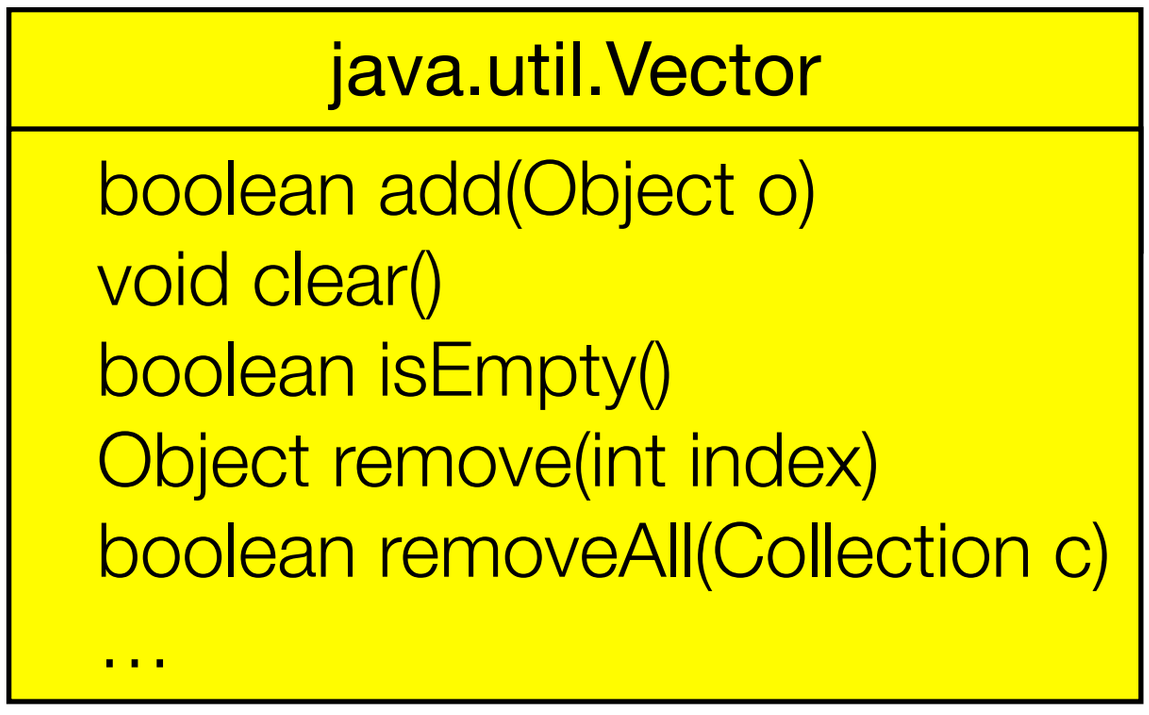
Mining Models of Object Behavior

Andrzej Wasylkowski, Saarland University
(with Valentin Dallmeier and Nicolas Bettenburg)

Advisors: Christian Lindig and Andreas Zeller

The problem

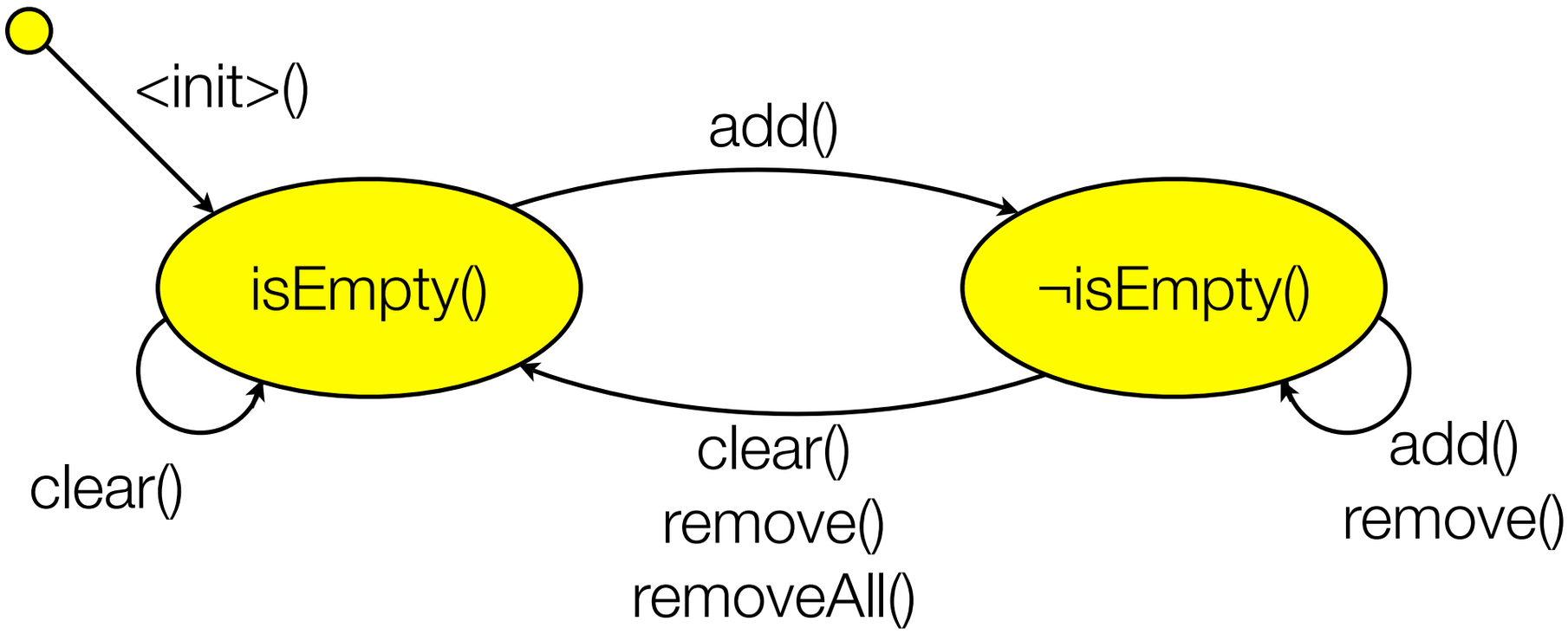


- What is the effect of calling ‘clear’?

- Can ‘remove’ be called at any time?

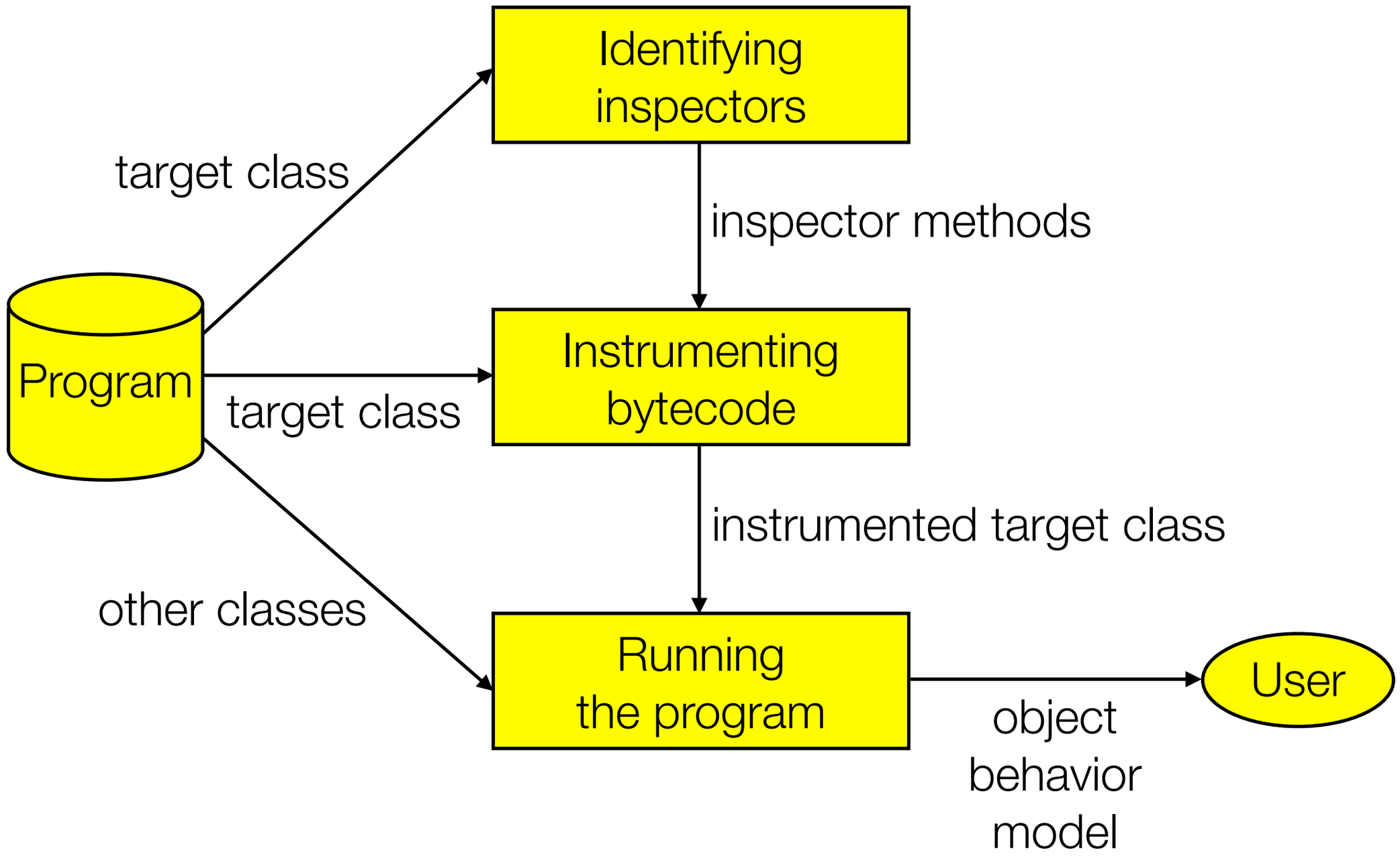
How to use a class correctly?

What do object behavior models look like?

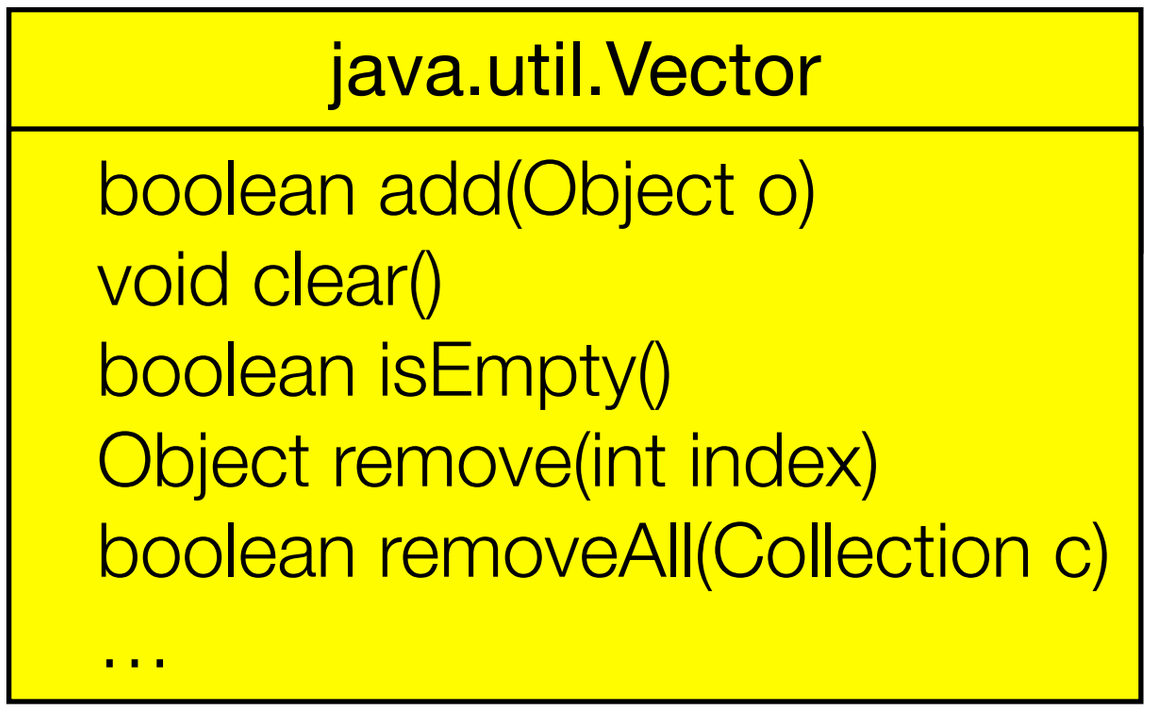


Object behavior model for the java.util.Vector class

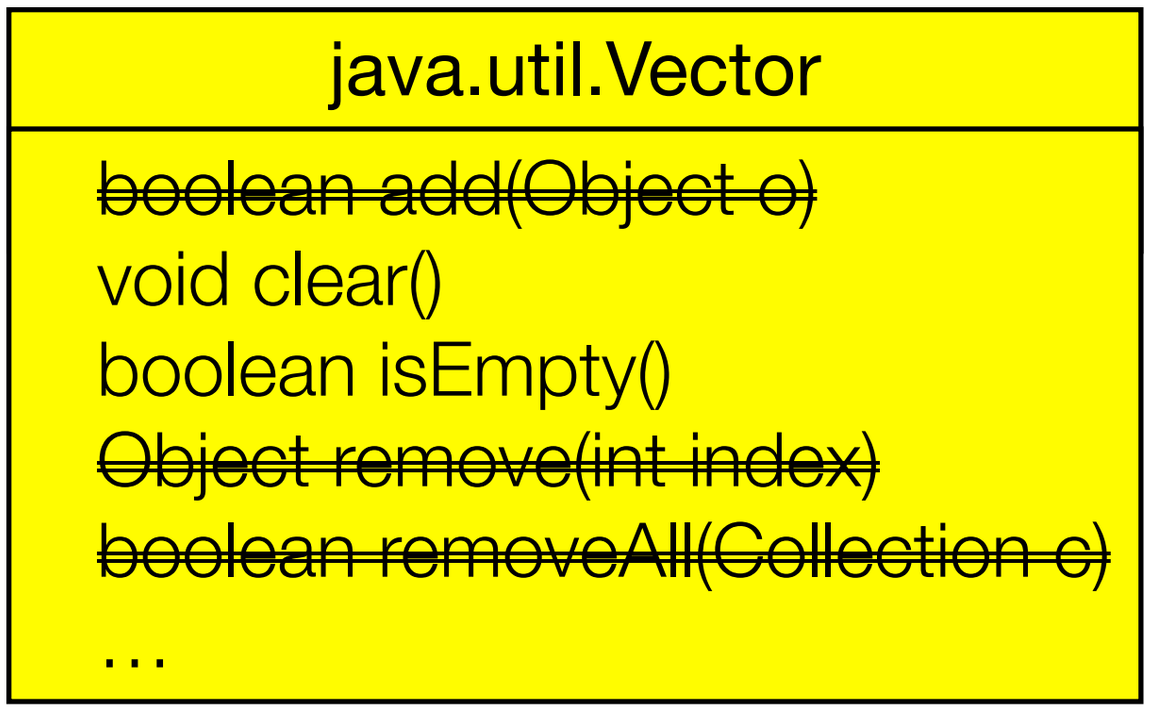
How can we mine object behavior models?



Identifying inspectors

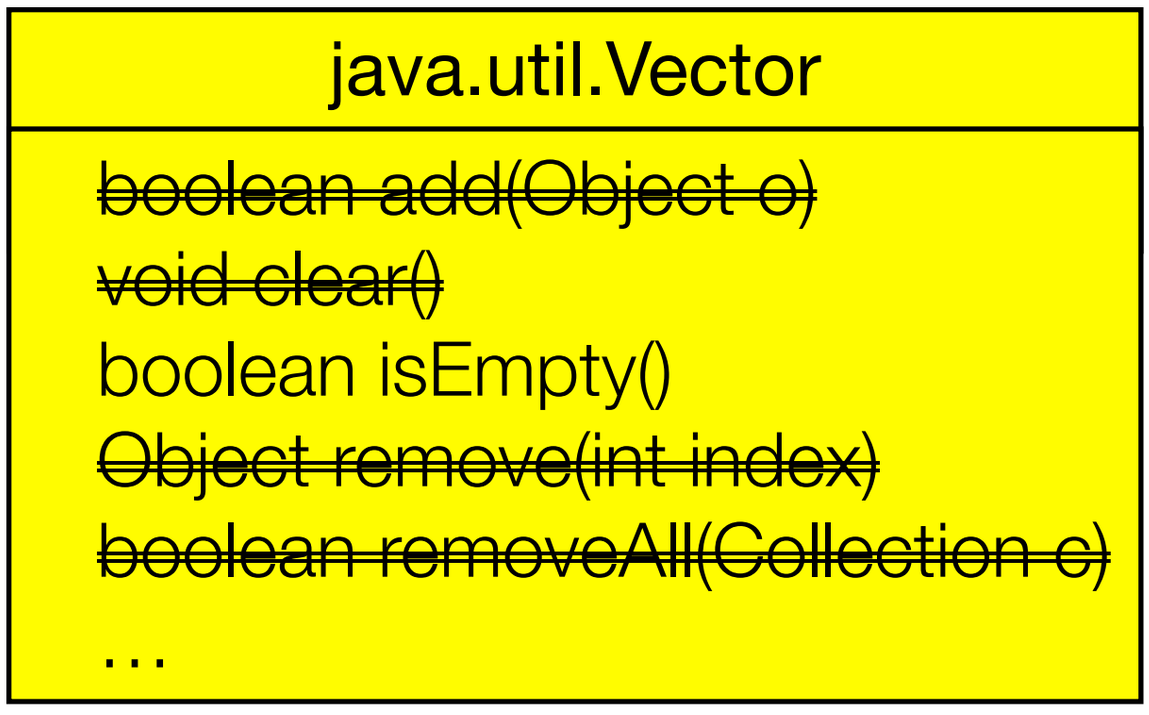


Identifying inspectors



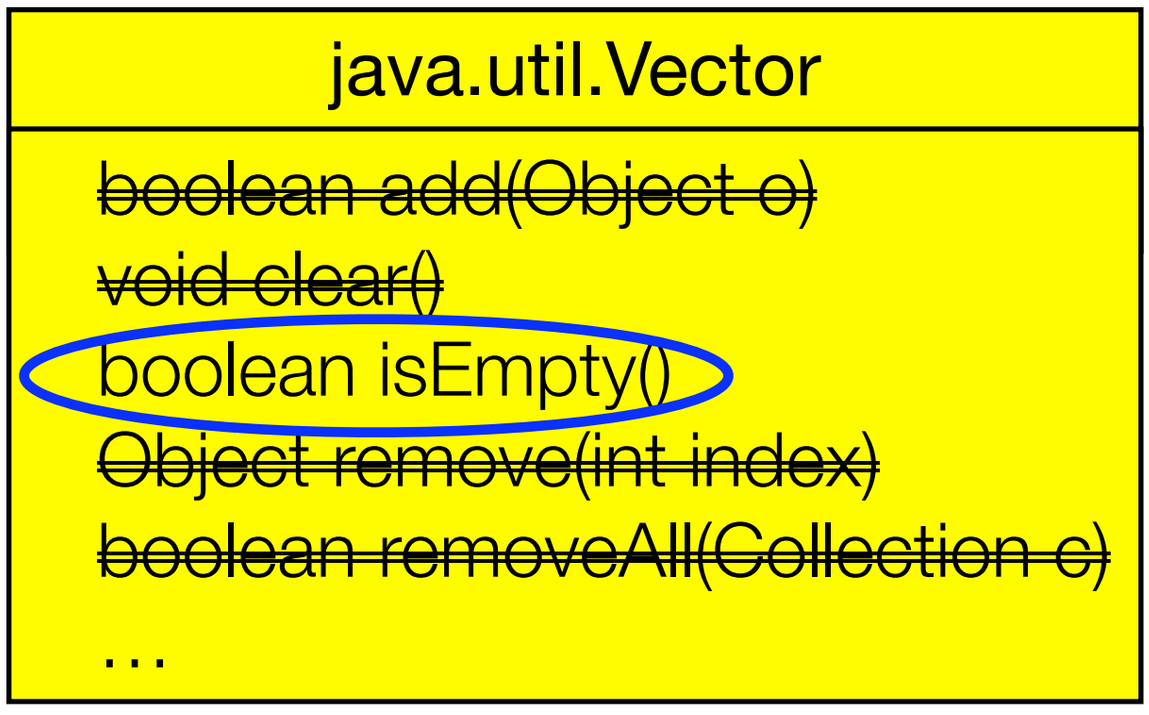
- No parameters

Identifying inspectors



- No parameters
- Is “pure”

Identifying inspectors



- No parameters
- Is “pure”
- Returns a value

Instrumenting bytecode

```
public class Vector {
    …
    public void clear() {
        …
    }
    …
}
```

Instrumenting bytecode

```
public class Vector {
    …
    public void clear() {
        State entryState = new State();
        entryState.addElement(”isEmpty()”, isEmpty());
                …
    }
    …
}
```

Instrumenting bytecode

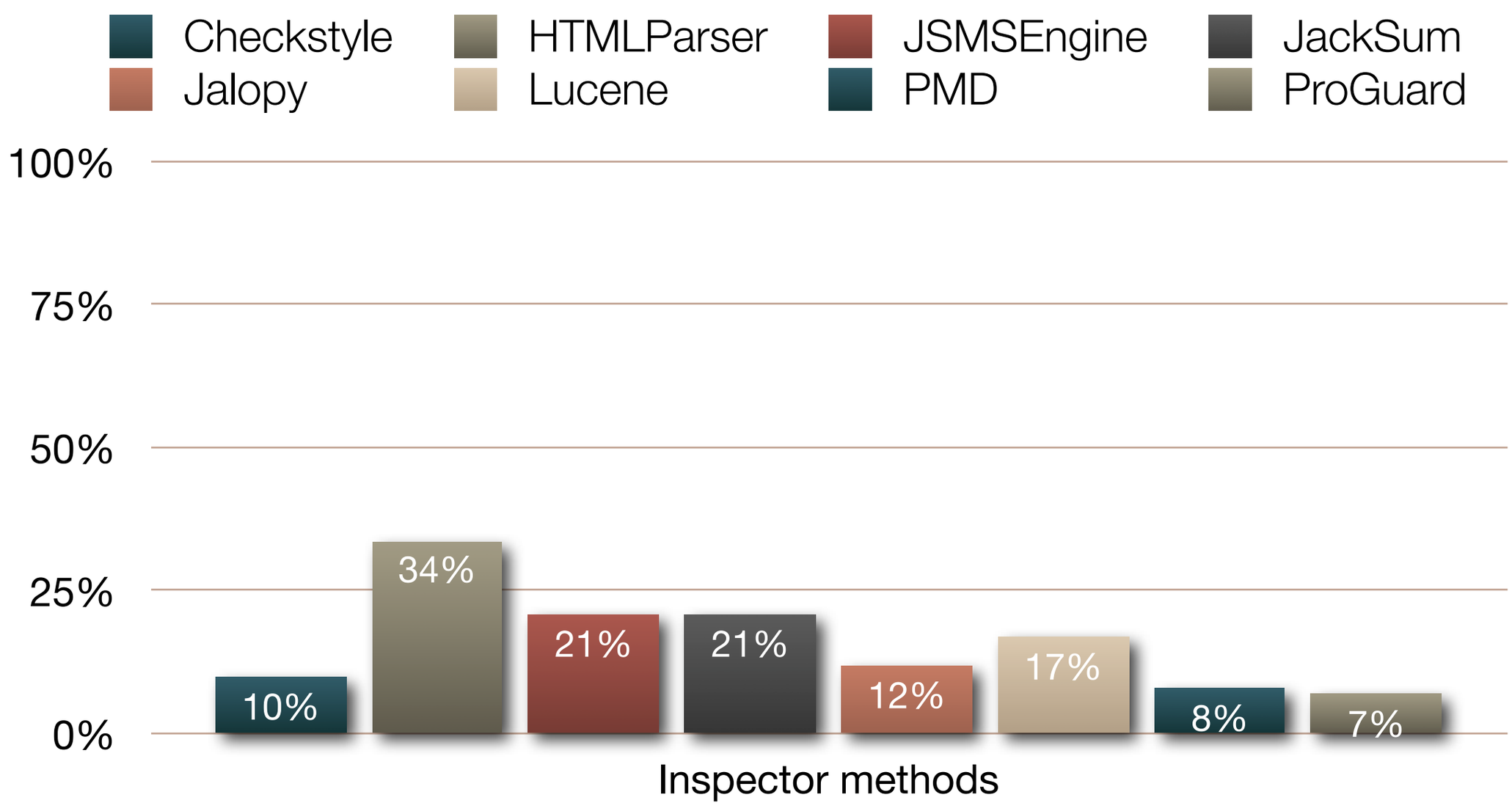
```
public class Vector {
    …
    public void clear() {
        State entryState = new State();
        entryState.addElement(”isEmpty()”, isEmpty());
        try {
            …
        }
        finally {
            State exitState = new State();
            exitState.addElement(”isEmpty()”, isEmpty());

        }
    }
    …
}
```

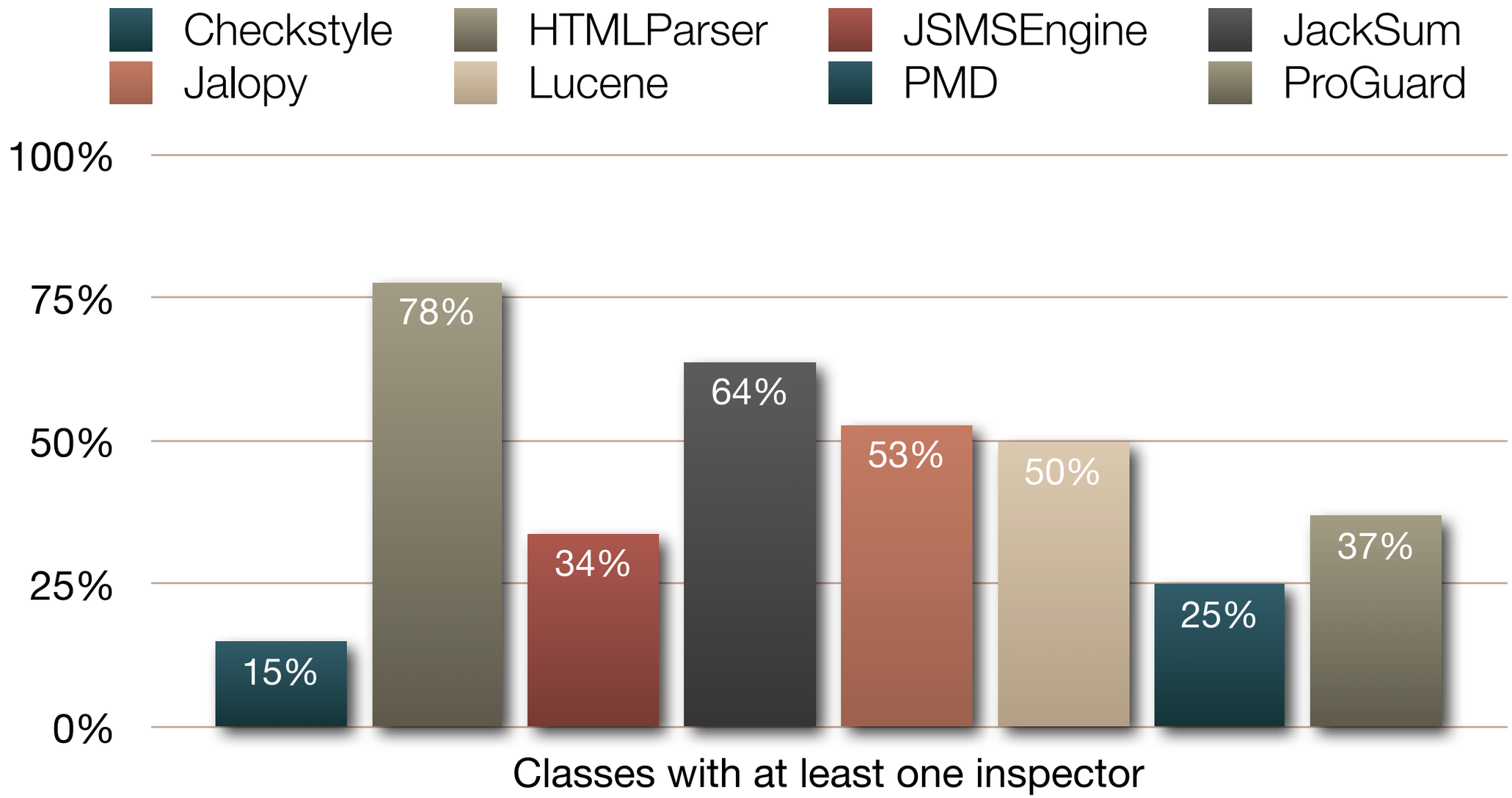
Instrumenting bytecode

```
public class Vector {
    …
    public void clear() {
        State entryState = new State();
        entryState.addElement(”isEmpty()”, isEmpty());
        try {
            …
        }
        finally {
            State exitState = new State();
            exitState.addElement(”isEmpty()”, isEmpty());
            model.addTransition(entryState, exitState, “clear()”);
        }
    }
    …
}
```

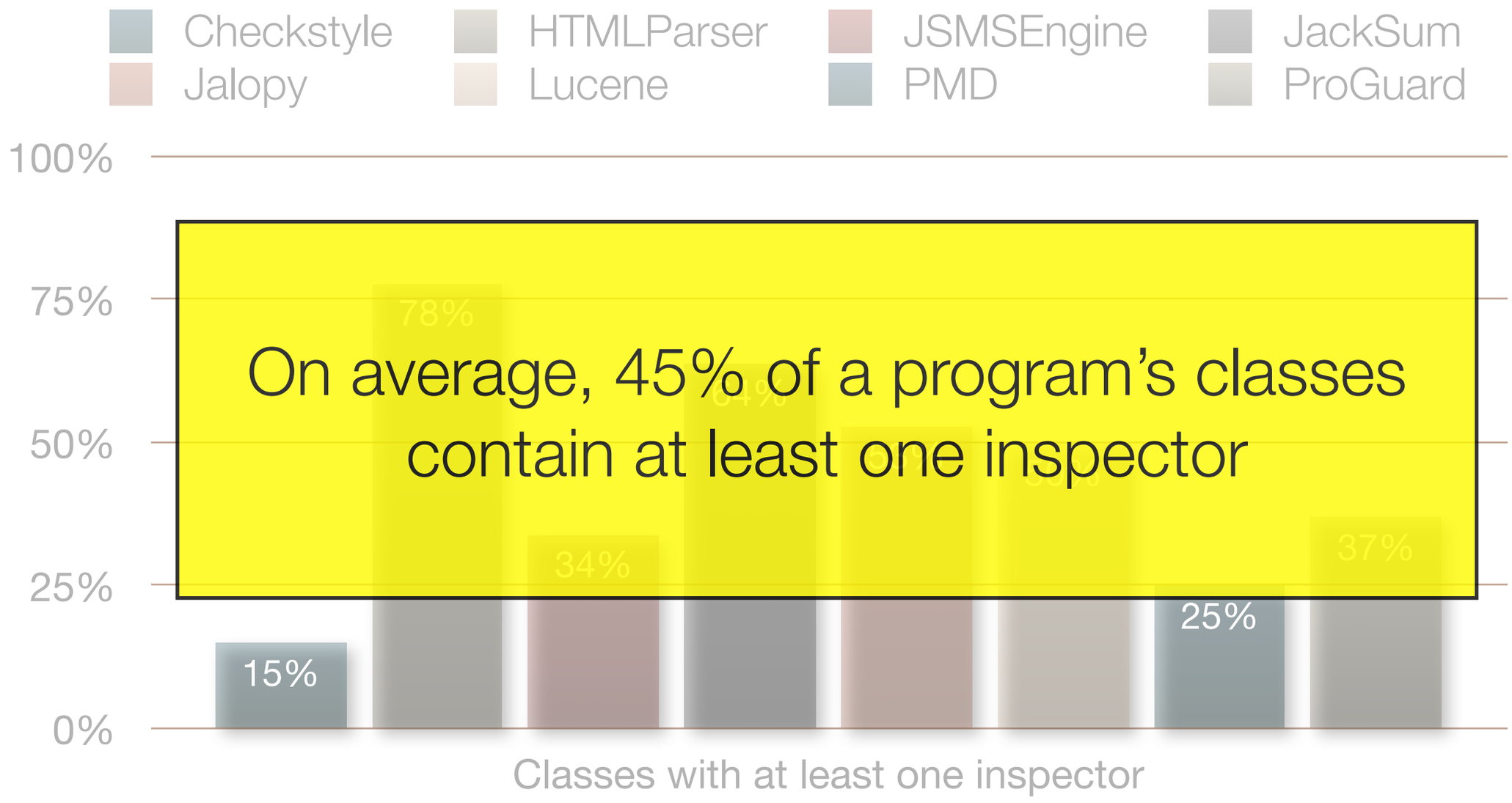
How common are inspectors?



How common are classes with inspectors?



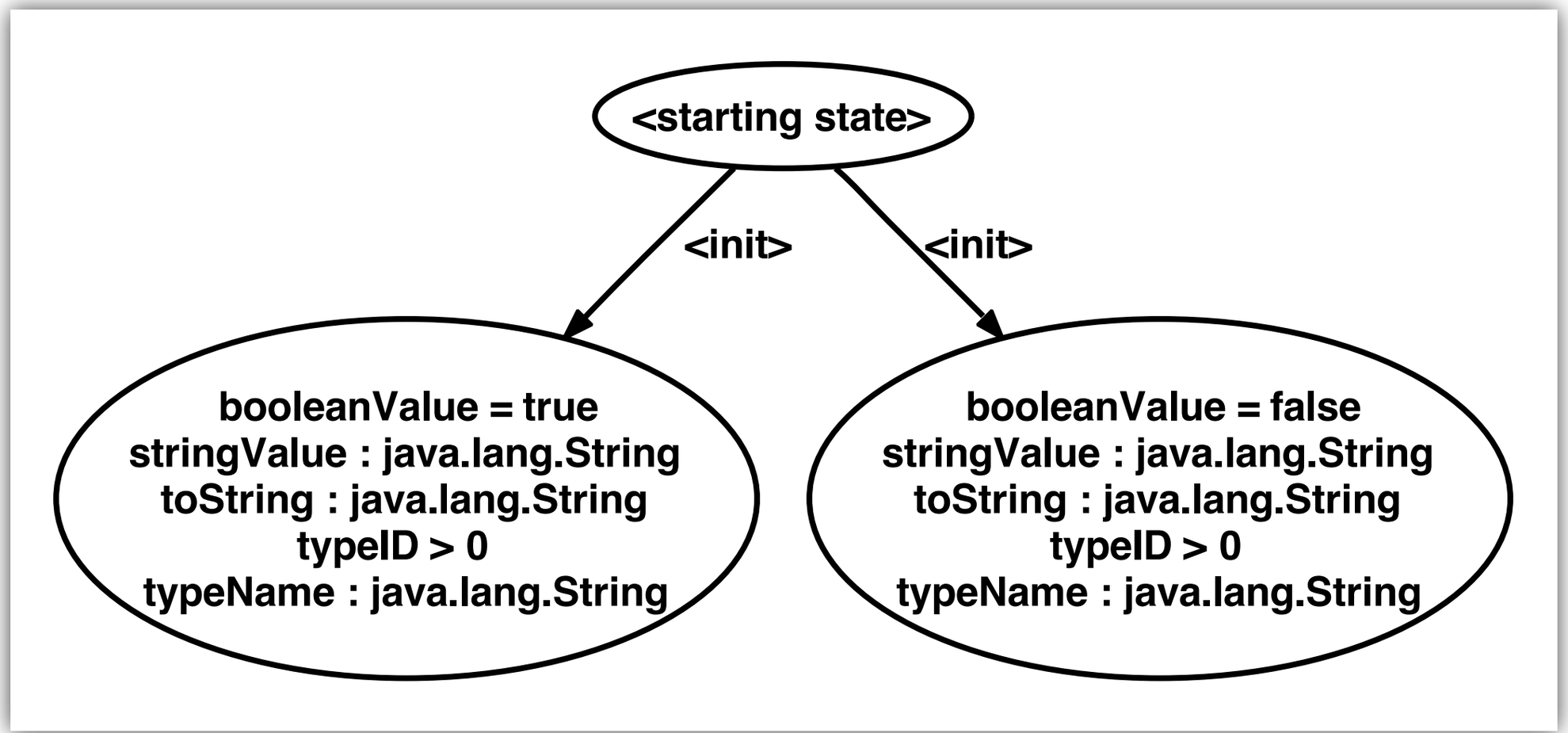
How common are classes with inspectors?



On average, 45% of a program's classes contain at least one inspector

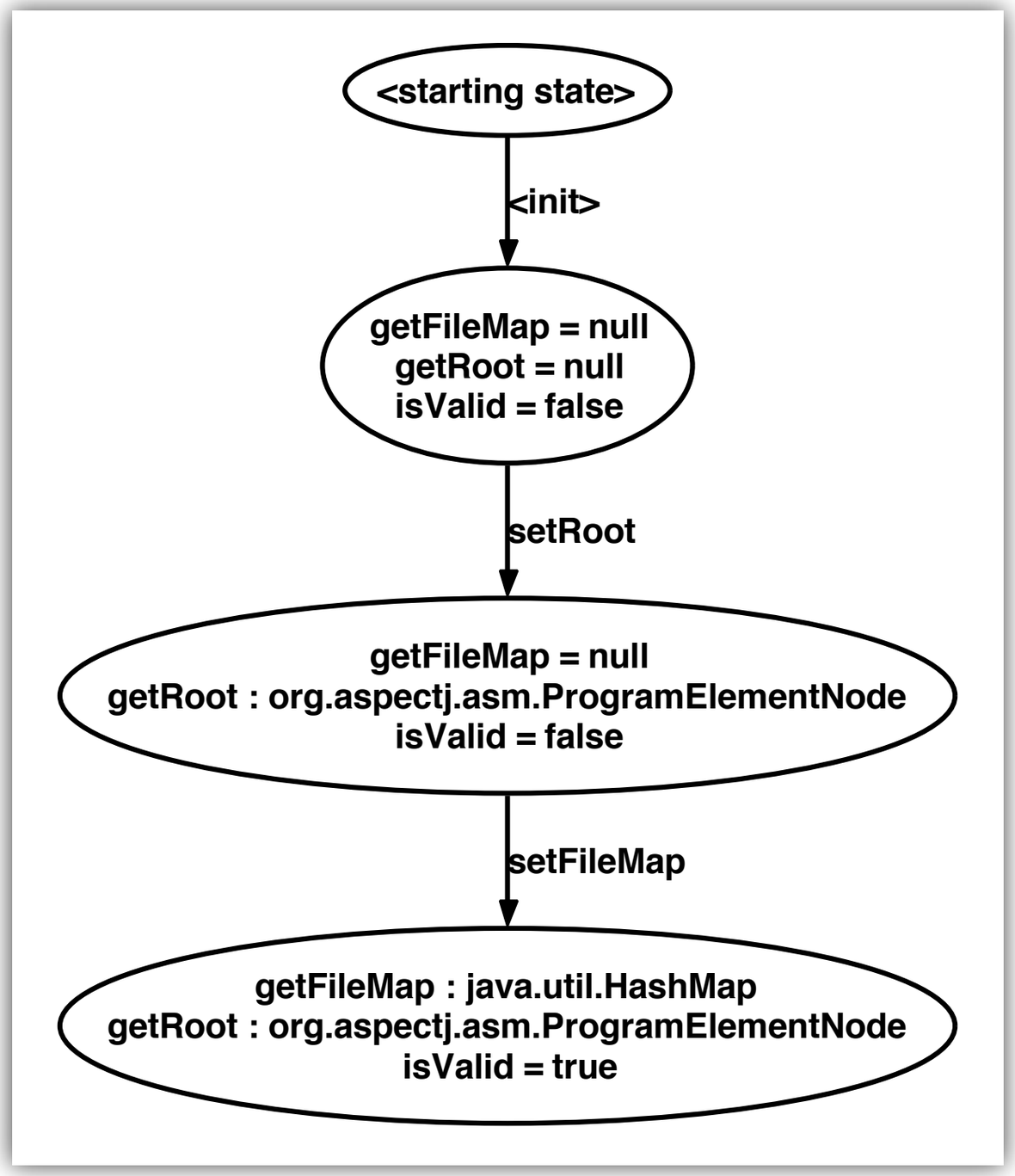
Application: understanding behavior of a class

org.eclipse.jdt.internal.compiler.impl.BooleanConstant



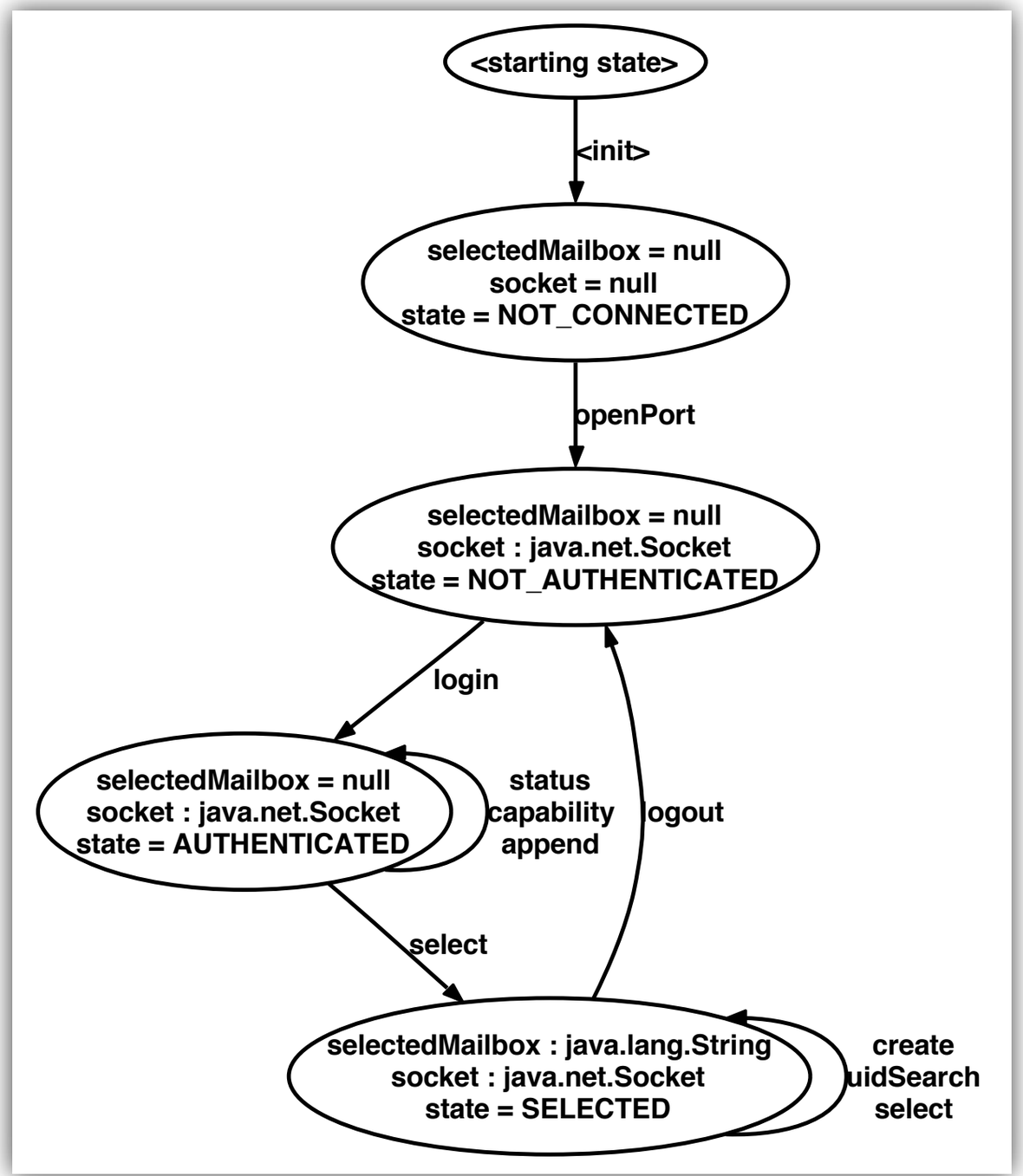
Application: understanding behavior of a class

org.aspectj.asm.StructureModel



Application: understanding behavior of a class

org.columba.ristretto.imap.IMAPProtocol

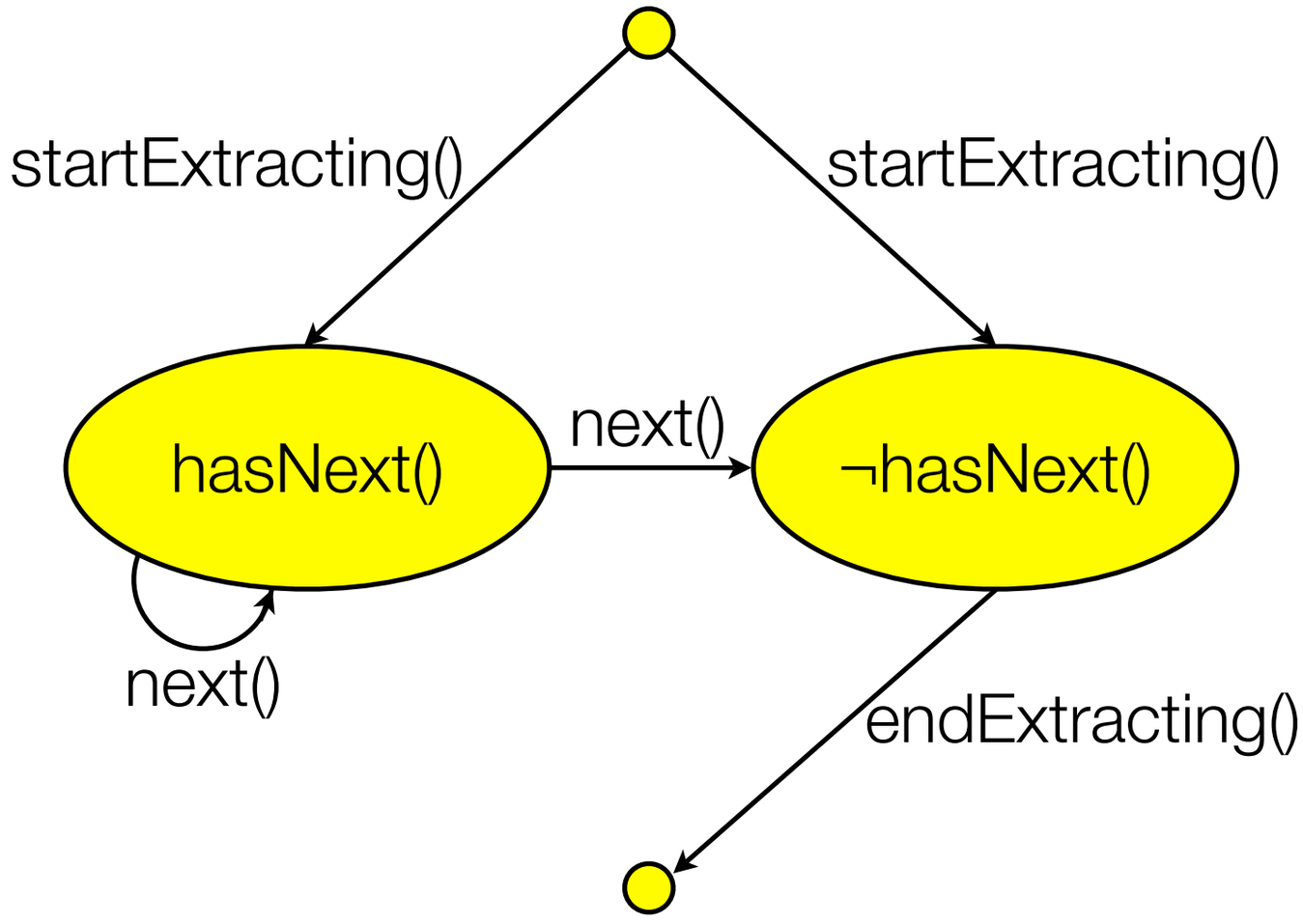


Other applications

- Checking behavior of objects with respect to models at runtime
- Finding possible model violations in a source code
- Suggesting method calls during development

Future work: mining models statically

```
public void foo (Data data) {
    data.startExtracting ();
    while (data.hasNext ())
        process (data.next ());
    data.endExtracting ();
}
```



Mining models statically: challenges

- How to recognize object's state?
  ```
  while (a.foo () || b.bar ()) {    // what is the state of a?
  ```

- How to merge models based on a control flow?
  ```
  public void foo (Data data) {
      ...
      foo (data);
      ...
  }
  ```

- How can we deal with possible aliases? Points-to analysis?
  ```
  a.foo (); b.bar ();
  ```

- Can we combine models mined statically with those mined dynamically?

Conclusions

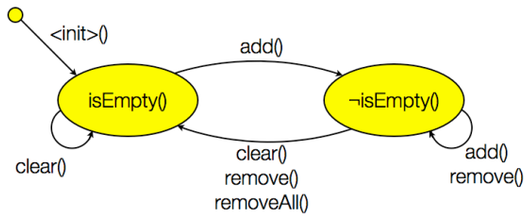
The problem

java.util.Vector
boolean add(Object o)
void clear()
boolean isEmpty()
Object remove(int index)
boolean removeAll(Collection c)
…

- What is the effect of calling ‘clear’?
- Can ‘remove’ be called at any time?

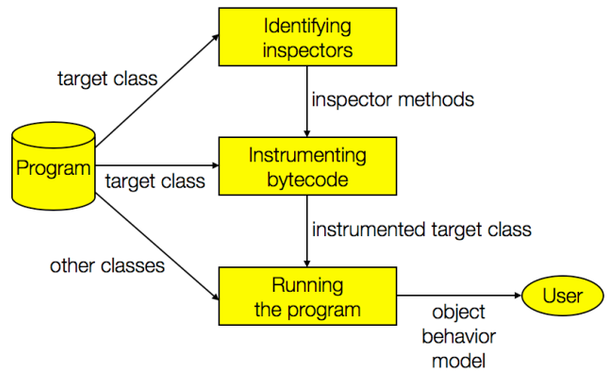
How to use a class correctly?

What do object behavior models look like?

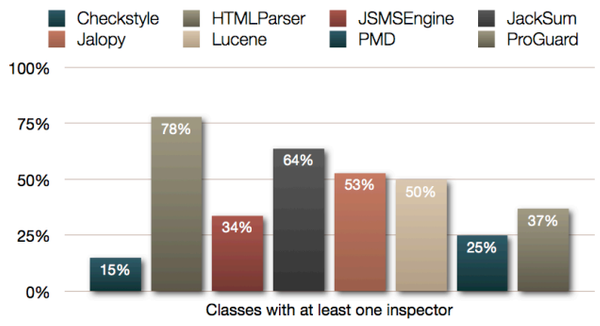


Object behavior model for the java.util.Vector class

How can we mine object behavior models?

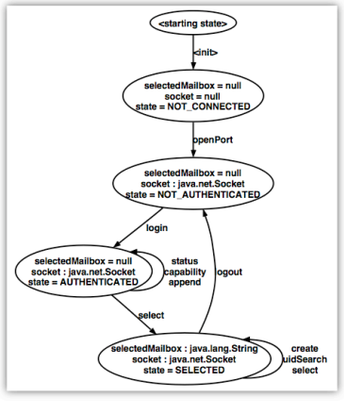


How common are classes with inspectors?



Application: understanding behavior of a class

org.columba.ristretto.imap.IMAPProtocol



Future work: mining models statically

```
public void foo (Data data) {
   data.startExtracting ();
   while (data.hasNext ())
      process (data.next ());
   data.endExtracting ();
}
```