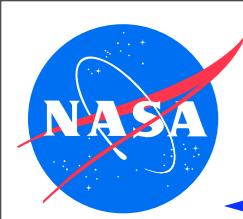


A light blue coordinate system is centered on the slide. It consists of a horizontal axis with a blue arrow pointing right, a vertical axis with a blue arrow pointing up, and a diagonal axis with a blue arrow pointing up-right. The axes intersect at the bottom center of the slide area.

# The State of JPF

## JavaPathfinder Workshop 2011

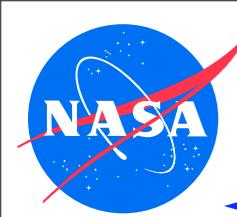
Peter C. Mehlitz  
SGT / NASA Ames Research Center  
[peter.c.mehlitz@nasa.gov](mailto:peter.c.mehlitz@nasa.gov)



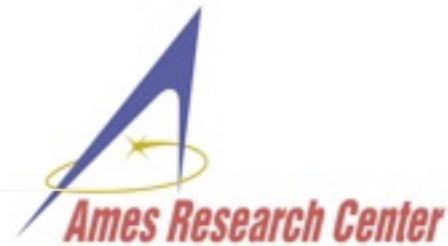
# Roadmap



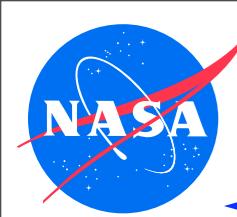
- ◆ JPF is more than jpf-core - what state(s) are we talking about?
- ◆ core & infrastructure - selected changes
- ◆ future plans



# What State? What JPF?



- ◆ JPF is more than core
- ◆ for outsiders,  $JPF = \sum$  public JPF projects
- ◆ project modularity good from technical perspective
- ◆ less good for giving a consistent, coherent view
- ◆ all project maintainers have to work for this view



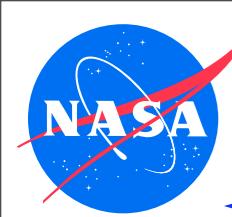
# What State? What JPF?



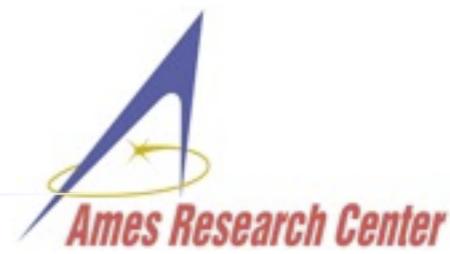
- ◆ JPF is more than core
- ◆ for outsiders,  $JPF = \sum$  public JPF projects
- ◆ project modularity good from technical perspective
- ◆ less good for giving a consistent, coherent view
- ◆ all project maintainers have to work for this view

.. but some can be extracted  
watching you





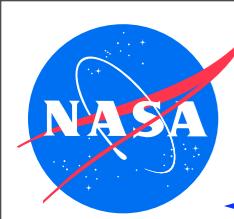
# State of JPF: Projects on Babelfish



- ◆ not all projects are created equal

We Need Status

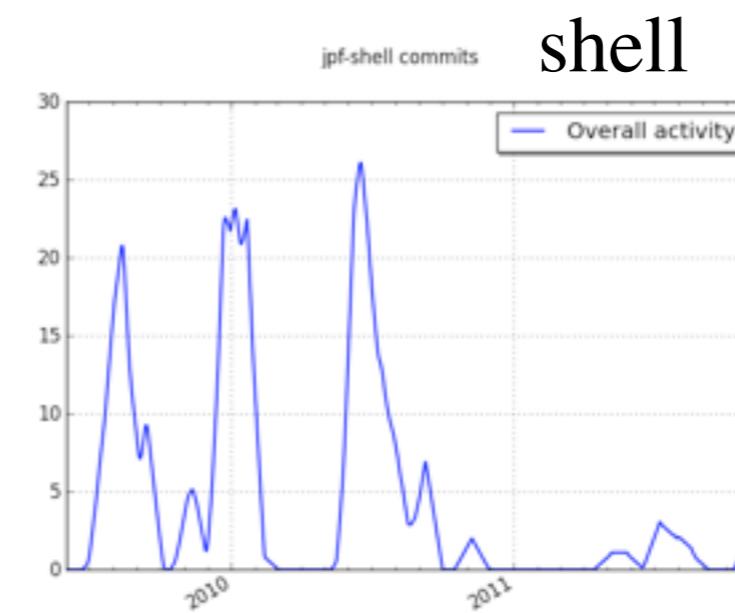
project	size	repo							
-----	-----	-----							
name	rev	file	loc	last	co	co	lines	-----	+/-
jpf-actor	r7:	123,	11017,	2011-06-01,	5,	33+,	29-,	(1.1)	
jpf-aprop	r55:	156,	17137,	2011-11-05,	19,	6869+,	1161-,	(5.9)	
jpf.awt	r55:	57,	4961,	2011-11-05,	37,	1747+,	295-,	(5.9)	
jpf.awt-shell	r42:	11,	1917,	2011-11-09,	9,	225+,	57-,	(3.9)	
jpf.concurrent	r218:	97,	16823,	2011-09-29,	14,	606+,	971-,	(0.6)	
jpf.core	r617:	837,	101565,	2011-11-06,	289,	74175+,	35332-,	(2.1)	
jpf.cv	r21:	34,	4408,	2011-05-24,	2,	25+,	10-,	(2.5)	
jpf.delayed	r2:	142,	8720,	2011-04-06,	2,	188+,	685-,	(0.3)	
jpf.extended-test-gen	r16:	89,	13828,	2010-10-30,	0,	0+,	0-		
jpf.guided-test	r86:	311,	25414,	2011-11-05,	43,	16196+,	4380-,	(3.7)	
jpf.inspector	r60:	218,	16507,	2011-11-01,	53,	27497+,	12930-,	(2.1)	
jpf.mango	r625:	1516,	116896,	2011-11-08,	626,	1182290+,	997207-,	(1.2)	
jpf.net-iocache	r35:	80,	7473,	2011-10-14,	27,	227424+,	50678-,	(4.5)	
jpf.numeric	r36:	34,	1532,	2011-11-05,	6,	286+,	222-,	(1.3)	
jpf.racefinder	r4:	83,	12128,	2010-10-18,	0,	0+,	0-		
jpf.rtembed	r17:	156,	13629,	2011-04-19,	3,	1982+,	224-,	(8.8)	
jpf.shell	r176:	81,	8463,	2011-11-09,	13,	2963+,	1374-,	(2.2)	
jpf.statechart	r21:	55,	10551,	2011-11-05,	7,	128+,	244-,	(0.5)	
jpf.symbc	r363:	576,	74742,	2011-11-07,	163,	75901+,	21244-,	(3.6)	
jpf.template	r24:	1,	313,	2011-11-07,	21,	837+,	629-,	(1.3)	
jpf.trace-server	r36:	102,	11770,	2011-11-01,	28,	20427+,	15742-,	(1.3)	
eclipse-jpf	r24:	13,	1720,	2011-11-05,	7,	1181+,	269-,	(4.4)	
netbeans-jpf	r22:	7,	1244,	2011-05-24,	2,	308+,	202-,	(1.5)	

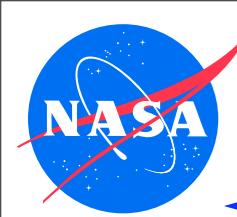


# Project State

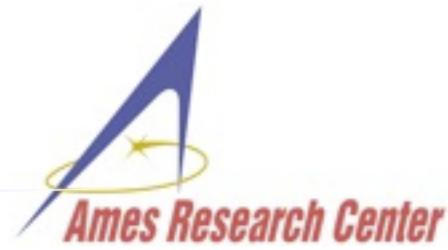


- ◆ projects come in various natures
  - signs of seasonal activity (papers, demos, internships)
  - actively developed / maintained (core sync) / abandoned
- ◆ not all can be automatically extracted from hg log  
(esp. if babelfish is not main site)
- ◆ notoriously missing: dependency project revisions  
are you in sync with core? → build.xml (test target)

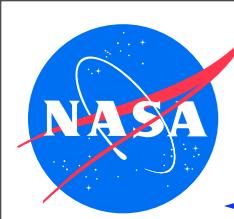




# Nuggets from “hg log”



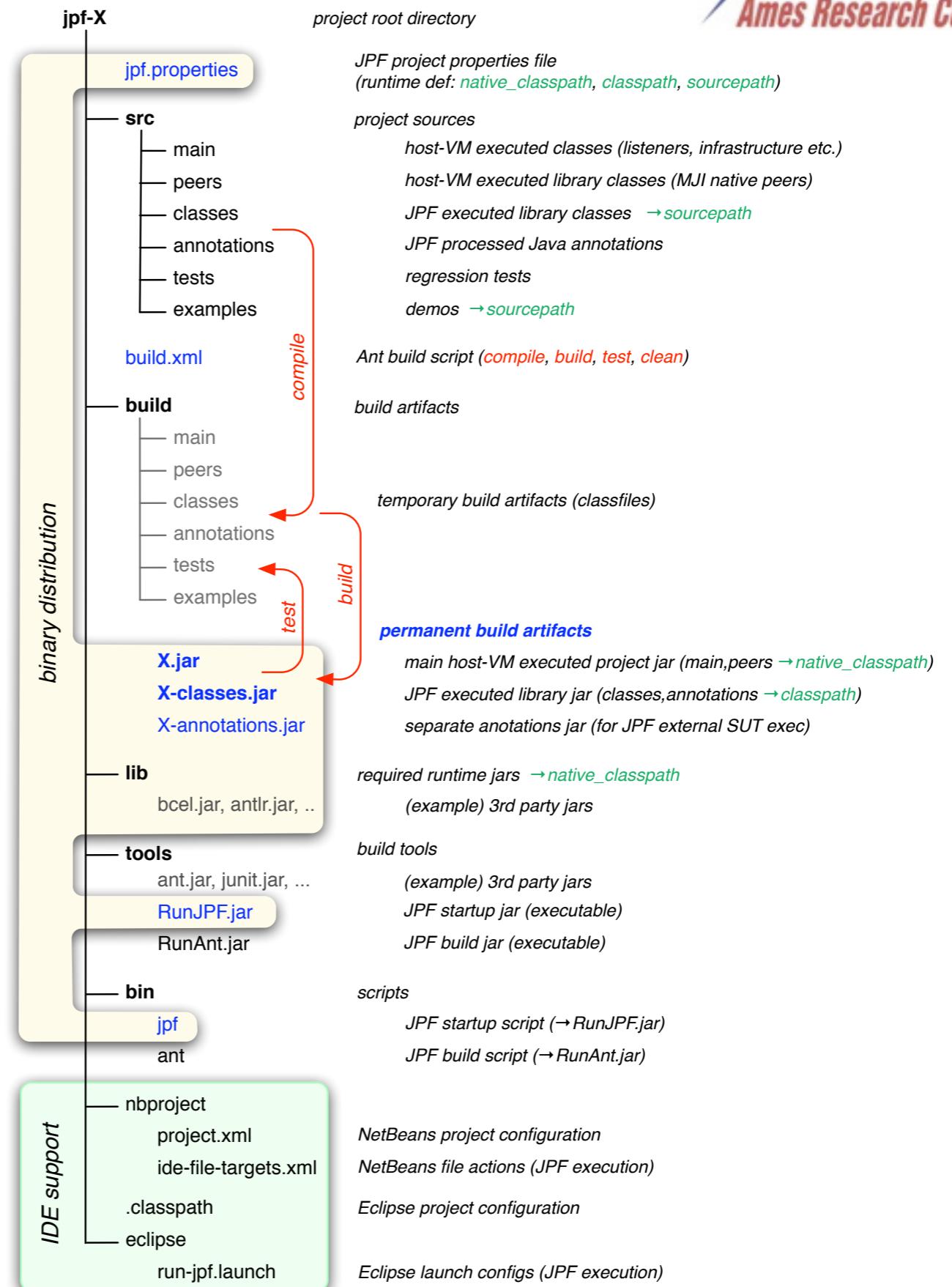
- ◆ list of changes that seem to have gone unnoticed
  - jpf-template and -addproject
  - gov.nasa.jpf.classfile infrastructure
  - ReleaseActions
  - (Multiple) Attributes
  - thread reference tracking & transition preemption  
(a nugget that can bite you)
  - ... more we can't cover here

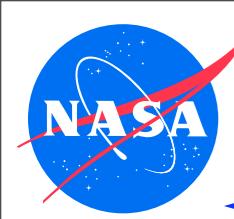


# Nuggets: Project Creation & Configuration

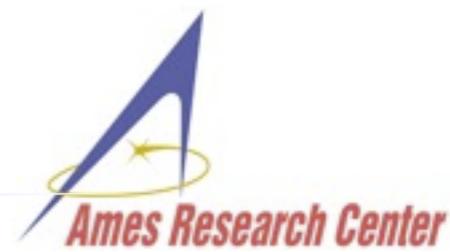


- ◆ jpf-template extension:  
`bin/create_project <pathname>`
- ◆ eclipse-jpf  
`New/Other/JPF/JPF-Project`
- ◆ now supports site configuration  
(major source of support Q's)
- ◆ RunJPF.jar (SiteUtilProperties)  
`bin/jpf -addproject [init]`





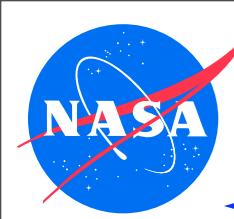
# Nuggets: What's in a Class File?



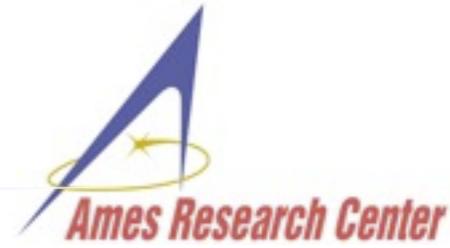
- ◆ gov.nasa.jpf.classfile good performance boost, but more (future proof)
- ◆ added infrastructure: ByteCodeReader, ClassFileReader + Adapters
- ◆ Example: ClassFilePrinter utility (ever wondered why your local var lookup fails?): [jpf-core/bin/print\\_class <classfile>](#)

```
----- constpool
[1]: constant_class {name=#2("ContractViolation")}
...
[12]: methodref {class=#3("ContractViolationBase"),nameType=#13("<init>","()V")}
...
----- methods ...
[1]: getLoopCount(I)I, flags=0x1, attr count=2
  [0]: RuntimeVisibleAnnotations count=2
    [0]: Lgov/nasa/jpf/annotation/Requires; valueCount=1
      [0]: value="c > 0"
[1]: Code, maxStack=2,maxLocals=2,length=4
  0: iload_1
  1: iconst_3
  2: isub
  3: ireturn
code attribute count=2
...
[1]: LocalVariableTable, localVar count=2
  [0]: this, type=LContractViolation;, scope=[0..3], slot=0
  [1]: c, type=I, scope=[0..3], slot=1
```

shows it all

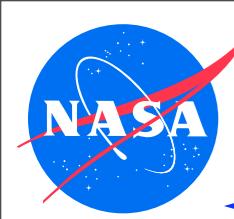


# Nuggets: Housekeeping? ReleaseActions



- ◆ interface `ReleaseAction.release(ElementInfo ei)`
- ◆ called by `ElementInfo.processReleaseActions(this)`
- ◆ type-based via `ClassInfo.addReleaseAction()` (classLoaded notification)
- ◆ instance based via attributes
- ◆ good to clean up native objects associated with live SUT objects:

```
protected void registerThreadListCleanup(){  
    ClassInfo ciThread = ClassInfo.tryGetResolvedClassInfo("java.lang.Thread");  
    assert ciThread != null : "java.lang.Thread not loaded yet";  
  
    ciThread.addReleaseAction( new ReleaseAction(){  
        public void release(ElementInfo ei) {  
            ThreadList tl = getThreadList();  
            int objRef = ei.getObjectRef();  
            ThreadInfo ti = tl.getThreadInfoForObjRef(objRef);  
            if (tl.remove(ti)){  
                getKernelState().changed();  
            }  
        }  
    });  
}
```



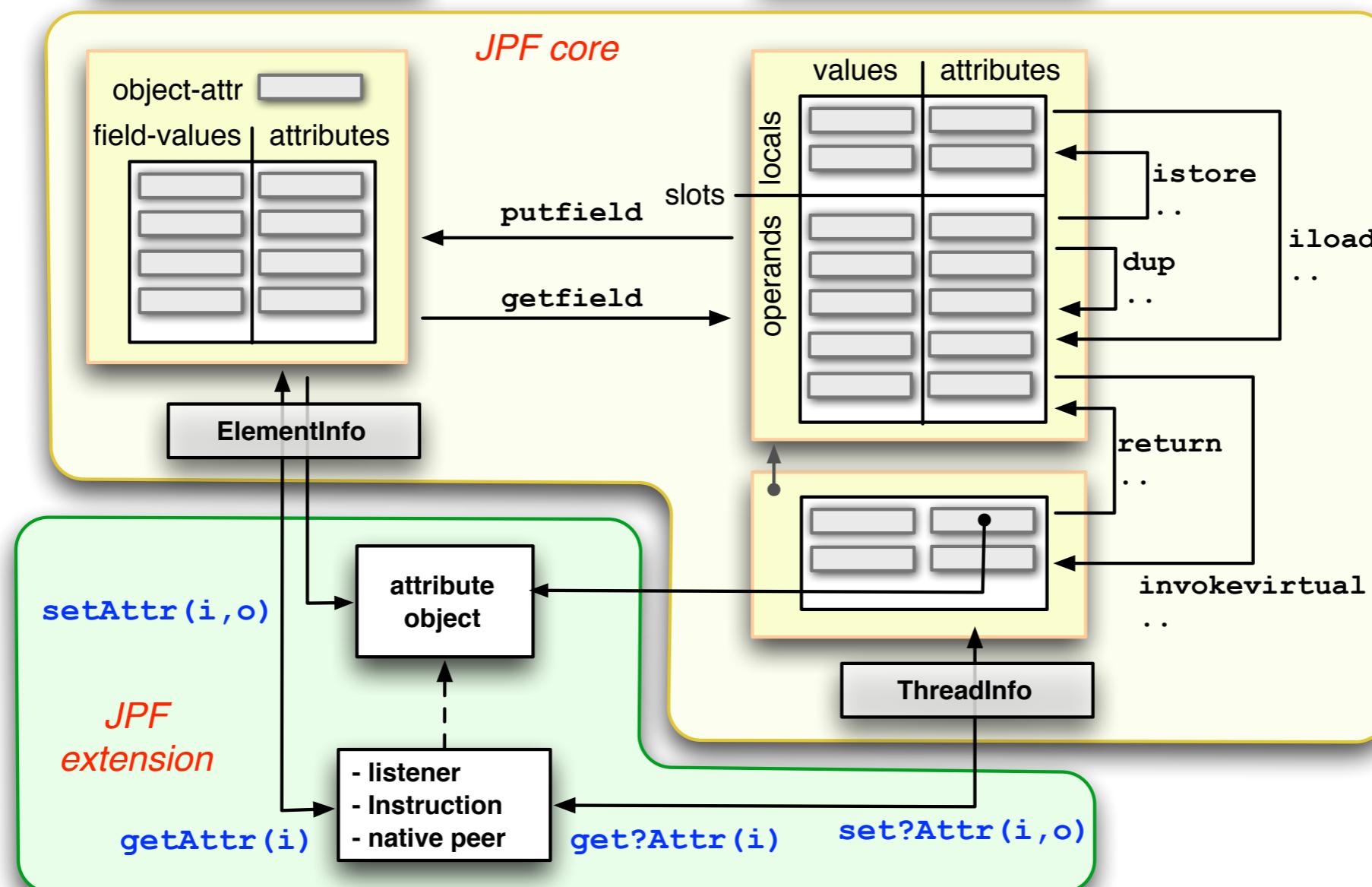
# Nuggets: Got Attributes?

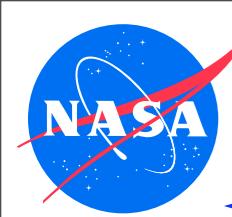


recap..

Fields	StackFrame
int[] values	int[] locals
Object[] fieldAttrs	Object[] localAttr
Object objectAttr	int[] operands
getIntValue(idx), ...	Object[] operandAttr
setIntValue(idx, v), ...	dup(), push(), pop(), ..
-----	-----
getFieldAttr(idx)	getOperandAttr(idx)
setFieldAttr(idx,obj)	setOperandAttr(idx,obj)
getObjectAttr()	getLocalAttr(idx)
setObjectAttr(obj)	setLocalAttr(idx,obj)

*attribute API*

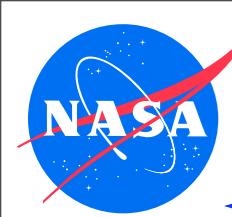




# Nuggets: Now you've got Multiple Attributes



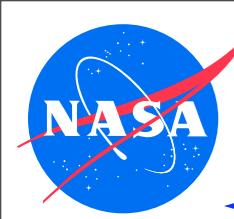
- ◆ like cascaded CGs a matter of coexistence of JPF extensions
- ◆ uses `gov.nasa.util.ObjectList` API (untyped, low overhead)
- ◆ more constructs can be attributed
  - `ElementInfo` (`Fields`)
    - ▶ `add/set/getObjectAttr[NoClone]()`
    - ▶ `add/set/getFieldAttr[NoClone]()`
  - `ThreadInfo`, `StackFrame`
    - ▶ `add/set/get[Long]OperandAttr[NoClone]()`
    - ▶ `add/set/getLocalAttr[NoClone]()`
    - ▶ `StackFrame.add/set/getFrameAttr()`
  - `MethodInfo`, `FieldInfo`, `ClassInfo`, `ChoiceGenerator`, `Instruction`
    - ▶ `add/set/getAttr()`
- ◆ `setAttr()` vs. `addAttr()` - be a nice JPF citizen
- ◆ can be enumerated but should be retrieved by (private) type:  
`T getAttr(Class<T>type)`
- ◆ be aware:
  - attributes are not state matched (by jpf-core serializers)
  - attribute references are restored, but not attribute values (can't clone Object)



# Nuggets: Now you've got Multiple Attributes



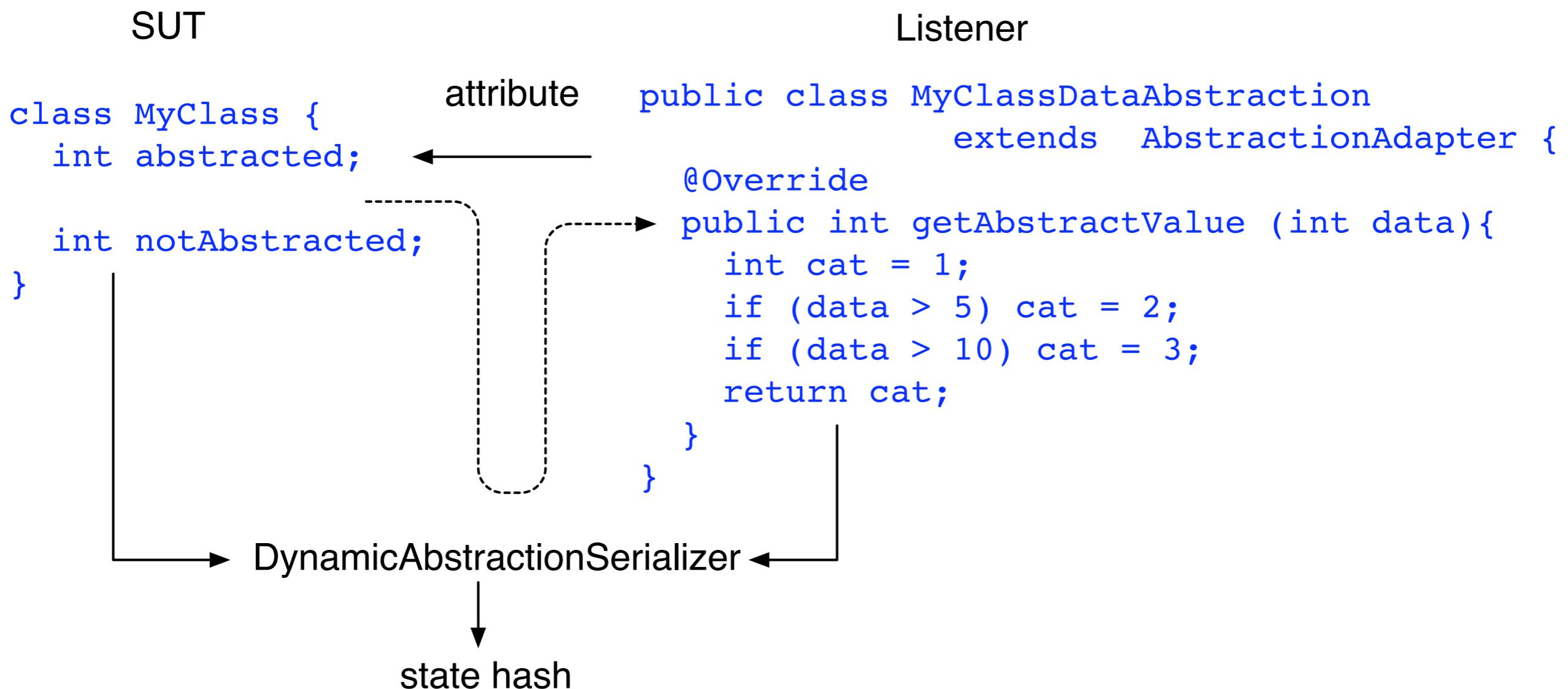
- ◆ like cascaded CGs a matter of coexistence of JPF extensions
- ◆ uses `gov.nasa.util.ObjectList` API (untyped, low overhead)
- ◆ more constructs can be attributed
  - `ElementInfo` (Fields)
    - ▶ `add/set/getObjectAttr[NoClone]()`
    - ▶ `add/set/getFieldAttr[NoClone]()`
  - `ThreadInfo`, `StackFrame`
    - ▶ `add/set/get[Long]OperandAttr[NoClone]()`
    - ▶ `add/set/getLocalAttr[NoClone]()`
    - ▶ `StackFrame.add/set/getFrameAttr()`
  - `MethodInfo`, `FieldInfo`, `ClassInfo`, `ChoiceGenerator`, `Instruction`
    - ▶ `add/set/getAttr()`
- ◆ `setAttr()` vs. `addAttr()` - be a nice JPF citizen
- ◆ can be enumerated but should be retrieved by (private) type:  
`T getAttr(Class<T>type)`
- ◆ be aware:
  - attributes are not state matched (by jpf-core serializers)
  - attribute references are restored, but not attribute values (can't clone Object)

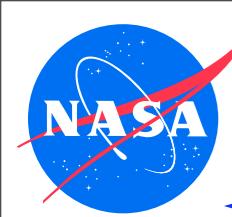


# Nuggets: But what to do with Attributes?



- ◆ mostly data flow (see StopWatchFuzzer example)
- ◆ mark features at class load time (class, method, field)
- ◆ can be more: state abstractions on a per-object/field basis  
(see [.test.mc.data.DynamicAbstractionTest](#))

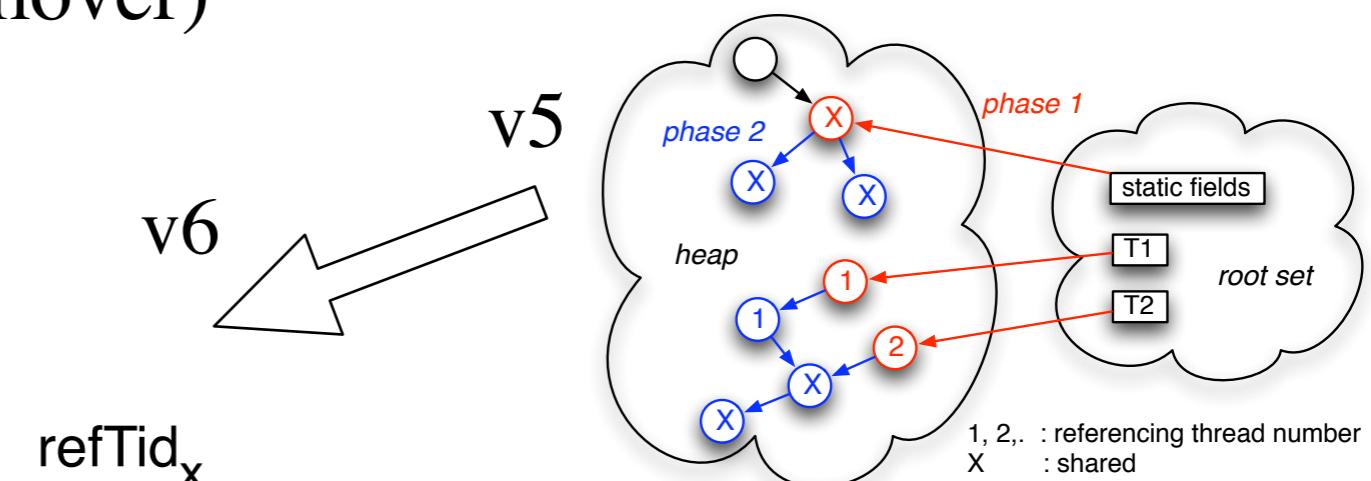
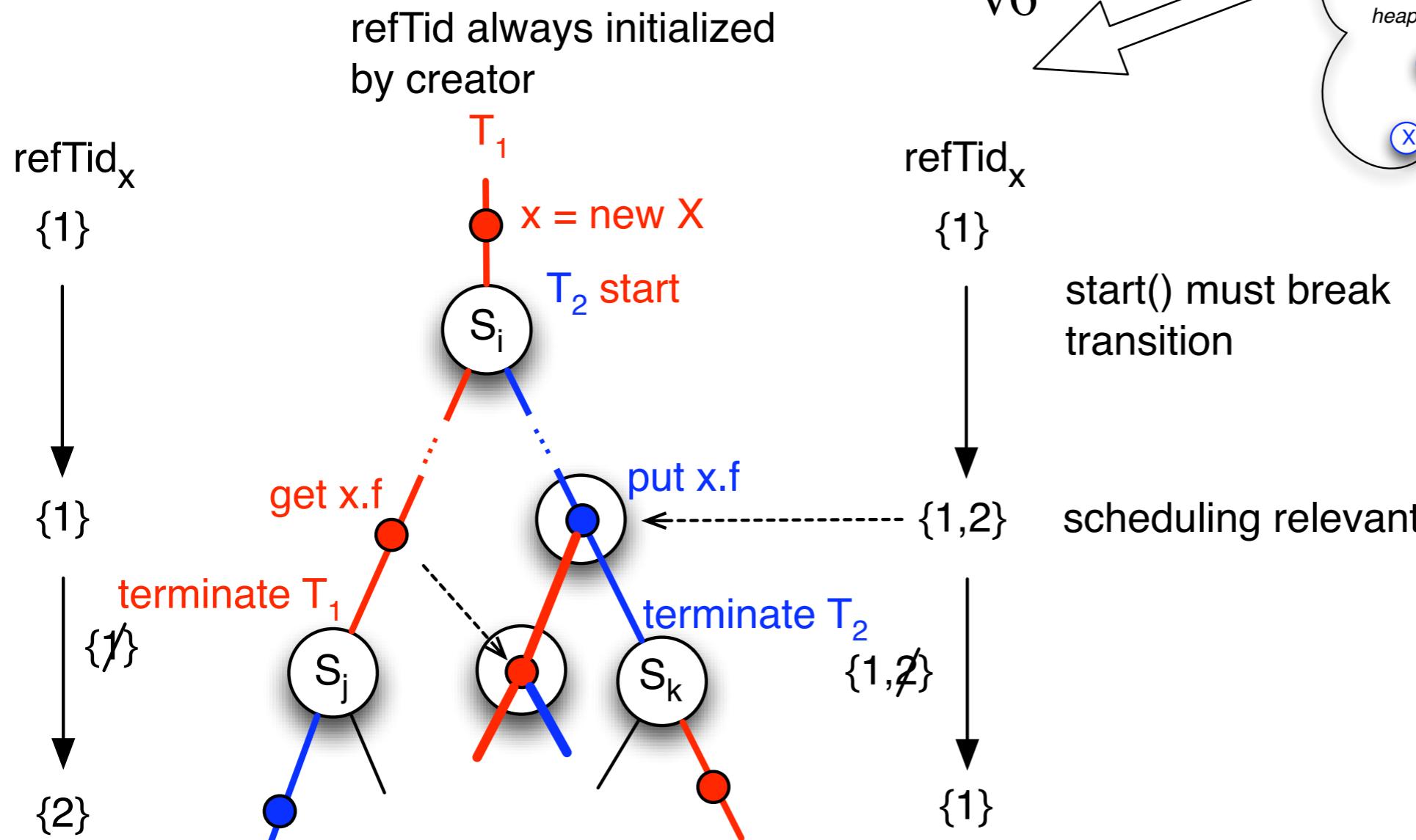


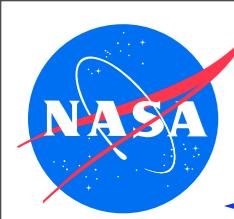


# Nuggets: What's Sharedness



- ◆ old sharedness: attribute set by conservative GC reachability analysis
- ◆ now per-object **refTid** set of referencing threads (put/get ops)  
(`Thread.start()` has to be right mover)
- ◆ new sharedness:  $| \text{refTid} | > 1$





# Not So Nugget: The Problem(s) with refTid



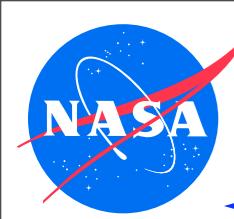
- ◆ implementation: FixedBitSet & ThreadInfo.id  
can't grow - some apps have to use vm.reuse\_tid + termination cleanup
- ◆ algorithm: “why does JPF hang - my example is so *simple*”

```
static MyObject myObj = new MyObject(0);

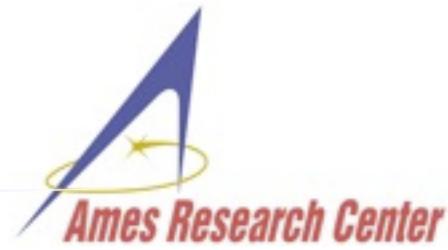
static void main(String[] args){
    new Thread(){
        public void run(){
            while (true){
                if (myObj.x > 10) throw ...
            }
        }
    }.start();

    new Thread(){
        public void run(){
            myObj.x = 42;
        }
    }.start();
}
```

pathological path:



# Not So Nugget: The Problem(s) with refTid



- ◆ implementation: FixedBitSet & ThreadInfo.id  
can't grow - some apps have to use vm.reuse\_tid + termination cleanup
- ◆ algorithm: “why does JPF hang - my example is so *simple*”

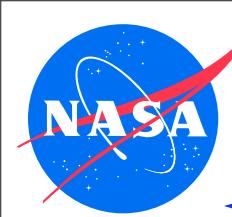
```
static MyObject myObj = new MyObject(0);

static void main(String[] args){
    new Thread(){
        public void run(){
            while (true){
                if (myObj.x > 10) throw ...
            }
        }
    }.start();

    new Thread(){
        public void run(){
            myObj.x = 42;
        }
    }.start();
}
```

pathological path:

main terminated +  
T2 ‘put’ not yet reached  
⇒ ‘myObj’ is not shared



# Not So Nugget: The Problem(s) with refTid



- ◆ implementation: FixedBitSet & ThreadInfo.id can't grow - some apps have to use vm.reuse\_tid + termination cleanup
- ◆ algorithm: “why does JPF hang - my example is so *simple*”

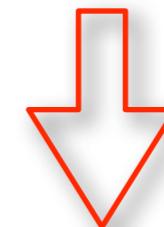
```
static MyObject myObj = new MyObject(0);

static void main(String[] args){
    new Thread(){
        public void run(){
            while (true){
                if (myObj.x > 10) throw ...
            }
        }
    }.start();

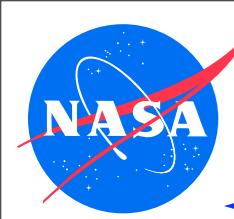
    new Thread(){
        public void run(){
            myObj.x = 42;
        }
    }.start();
}
```

pathological path:

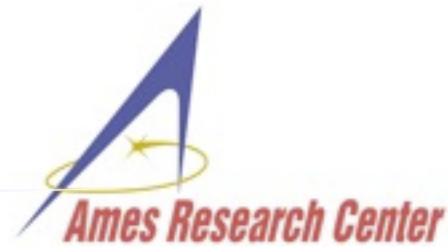
main terminated +  
T2 ‘put’ not yet reached  
⇒ ‘myObj’ is not shared



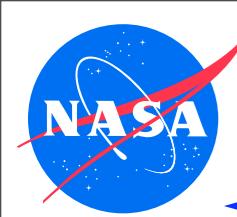
T1 gets stuck in  
endless transition



# Nuggets: (Almost) a Solution for Bad Path



- ◆ transition preemption to terminate loop through state matching
- ◆ naive approach: break after x number of instructions  
(good luck with matching states)
- ◆ better approach: break after x instructions at backjump (loop)
- ◆ compatible with IdleFilter
- ◆ BUT: can produce lots of new states
- ◆ breaks fragile tests in some projects
- ◆ controlled by new option: `vm.max_transition_length`  
(default 5000, set to MAX to avoid preemption)



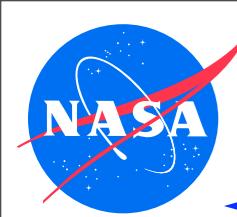
# Outlook



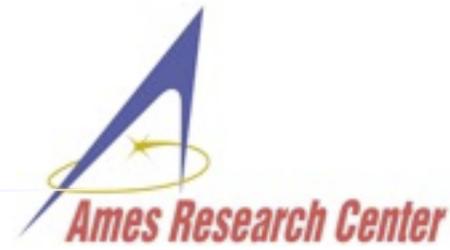
- ◆ intentionally vague

**Subject to the Requirements of the Service**

- ◆ only one big topic remaining from '03 list:
  - ClassLoader modeling (distributed applications)
- ◆ but new ones as we go:
  - Separate public project sites? Maybe ([code.google.com/p/jpf-core](http://code.google.com/p/jpf-core))
  - Project documentation tools
  - External trace analysis (trace server)
  - fast JMM awareness (sync with volatile vars)
  - ... (wait for closing session)



# Outlook



- ◆ intentionally vague

**Subject to the Requirements of the Service**

- ◆ only one big topic remaining from '03 list:
  - ClassLoader modeling (distributed applications)
- ◆ but new ones as we go:
  - Separate public project sites? Maybe ([code.google.com/p/jpf-core](http://code.google.com/p/jpf-core))
  - Project documentation tools
  - External trace analysis (trace server)
  - fast JMM awareness (sync with volatile vars)
  - ... (wait for closing session)

Thank You