Detecting Object Usage Anomalies

Andrzej Wasylkowski
Andreas Zeller
Christian Lindig

Saarland University



Some AspectJ method

```
private boolean verifyNIAP (…) {
    …
    Iterator iter = …;
    while (iter.hasNext()) {
        … = iter.next();
        …
        return verifyNIAP (…);
    }
    return true;
}
```

Some AspectJ method

```
private boolean verifyNIAP (…) {
    …
    Iterator iter = …;
    while (iter.hasNext()) {
        … = iter.next();
        …
        return verifyNIAP (…);
    }
    return true;
}
```

Never loops!

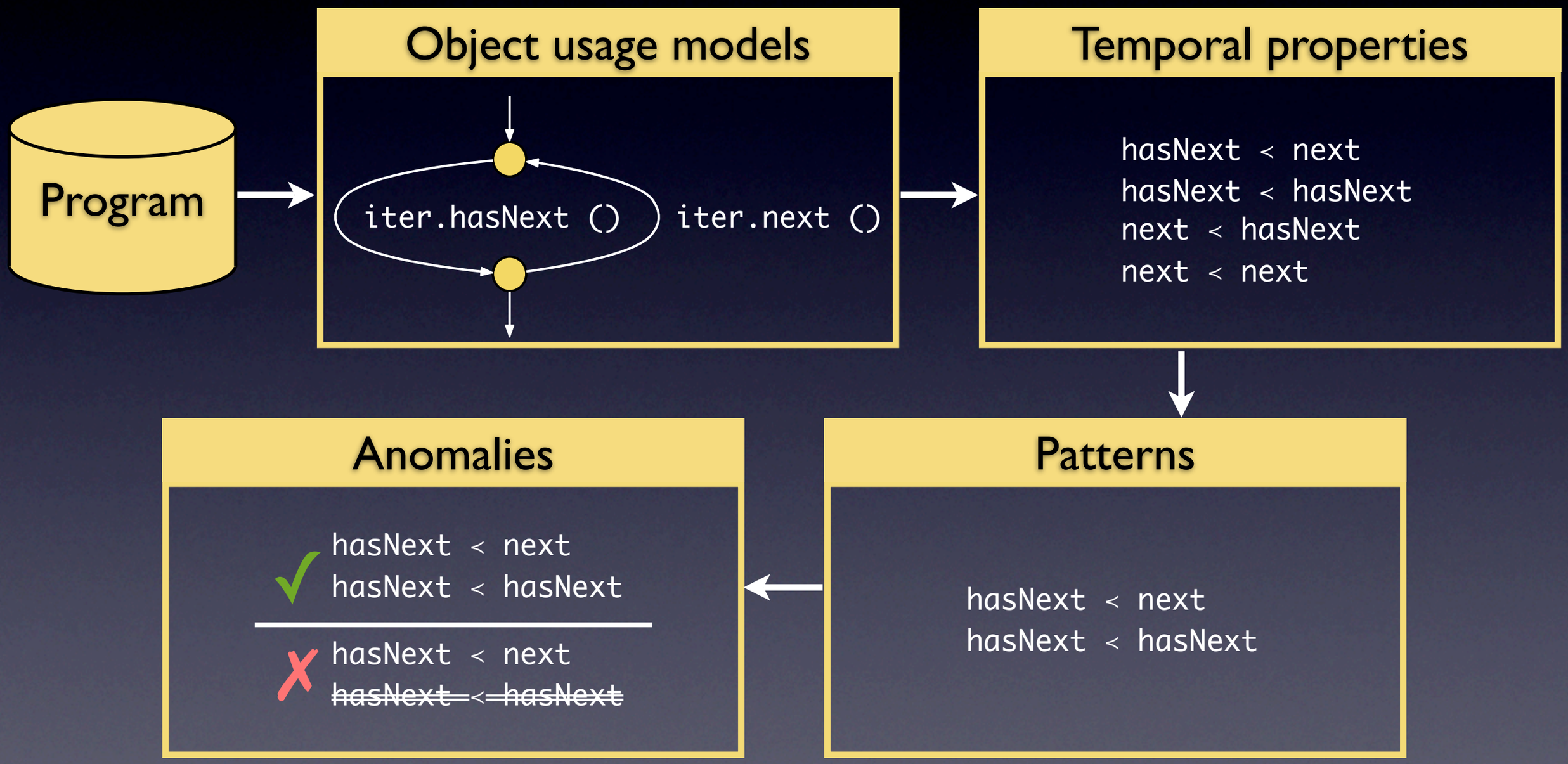
Some AspectJ method

```
private boolean verifyNIAP (…) {
    …
    Iterator iter = …;
    while (iter.hasNext()) {
        … = iter.next();
        …
        return verifyNIAP (…);
    }
    return true;
}
```

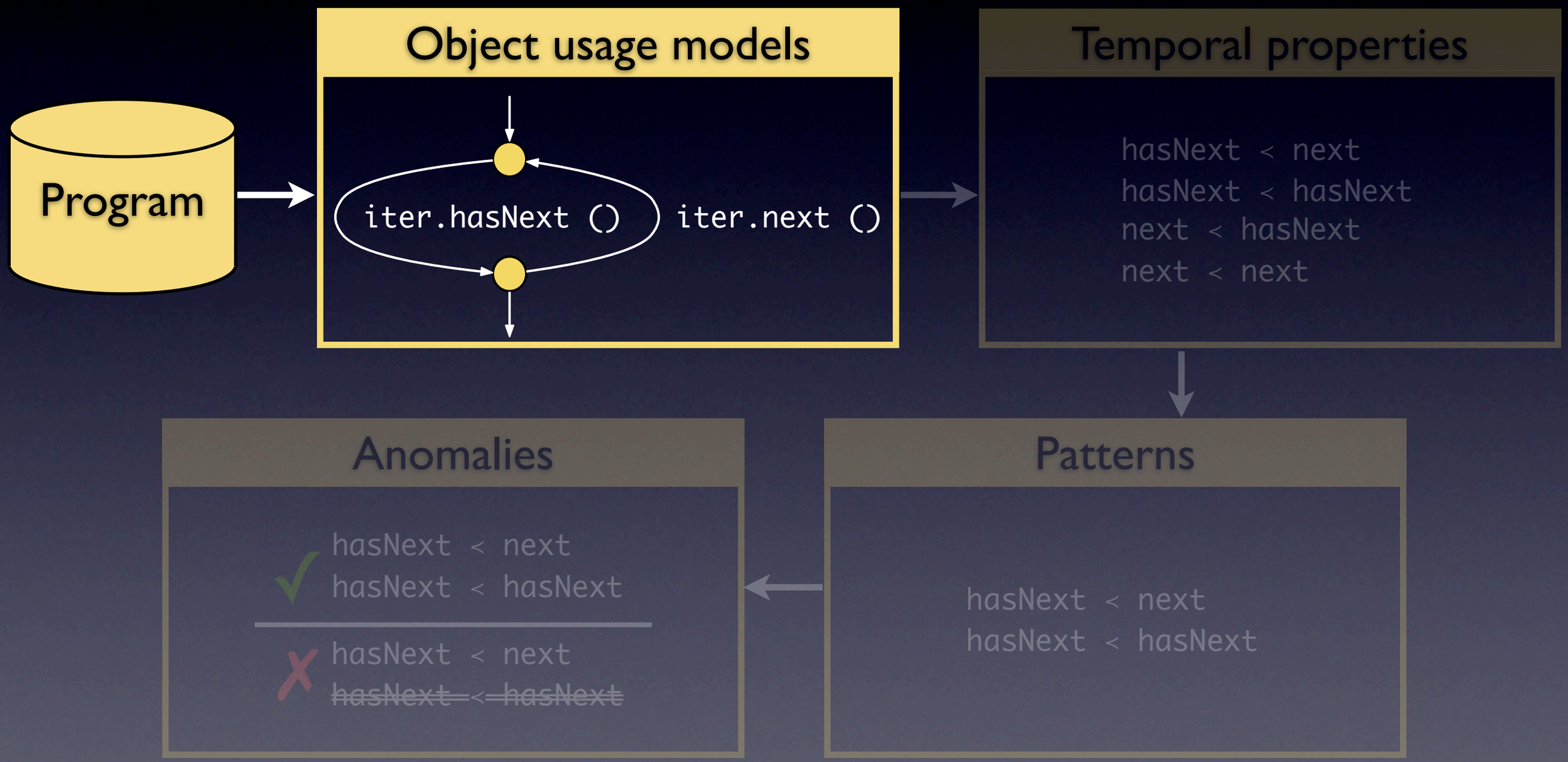
Never loops!

Violates typical iterator usage pattern

JADET


Program
Object usage models
iter.hasNext ()
iter.next ()
Temporal properties
hasNext ≺ next
hasNext ≺ hasNext
next ≺ hasNext
next ≺ next
Patterns
hasNext ≺ next
hasNext ≺ hasNext
Anomalies
hasNext ≺ next
hasNext ≺ hasNext
hasNext ≺ next
hasNext ≺ hasNext

JADET


Program
Object usage models
iter.hasNext ()
iter.next ()
Temporal properties
hasNext < next
hasNext < hasNext
next < hasNext
next < next
Patterns
hasNext < next
hasNext < hasNext
Anomalies
hasNext < next
hasNext < hasNext
hasNext < next
hasNext < hasNext

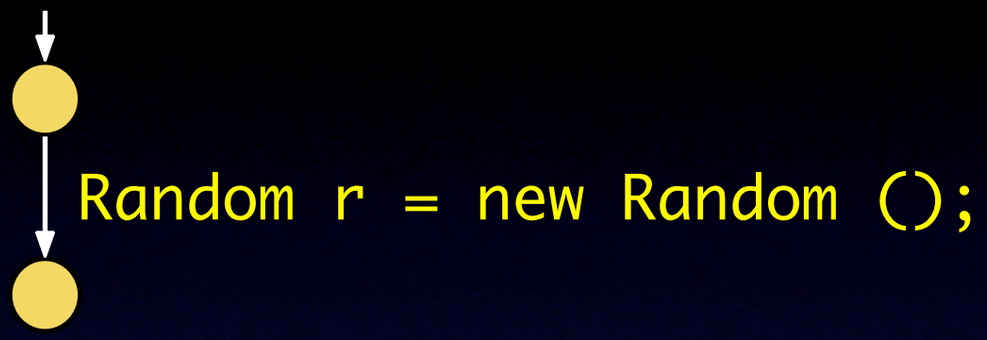
Creating a method model

```
public Stack createStack () {
  Random r = new Random ();
  int n = r.nextInt ();
  Stack s = new Stack ();
  int i = 0;
  while (i < n) {
    s.push (rand (r));
    i++;
  }
  s.push (-1);
  return s;
}
```

Creating a method model

```
public Stack createStack () {
  Random r = new Random ();
  int n = r.nextInt ();
  Stack s = new Stack ();
  int i = 0;
  while (i < n) {
    s.push (rand (r));
    i++;
  }
  s.push (-1);
  return s;
}
```

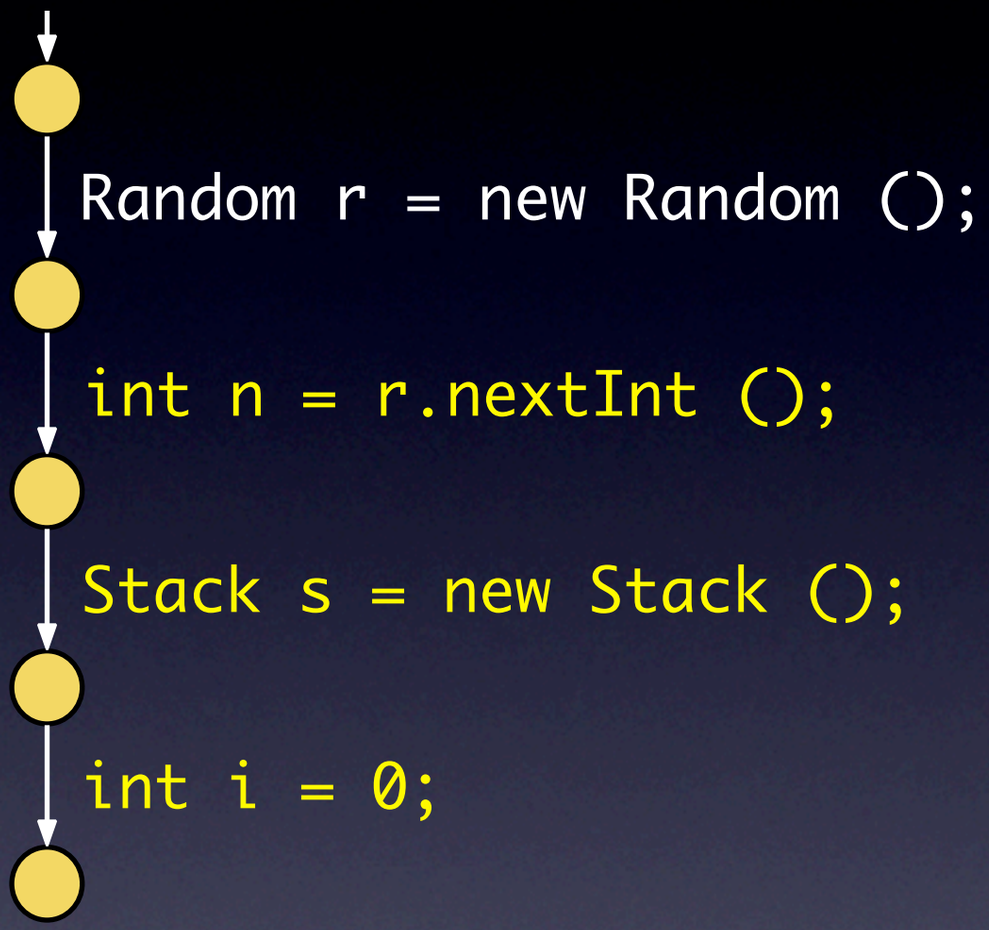
Creating a method model




```
public Stack createStack () {
  Random r = new Random ();
  int n = r.nextInt ();
  Stack s = new Stack ();
  int i = 0;
  while (i < n) {
    s.push (rand (r));
    i++;
  }
  s.push (-1);
  return s;
}
```

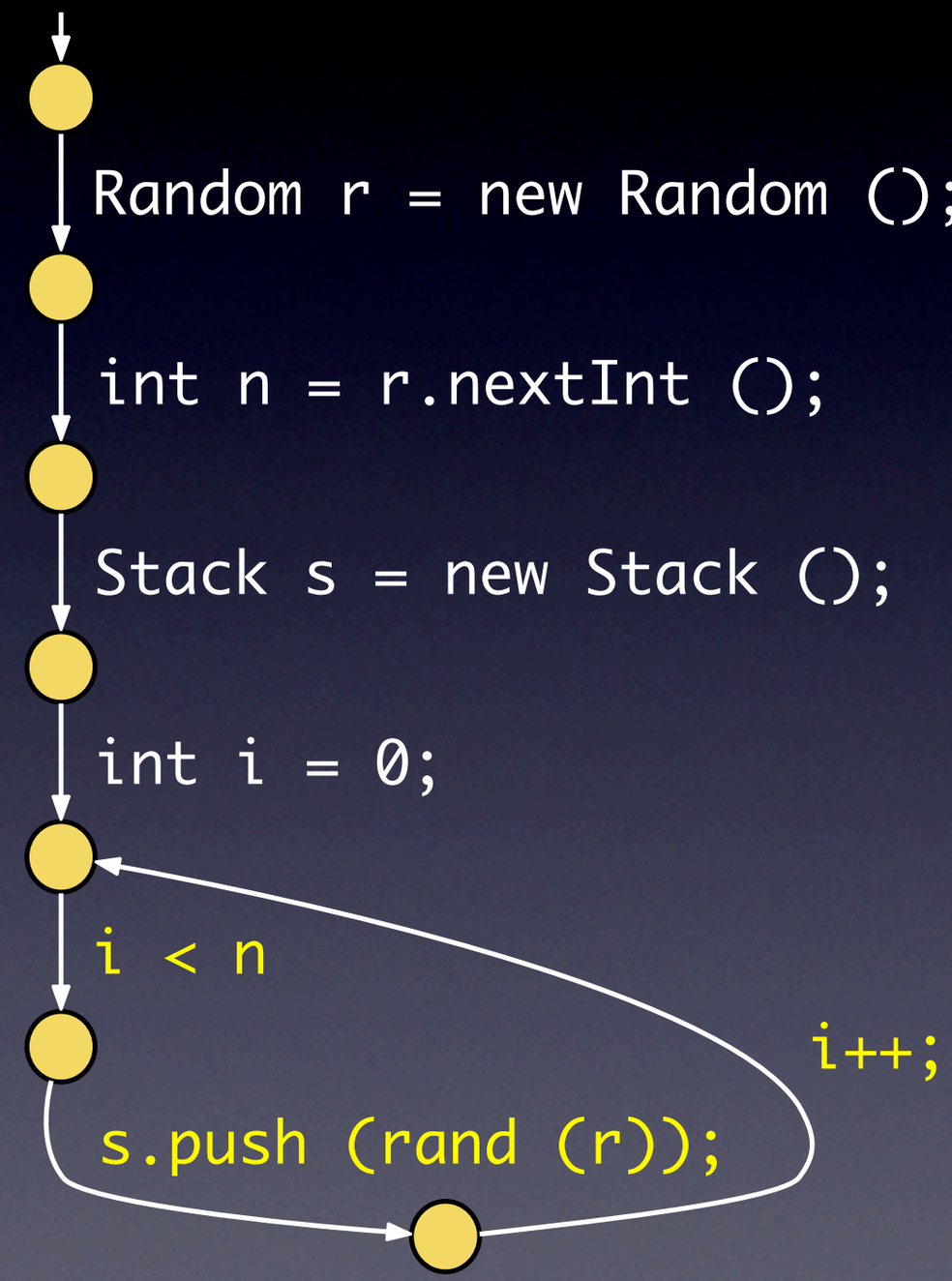
Creating a method model

```
public Stack createStack () {
  Random r = new Random ();
  int n = r.nextInt ();
  Stack s = new Stack ();
  int i = 0;
  while (i < n) {
    s.push (rand (r));
    i++;
  }
  s.push (-1);
  return s;
}
```



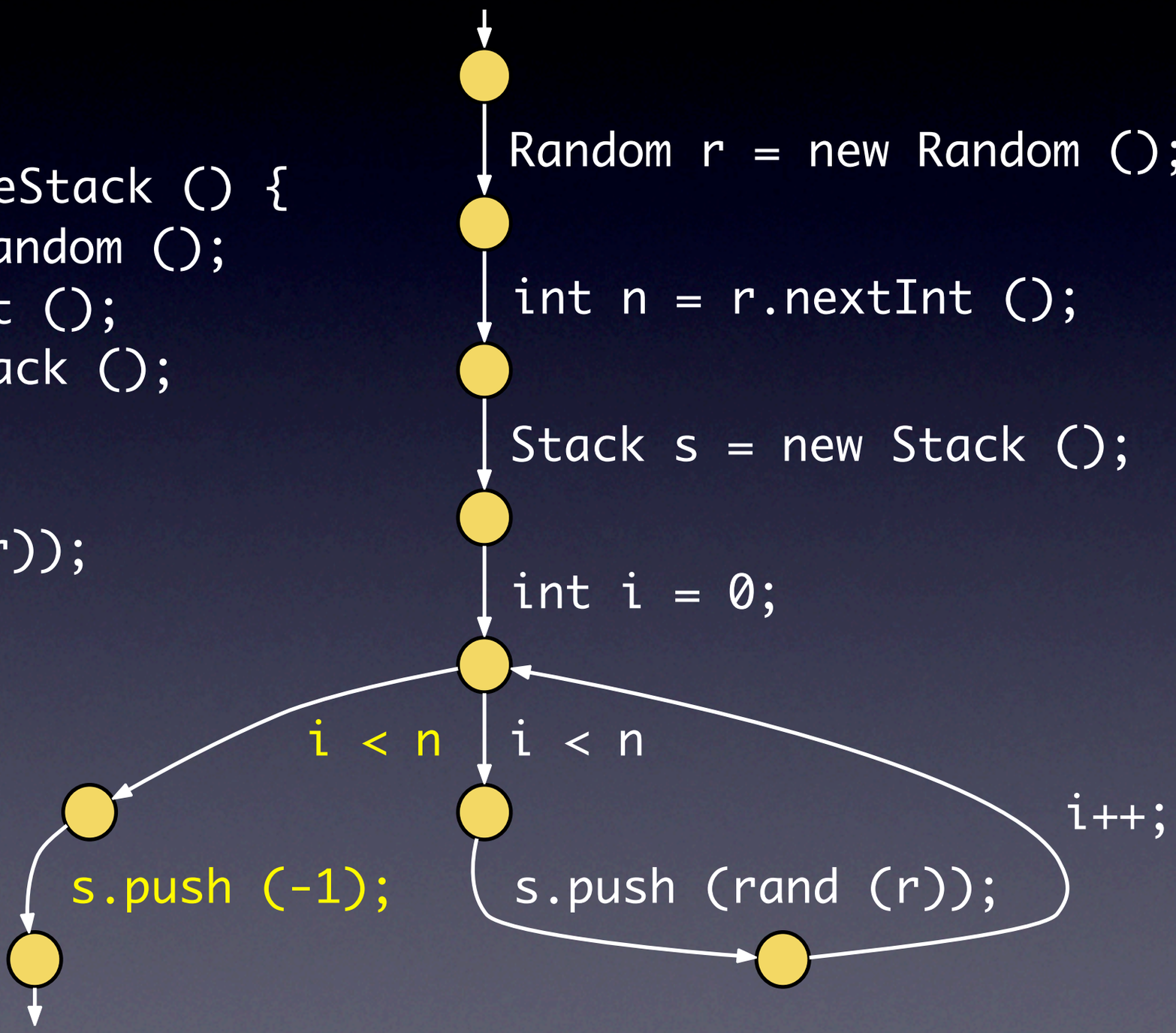
Creating a method model

```
public Stack createStack () {
  Random r = new Random ();
  int n = r.nextInt ();
  Stack s = new Stack ();
  int i = 0;
  while (i < n) {
    s.push (rand (r));
    i++;
  }
  s.push (-1);
  return s;
}
```



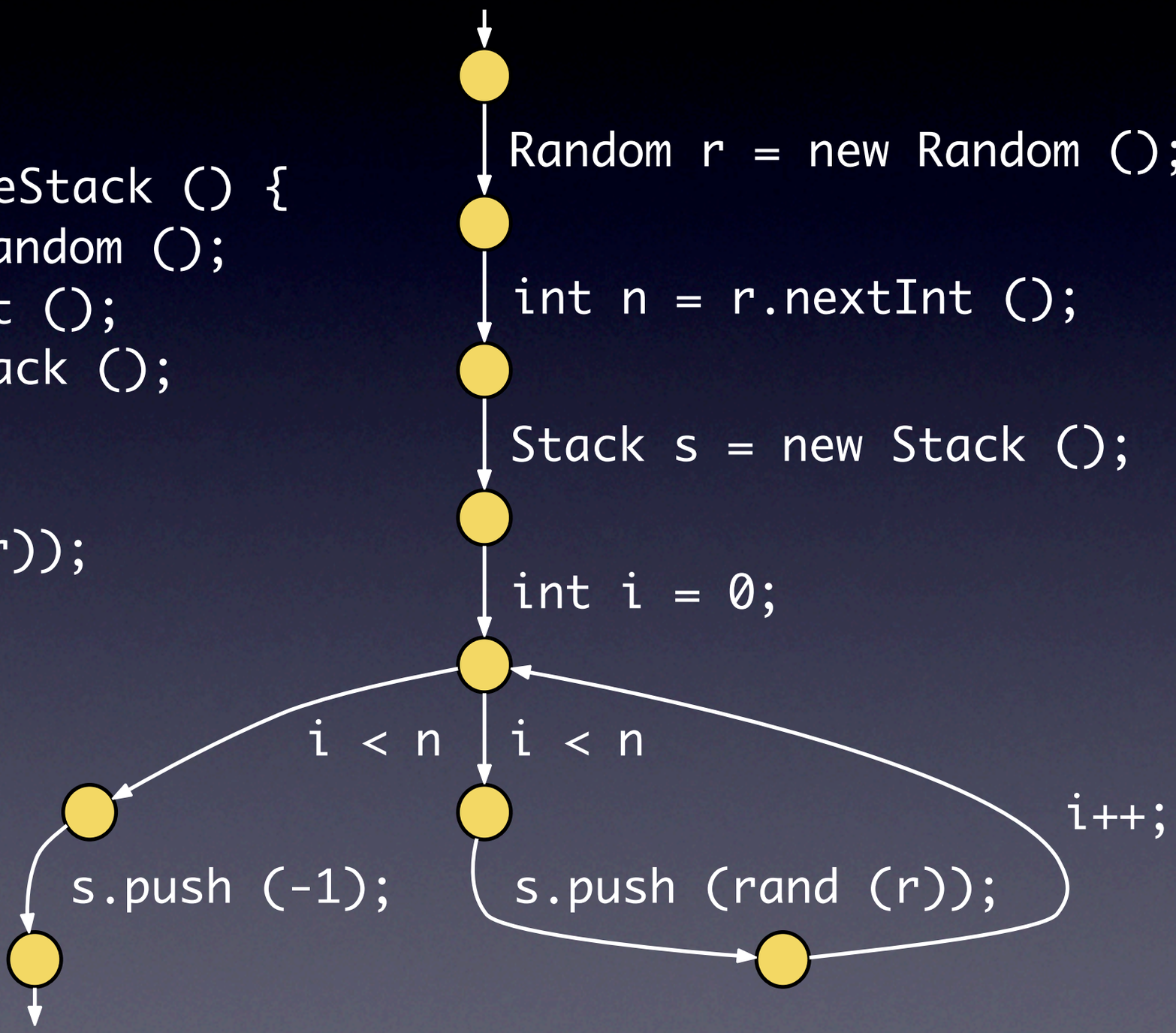
Creating a method model

```
public Stack createStack () {
  Random r = new Random ();
  int n = r.nextInt ();
  Stack s = new Stack ();
  int i = 0;
  while (i < n) {
    s.push (rand (r));
    i++;
  }
  s.push (-1);
  return s;
}
```

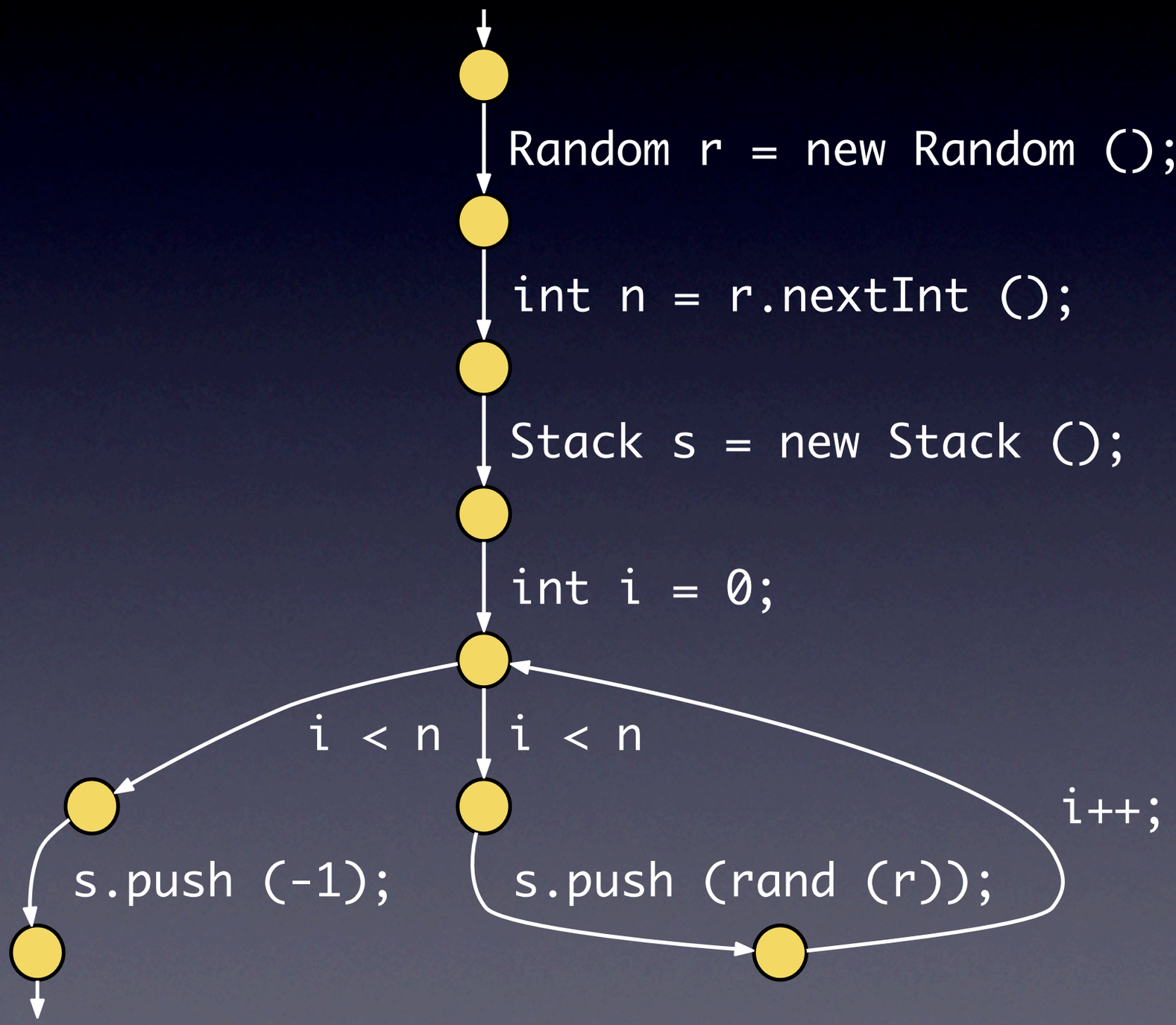


Creating a method model

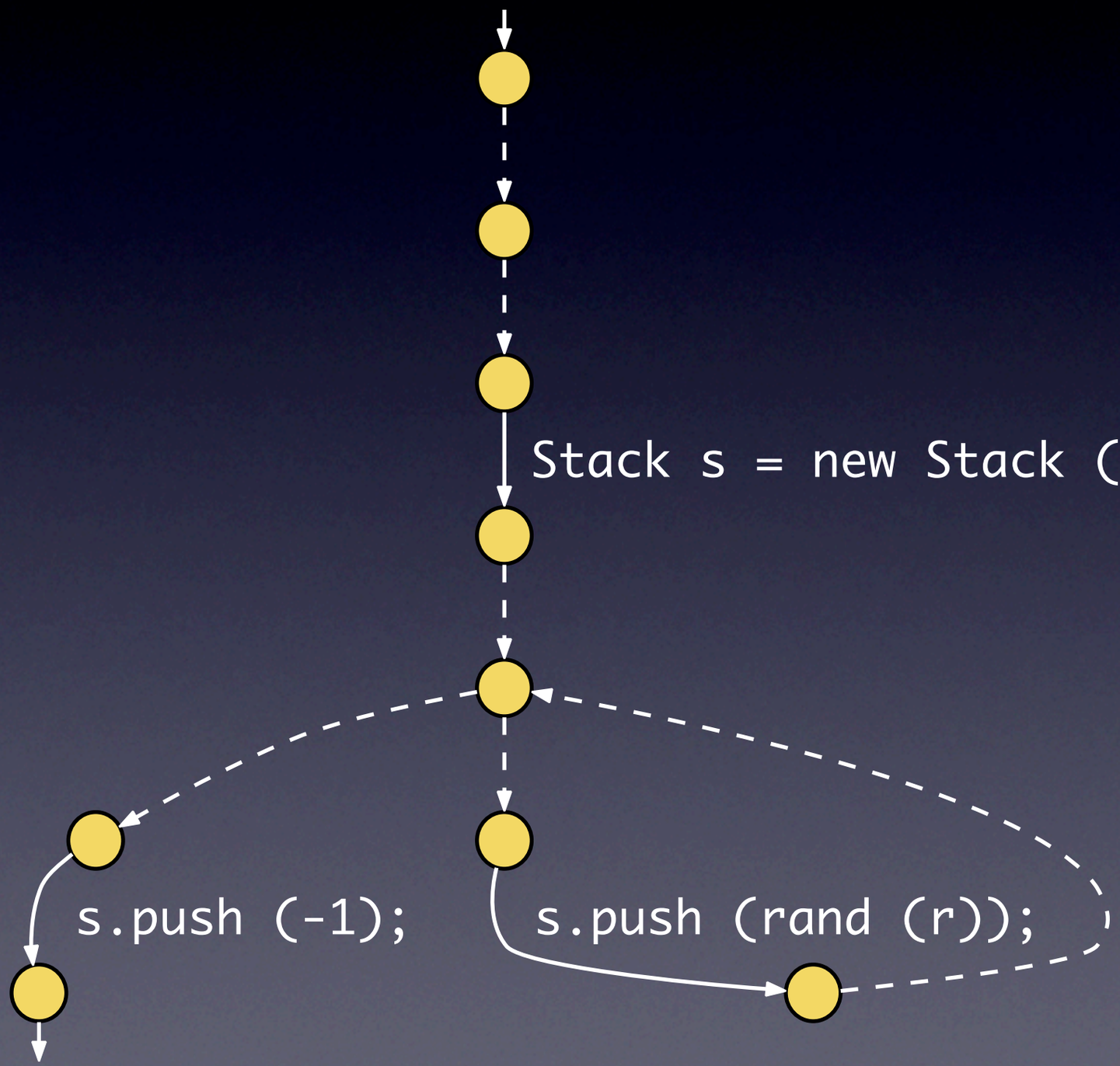
```
public Stack createStack () {
  Random r = new Random ();
  int n = r.nextInt ();
  Stack s = new Stack ();
  int i = 0;
  while (i < n) {
    s.push (rand (r));
    i++;
  }
  s.push (-1);
  return s;
}
```



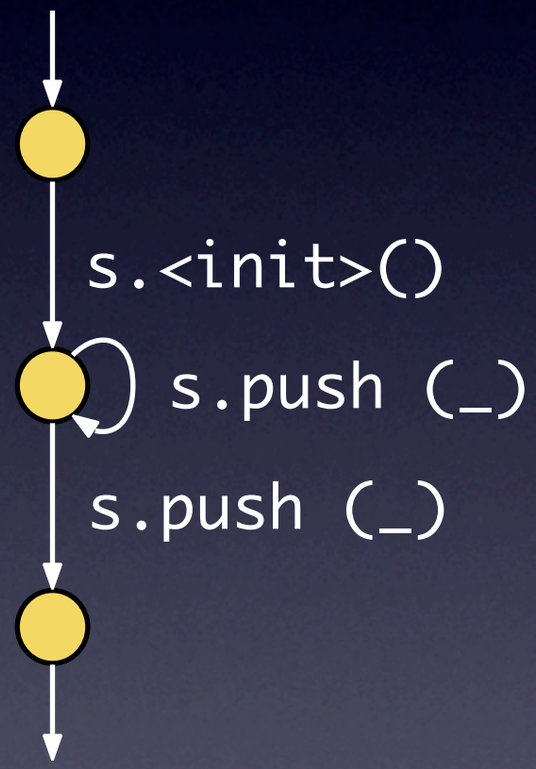
Creating a usage model



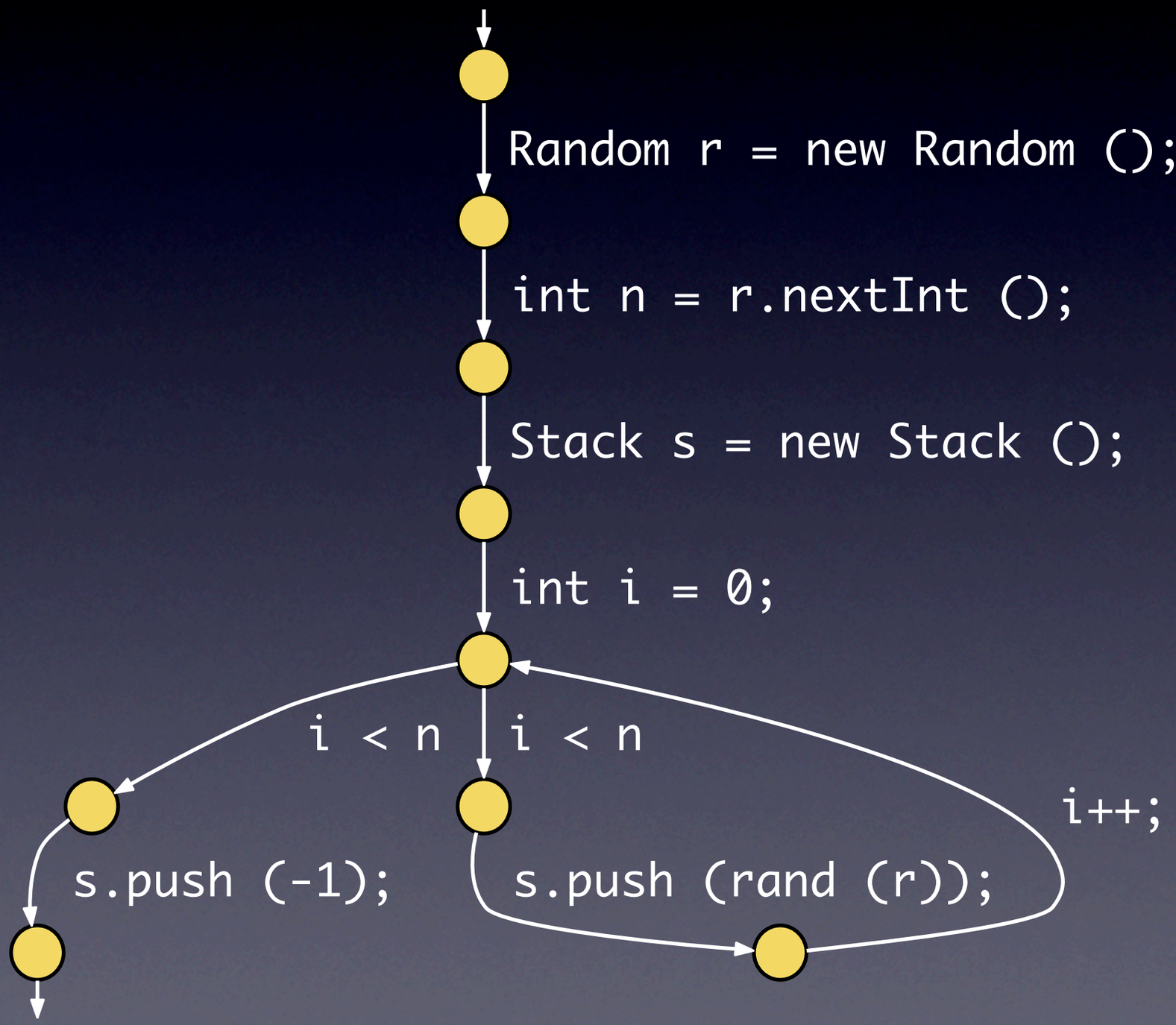
Creating a usage model



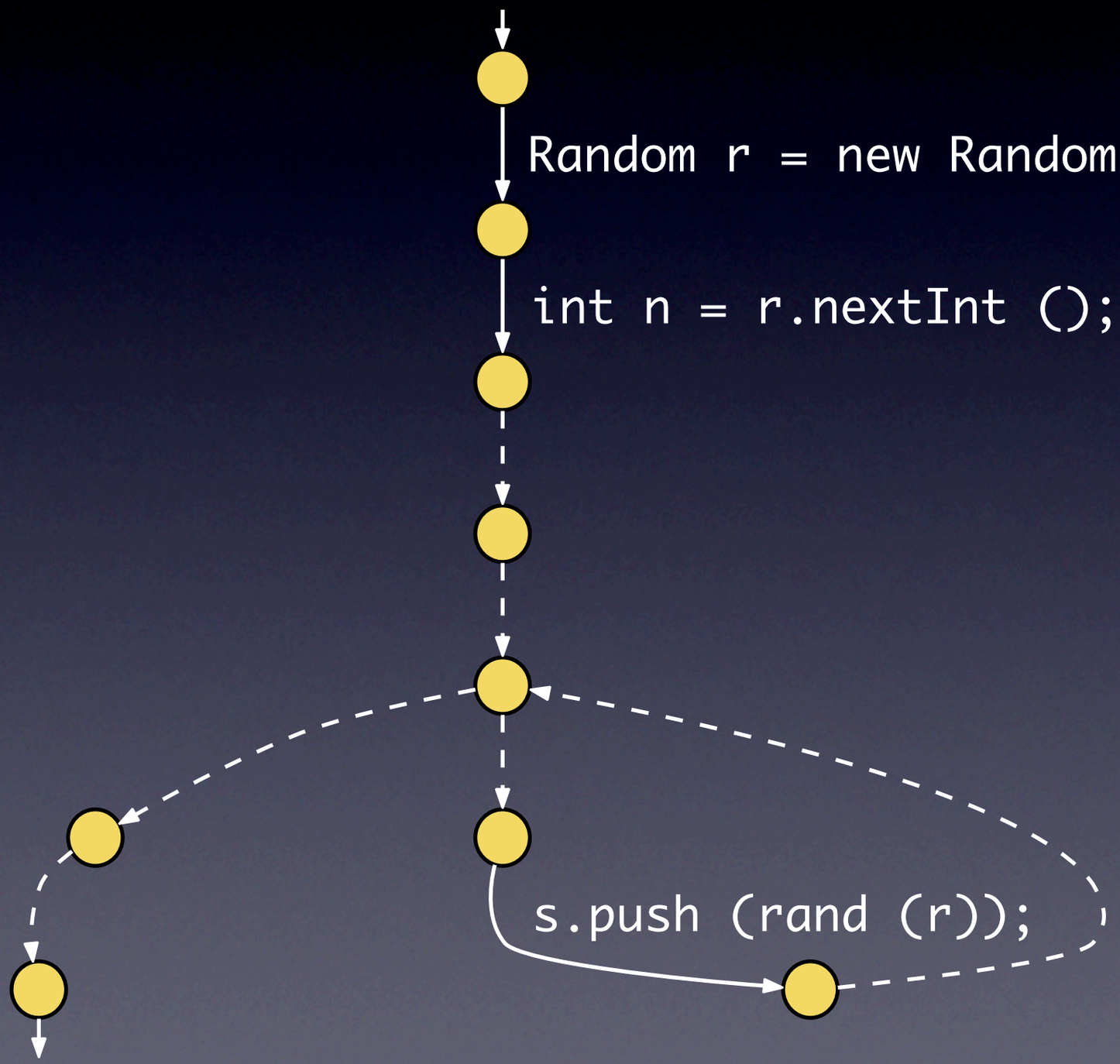
Creating a usage model



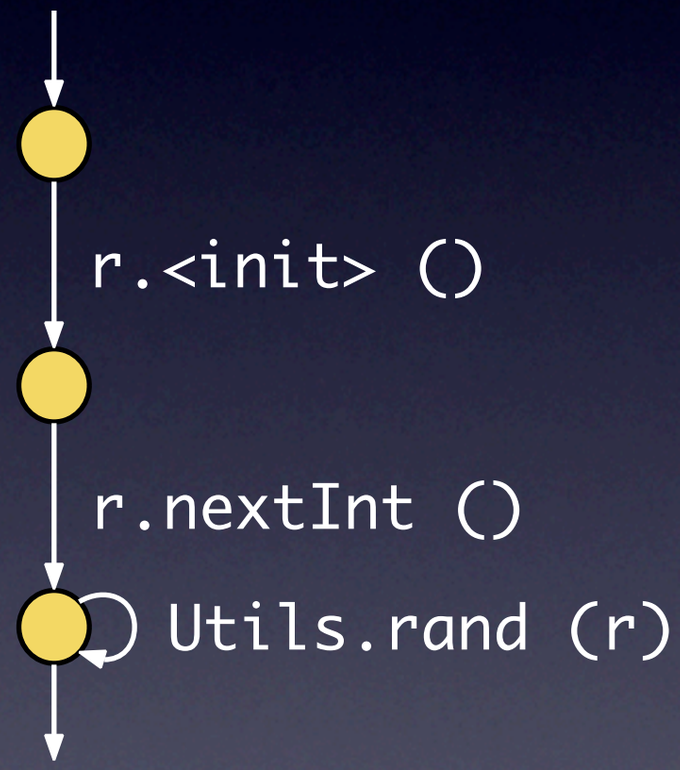
Creating a usage model



Creating a usage model

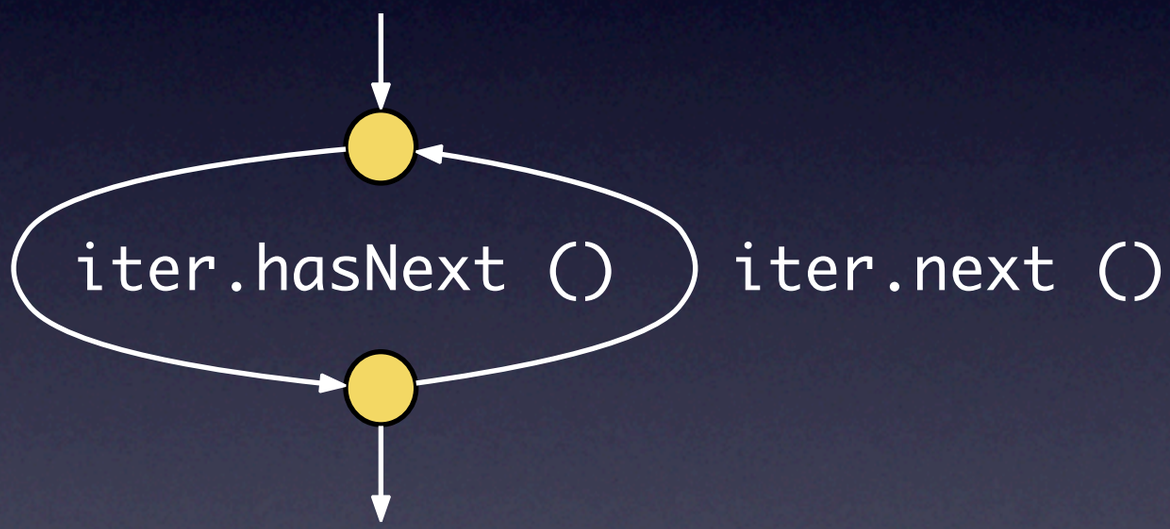


Creating a usage model



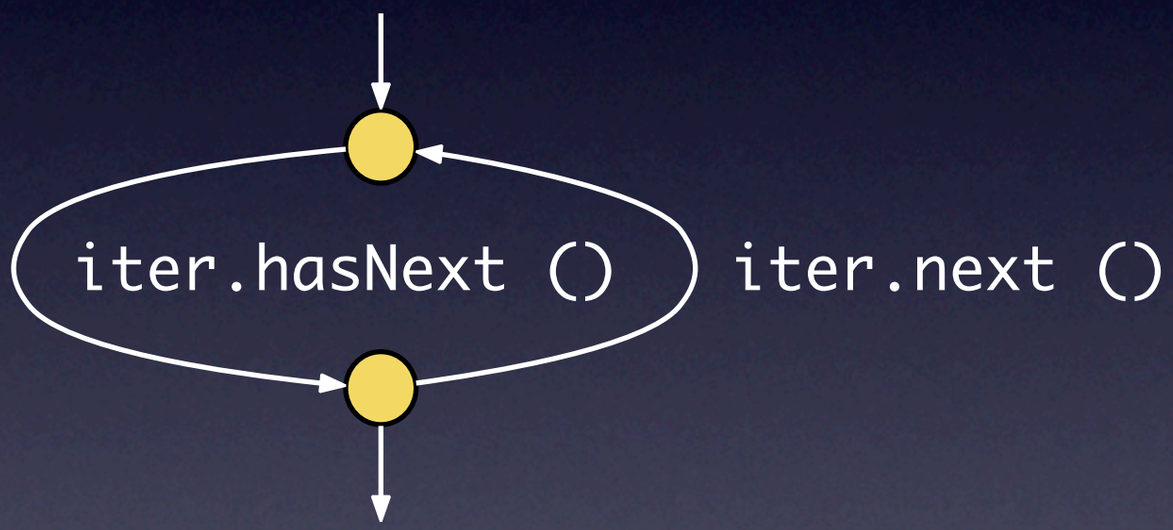
Iterator models

Typical



Iterator models

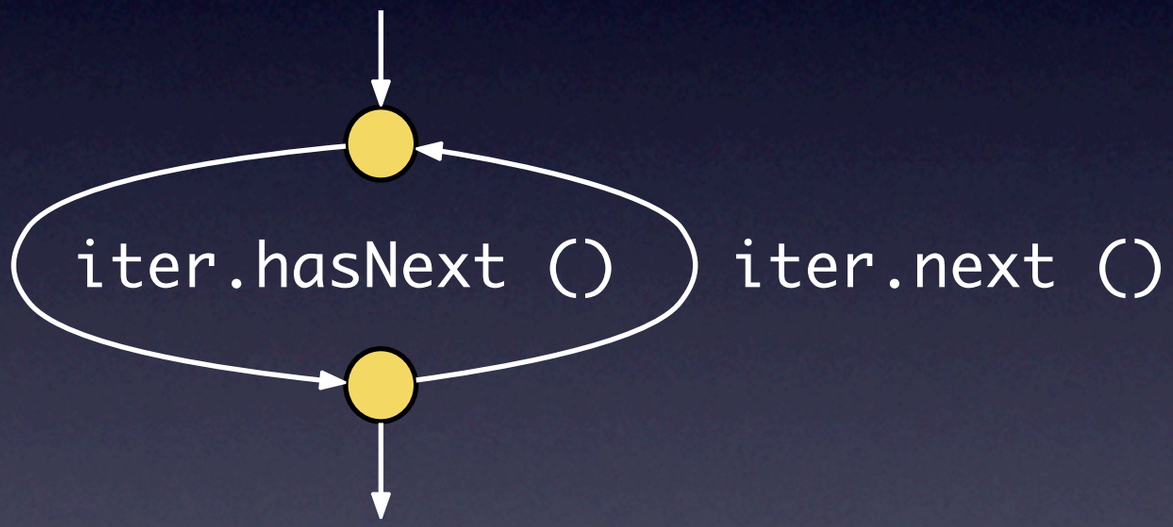
Typical



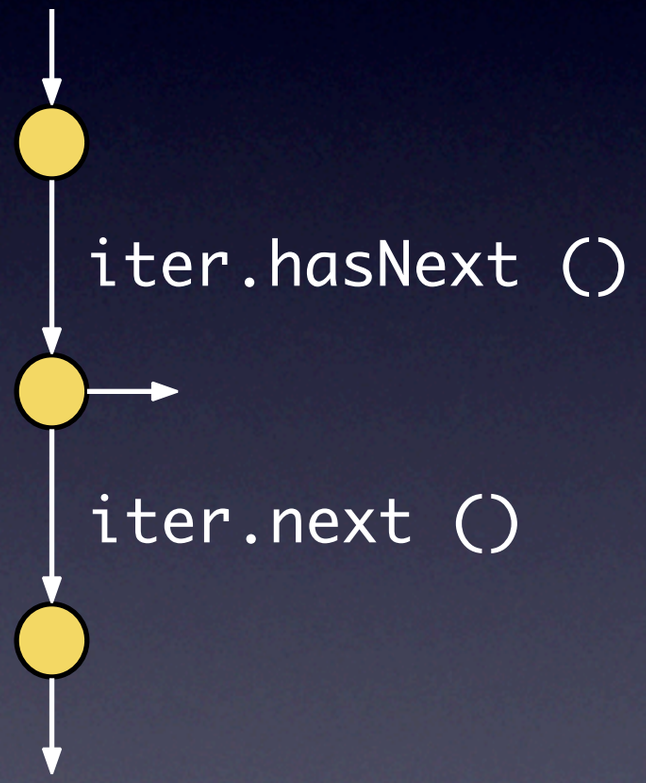

```
private boolean verifyNIAP (…) {
    …
    Iterator iter = …;
    while (iter.hasNext()) {
        … = iter.next();
        …
        return verifyNIAP (…);
    }
    return true;
}
```

Iterator models

Typical

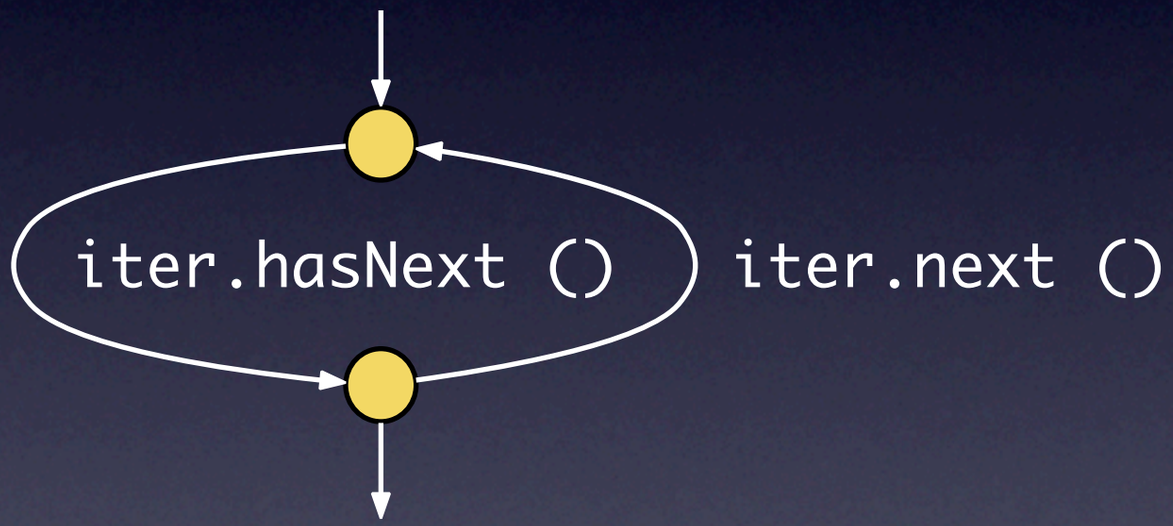


Anomalous

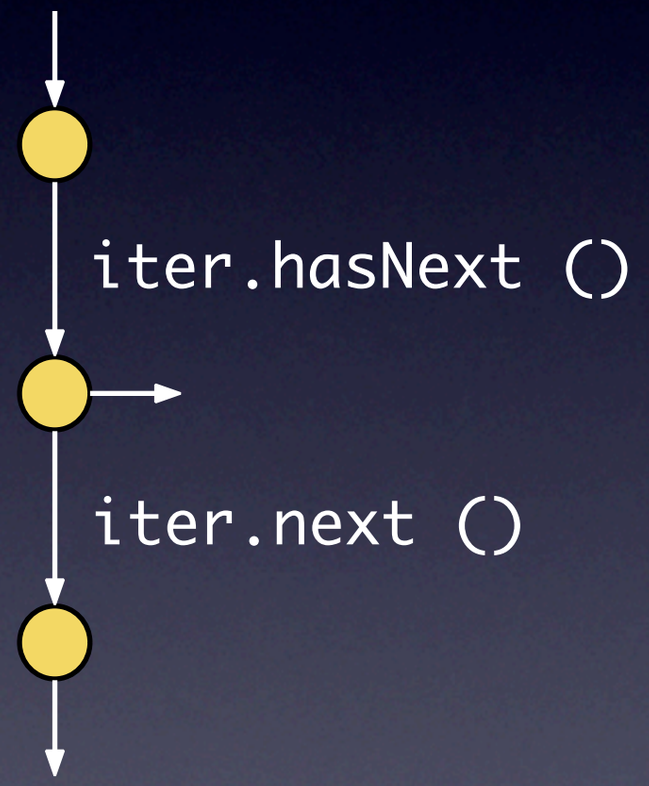


Iterator models

Typical

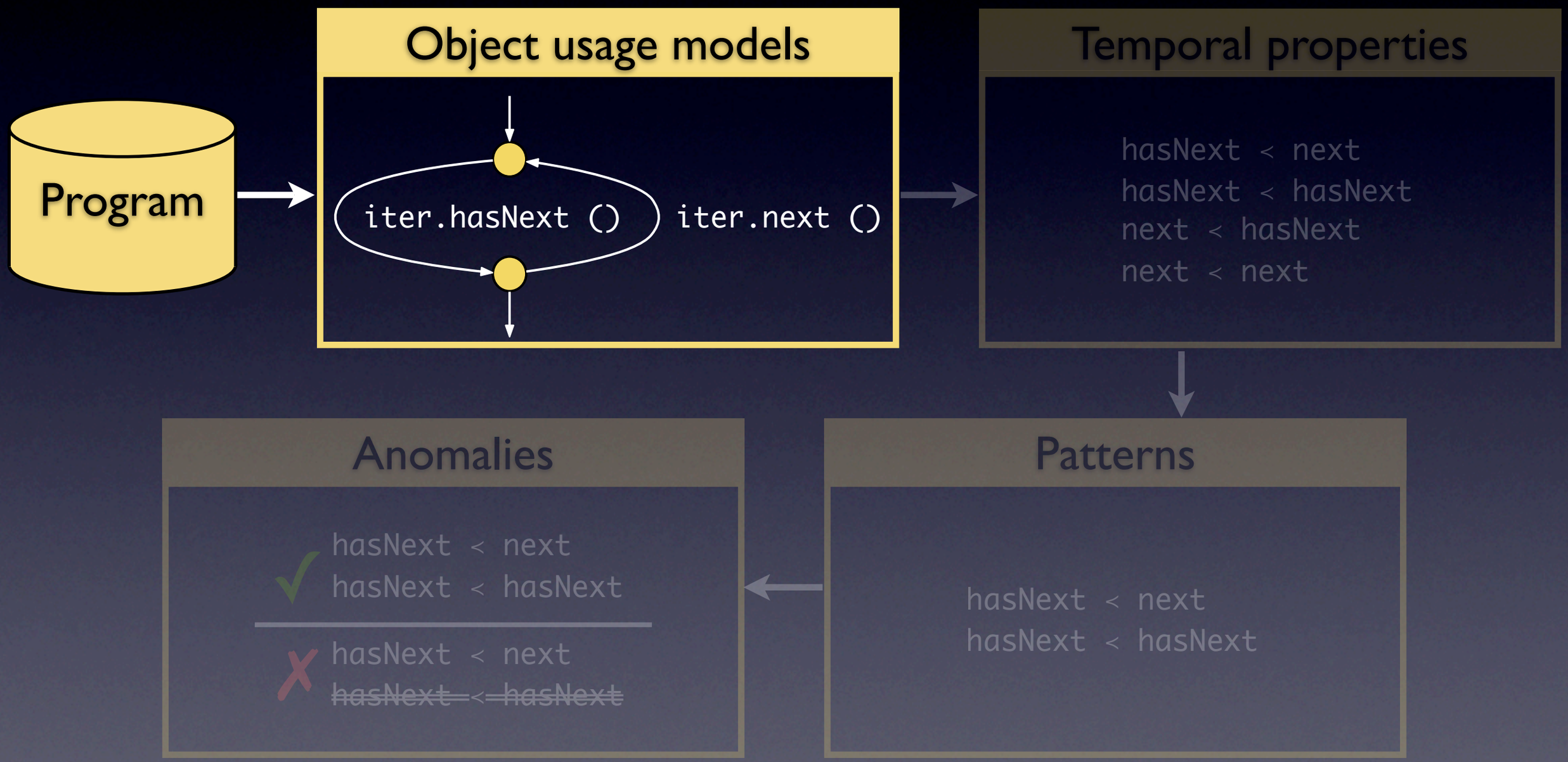


Anomalous

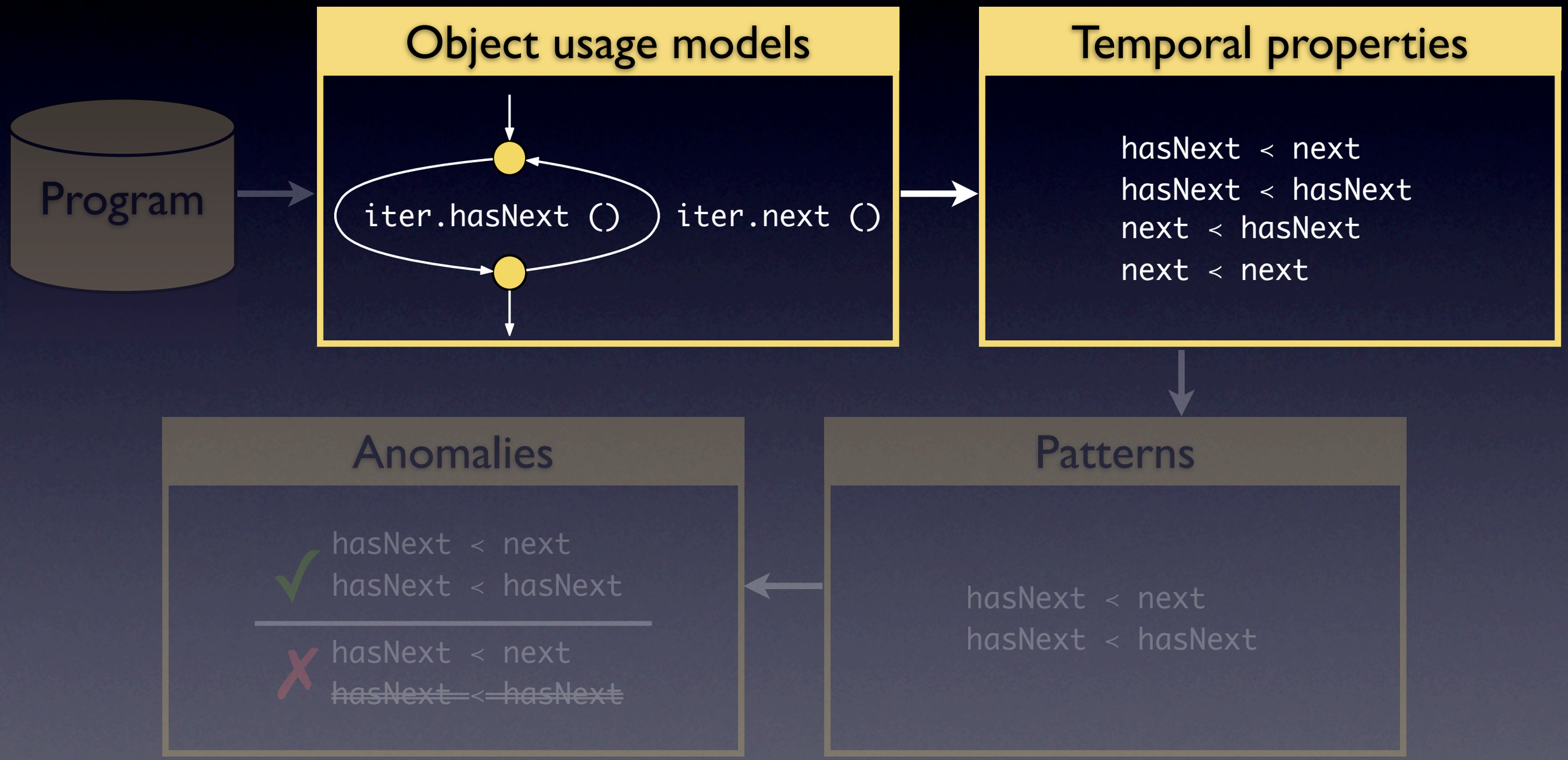


How do we quantify such difference?

JADET


Program
Object usage models
iter.hasNext ()
iter.next ()
Temporal properties
hasNext < next
hasNext < hasNext
next < hasNext
next < next
Patterns
hasNext < next
hasNext < hasNext
Anomalies
hasNext < next
hasNext < hasNext
hasNext < next
hasNext < hasNext

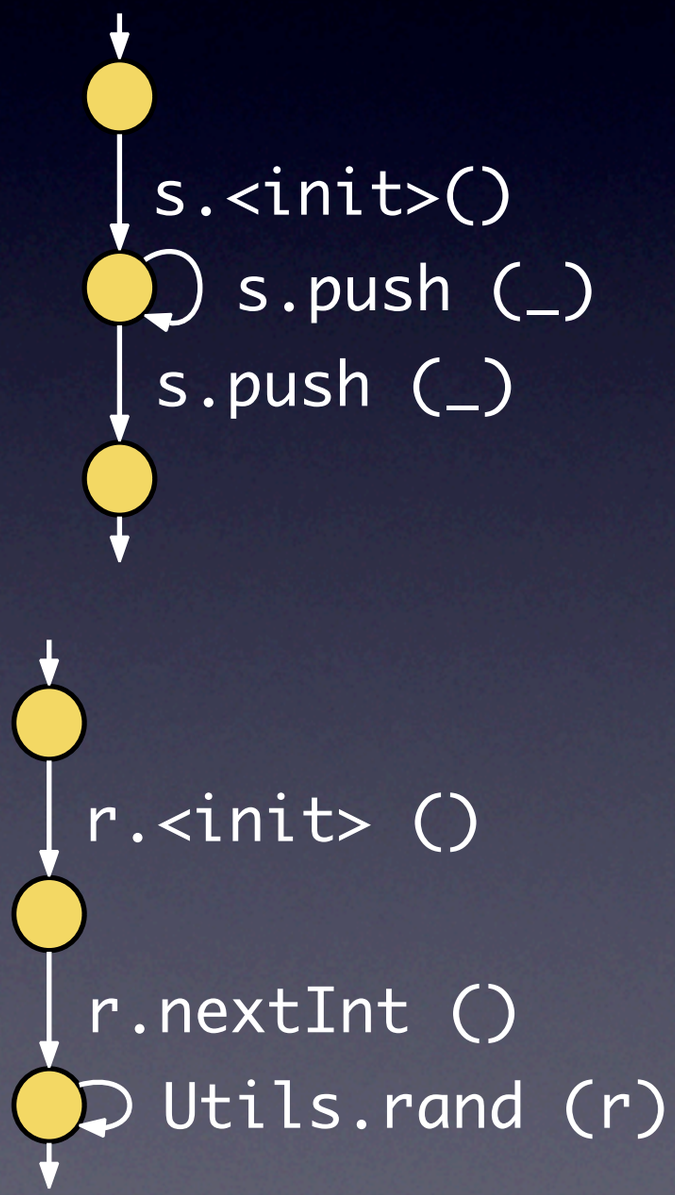
JADET


Program
Object usage models
iter.hasNext ()
iter.next ()
Temporal properties
hasNext ≺ next
hasNext ≺ hasNext
next ≺ hasNext
next ≺ next
Patterns
hasNext ≺ next
hasNext ≺ hasNext
Anomalies
hasNext ≺ next
hasNext ≺ hasNext
hasNext ≺ next
hasNext ≺ hasNext

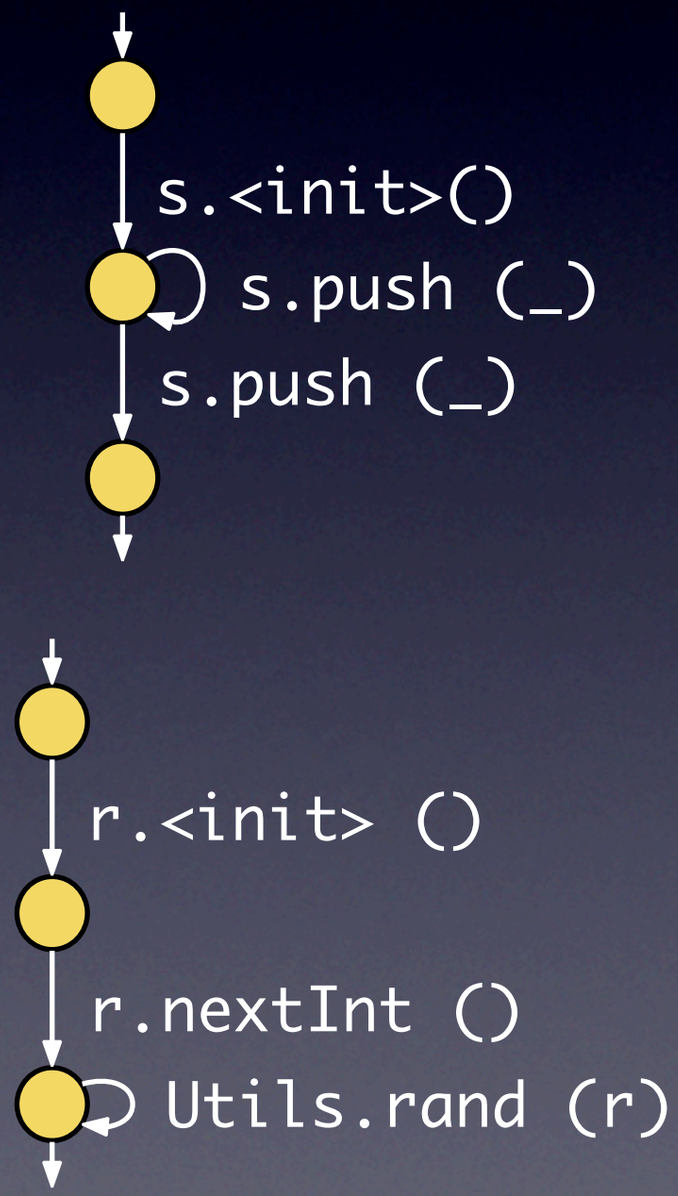
Extracting temporal properties

```
public Stack createStack () {
  Random r = new Random ();
  int n = r.nextInt ();
  Stack s = new Stack ();
  int i = 0;
  while (i < n) {
    s.push (rand (r));
    i++;
  }
  s.push (-1);
  return s;
}
```

Extracting temporal properties

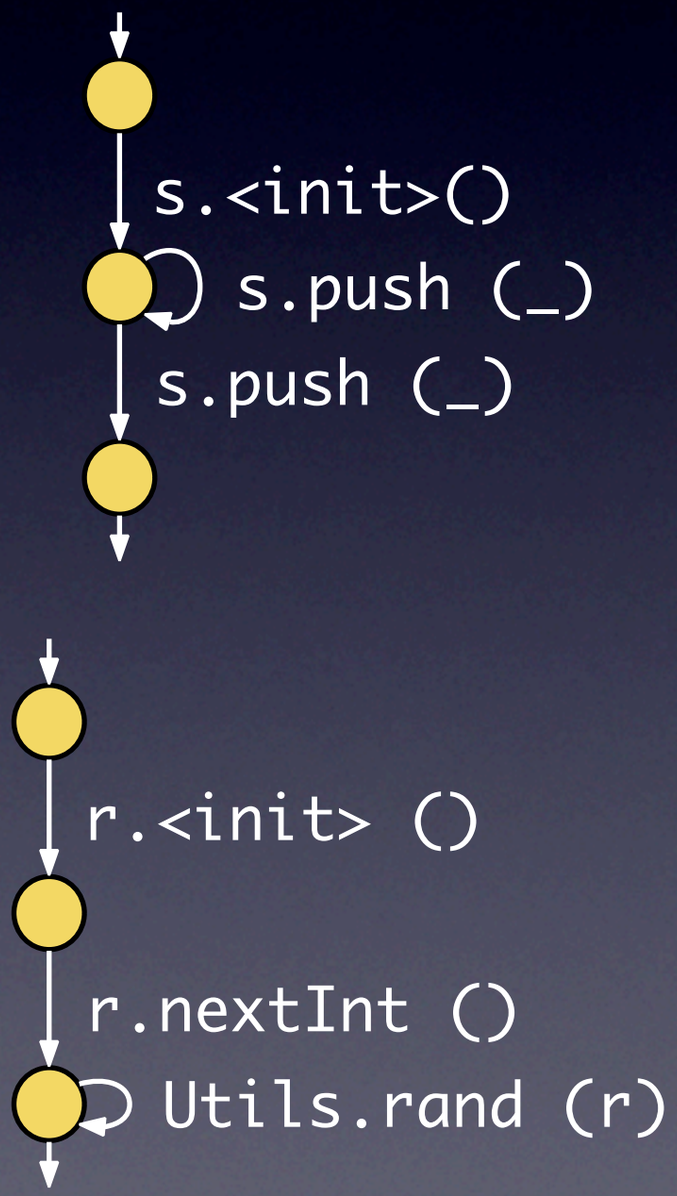


Extracting temporal properties




```
Stack.<init> ≺ Stack.push
Stack.push ≺ Stack.push
```

Extracting temporal properties




```
Stack.<init> ≺ Stack.push
Stack.push ≺ Stack.push
```

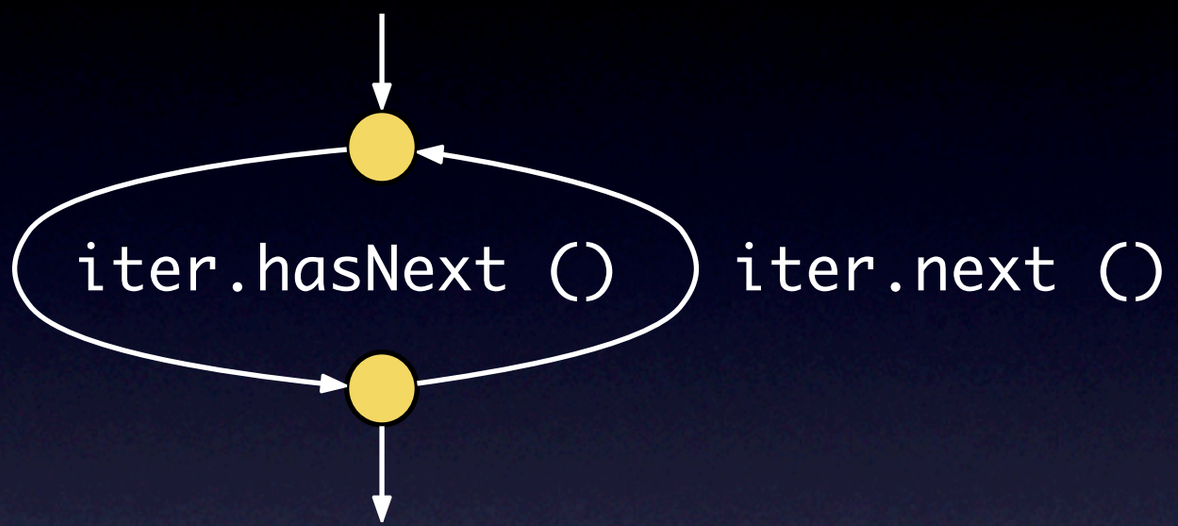
```
Random.<init> ≺ Random.nextInt
Random.<init> ≺ Utils.rand
Random.nextInt ≺ Utils.rand
Utils.rand ≺ Utils.rand
```

Extracting temporal properties

```
public Stack createStack () {
  Random r = new Random ();
  int n = r.nextInt ();
  Stack s = new Stack ();
  int i = 0;
  while (i < n) {
    s.push (rand (r));
    i++;
  }
  s.push (-1);
  return s;
}
```

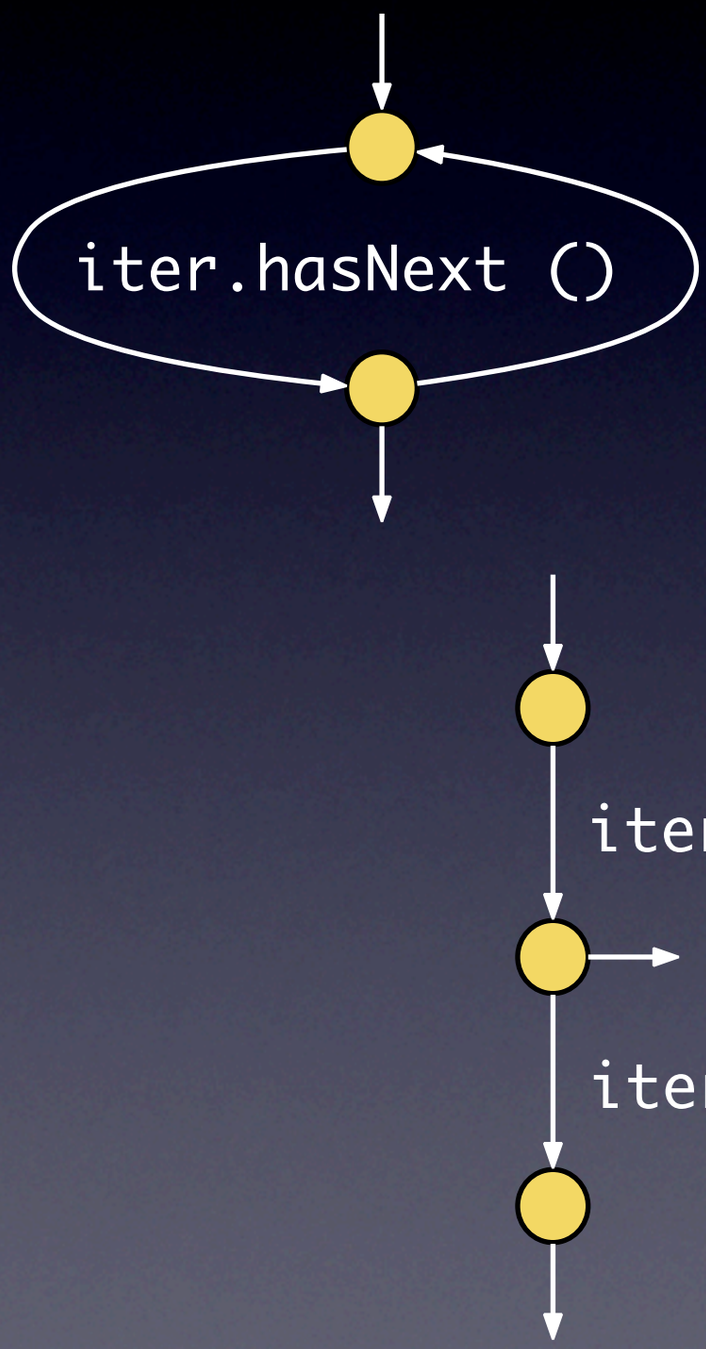
```
Stack.<init> ≺ Stack.push
Stack.push ≺ Stack.push
Random.<init> ≺ Random.nextInt
Random.<init> ≺ Utils.rand
Random.nextInt ≺ Utils.rand
Utils.rand ≺ Utils.rand
```

The Iterator properties




```
hasNext < next
hasNext < hasNext
next < hasNext
next < next
```

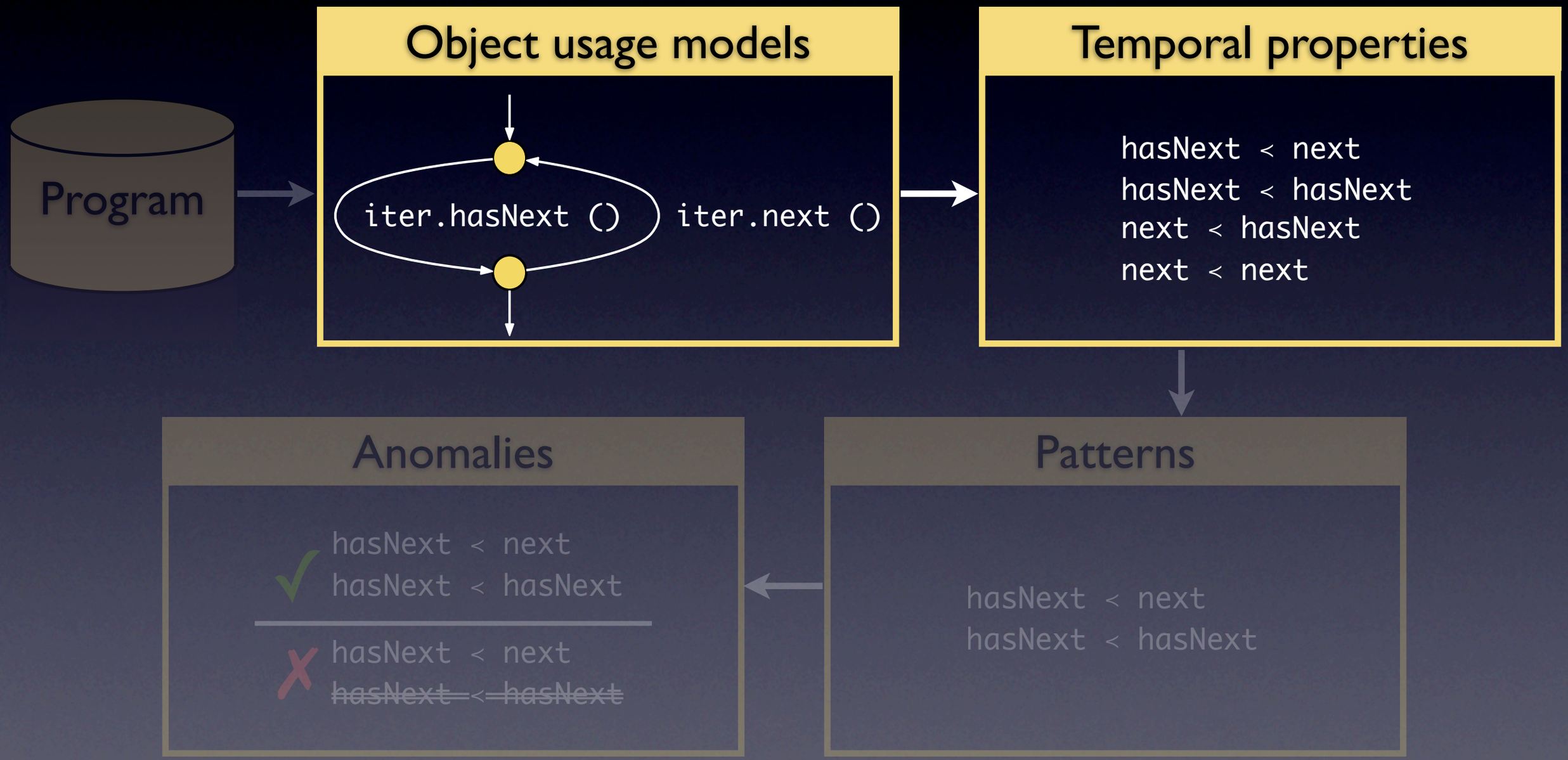
The Iterator properties



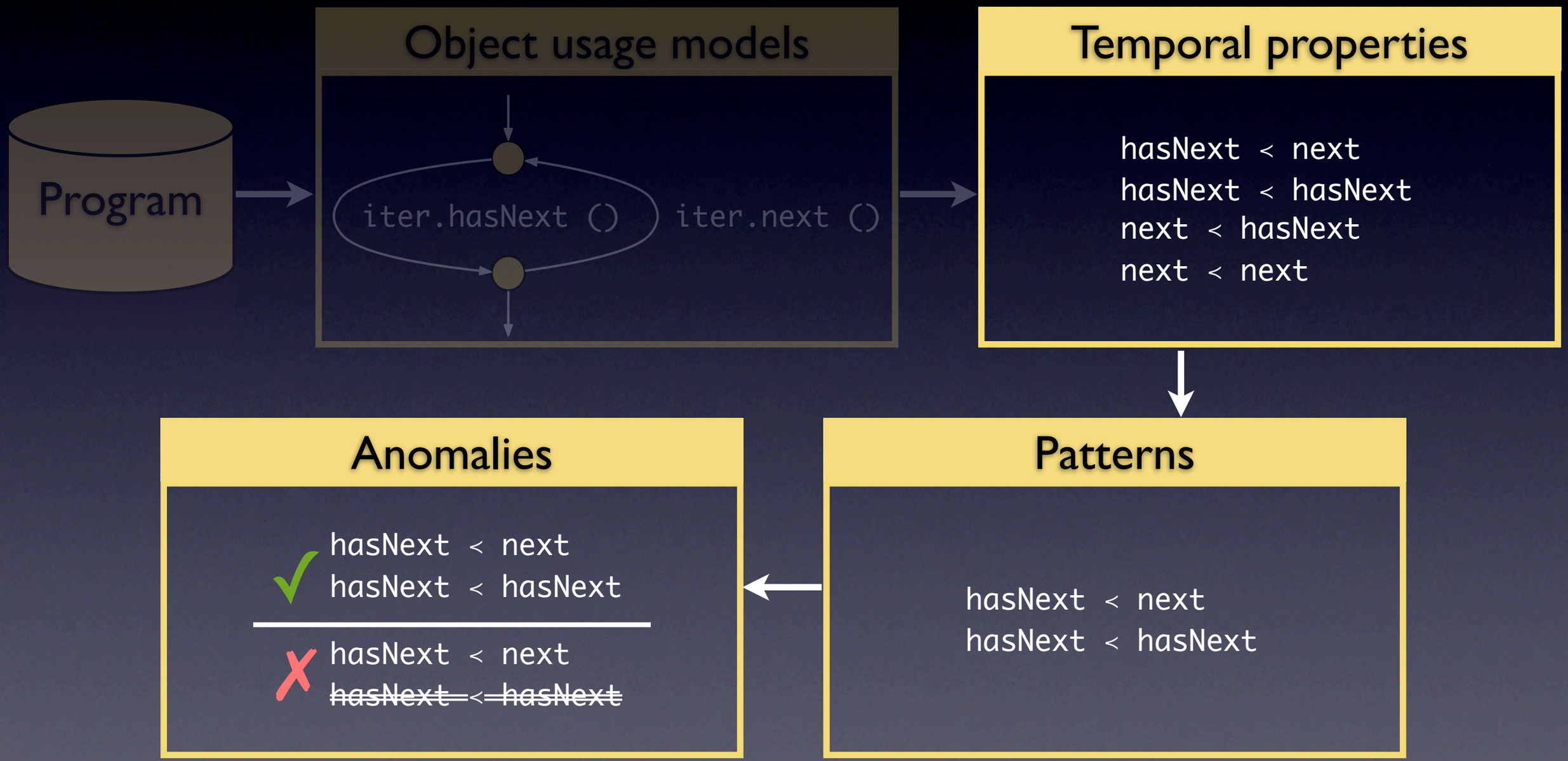

```
hasNext < next
hasNext < hasNext
next < hasNext
next < next
```

```
hasNext < next
```

JADET


Program
Object usage models
iter.hasNext ()
iter.next ()
Temporal properties
hasNext ≺ next
hasNext ≺ hasNext
next ≺ hasNext
next ≺ next
Patterns
hasNext ≺ next
hasNext ≺ hasNext
Anomalies
hasNext ≺ next
hasNext ≺ hasNext
hasNext ≺ next
hasNext ≺ hasNext

JADET

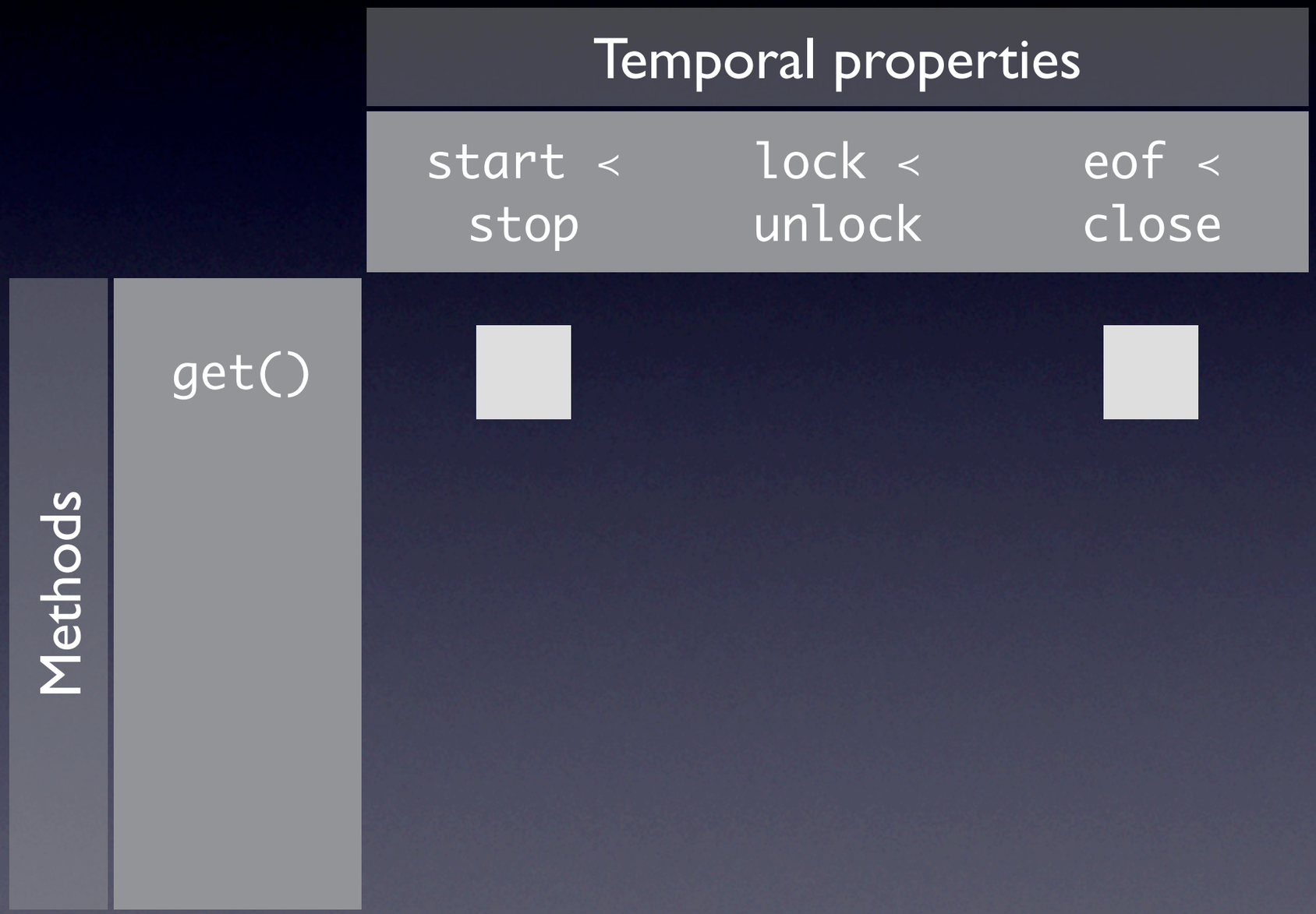

Program
Object usage models
iter.hasNext ()
iter.next ()
Temporal properties
hasNext ≺ next
hasNext ≺ hasNext
next ≺ hasNext
next ≺ next
Patterns
hasNext ≺ next
hasNext ≺ hasNext
Anomalies
hasNext ≺ next
hasNext ≺ hasNext
hasNext ≺ next
hasNext ≺ hasNext

Methods vs. properties

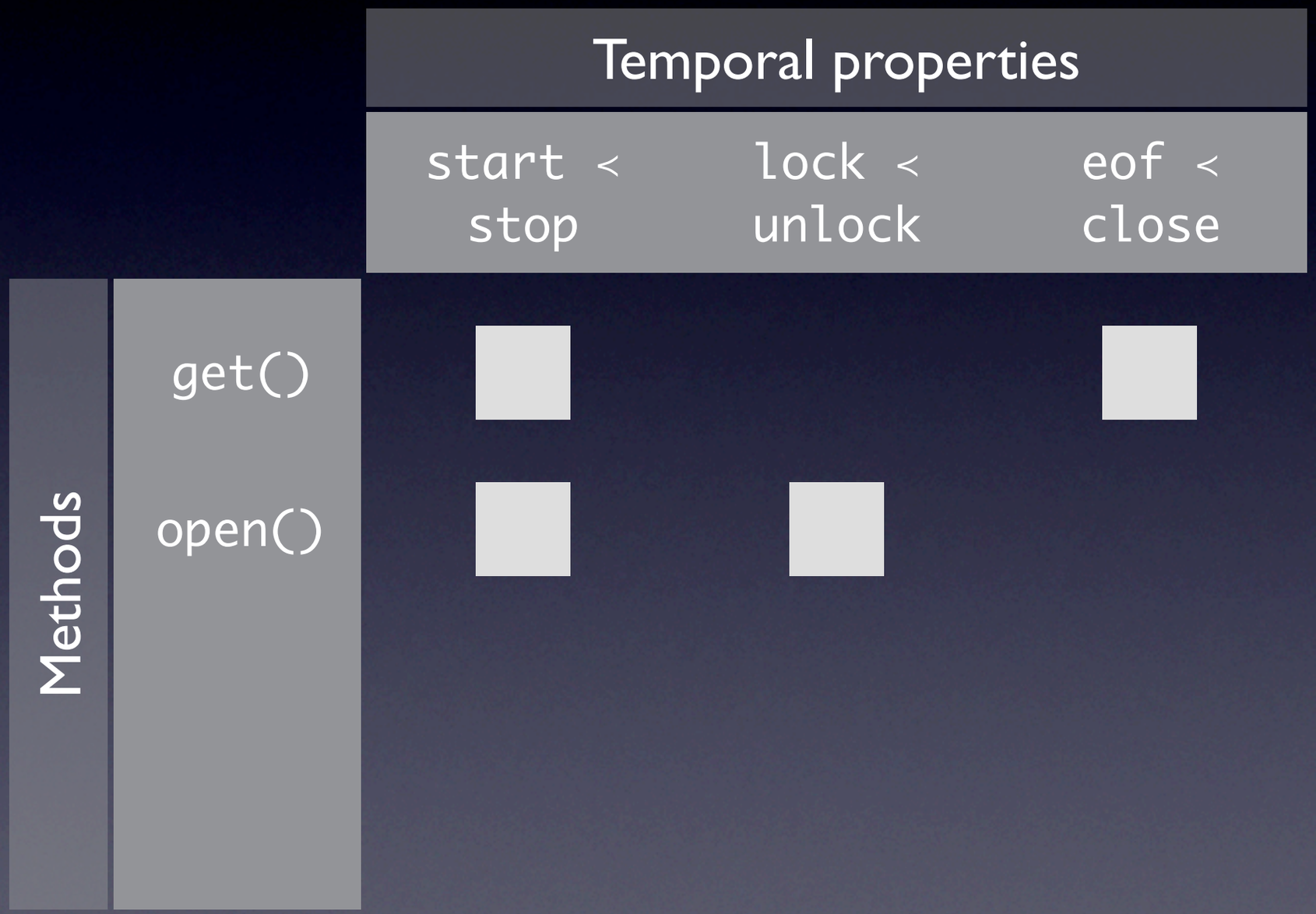
Temporal properties		
`start <` `stop`	`lock <` `unlock`	`eof <` `close`

Methods

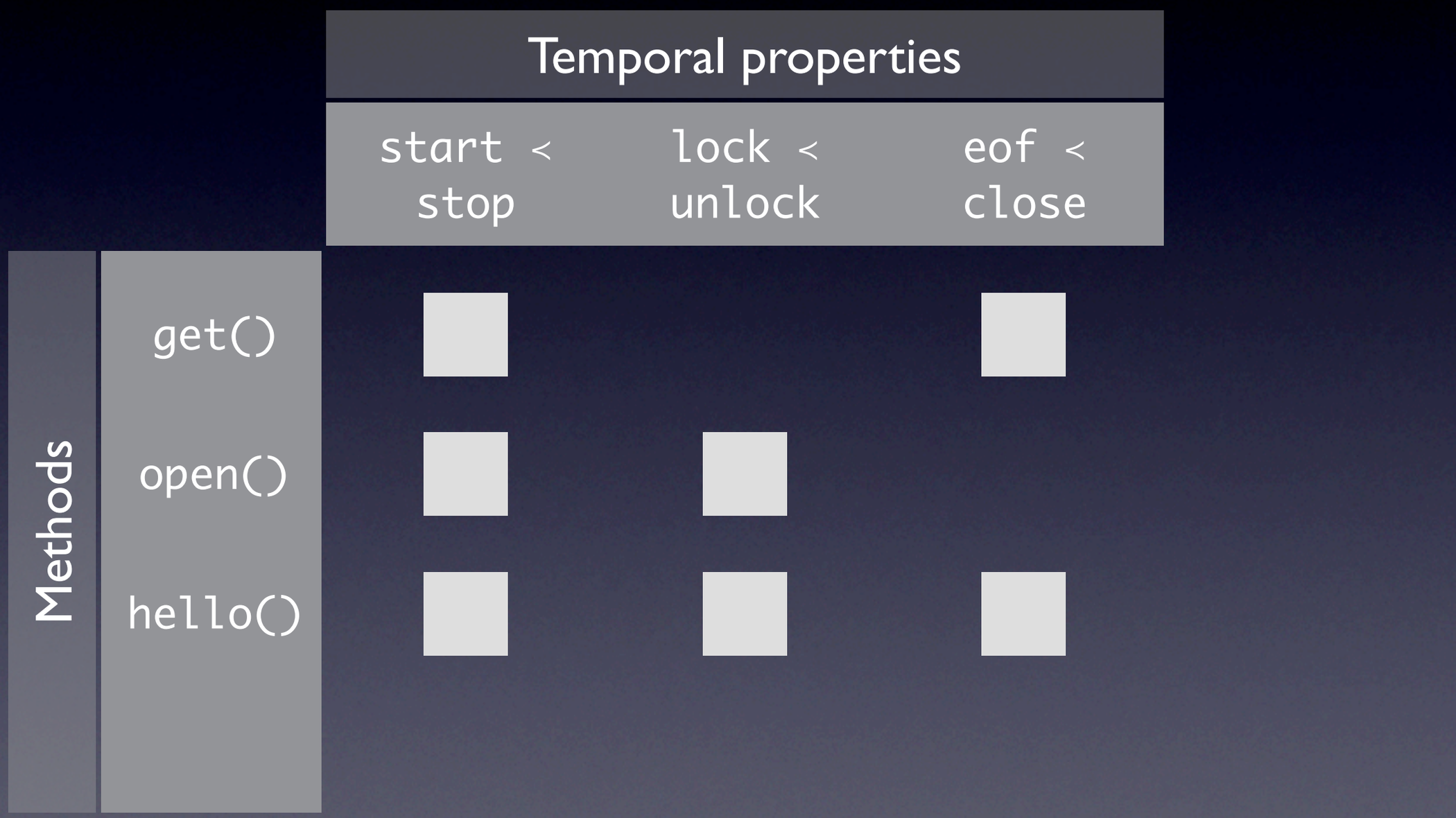
Methods vs. properties



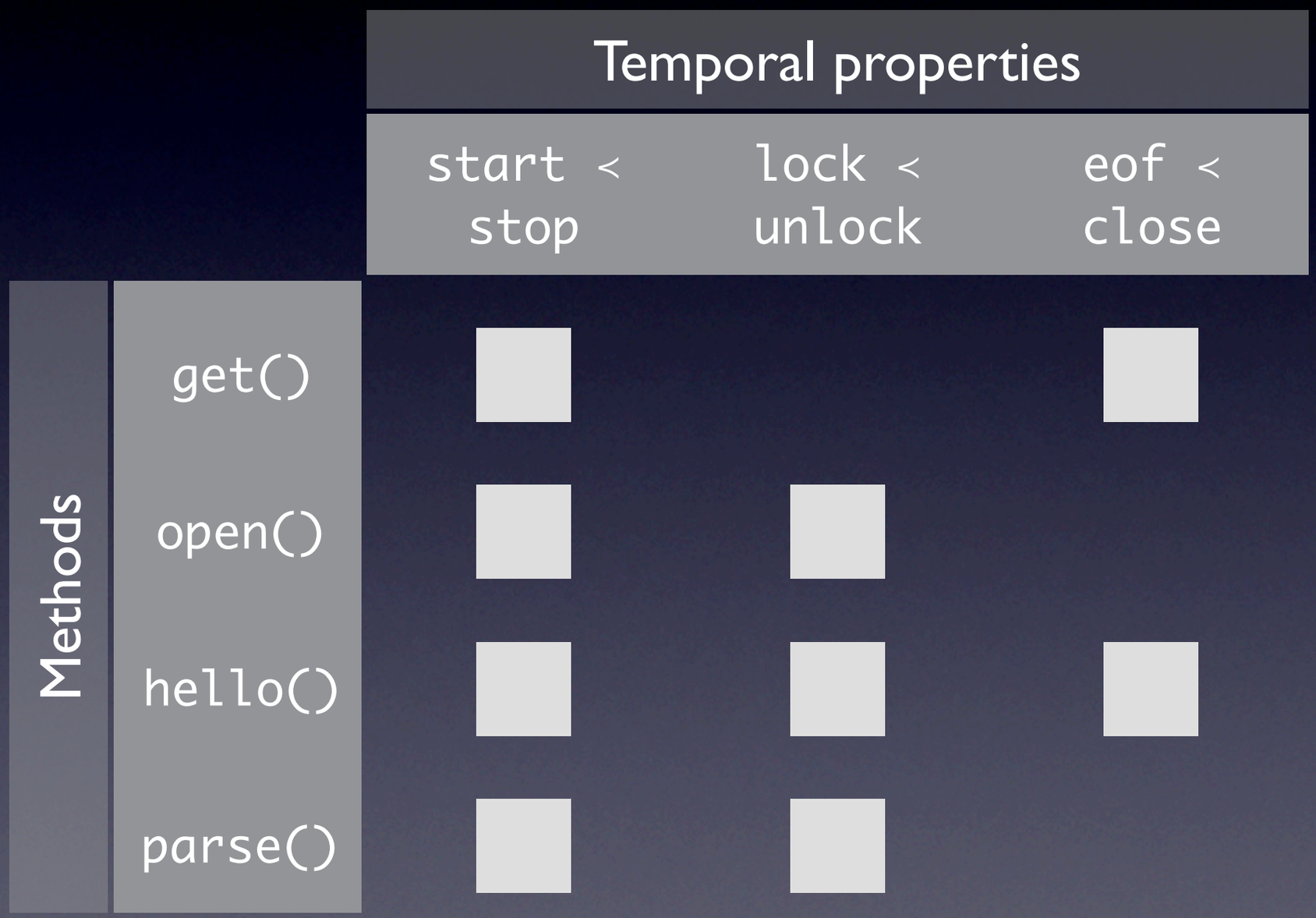
Methods vs. properties



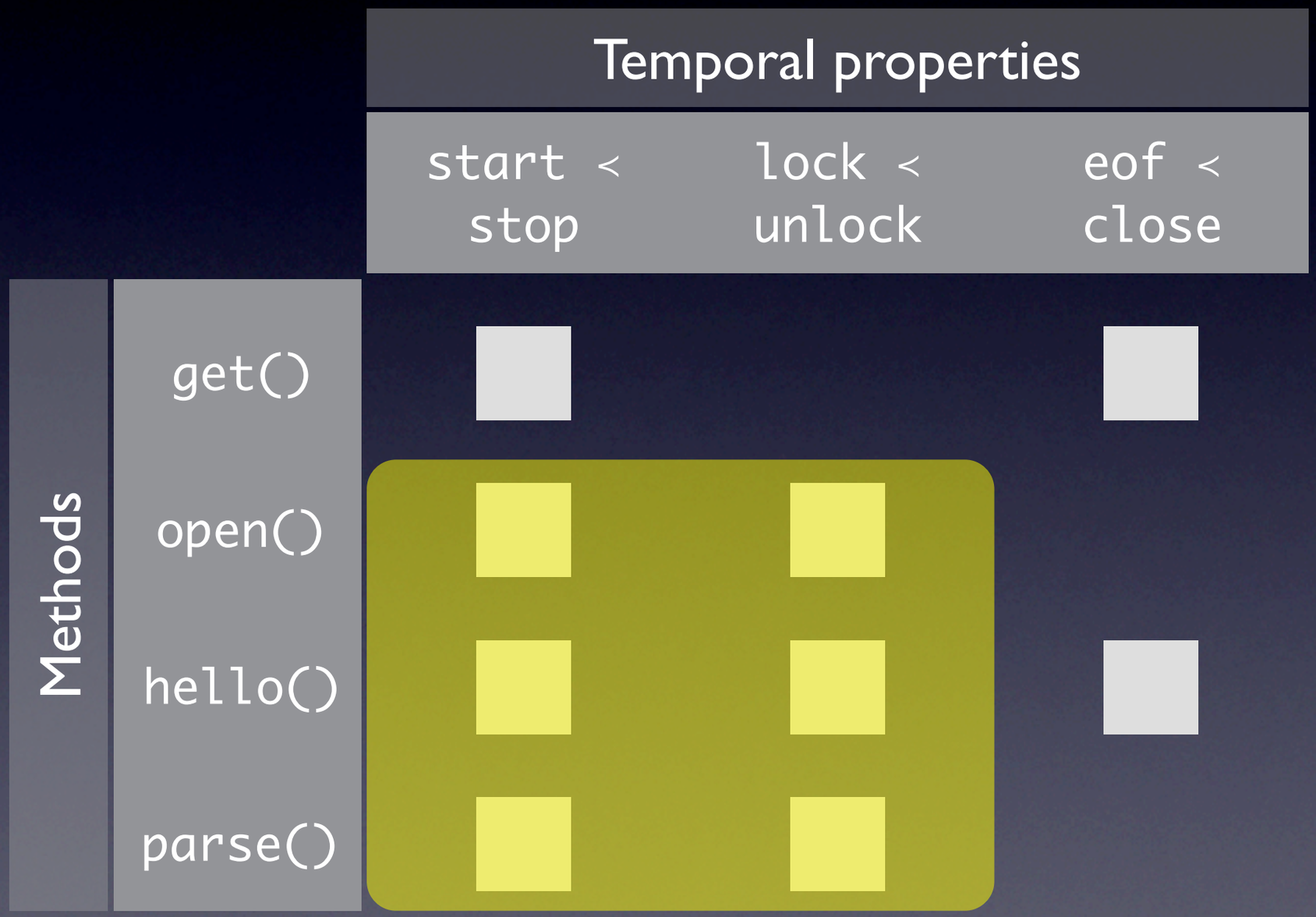
Methods vs. properties



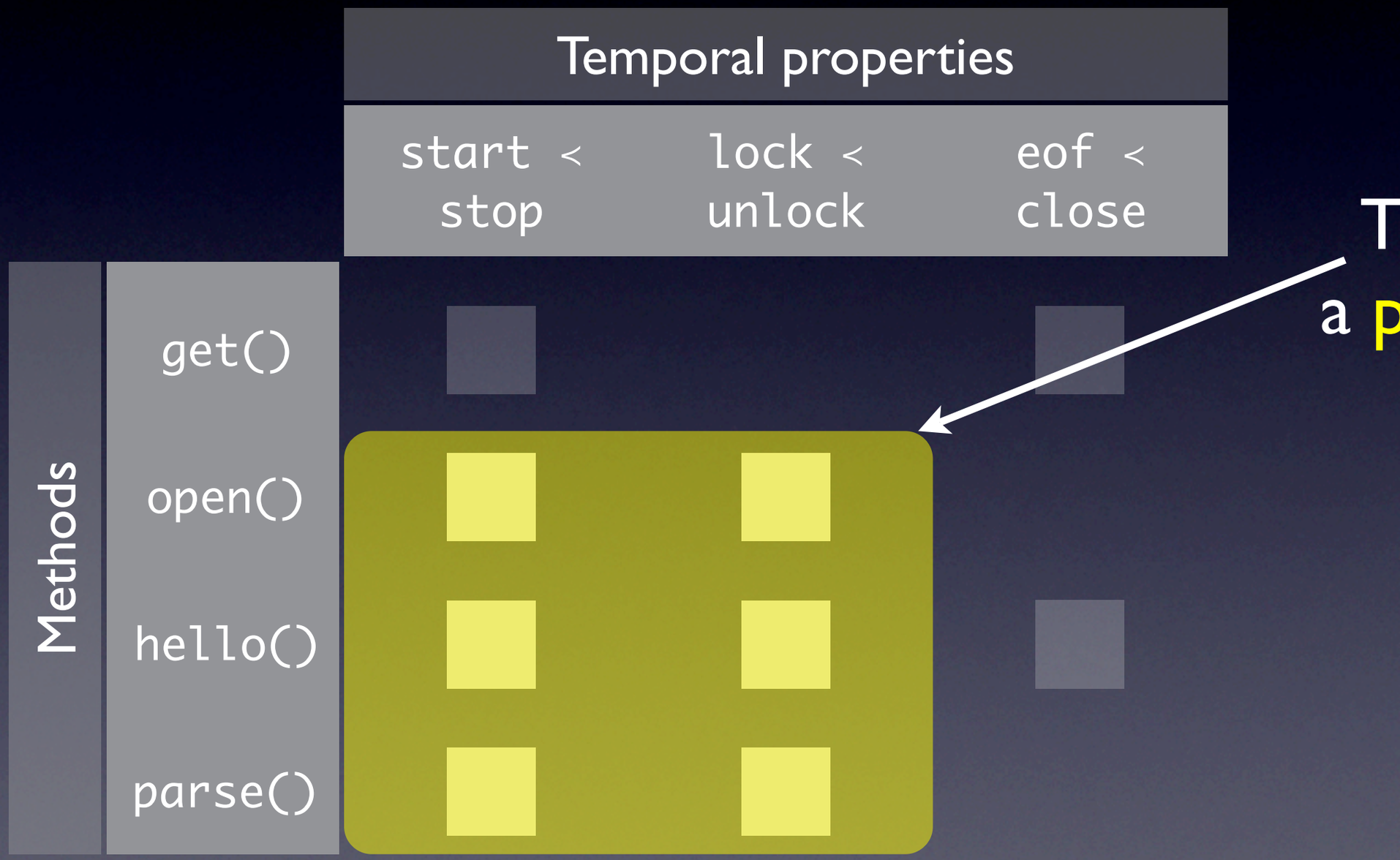
Methods vs. properties



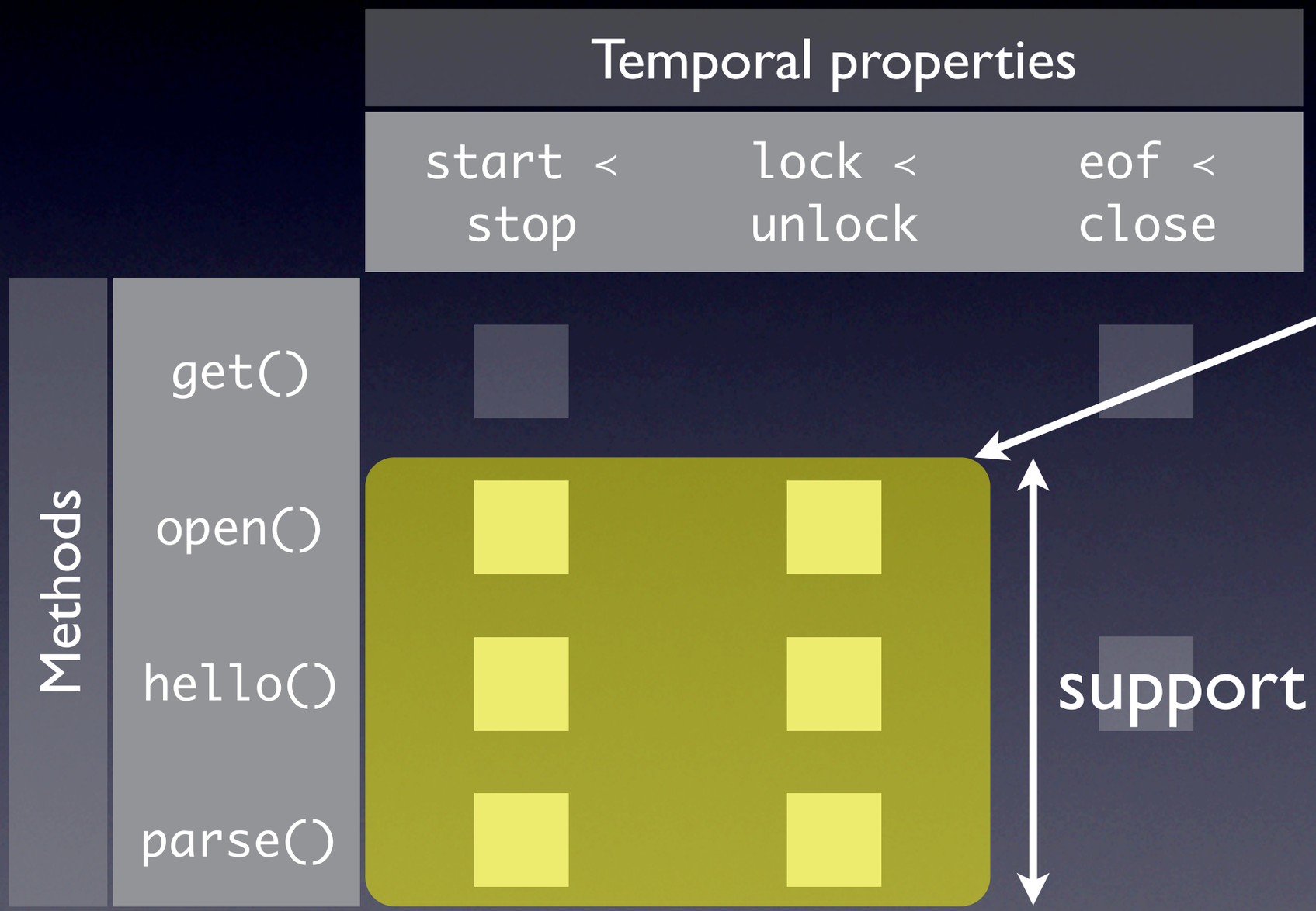
Methods vs. properties



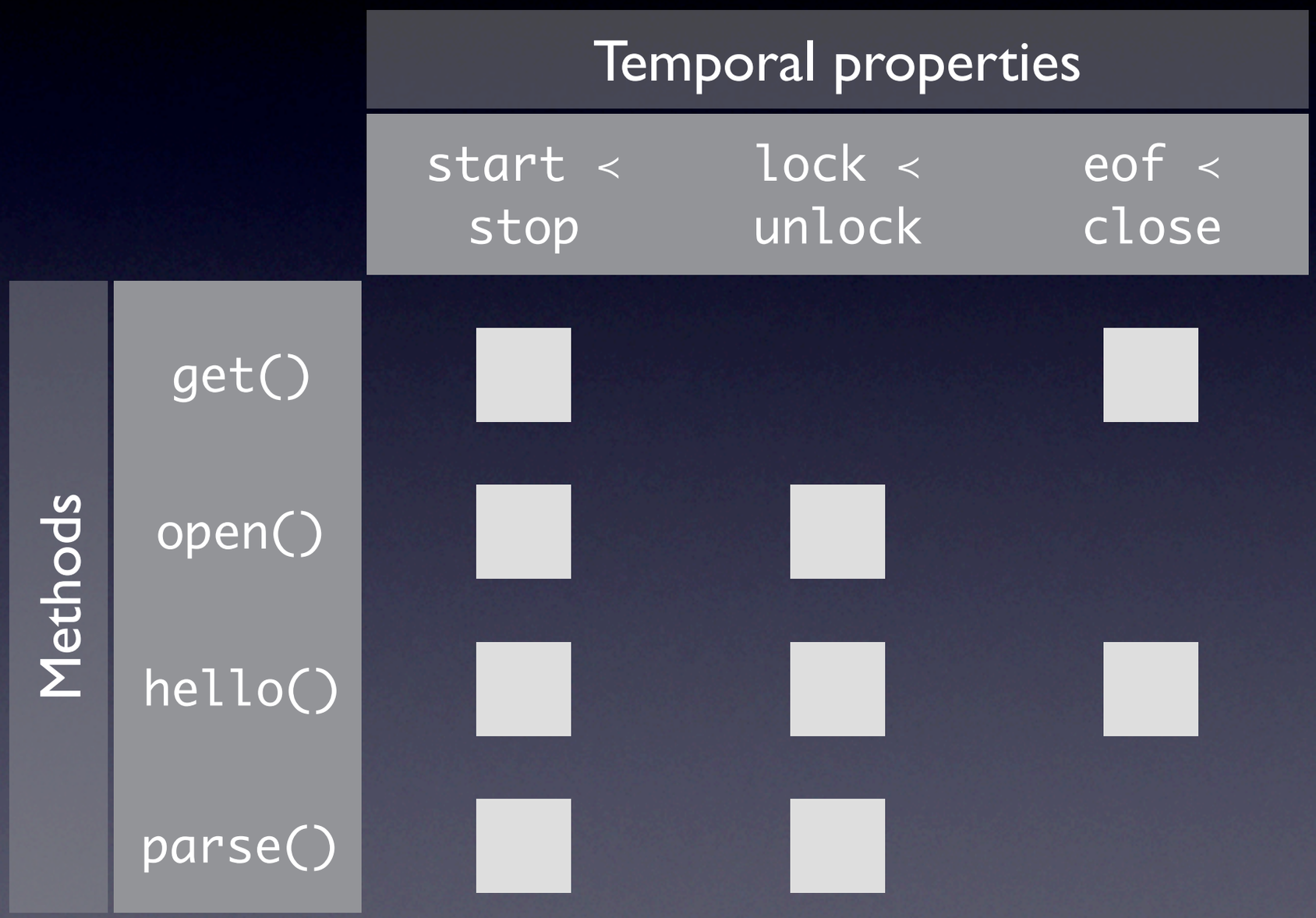
Methods vs. properties



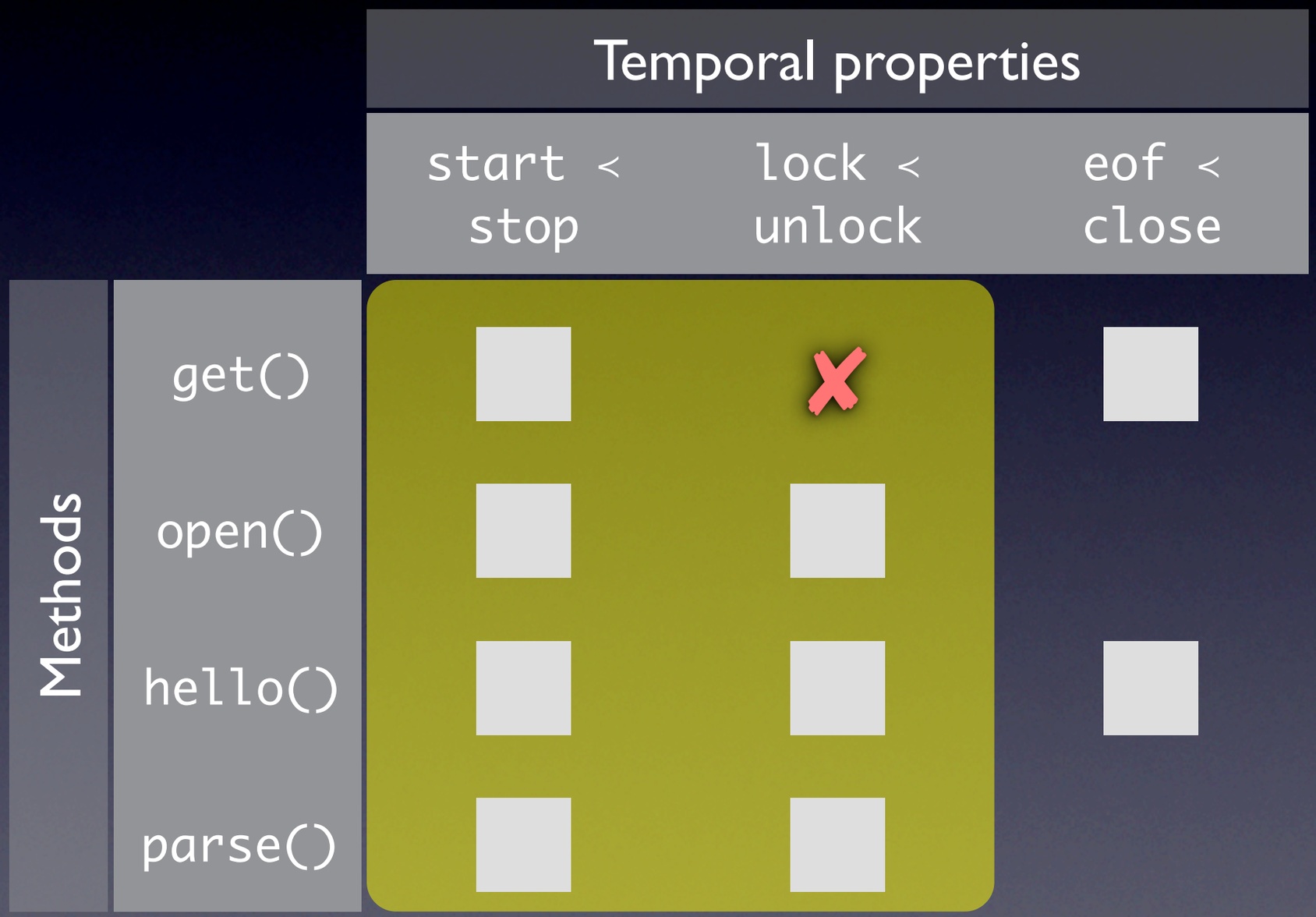
Methods vs. properties



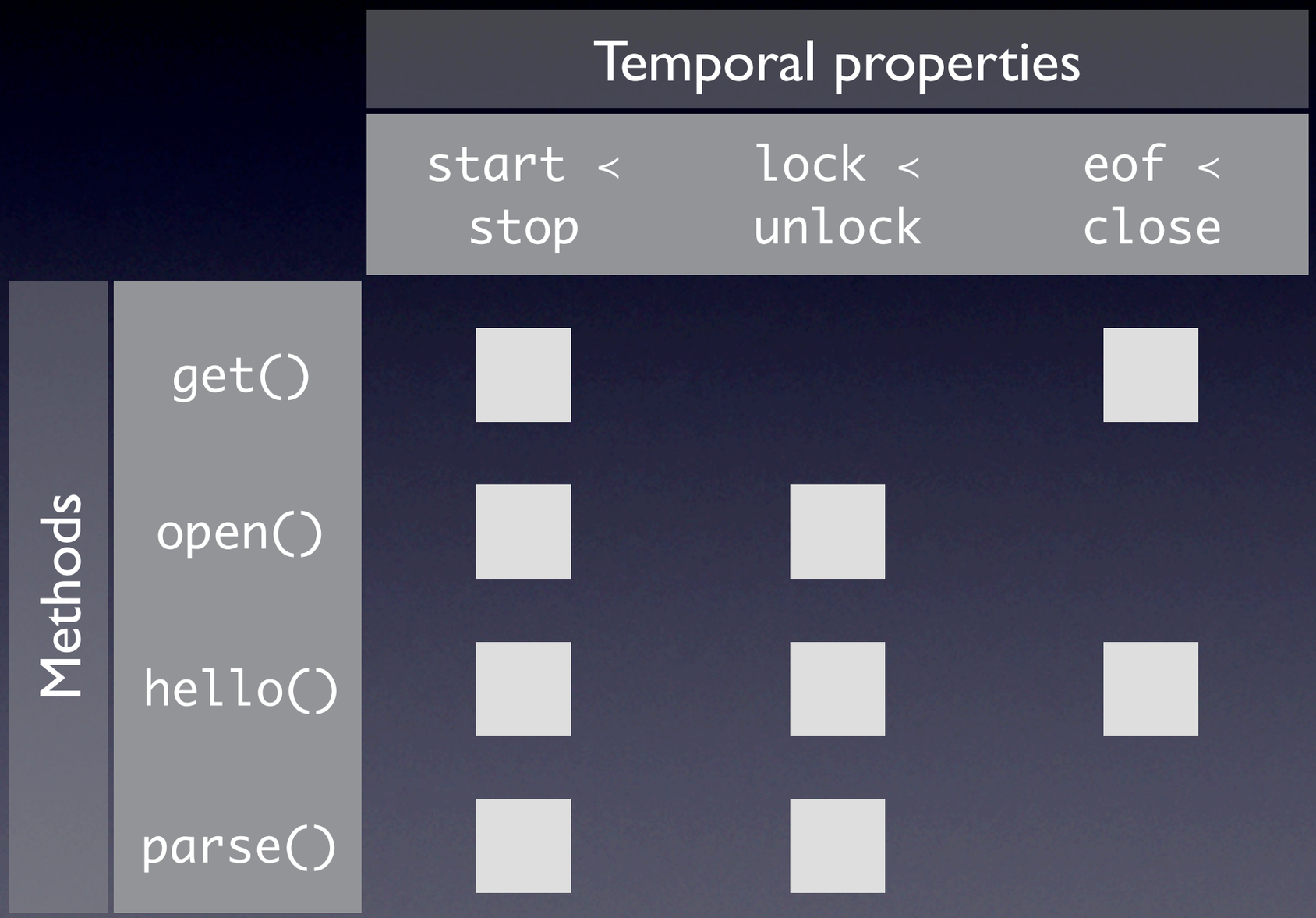
Detecting anomalies



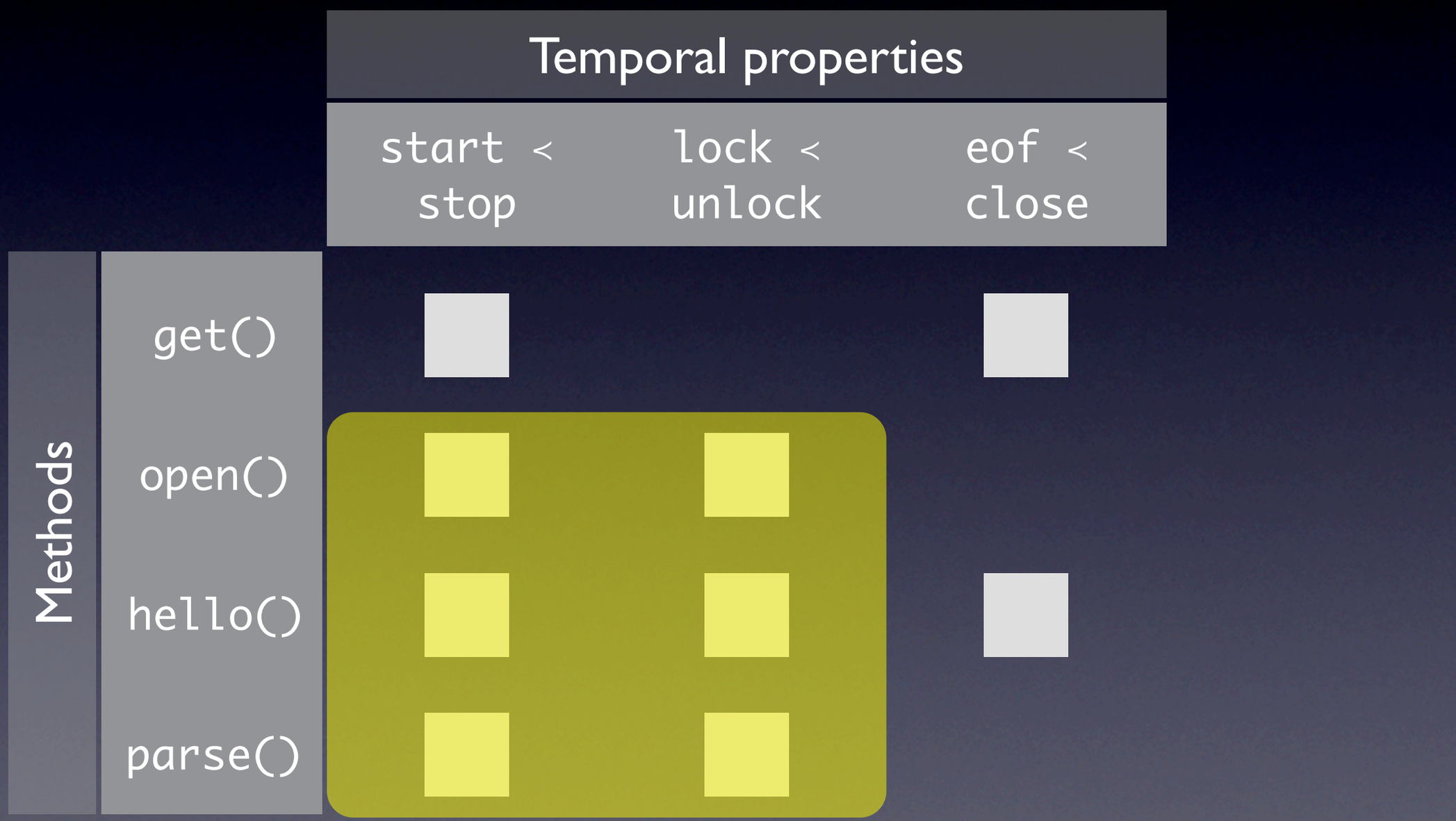
Detecting anomalies



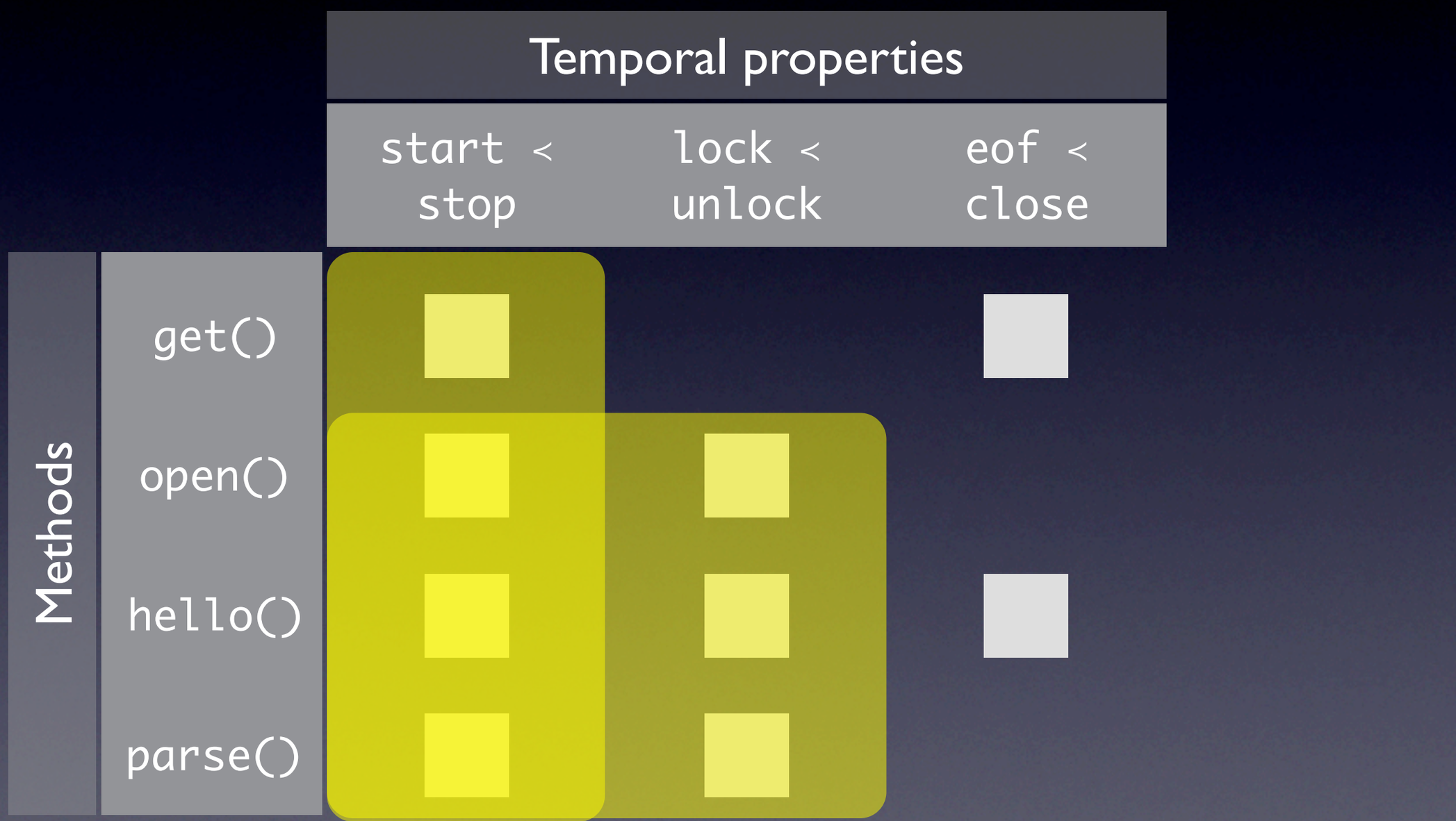
Detecting anomalies



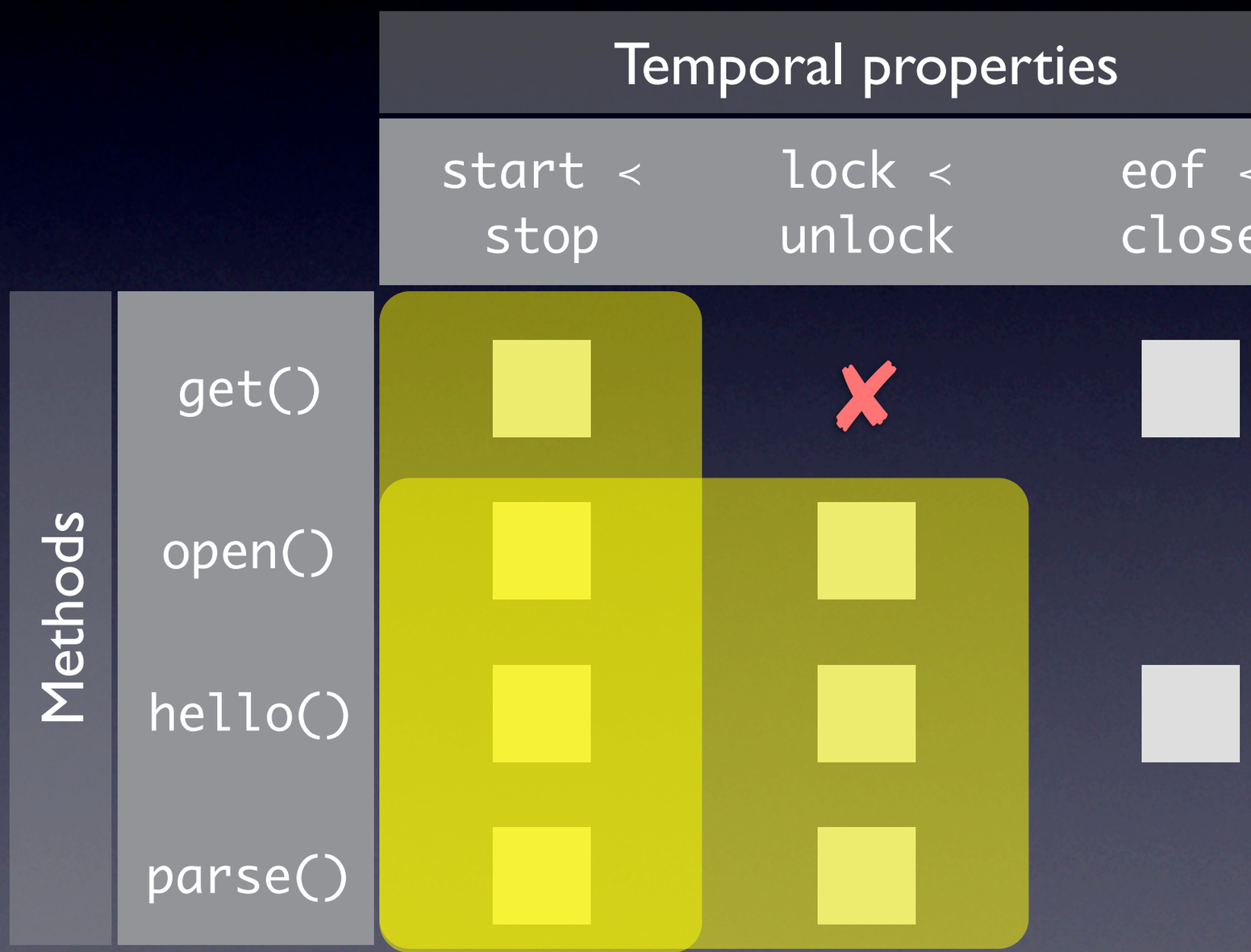
Detecting anomalies



Detecting anomalies



Detecting anomalies



Case study: AspectJ

- Compiler for the AspectJ language
- 36,045 methods in 2,957 classes
- JADET analysis results:
  - 253,084 object usage models created
    Time: 13:19
  - 666 patterns with 276 anomalies found
    Time: 3:47

Ranking anomalies

Top-ranked
(potential defects)

Bottom-ranked
(potential false positives)

Ranking anomalies

Top-ranked
(potential defects)

High support

Bottom-ranked
(potential false positives)

Ranking anomalies

Top-ranked
(potential defects)

High support
High confidence

Bottom-ranked
(potential false positives)

Ranking anomalies

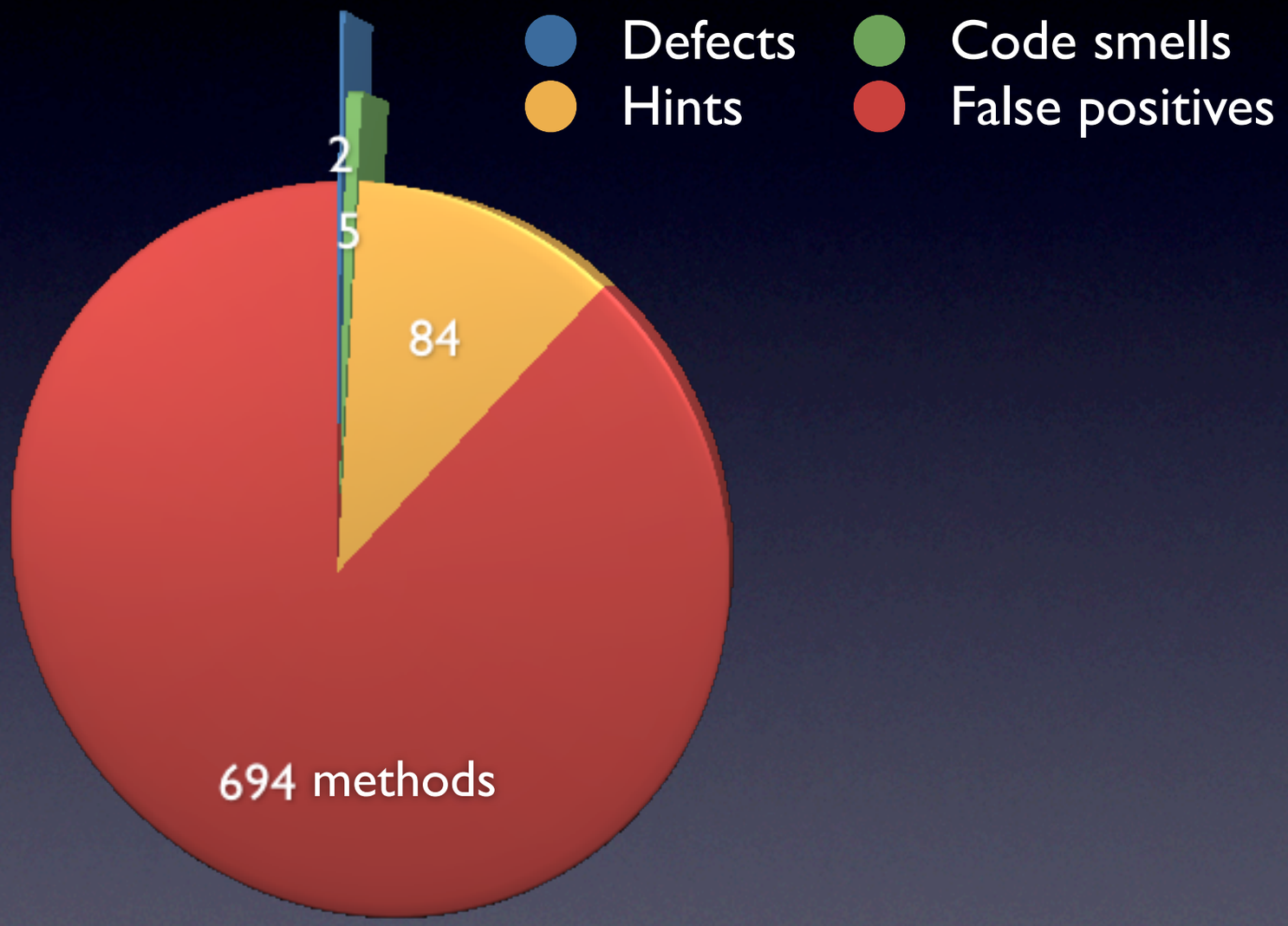
Top-ranked
(potential defects)

High support
High confidence

Bottom-ranked
(potential false positives)

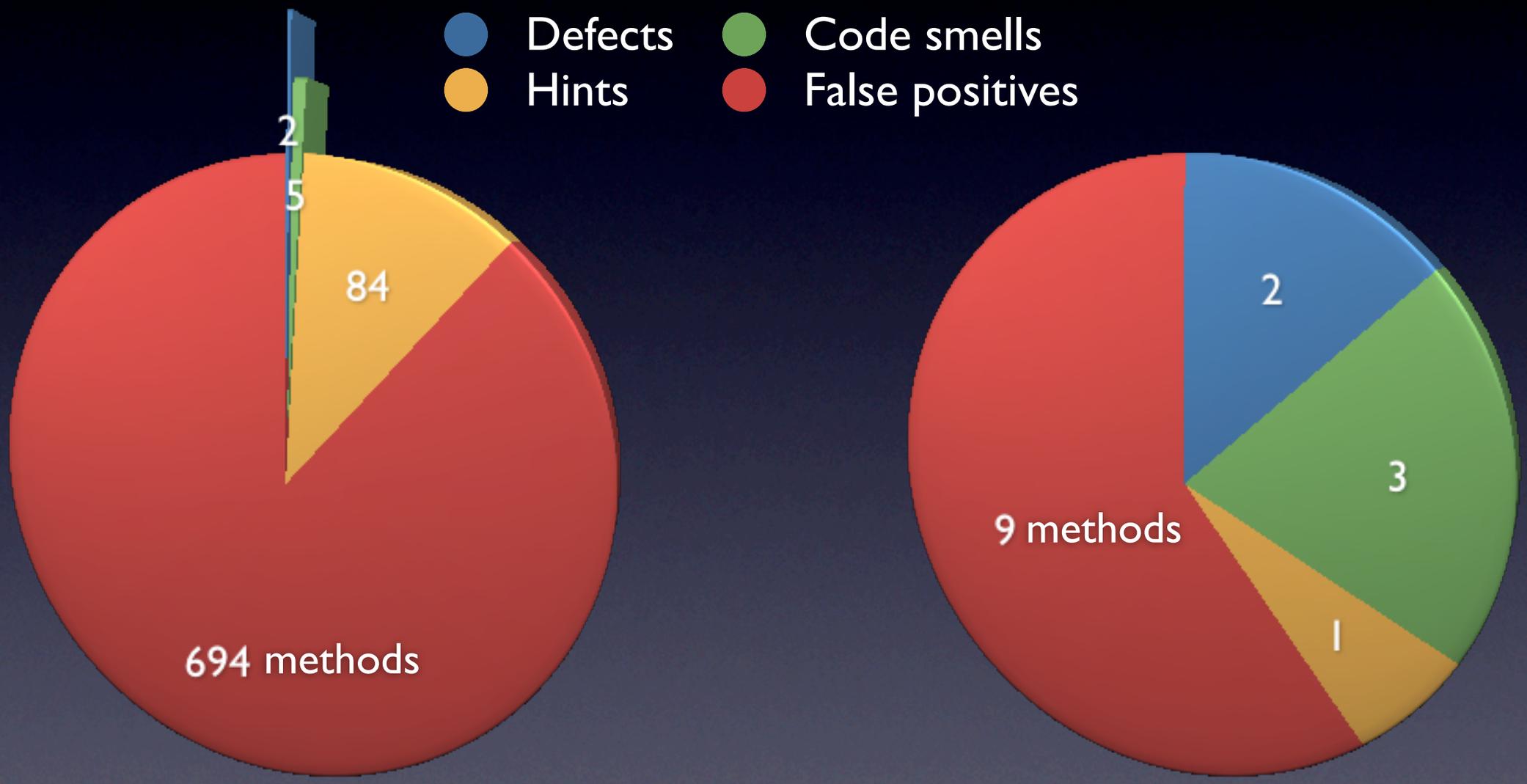
Many similar violations

AspectJ violations



All anomalies
785 methods

AspectJ violations



All anomalies
785 methods

Top 10 anomalies
15 methods

Violation ranked #1

```
private boolean verifyNIAP (…) {
    …
    Iterator iter = …;
    while (iter.hasNext()) {
        … = iter.next();
        …
        return verifyNIAP (…);
    }
    return true;
}
```

Violation ranked #1

```
private boolean verifyNIAP (…) {
    …
    Iterator iter = …;
    while (iter.hasNext()) {
        … = iter.next();
        …
        return verifyNIAP (…);
    }
    return true;
}
```

Never loops!

Violation ranked #2

```
public String getRetentionPolicy () {
    …
    for (Iterator it = …; it.hasNext();) {
        … = it.next();
        …
        return retentionPolicy;
    }
    …
}
```

Violation ranked #2

```
public String getRetentionPolicy () {
    …
    for (Iterator it = …; it.hasNext();) {
        … = it.next();
        …
        return retentionPolicy;
    }
    …
}
```

Never loops!

Violation ranked #2

```
public String getRetentionPolicy () {
    …
    for (Iterator it = …; it.hasNext();) {
        … = it.next();
        …
        return retentionPolicy;
    }
    …
}
```

Never loops!

Fortunately, there can be at most one retention policy!

Violation ranked #7

```
public void visitCALOAD (CALOAD o) {
    Type arrayref = stack().peek(1);
    Type index = stack().peek(0);
    indexOfInt(o, index);
    arrayrefOfArrayType(o, arrayref);
}
```

Violation ranked #7

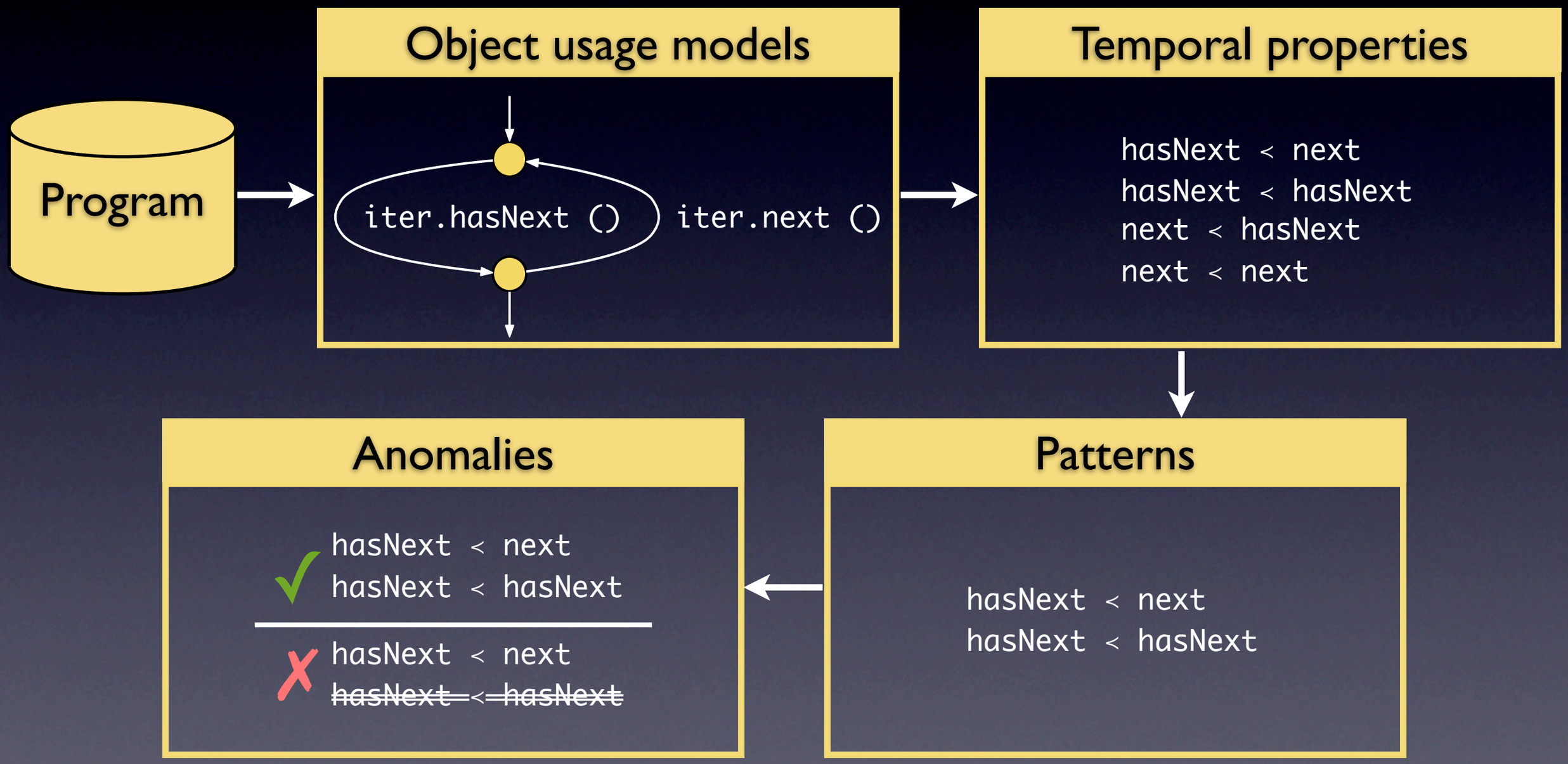
```
public void visitCALOAD (CALOAD o) {
    Type arrayref = stack().peek(1);
    Type index = stack().peek(0);
    indexOfInt(o, index);
    arrayrefOfArrayType(o, arrayref);
}
```

Doesn’t check if arrayref holds chars!

Related work

- FindBugs by Hovemeyer and Pugh (SIGPLAN Notices Dec. 2004)
- PR-Miner by Li and Zhou (ESEC/FSE 2005)
- DynaMine by Livshits and Zimmermann (ESEC/FSE 2005)
- Perracotta by Yang and Evans (ICSE 2006)
- Chronicler by Ramanathan et al. (ICSE 2007)

JADET


Program
Object usage models
iter.hasNext ()
iter.next ()
Temporal properties
hasNext ≺ next
hasNext ≺ hasNext
next ≺ hasNext
next ≺ next
Patterns
hasNext ≺ next
hasNext ≺ hasNext
Anomalies
hasNext ≺ next
hasNext ≺ hasNext
hasNext ≺ next
hasNext ≺ hasNext

JADET

- creates models that accurately reflect structure of the code
- detects anomalies in sequences of method calls
- scales to industrial code
- found two new defects in AspectJ
- http://www.st.cs.uni-sb.de/models/