Formal Methods for Java

Lecture 26: Properties, Listener and Java Pathfinder

Jochen Hoenicke

Software Engineering
Albert-Ludwigs-University Freiburg

Feb 03, 2012

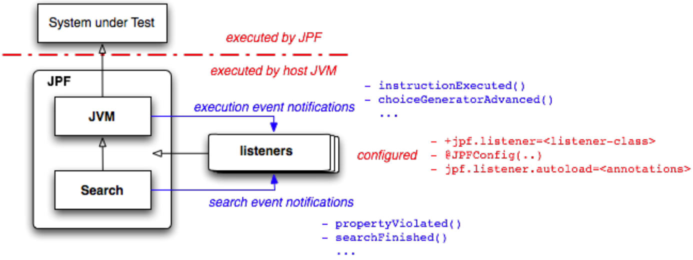
Properties

- Configured with `search.properties`
- Evaluated after every transition
- Base class: *gov.nasa.jpf.Property*
- Properties shipped with JPF Core:
  - *gov.nasa.jpf.jvm.IsEndStateProperty*
  - *gov.nasa.jpf.jvm.NoOutOfMemoryErrorProperty*
  - *gov.nasa.jpf.jvm.NotDeadlockedProperty*
  - *gov.nasa.jpf.jvm.NoUncaughtExceptionsProperty*

Listener

- Configured with `listener` and `listener.autoload`
- Different types:
  - *VMListener* notified about executed instructions, threads state changes, loaded classes, created objects, object monitor events, garbage collections, choice generators, and method enter and exit events
  - *SearchListener* notified about state changes, property violations, and search related events
- Implementation basis for many extensions
- Idea: JPF can check what you can program
- JPF Core comes with many listeners in package *gov.nasa.jpf.listener*

How Listeners Work



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

- VM or search notifies listener about next or previous event.
- Listener can act upon this event.
- Listeners can influence VM or search.
- Can annotate objects, fields, operands, and variables with attributes

Writing Our First Listener

Desired Property

A user-specified set of fields and variables should never be assigned to `null`*.*

Chopped into Pieces

- configurable field and variable description
- check for variable and field assignment

JPF Property vs. Listener

- Desired property can be violated by writing a field or variable.
- This does not necessarily break a transition.
- ➥ We need a listener to break the transition and report an error.

Using Utilities (1/2)

gov.nasa.jpf.util.FieldSpec

Utility for specifying field descriptions:

x.y.Foo.bar	field *bar* in class *x.y.Foo*
x.y.Foo+.bar	all *bar* fields in *x.y.Foo* and all its supertypes
x.y.Foo.*	all fields of *x.y.Foo*
*.myData	all fields names *myData*
!x.y.*	all fields of types outside types in package *x.y*

Using Utilities (2/2)

gov.nasa.jpf.util.MethodSpec

Utility for specifying methods:

exact method signature, or:

x.y.Foo.*	all methods of class *x.y.Foo*
.(x.y.MyClass)	all methods that take exactly one parameter which is of type *x.y.MyClass*
!x.y.*.*(int)	no method of any class in package *x.y* or any subpackage that takes exactly one argument that is an `int`

gov.nasa.jpf.util.VarSpec

Utility for specifying local variable descriptions:

Syntax: `MethodSpec`:`VariableName`

Initializing our Listener

```
public NonNullChecker(Config conf) {
  Set<String> spec = conf.getStringSet("nnc.fields");
  if (spec == null)
    spec = Collections.emptySet();
  nonNullableFields = new FieldSpec[spec.size()];
  int i = -1;
  for (String field : spec)
    nonNullableFields[++i] = FieldSpec.createFieldSpec(field);
  spec = conf.getStringSet("nnc.vars");
  if (spec == null)
    spec = Collections.emptySet();
  nonNullableVars = new VarSpec[spec.size()];
  i = -1;
  for (String var : spec)
    nonNullableVars[++i] = VarSpec.createVarSpec(var);
}
```

Checking the Desired Property Part 1: Fields

Observation

Only two instructions can assign `null` to a field:

- putfield
- putstatic

Basic Idea

If such an instruction wrote to a field we are interested in, check value of that field.

➥ *instructionExecuted* notification

Field Checks

```
private void checkFieldInsn(FieldInstruction insn) {
  if (isRelevantField(insn)) {
    if (isNullFieldStore(insn)) {
      storeError(vm, insn);
      vm.breakTransition();
    }
  }
}
private boolean isRelevantField(FieldInstruction insn) {
  if (!insn.isReferenceField())
    return false;
  FieldInfo fi = insn.getFieldInfo();
  for (FieldSpec fieldSpec : nonNullableFields) {
    if (fieldSpec.matches(fi)) {
      return true;
    }
  }
  return false;
}
private boolean isNullFieldStore(FieldInstruction insn) {
  FieldInfo fi = insn.getFieldInfo();
  ElementInfo ei = insn.getLastElementInfo();
  return ei.getFieldValueObject(fi.getName()) == null;
}
```

Checking the Desired Property Part 2: Local Variables

Observation

Only one instruction can assign `null` to a local variable:

- astore

We can use our method from before to check that.

Local Variable Checks

```
private void checkLocalVarInsn(ASTORE insn) {
  if (isRelevantVar(insn)) {
    if (isNullVarStore(insn)) {
      storeError(vm, insn);
      vm.breakTransition();
    }
  }
}
private boolean isRelevantVar(ASTORE insn) {
  int slotIdx = insn.getLocalVariableIndex();
  MethodInfo mi = insn.getMethodInfo();
  int pc = insn.getPosition() + 1;

  for (VarSpec varSpec : nonNullableVars) {
    if (varSpec.getMatchingLocalVarInfo(mi, pc, slotIdx) != null)
      return true;
  }
  return false;
}
private boolean isNullVarStore(ASTORE insn) {
  ThreadInfo ti = vm.getLastThreadInfo();
  int slotIdx = insn.getLocalVariableIndex();
  return ti.getObjectLocal(slotIdx) == null;
}
```

Demo