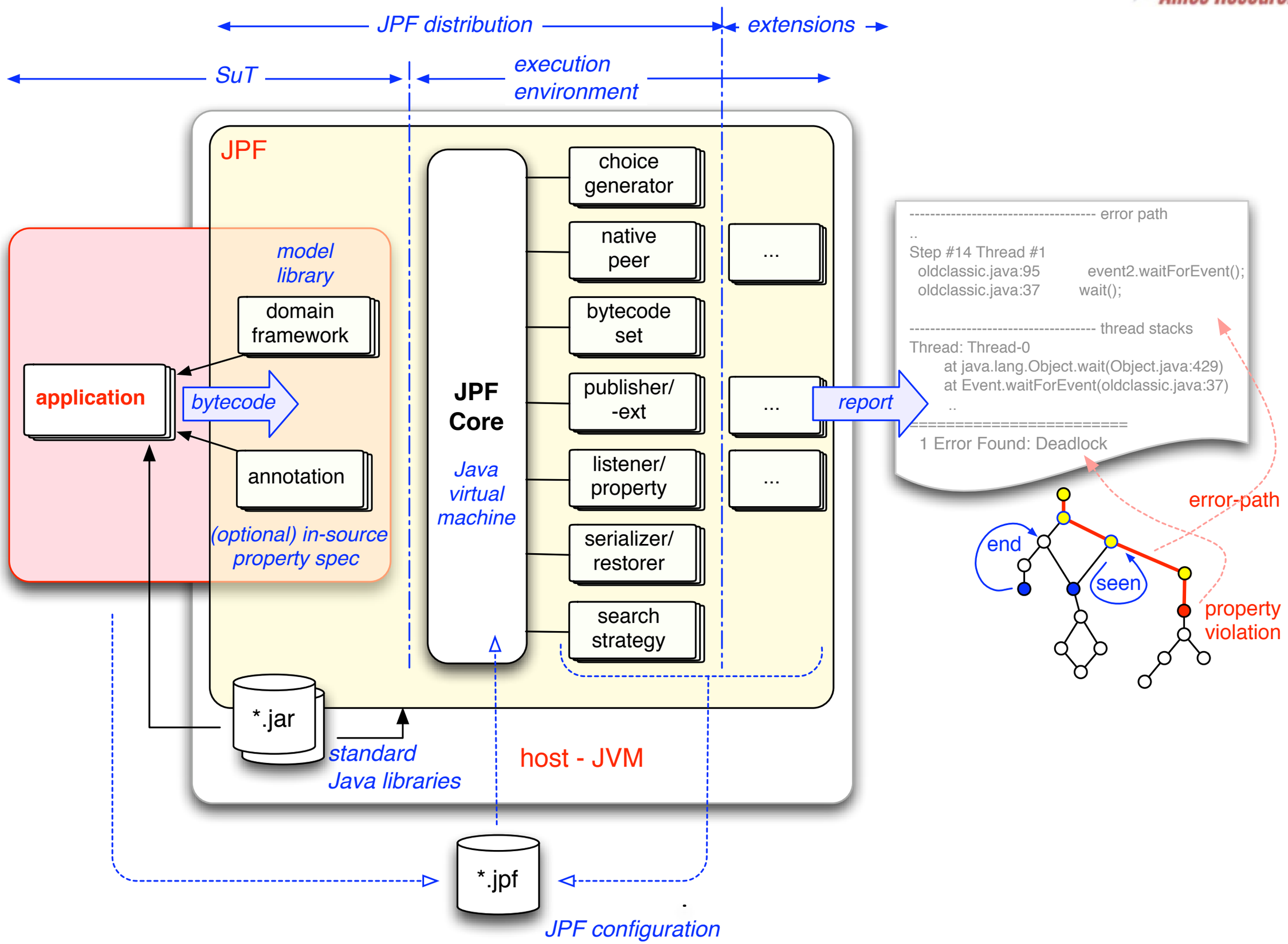


- ..and the answer is: both, and more - it depends on you
- *not* a monolithic, black box tool

> No “one size fits all” - Extensibility is Paramount

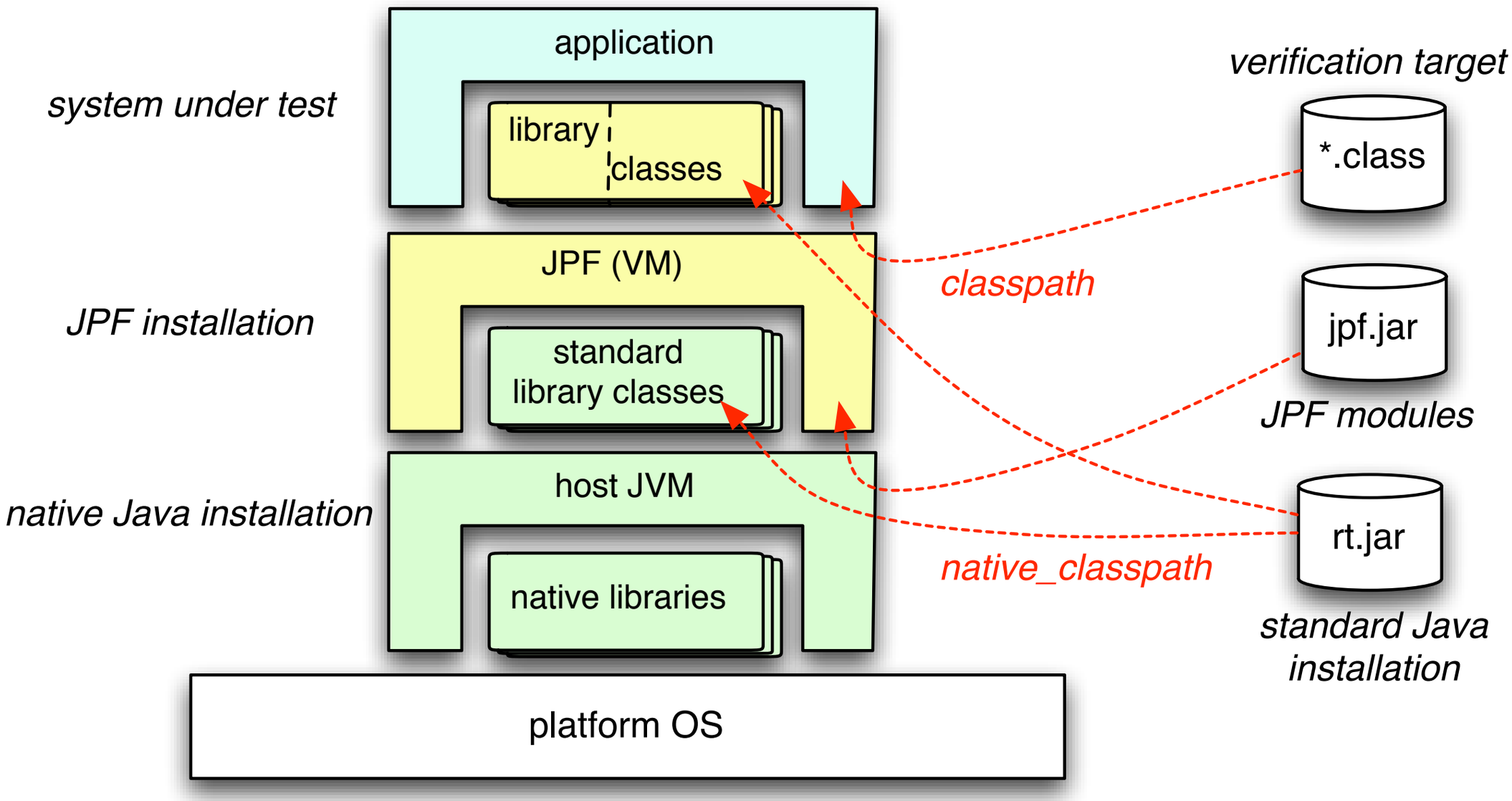
- the quest of today: learn what is in the toolbox to find out how you can adapt JPF to *your* needs

Basics: JPF Components



Basics: a VM running inside a VM

- main stumbling block is recursive nature of JPF
- verified Java program is executed by JPF, which is a virtual machine implemented in Java, i.e. runs on top of a host JVM
- need for two separate classpaths (host VM and JPF)



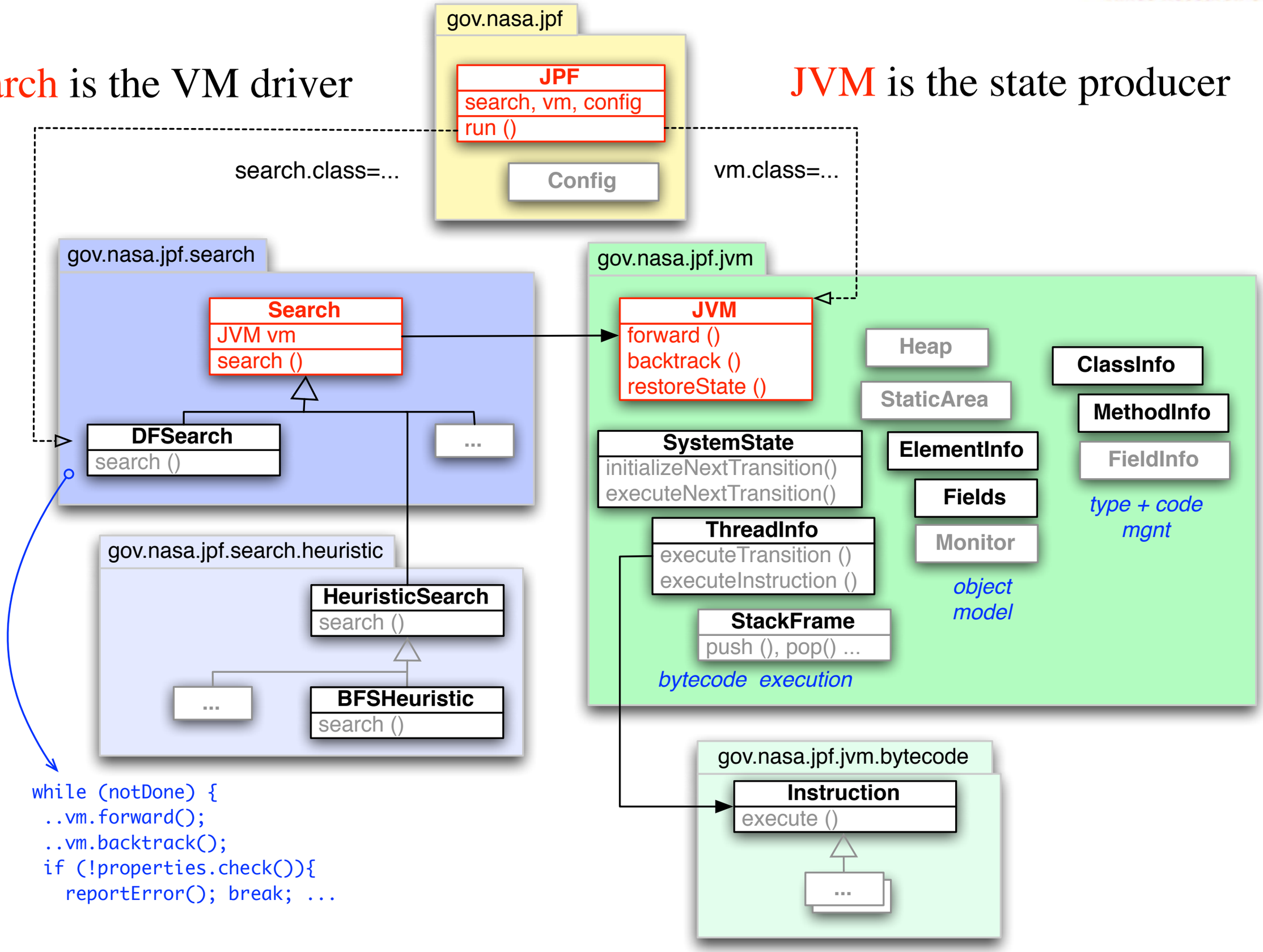
Basics: Concepts and Extension Mechanisms

- Toplevel Design
- ChoiceGenerators - Transitions and States
- VM and Search Listeners - the JPF “plugins”
- Object Attributes - data flow
- Native Peers - Model Java Interface (MJI)
- Instruction Factories - execution semantics
- ... and many more (Serializers, Publishers, PublisherExtensions, ...)

Basics: JPF Toplevel Design

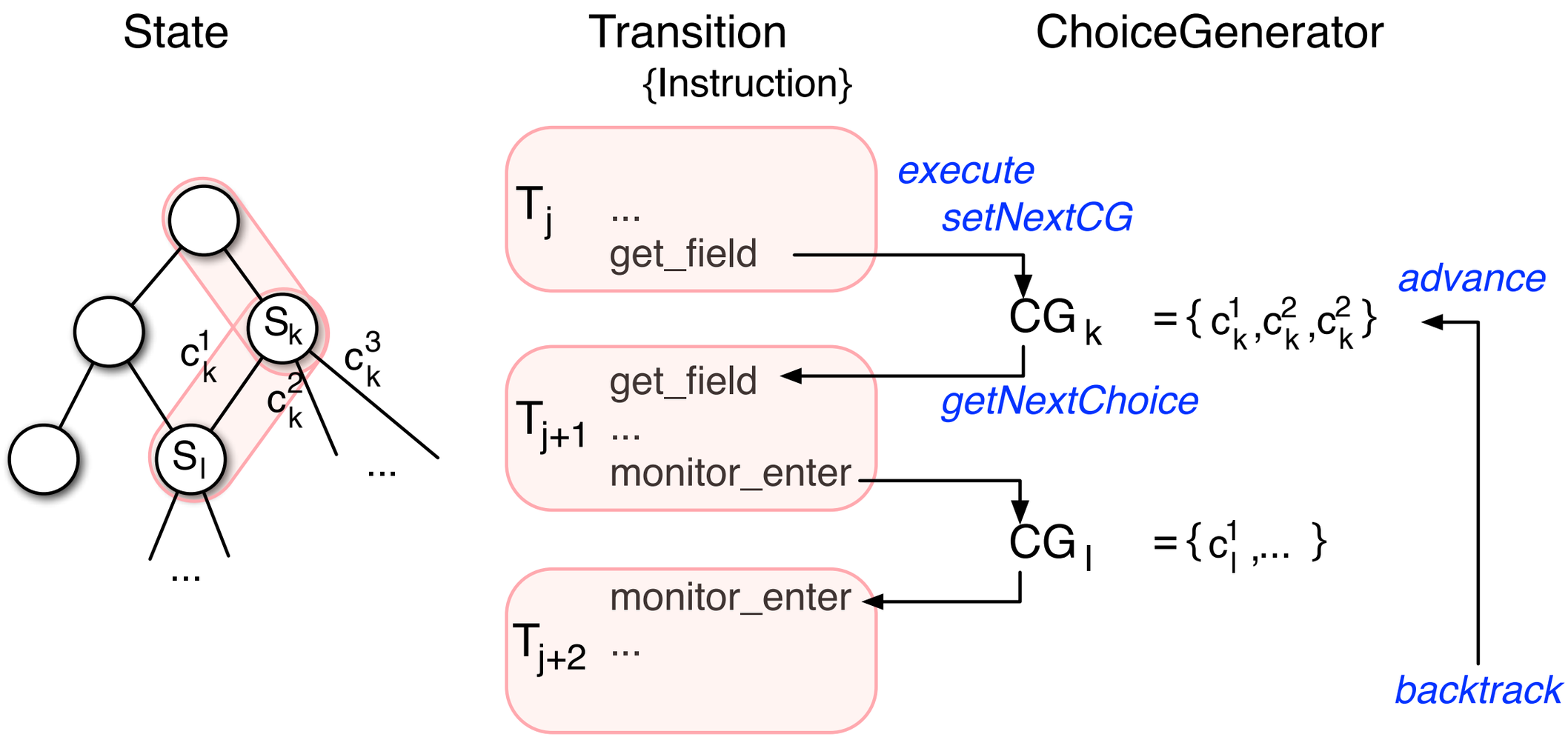
Search is the VM driver

JVM is the state producer



Basics: ChoiceGenerators, States, Transitions

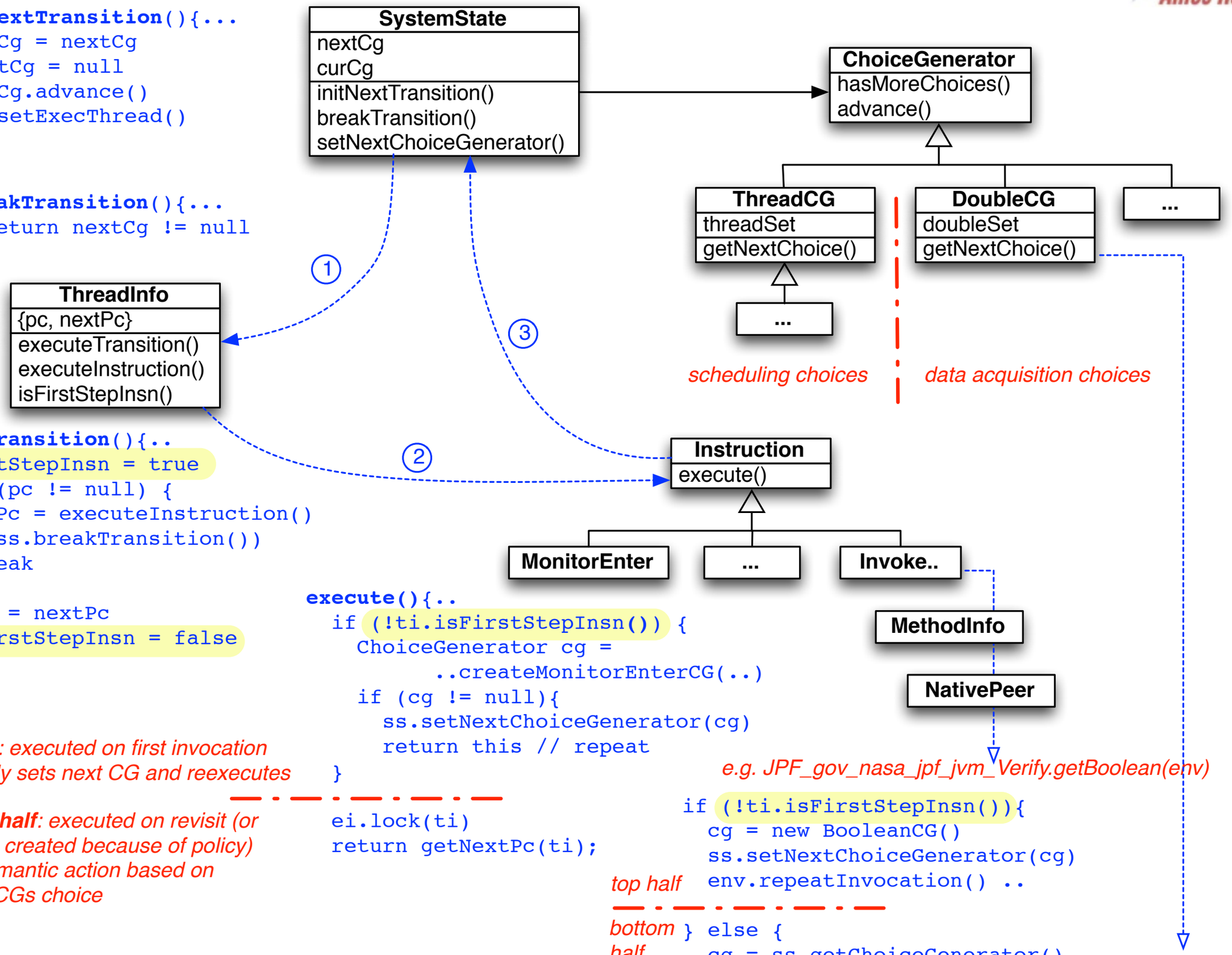
- transitions begin with a choice and extend until the next ChoiceGenerator (CG) is set (by instruction, native peer or listener)
- 'advance' positions the CG on the next unprocessed choice (if any)
- 'backtrack' goes up to the next CG with unprocessed choices



Basics: ChoiceGenerator Implementation

```
initNextTransition(){...
  curCg = nextCg
  nextCg = null
  curCg.advance()
   ..setExecThread()
  ...
```

```
breakTransition(){...
  return nextCg != null
```




```
executeTransition(){..
  isFirstStepInsn = true
  while (pc != null) {
    nextPc = executeInstruction()
    if (ss.breakTransition())
      break
    else
      pc = nextPc
  isFirstStepInsn = false
```

```
execute(){..
  if (!ti.isFirstStepInsn()) {
    ChoiceGenerator cg =
          ..createMonitorEnterCG(..)
    if (cg != null){
      ss.setNextChoiceGenerator(cg)
      return this // repeat
  }
  ei.lock(ti)
  return getNextPc(ti);
```

```
if (!ti.isFirstStepInsn()){
  cg = new BooleanCG()
  ss.setNextChoiceGenerator(cg)
  env.repeatInvocation() ..
} else {
  cg = ss.getChoiceGenerator()
  return ((BooleanCG)cg).getNextChoice()
}
```

top half

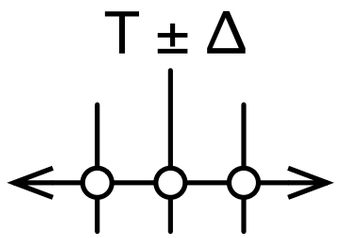
bottom half

***top half**: executed on first invocation optionally sets next CG and reexecutes*

***bottom half**: executed on revisit (or if no CG created because of policy) does semantic action based on current CGs choice*

Basics: ChoiceGenerator Example

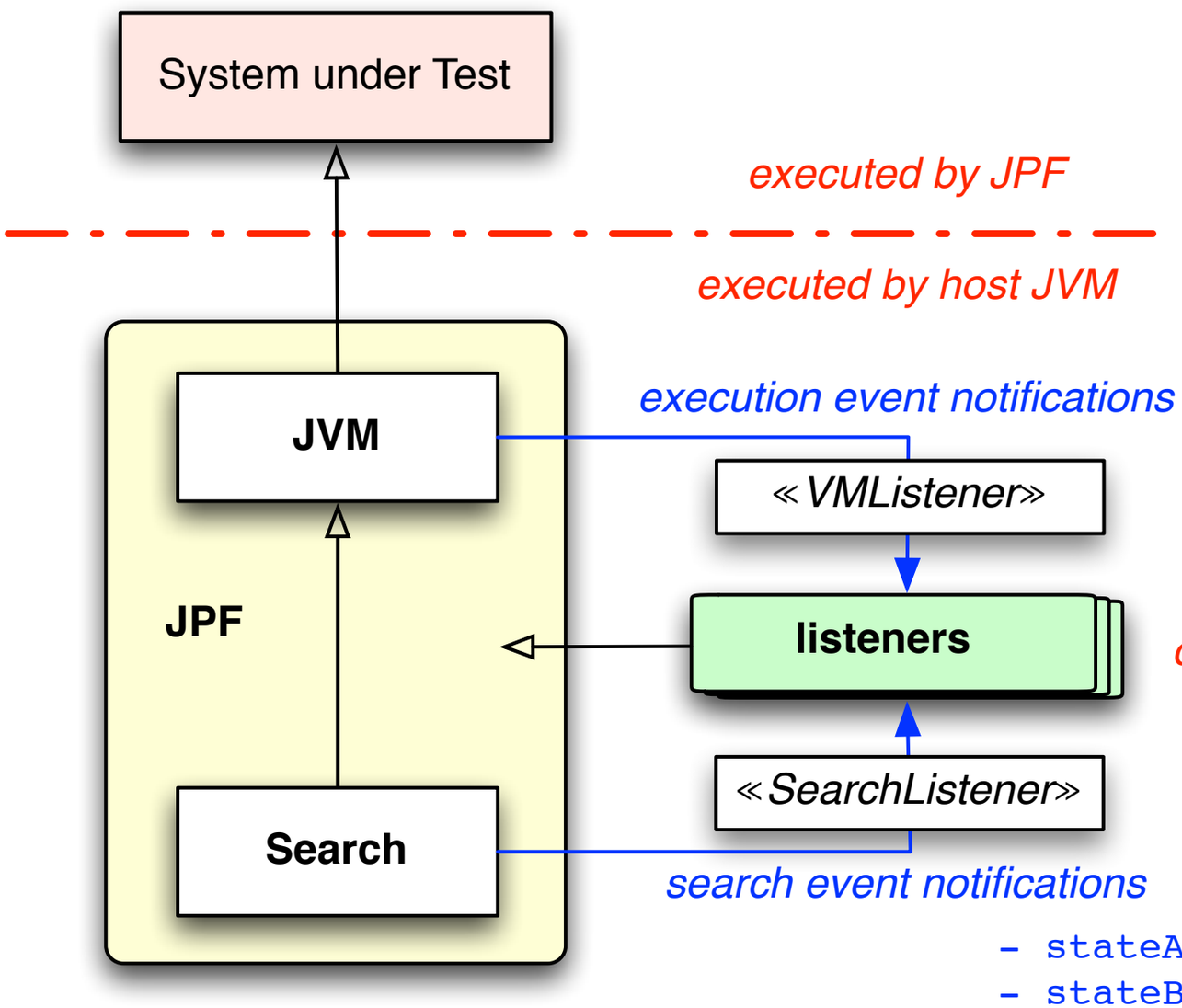
```
public class DoubleThresholdGenerator extends DoubleChoiceGenerator {
  double[] values = new double[3];
  int count;

  public DoubleThresholdGenerator(Config conf, String id) {
    super(id);
    values[0] = conf.getDouble(id + ".low");
    values[1] = conf.getDouble(id + ".threshold");
    values[2] = conf.getDouble(id + ".high");
    count = -1;
  }

  public boolean hasMoreChoices () {
    return !isDone && (count < 2);
  }

  public Double getNextChoice () {
    return (count >+ 0) ? new Double(values[count]) :  new Double(values[0]);
  }

  public void advance () {
    if (count < 2) count++;
  }

  public int getTotalNumberOfChoices () { return 3; }
  public int getProcessedNumberOfChoices () { return count + 1; }
  public void reset () { count = -1; }
```



C = { T-Δ, T, T+ Δ }

Basics: Listeners - the JPF plugins

- executed at host VM level, can access all exported JPF features
- includes Search and JVM events, both high- and low-level
- dynamically configured at runtime



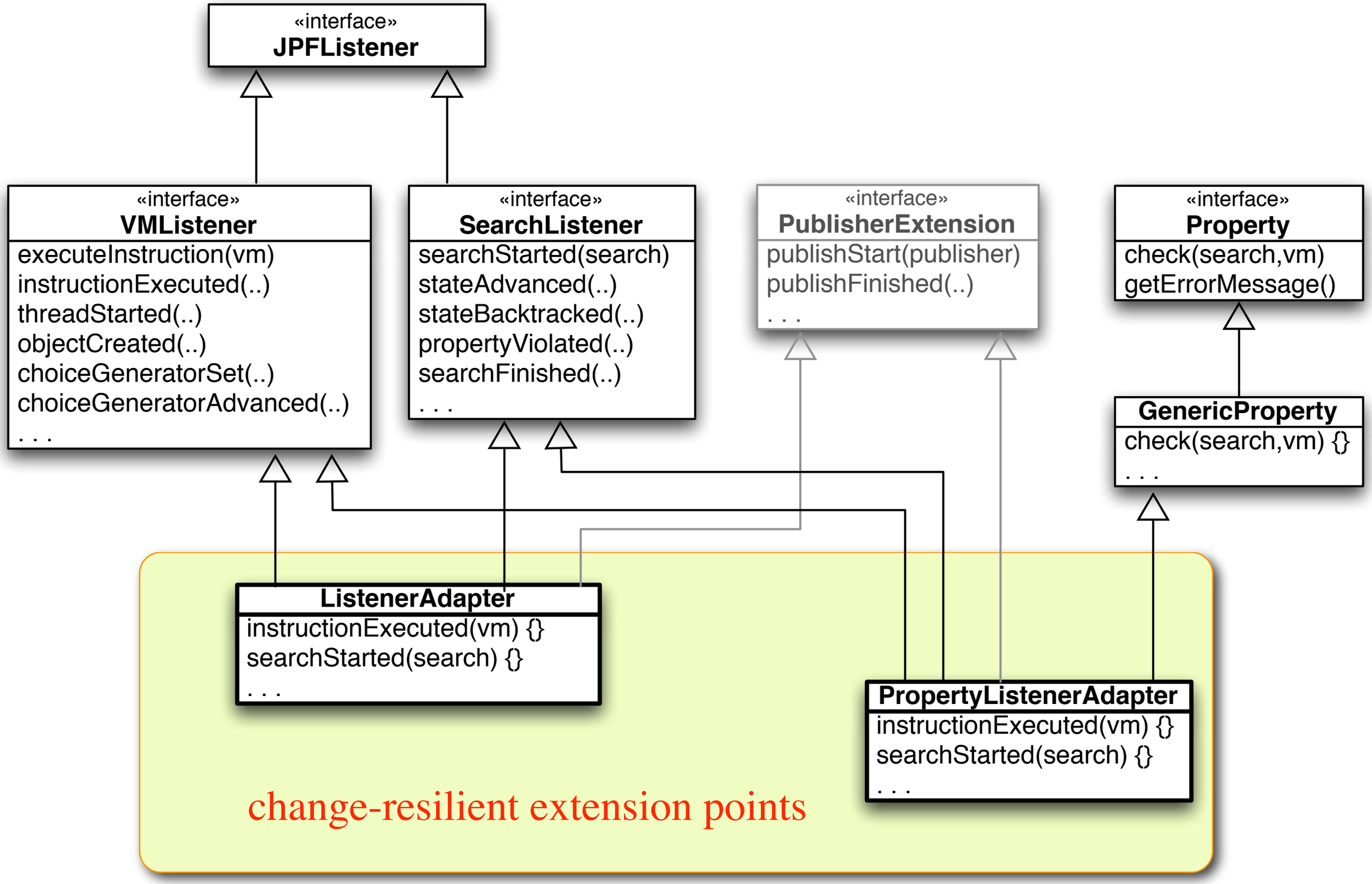

```
- classLoaded (vm)
- threadScheduled (vm)
- threadNotified (vm)
  ...
- executeInstruction (vm)
- instructionExecuted (vm)
  ...
- objectCreated (vm)
  ...
- exceptionThrown(vm)
  ...
- choiceGeneratorAdvanced (vm)
  ...
```

configured

```
- +listener=<listener-class>
- @JPFConfig(..)
- listener.autoload=<annotations>
- jpf.addListener(..)
  ...
```

```
- stateAdvanced (search)
- stateBacktracked (search)
- propertyViolated (search)
- searchFinished (search)
  ...
```

Basics: Listener Implementation



Basics: Listener Example

```
public class NonnullChecker extends ListenerAdapter {
  ...
  public void executeInstruction (JVM vm){

    Instruction insn = vm.getLastInstruction();
    ThreadInfo ti = vm.getLastThreadInfo();

    if (insn instanceof ARETURN) {  // check @NonNull method returns
      ARETURN areturn = (ARETURN)insn;
      MethodInfo mi = insn.getMethodInfo();

      if (areturn.getReturnValue(ti) == null) {
        if (mi.getAnnotation("javax.annotation.Nonnull") != null) {
          Instruction nextPc =
              ti.createAndThrowException("java.lang.AssertionError",
               "null return from @Nonnull method: "+mi.getCompleteName());
          ti.setNextPC(nextPC);
          return;
        }
      }
    ...
```

```
@NonNull
String getId(){
  ...return s;
}
```

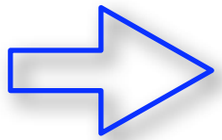
Basics: Listener Patterns

- cascaded `instanceof` bad if many alternatives, use Visitor pattern

```
public class MyListener extends gov.nasa.jpf.ListenerAdapter {
  class InsnDispatcher
     extends gov.nasa.jpf.jvm.bytecode.InstructionVisitorAdapter {
    ...
    void visit (ARETURN insn){
      ThreadInfo ti = vm.getLastThreadInfo();
      .. if (areturn.getReturnValue(ti) ..
    }
  }
  ...
  VM vm;
  InsnDispatcher dispatcher;

  public MyListener(Config conf){ .. dispatcher = new InsnDispatcher(); ..}

  public void executeInstruction (JVM vm){
    this.vm = vm;
    vm.getLastInstruction().accept(dispatcher);
  }
}
```

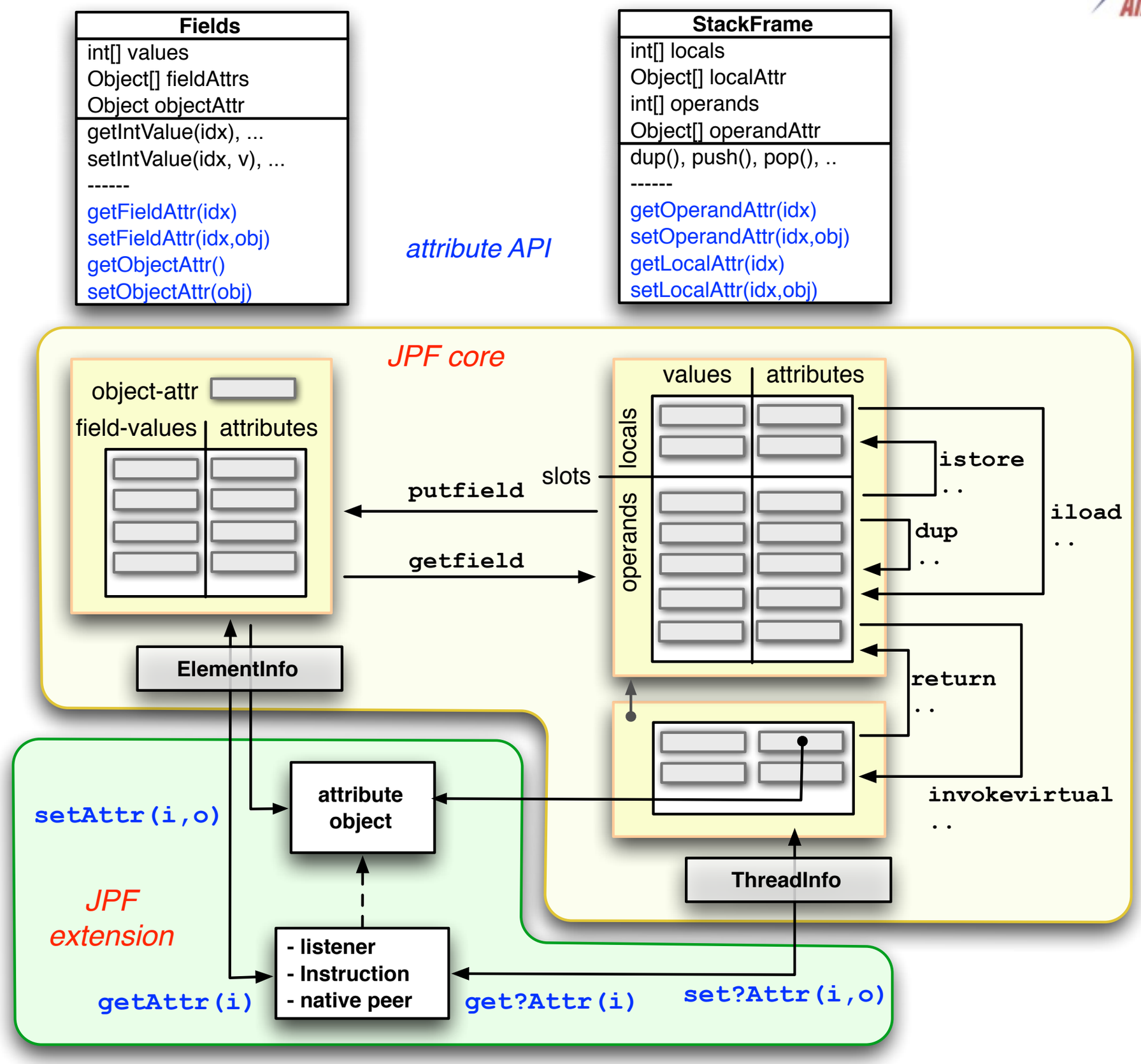
Basics: Object Attributes

- user defined (host-VM) objects that can be attached to JPF values (SUT objects, fields, local vars, but also JPF internals: methods, ..)
- attributes travel automatically with values (stack↔stack, stack↔heap)
- set/processed by extensions (listeners, native peers)
- good for data flow & data quality properties
- supports multiple attributes for objects, fields, operands and local vars
  - ThreadInfo, StackFrame
    - hasOperandAttr(type), hasLocalAttr(idx,type)
    - getOperandAttr(type), getLocalAttr(idx,type)
    - operandAttrIterator(), localAttrIterator(idx)
    - addOperandAttr(attr), addLocalAttr(idx, attr)
    - setOperandAttr(attr), setLocalAttr(idx, attr)
    - ...
  - ElementInfo, Fields
    - hasObjectAttr(type), hasFieldAttr(type)
    - getObjectAttr(type), getFieldAttr(field,type)
    - addObjectAttr(attr), addFieldAttr(field, attr)
    - setObjectAttr(attr), setFieldAttr(field, attr)
    - ...



gov.nasa.jpf.util.ObjectList

Basics: Attribute Implementation



Basics: Attribute Example

```
public class NumericAttr {
  Apfloat value;

  public NumericAttr subtract (NumericAttr x){
    return new NumericAttr( value.subtract( x.value));
  }
  ...

public class DSUB extends gov.nasa.jpf.jvm.bytecode.DSUB {
  ...
  public Instruction execute (SystemState ss, KernelState ks, ThreadInfo ti) {
    NumericAttr a1 = ti.getLongOperandAttr(NumericAttr.class);
    double v1 = Types.longToDouble(ti.longPop());
    NumericAttr a2 = ..
    double v2 = ..
    double r = v2 - v1;
    NumericAttr ar = a2.subtract(a1);
    ...
    if ((error = checkAttr(r,ar)) != null){
      return throwException(ti,error);
    }

    ti.longPush(Types.doubleToLong(r));
    ti.addLongOperandAttrNoClone(ar);

    return getNext(ti);
  }
...
```

Basics: Native Peers

- what to do with (host-VM) native methods?
  - state lives outside of Java (files, network, windows, ..)
  - how can we backtrack?
- solution: intercept native method calls and replace with calls to *NativePeer* methods that are executed by host VM
- association done at class load time
- native peer methods can be used for more than native code:
  - atomic
  - not automatically state tracked
  - can directly interact with JPF (ChoiceGenerator creation etc.)
- can be venerable optimization
- main challenge is to translate between different object models: MJI

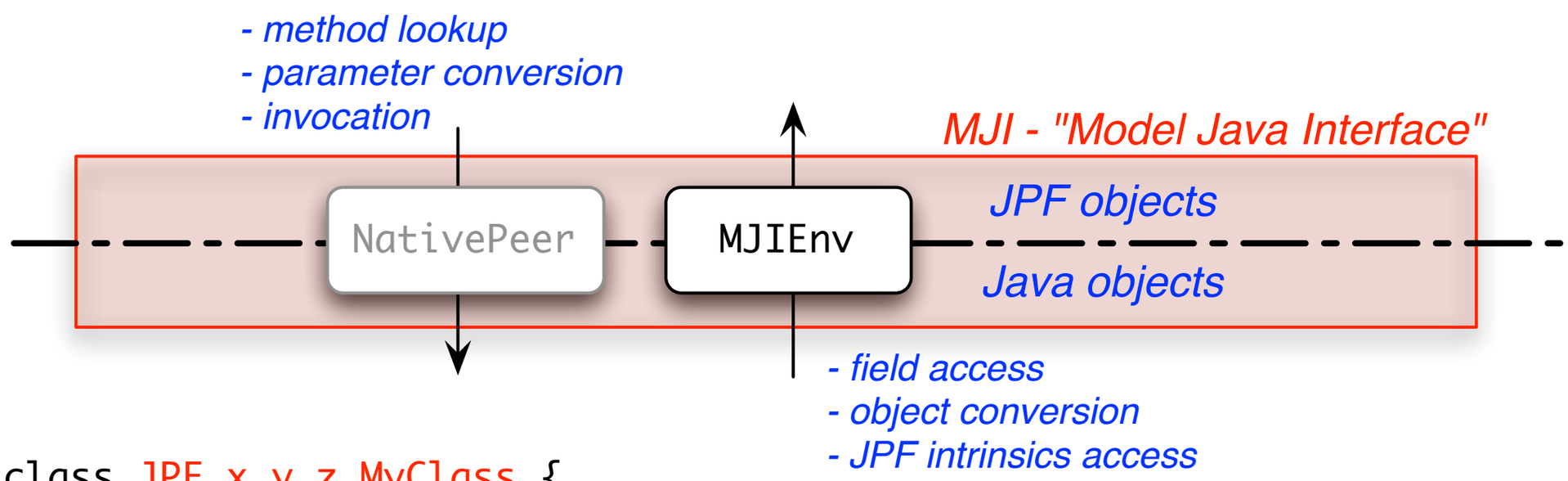
Basics: MJI - Model Java Interface

- transfer from JPF executed code into JVM executed code

"Model" Class

JPF executed

```
package x.y.z;
class MyClass {
  ..
  native String foo (int i, String s);
}
```




```
class JPF_x_y_z_MyClass {
  public static int
      foo__ILjava_lang_String_2__Ljava_lang_String_2 (MJIEnv env, int objRef,
                                                      int i, int sRef) {
    String s = env.getStringObject(sRef);
    ..
    int ref = env.newString(..);
    return ref;
  }
}
```

host VM executed

"NativePeer" Class

Basics: Native Peer Example

```
public class JPF_java_lang_String {
  ...
  public static int indexOf__I__I (MJIEnv env, int objref, int c) {
    int vref = env.getReferenceField(objref, "value");
    int off = env.getIntField(objref, "offset");
    int len = env.getIntField(objref, "count");

    for (int i=0, j=off; i<len; i++, j++){
      if ((int)env.getCharArrayElement(vref, j) == c) return i;
    }
    return -1;
  }

  public static int toCharArray____3C (MJIEnv env, int objref){
    ...
    int cref = env.newCharArray(len);
    for (int i=0, j=off; i<len; i++, j++){
      env.setCharArrayElement(cref, i, env.getCharArrayElement(vref, j));
    }
    return cref;
  }

  public static boolean matches__Ljava_lang_String_2__Z (MJIEnv env, int objRef,
                                                          int regexRef){
    String s = env.getStringObject(objRef);
    String r = env.getStringObject(regexRef);
    return s.matches(r);
  }
```

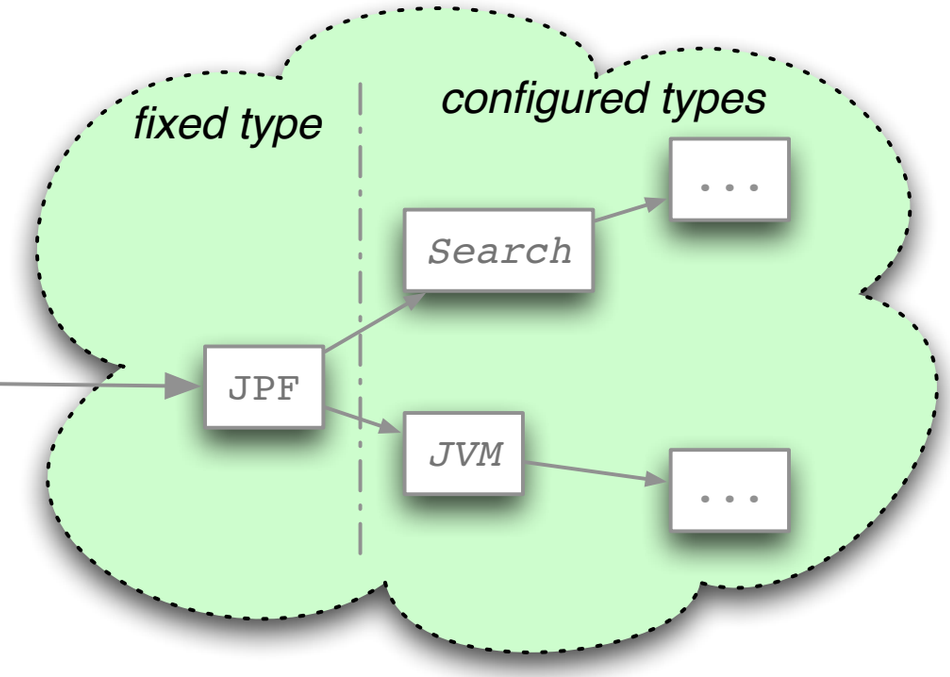
Basics: Configuration (1)

- open - easy to extend, easy to get lost

java.util.Properties instance

Config object

```
..
some_value=42
search.class=..BFSHeuristic
vm.class=..JVM
..
```




```
> bin/jpf +some_value=42  ...
```

'+' command line arguments

```
...
```

```
..
search.class = .search.heuristic.BFSHeuristic
..
```

```
..
vm.class = gov.nasa.jpf.jvm.JVM
search.class = gov.nasa.jpf.search.DFSearch
..
```

hierarchy of Java property files

depends on installed JPF (extension) projects

Basics: Configuration (2)

command line property arguments

```
> bin/jpf [-log][-show] {+log.info=..} .../RobotManager.jpf
```

application properties

```
target = RobotManager
target_args = ...

@using = jpf-aprop
@import = ./my.properties

shell = .shell.basicshell.BasicShell
listener = .aprop.listener.SharedChecker
...
```

jpf.properties in current directory

```
jpf-core = ${config_path}

jpf-core.native_classpath=\
  ${jpf-core}/build/jpf.jar;\
  ...
  ${jpf-core}/lib/bcel.jar;

jpf-core.classpath=\
  build/jpf-classes.jar

jpf-core.test_classpath=\
  build/tests

jpf-core.sourcepath=\
  src/classes
...
```

```
jpf-awt-shell = ${config_path}

@using = jpf-awt

jpf-awt-shell.native_classpath=...
jpf-awt-shell.classpath=...
...
```

site properties

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

jpf-core = ${jpf.home}/jpf-core
jpf-awt = ${jpf.home}/awt
jpf-shell = ${jpf.home}/jpf-shell
jpf-aprop = ...
...
extensions = ${jpf-core},${jpf-shell}
```

all jpf.properties in order of *extensions*

4. command line
   - debugging

3. application properties

   `<project>/.../*.jpf`

   - system-under-test
   - listeners, shells

2. project properties

   `<project>/jpf.properties`

   - project class paths
   - project dependencies

1. site properties

   `~/.jpf/site.properties`

   - project locations
   - pre-loaded projects

Basics: JPF Project Layout

- all JPF projects share uniform directory layout
- use ANT based build system
- use same configuration scheme
- binary distributions are slices of source distributions (interchangeable)
- 3rd party tools & libraries can be included (self-contained)
- projects should have examples and regression test suites
- projects have out-of-the-box IDE configuration (NB,Eclipse)

jpf-proj	***project root directory***
jpf.properties	*JPF project properties file (runtime def: native_classpath, classpath, sourcepath)*
src	***project sources***
main	*host-VM executed classes (listeners, infrastructure etc.)*
peers	*host-VM executed library classes (MJI native peers)*
classes	*JPF executed library classes →sourcepath*
annotations	*JPF processed Java annotations*
tests	*regression tests*
examples	*demos →sourcepath*
build.xml	*Ant build script (compile, build, test, clean)*
build	***build artifacts***
main	*temporary build artifacts (classfiles)*
...	
	permanent build artifacts
jpf-proj.jar	*host-VM executed jar (main,peers →native_classpath)*
jpf-proj-classes.jar	*JPF executed library jar (classes,annotations →classpath)*
jpf-proj-annotations.jar	*separate anotations jar (for JPF external SUT exec)*
lib	***required runtime jars*** *→native_classpath*
tools	***build & run tools***
RunJPF.jar	*JPF startup jar (executable)*
RunTest.jar	*JPF testing jar (executable)*
RunAnt.jar	*JPF build jar (executable)*
bin	***scripts***
jpf	*JPF startup script (→RunJPF.jar)*
test	*JPF build script (→RunTest.jar)*
ant	*JPF build script (→RunAnt.jar)*

...

Basics: Where to get Help - Mailing List

- http://groups.google.com/group/java-pathfinder
- anyone can join & read, member only post
- moderate volume (~50 messages/month)
- subscribe on http://groups.google.com/group/java-pathfinder/subscribe

Basics: Where to get Help - Wiki

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

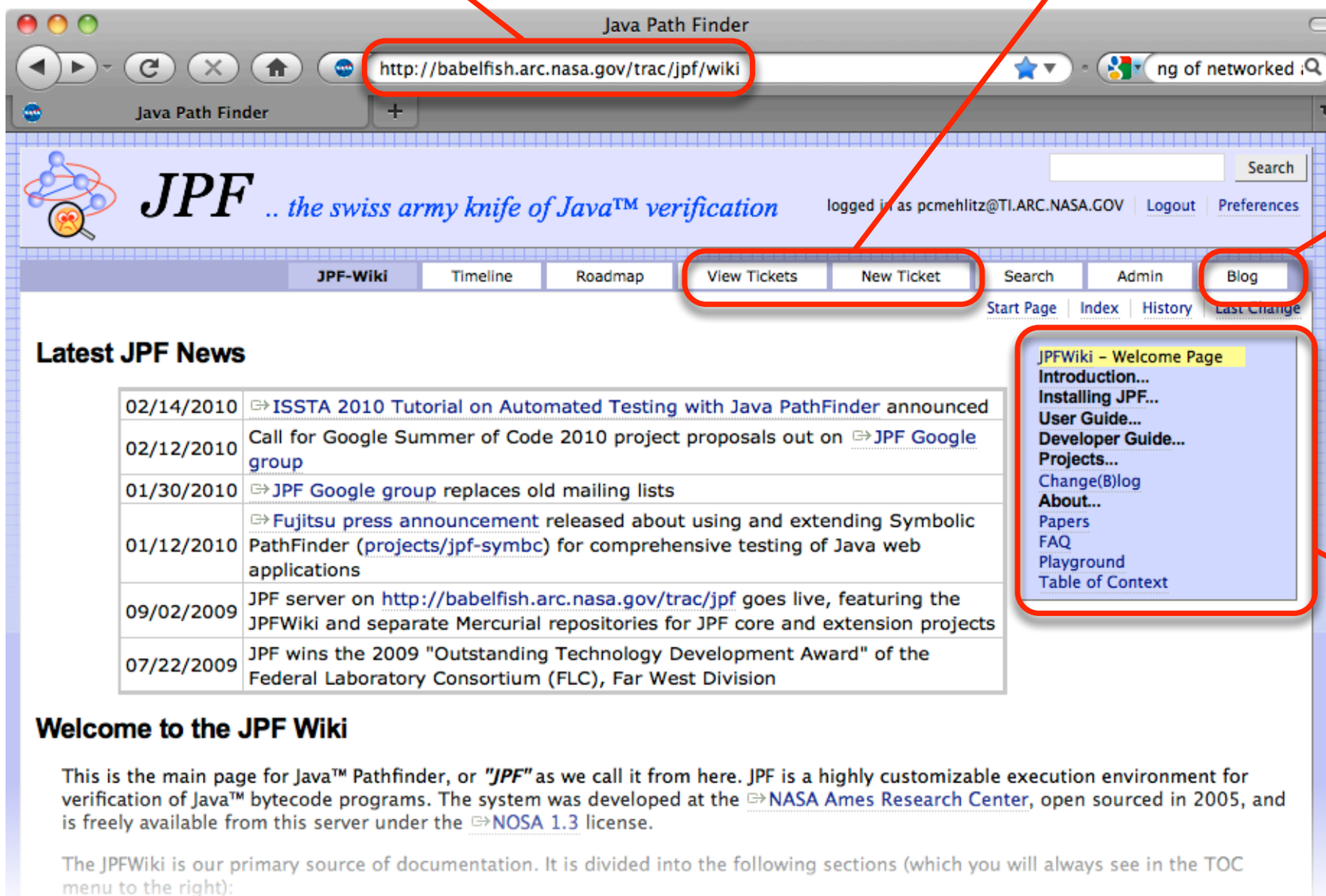
- public read access
- edit for account holders (also non-NASA)

bug tracking

- Trac ticket system

project blog

- announcements
- important changes



hierarchical navigation menu

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

Tutorial Part 2: Install & Run JPF

- install (all JPF projects have same procedure)
  - start with `jpf-core`
    - download
    - build
    - test
    - configure
  - repeat for JPF extension projects (`jpf-symbc, jpf-numeric, ...`)
  - for available projects, see “Projects” navigation bar on wiki

- running JPF
  - several alternative ways to run JPF
    - `java` from command line
    - using JPF scripts / Run.. jars
    - using IDEs (Eclipse, NetBeans)
  - examples

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
    - MercurialEclipse update site: http://cbes.javaforge.com/update
  - NetBeans: http://www.netbeans.org

Install: Getting 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)`
- (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

- get & build jpf-core sources
- run from within jpf-core directory
  `bin/jpf -addproject`
- this will create a ${user.home}/.jpf/site.properties file
- repeat same procedure for JPF extension projects
  - download
  - build
  - run `bin/jpf -addproject [init]` from within project root directory
- .. or just edit ~/.jpf/site.properties manually:

project name

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

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

project path

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 [+<jpf-option>=..] MySUT-verify.jpf`
  - using scripts **`>bin/jpf ... MySUT-verify.jpf`**

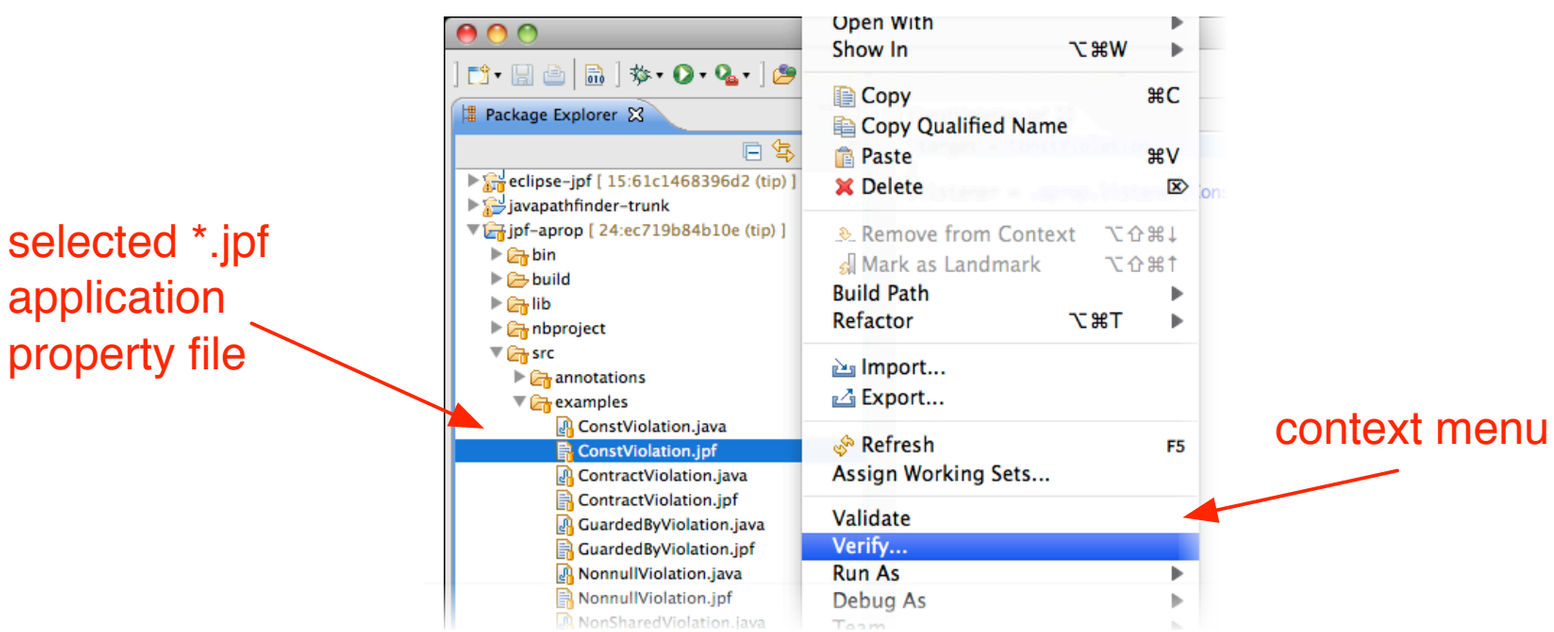
- running JPF from within JUnit
- running JPF from your program (tools using JPF)

- 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 Tutorial: Running from Eclipse

debugging

- 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
- 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



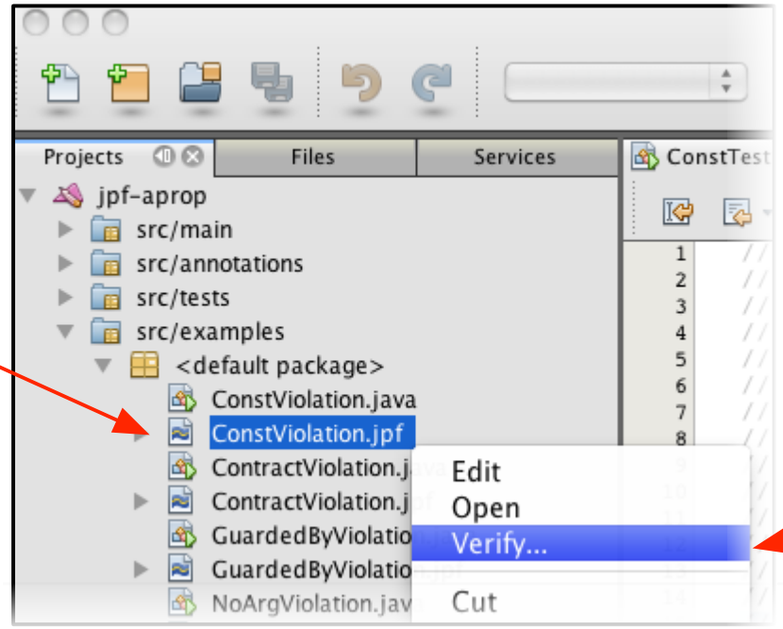
JPF Tutorial: Running from NetBeans

- 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

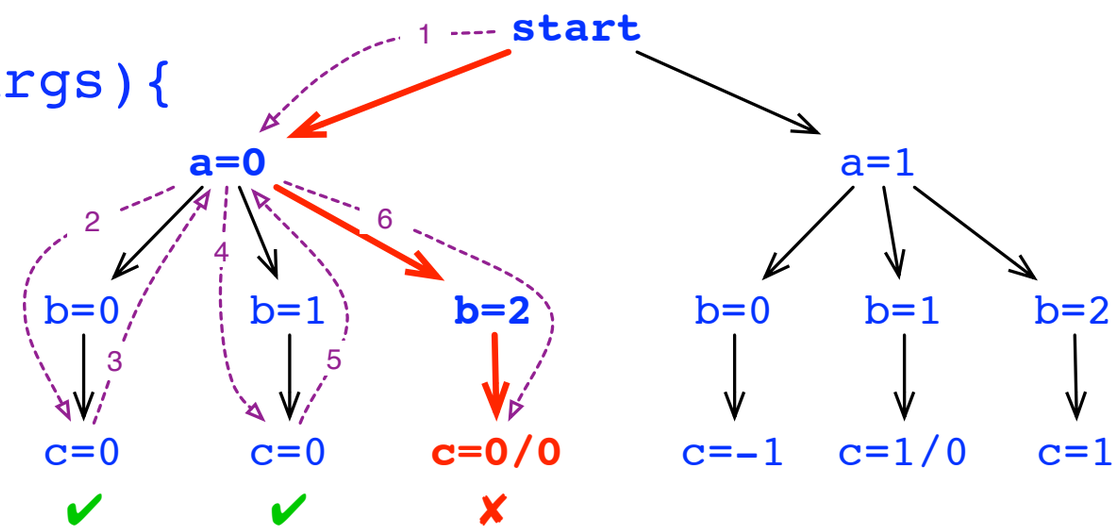


context menu

Examples: Random Data

- src/examples/Rand.java

```
public static void main(String[] args){
    Random random = new Random();
    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)
```

Examples: Data Race (1)

- 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
```

Examples: Data Race (1)

- 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
```

Examples: Data Race (2)

- 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)
...
```

Examples: Examine Execution with Listener

- 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
...
```

Examples: Numeric Extension (1)

- get jpf-numeric extension

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

- build it

  `bin/ant`

- run example (from inside jpf-numeric dir)

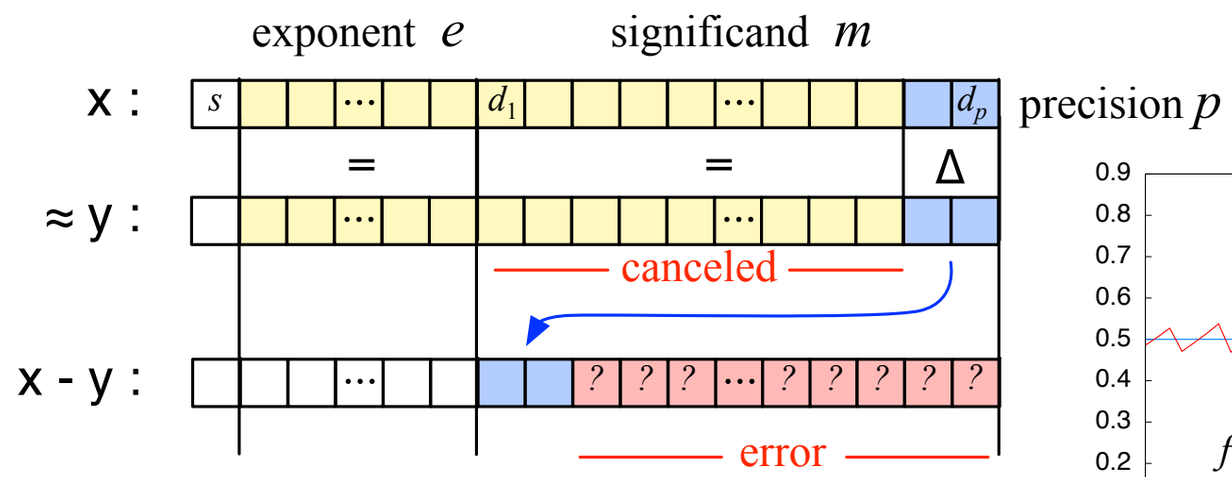
  `bin/jpf src/examples/CatastrophicCancellation.jpf`

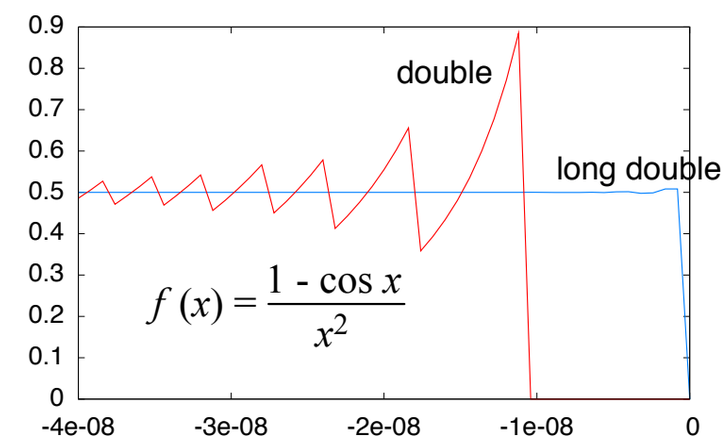
- try to find this with testing..

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

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

Examples: Numeric Extension (2)

- errors and required inspection can be a lot more intricate than NPEs
- e.g.: amplification of previous operand errors due to cancellation of identical leading bits in significands





Annie Cuyt et al. "A remarkable example of catastrophic cancellation unraveled"

```
double a = 77617.0;
double b = 33096.0;
res = 333.75*pow(b,6) + pow(a,2)*(11*pow(a,2)*pow(b,2) -
                    pow(b,6) - 121*pow(b,4) - 2) + 5.5*pow(b,8) + a/(2*b);
```

.numeric.bytecode.InstructionFactory

```
res=-1.1805916207174113e21 (should be -0.827396...)
...
========================================================= error #1
gov.nasa.jpf.jvm.NoUncaughtExceptionsProperty
        java.lang.ArithmeticException: cancellation of:
              -7.917111340668963E36 + 7.917111340668962E36
                    = -1.1805916207174113E21
```

jpf-numeric/src/examples/CatastrophicCancellation.jpf

Examples: Numeric Extension (3)

- InstructionFactory with specialized FP instructions

```
package gov.nasa.jpf.numeric.bytecode;

public class NumericInstructionFactory extends InstructionFactory {
  ...
  @Override public Instruction dsub() { return new DSUB(); }
}

public class DSUB extends Instruction {...
  @Override public Instruction execute (ThreadInfo ti) {
    long l1 = ti.longPop();
    double d1 = Double.longBitsToDouble(l1);

    long l2 = ti.longPop();
    double d2 = Double.longBitsToDouble(l1);

    double r = d2 - d1;
    ti.longPush(Double.doubleToLongBits(r);

    if (checkCancellation(l1,l2)){
      reportCancellation(l1,l2);
    }

    return getNextInsn(ti);
  }
```

Examples: MultiThreaded GUI (1)

- get jpf-awt extension

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

- build it

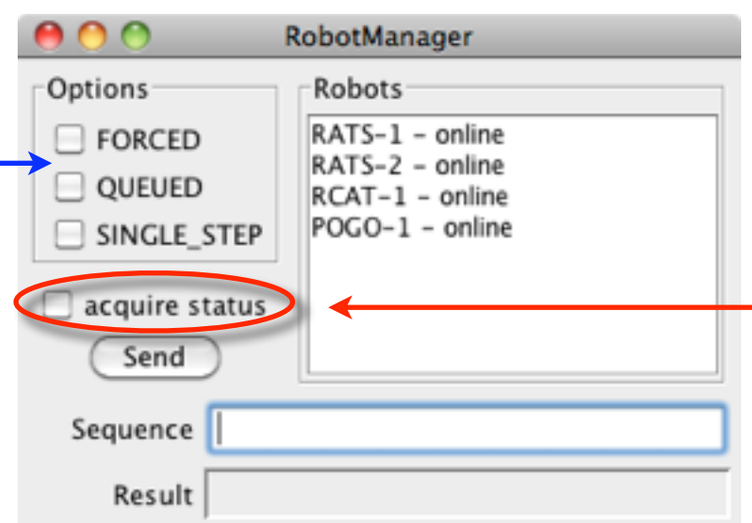
  `bin/ant`

- run example

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

- GUI application that uses concurrent data acquisition:

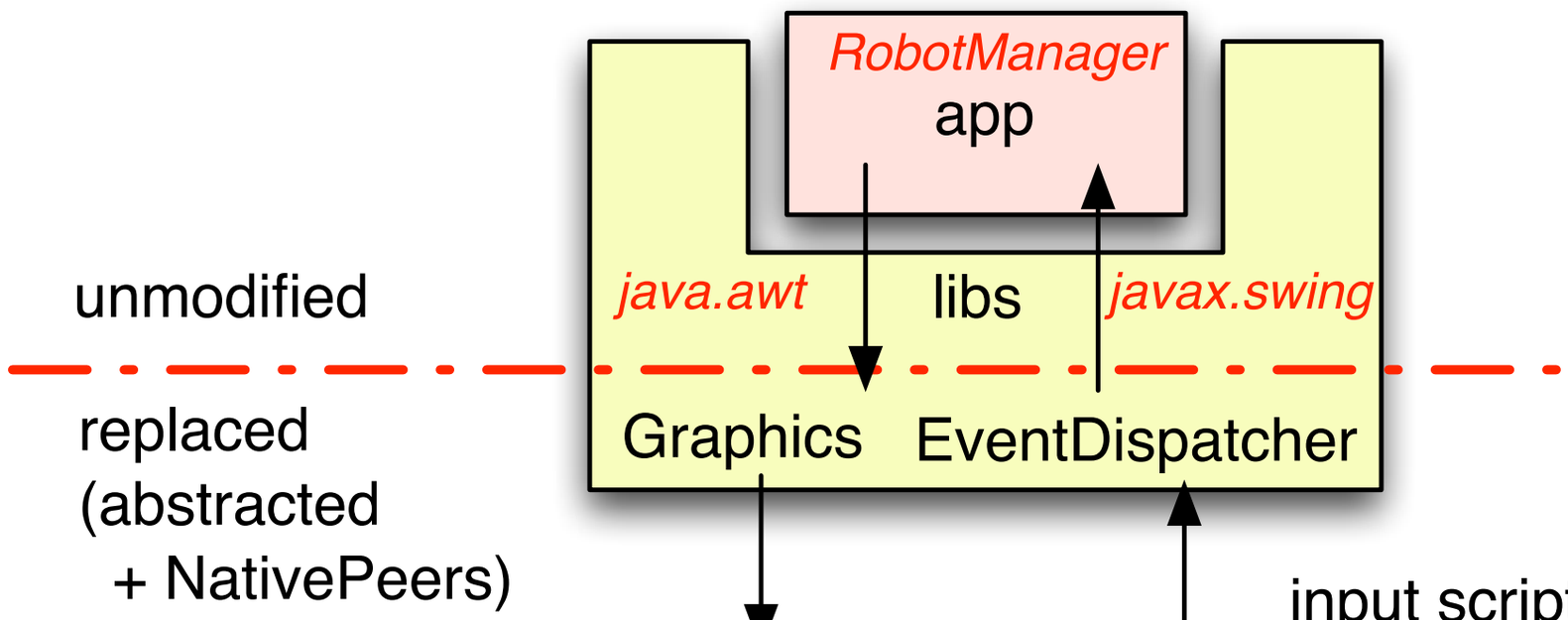
Thread 1:
user input

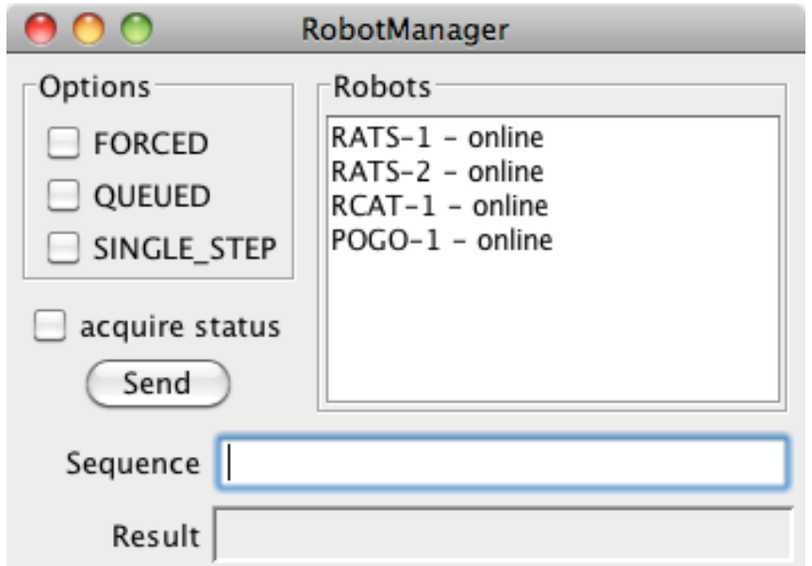


Thread 2:
data acquisition

Examples: GUI Model Checking (2)

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





input script (user model)

```
ANY { $<FORCED|QUEUED>.doClick() }
$Send.doClick()
...
```

input event generator

```
FORCED.doClick()
Send.doClick()
...
```

```
QUEUED.doClick()
Send.doClick()
...
```

...

Examples: GUI Model Checking (3)

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

JPF Shell – RobotManager-thread.jpf

Properties | Report | Test Output | Verify Output | Config View

Showing Streams: ☑ STDOUT ☑ STDERR ☑ STDIN

```
------------------------------------------------------ transition #35 thread: 3
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet[id="sharedField",isCascaded:false,{AWT-Event
  RobotManager.java:44            : return id;
  RobotManager.java:246           : onlineRobots.remove(robot.getId());
      [98 insn w/o sources]
  RobotManager.java:246           : onlineRobots.remove(robot.getId());
  RobotManager.java:248           : listModel.changed(robot);
------------------------------------------------------ transition #36 thread: 2
gov.nasa.jpf.jvm.choice.ThreadChoiceFromSet[id="sharedField",isCascaded:false,{>AWT-Even
  RobotManager.java:260           : return onlineRobots.get(robotName);
      [47 insn w/o sources]
  RobotManager.java:260           : return onlineRobots.get(robotName);
  RobotManager.java:566           : Robot robot = model.getOnlineRobot(selectedRobotName)
  RobotManager.java:567           : String result = model.sendSequence(robot, sequence);
  RobotManager.java:269           : return robot.processSequence(sequence);

====================================================== statistics
elapsed time:       0:00:03
states:             new=638, visited=207, backtracked=808, end=0
search:             maxDepth=68, constraints hit=0
choice generators:  thread=41 (signal=0, lock=10, shared ref=28), data=672
heap:               new=11358, released=5263, max live=3333, gc-cycles=843
instructions:       879891
max memory:         81MB
loaded code:        classes=404, methods=4959
```

Type here to send text to the program's STDIN

~1500 insn/transition,
error at depth 36
⇒ trace too long

Examples: GUI Model Checking (4)

- error 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)
         ...
```

- generic trace analysis listener can help
- still too time consuming in many cases

run with additional option:
+listener=.listener.OverlappingMethodAnalyzer

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

thread 1

thread 2

Examples: GUI Model Checking (5)

- better approach - get specific properties from SUT annotations
- more efficient to check
- **self explaining** - no need for further analysis

```
// RobotManager instances are not thread-safe, don't use them concurrently
@NonShared
public class RobotManager {
  ...
```

add to .. RobotManager-thread.jpf:

@using jpf-aprop

listener=.aprop.listener.SharedChecker

```
========================================================= error #1
gov.nasa.jpf.jvm.NoUncaughtExceptionsProperty
java.lang.AssertionError: NonShared object: RobotManager@13d
          accessed in live thread cycle: AWT-EventQueue-0,Thread-2,AWT-EventQueue-0,main
    at RobotManager$ListModel.getSize(RobotManager.java:173)
    at javax.swing.JList.setSelectedIndex(JList.java:280)
    at java.awt.EventDispatchThread.processScriptAction(JPF_java_awt_EventDispatchThread)
    at java.awt.EventDispatchThread.run(EventDispatchThread.java:65)

========================================================= statistics
elapsed time:       0:00:02
states:             new=531, visited=0, backtracked=512, end=0
search:             maxDepth=19, constraints hit=0
choice generators:  thread=13 (signal=0, lock=8, shared ref=2), data=604
```

Running JPF Revisited: The Shell (1)

- sometimes JPF reports too long for IDE Output view or console
- some applications / properties might require specific visualization
- solution: *JPF shell*
- get & build + install jpf-shell extension (plus jpf-awt-shell for example)
- specify shell class and views in your application property file (*.jpf)
- launch normally through "Verify" context menu
- JPF shell shows as separate window
- sources can be shown in IDE editors by clicking links in JPF shell window

```
# JPF application property file using generic JPF shell
target=oldclassic
...
shell=.shell.basicshell.BasicShell
shell.panels+=,searchgraph
...
```

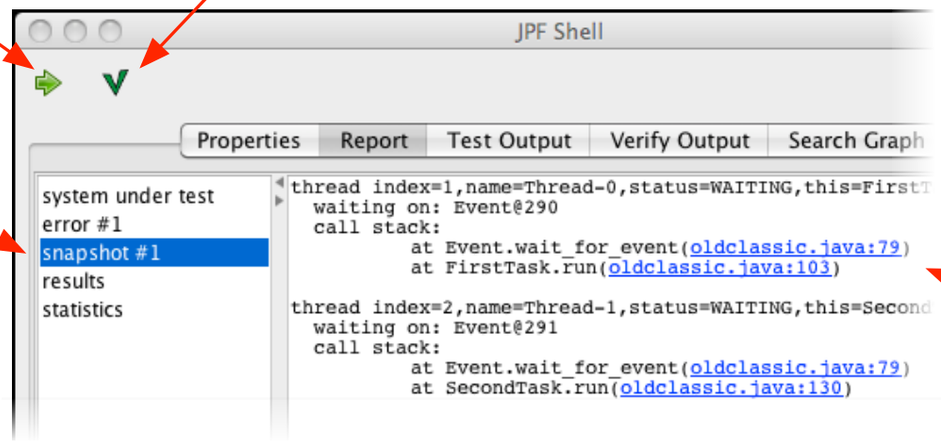
Running JPF Revisited: The Shell (2)

browse complex reports

run SUT (test)

run JPF (verify)

browse
report topics

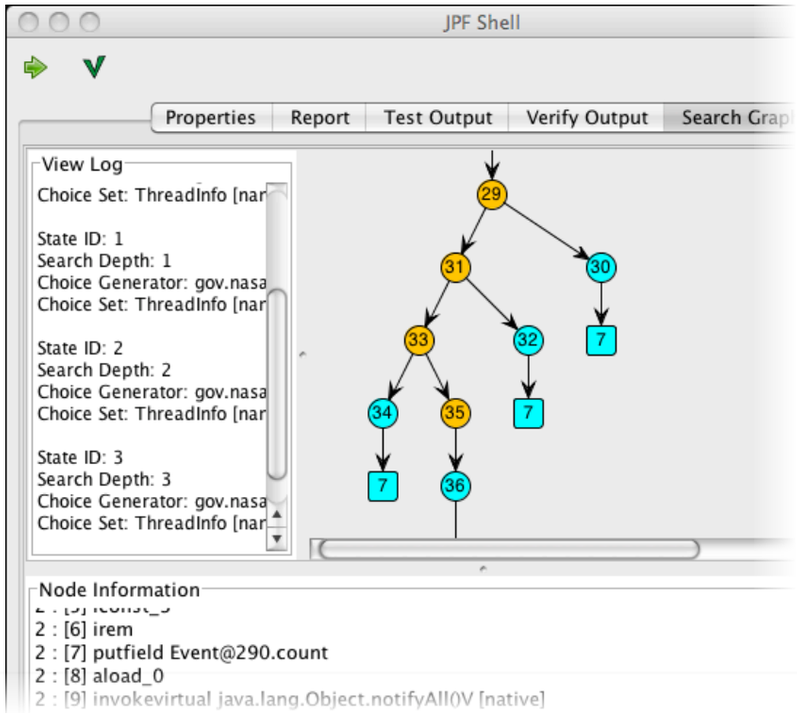


configurable
view tabs

```
shell=.shell.basicshell.BasicShell
shell.panels=...
```

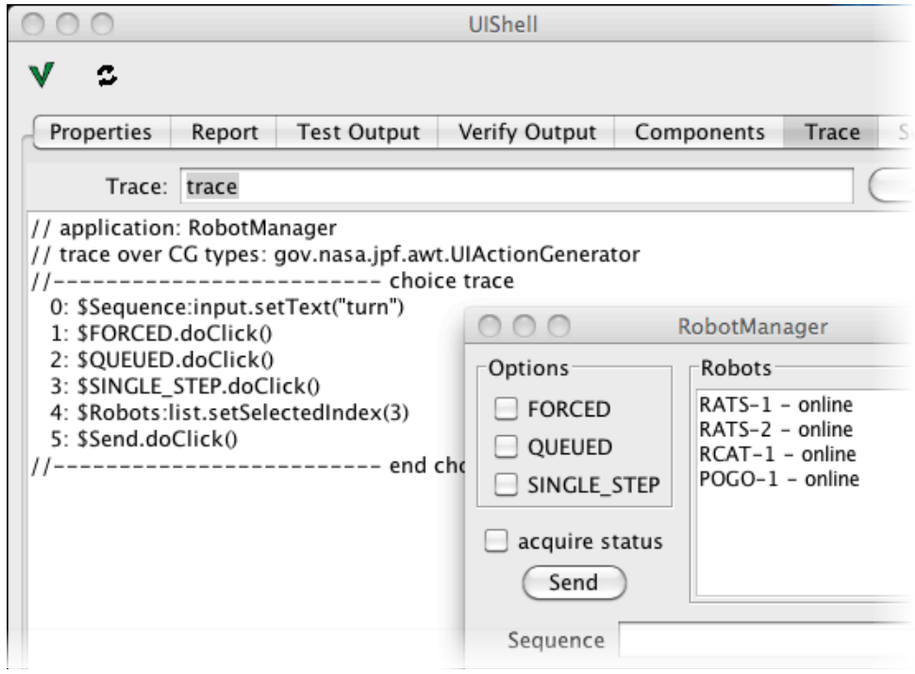
clickable
source links

inspect JPF run




```
.. shell.panels+=,searchgraph
```

application/domain specific views




```
shell=.shell.awt.UIShell
shell.panels+=,component,trace,script
```

Tutorial Part 3: Extending JPF

- hands on exercise - your task is to program JPF extensions
- implemented as JPF extension project
- Task 1: safety property listener
- Task 2: control flow variation via listener and data attributes

- sources on http://babelfish.arc.nasa.gov/trac/jpf/raw-attachment/wiki/presentations/start/jpf-tutorial-r0-src.zip

Exercises: Create Tutorial Project

- create your sandbox test project
- get & build jpf-template:
  `hg clone http://babelfish.arc.nasa.gov/hg/jpf/jpf-template`
- create project
  ```
  bin/create_project ../jpf-tutorial
  cd ../jpf-tutorial
  ```
- ..or use Eclipse-JPF wizard: New/Other.../JPF/JPF Project

Exercises: Task 1

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

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

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

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

Exercises: Task 1 - Create Test

- create example/test: src/examples/RangeCheckerTest.java

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

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

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

Exercises: Task 1 - Listener (1)

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

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

- initialize from Config

```
FieldSpec fieldSpec;
int min, max;

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);
```

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

Exercises: Task 1 - Listener (2)

- 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();
```

Exercises: Task 1 - Listener (3)

- 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);
}
```

Exercises: Task 1 - Listener (4)

- 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; }
```

Exercises: Task 1 - Listener Sources

```
package tutorial;

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;
  }
}
```

Exercises: Task 1 - Running

- create src/examples/RangeCheckerTest.jpf

```
target = RangeCheckerTest
listener=tutorial.RangeChecker
rc.field=RangeCheckerTest.data
rc.min=0
rc.max=42
```

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

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

- Congratulations - Your First Listener!

Exercises: Task 1 - Reference

- see NumericValueChecker listener in jpf-core
- sources in src/main/gov/nasa/jpf/listener/NumericValueChecker.java
- example in src/examples/NumericValueCheck.{java,jpf}
- run with

```
bin/jpf src/examples/NumericValueCheck.jpf
```

- also handles local variables and different types

Exercises: Task 2

- write a listener that detects time based comparisons and creates a ChoiceGenerator that executes both branches

```
long t1 = System.currentTimeMillis();
..
long t2 = System.currentTimeMillis();
if (t2 - t1 > MAX_TIME) {
  ①  println("disaster");
} else {
  ②  println("all fine");
}
```

Exercises: Task 2 - Bytecode Pattern

- approach hinges on bytecode pattern

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

obtain time values

compute duration

compare duration

```
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 MAX_TIME
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
```

Exercises: Task 2 - Create Test

- create example/test: src/examples/StopWatchTest.java

```
public class StopWatchTest {

  public static void main(String[] args){
    long tStart = System.currentTimeMillis();
    System.out.println("some lengthy computation..");
    long tEnd = System.currentTimeMillis();

    if (tEnd - tStart > 1000){
      throw new RuntimeException("it took too long..");
    }

    System.out.println("all fine, finished in time");
  }
```

Exercises: Task 2 - Approach

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

```
public class StopWatch
    extends gov.nasa.jpf.ListenerAdapter
```

- use JPF *value attribute objects* to identify time values and durations
- use JPF *ChoiceGenerator* to force control flow

```
return (invokestatic) native System.currentTimeMillis()
...
```
instructionExecuted() → create & set TimeVal operand attribute

```
lsub
...
```
executeInstruction() → propagate TimeVal attr to result operand

```
lcmp
ifle ..
...
```
executeInstruction() → use CG to push result operand

Exercises: Task 2 - Listener (1)

- instructionExecuted() → create & set TimeVal operand attribute

```
static class TimeVal {}
static TimeVal timeValAttr = new TimeVal();
...
@Override
public void instructionExecuted (JVM vm){
  Instruction insn = vm.getLastInstruction();

  if (insn instanceof NATIVERETURN){
    MethodInfo mi = ((NATIVERETURN)insn).getMethodInfo();
    if (mi.getUniqueName().equals("currentTimeMillis()J")){
      ThreadInfo ti = vm.getLastThreadInfo();
      ti.addLongOperandAttr( timeValAttr);
...
```

Exercises: Task 2 - Listener (2)

- executeInstruction() → propagate TimeVal attribute to result operand

```
...
@Override
public void executeInstruction (JVM vm){
  Instruction insn = vm.getLastInstruction();

  if (insn instanceof LSUB){
    ThreadInfo ti = vm.getLastThreadInfo();

    if (ti.hasOperandAttr(1, TimeVal.class) ||
        ti.hasOperandAttr(3, TimeVal.class)){
      long v2 = ti.longPop();
      long v1 = ti.longPop();
      ti.longPush(v1-v2);
      ti.addLongOperandAttr( timeValAttr);
      ti.skipInstruction( insn.getNext());
...
```

Exercises: Task 2 - Listener (3)

- executeInstruction() → top half creates CG

```
static final String CG_ID = “LCMP_choice”;
...
@Override
public void executeInstruction (JVM vm){
  Instruction insn = vm.getLastInstruction();
  ...
  if (insn instanceof LCMP){
    ThreadInfo ti = vm.getLastThreadInfo();
    if (!ti.isFirstStepInsn()){      // first time (top half)
      if (ti.hasOperandAttr(1, TimeVal.class) ||
          ti.hasOperandAttr(3, TimeVal.class)){
        IntChoiceFromSet cg =
           new IntChoiceFromSet(CG_ID, -1,0,1);
        vm.getSystemState().setNextChoiceGenerator(cg);
        ti.skipInstruction(insn);
      }
    } else { // second time (bottom half)
...
```

Exercises: Task 2 - Listener (4)

- executeInstruction() → bottom half pushes CG choice to be used by subsequent IFLE instruction

```
...
@Override
public void executeInstruction (JVM vm){
  Instruction insn = vm.getLastInstruction();
  ...
  if (insn instanceof LCMP){
    ThreadInfo ti = vm.getLastThreadInfo();
    if (!ti.isFirstStepInsn()){  // first time (top half)
      ...
    } else { // second time (bottom half)
      IntChoiceFromSet cg =
        vm.getCurrentChoiceGenerator(CG_ID,
                                     IntChoiceFromSet.class);
      if (cg != null){
        ti.longPop(); ti.longPop();
        ti.push( cg.getNextChoice());
        ti.skipInstruction( insn.getNext());
...
```

Exercises: Task 2 - Listener Sources

```
package tutorial;

import gov.nasa.jpf.*;
import gov.nasa.jpf.jvm.*;
import gov.nasa.jpf.jvm.choice.*;
import gov.nasa.jpf.jvm.bytecode.*;

public class StopWatch extends ListenerAdapter {
  static class TimeVal {}
  static TimeVal timeValAttr = new TimeVal();

@Override
public void instructionExecuted (JVM vm){
  Instruction insn = vm.getLastInstruction();
  if (insn instanceof NATIVERETURN){
    MethodInfo mi =((NATIVERETURN)insn).getMethodInfo();
    if (mi.getUniqueName().equals(
                  "currentTimeMillis()J")){
      ThreadInfo ti = vm.getLastThreadInfo();
      ti.addLongOperandAttr( timeValAttr);
    }
  }
}
```

```
static final String CG_ID = "LCMP_choice";

@Override
public void executeInstruction (JVM vm){
  Instruction insn = vm.getLastInstruction();

  if (insn instanceof LSUB){
    ThreadInfo ti = vm.getLastThreadInfo();
    if (ti.hasOperandAttr(1, TimeVal.class) ||
        ti.hasOperandAttr(3, TimeVal.class)){
      long v2 = ti.longPop();
      long v1 = ti.longPop();
      ti.longPush(v1-v2);
      ti.addLongOperandAttr( timeValAttr);
      ti.skipInstruction( insn.getNext());
    }

  } else if (insn instanceof LCMP){
    ThreadInfo ti = vm.getLastThreadInfo();
    if (!ti.isFirstStepInsn()){  // first time (top half)
      if (ti.hasOperandAttr(1, TimeVal.class) ||
          ti.hasOperandAttr(3, TimeVal.class)){
        IntChoiceFromSet cg =
           new IntChoiceFromSet(CG_ID, -1,0,1);
        vm.setNextChoiceGenerator(cg);
        ti.skipInstruction(insn);
      }
    } else { // second time (bottom half)
      IntChoiceFromSet cg =
        vm.getCurrentChoiceGenerator(CG_ID,
                                     IntChoiceFromSet.class);
      if (cg != null){
        ti.longPop(); ti.longPop();
        ti.push( cg.getNextChoice());
        ti.skipInstruction( insn.getNext());
      }
    }
  }
}
```

Exercises: Task 2 - Run Example

- create src/examples/StopWatchTest.jpf

```
target = StopWatchTest
listener=tutorial.StopWatch
```

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

```
...
============== search started: 10/23/11 11:28 PM
some lengthy computation..
all fine, finished in time
all fine, finished in time
```

? why repeated ?

```
============== error #1
gov.nasa.jpf.jvm.NoUncaughtExceptionsProperty
java.lang.RuntimeException: it took too long..
      at StopWatchTest.main(StopWatchTest.java:31)
...
```

Examples: Task 2 - Reference

- see StopWatchFuzzer listener in jpf-core
- sources in src/main/gov/nasa/jpf/listener/StopWatchFuzzer.java
- example in src/examples/StopWatch.{java,jpf}
- run with

```
bin/jpf src/examples/StopWatch.jpf
```

- more efficient way to detect System.currentTimeMillis() (uses classLoaded() notification)
- delegates LSUB execution

Conclusions

Conclusions

.. confused?

- 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!