Welcome to IWACO'17!

International Workshop on Aliasing, Capabilities and Ownership

Morning Schedule

- **Spencer: Tracing as a service**
  Stephan Brandauer

- **Are Your Incoming Aliases Really Necessary? Remembering the Cost of Object Ownership**
  Alex Potanin

- **Reference Capabilities in Practice: Examining Real-World Pony Code**
  Sylvan Clebsch

Lunch: 12:30-14:00

Schedule

- **Aliasing, Capabilities and Ownership in Rust**
  Felix Klock

- **Introducing Ownership Type Constraints to UML/OCL**
  Jagadeeswaran Thangaraj & Senthil Kumaran

- **Towards Reasonable Ownership**
  Anya Helene Bagge, Kristoffer Haugsbakk & Vadim Zaytsev

Coffee: 15:30-16:00

- **Adding Safe Manual Memory Management to .NET**
  Dimitrios Vytiniotis

Spencer:
Tracing as a Service

UPPSALA UNIVERSITET

Stephan Brandauer, Tobias Wrigstad
http://stbr.me/spencer
sbrandauer


Spencer

- Web service: analyse pre-recorded program traces

- Web service: analyse pre-recorded program traces
- Focus on side effects, heap structure, aliasing, …

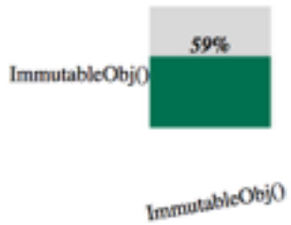
- Web service: analyse pre-recorded program traces
- Focus on side effects, heap structure, aliasing, …
- Domain specific language (DSL) for trace analysis

Query

✔ ImmutableObj() ¬ *Objects that are never changed outside their constructor.*

---

refine query ›



ImmutableObj()

22262, 55290, 108807, 40448, 8533, 66609, 9243, 10627, 41308, 54793, 66093, 48678, 56196, 77345, 128584, 54731, 16278, 29801, 96673, 39154, 85392, 68518, 16408, 141228, 54121, 139916, 119341, 62215, 40253, 49056, 79561, 96317, 65342, 63428, 99460, 124764, 115231, 101095, 65977, 84802, 47793, 34937, 64994, 75497, 8716, 146874, 18539, 64090, 36016, 38397, 55562, 11200, 99036, 24838, 62344, 12074, 45433, 106361, 32199, 85783, 98466, 74916, 55009, 71290, 27721, 143788, 49658, 40971, 143747, 142504, 74381, 44885, 28015, 18659, 112833, 43279, 49995, 71224, 76333, 55733, 99476, 124139, 100185, 31281, 44301, 82081, 139543, 76363, 82299, 146466, 114103, 19130, 6700, 86022, 50274, 145559, 121670, 56830, 45940 77441 total

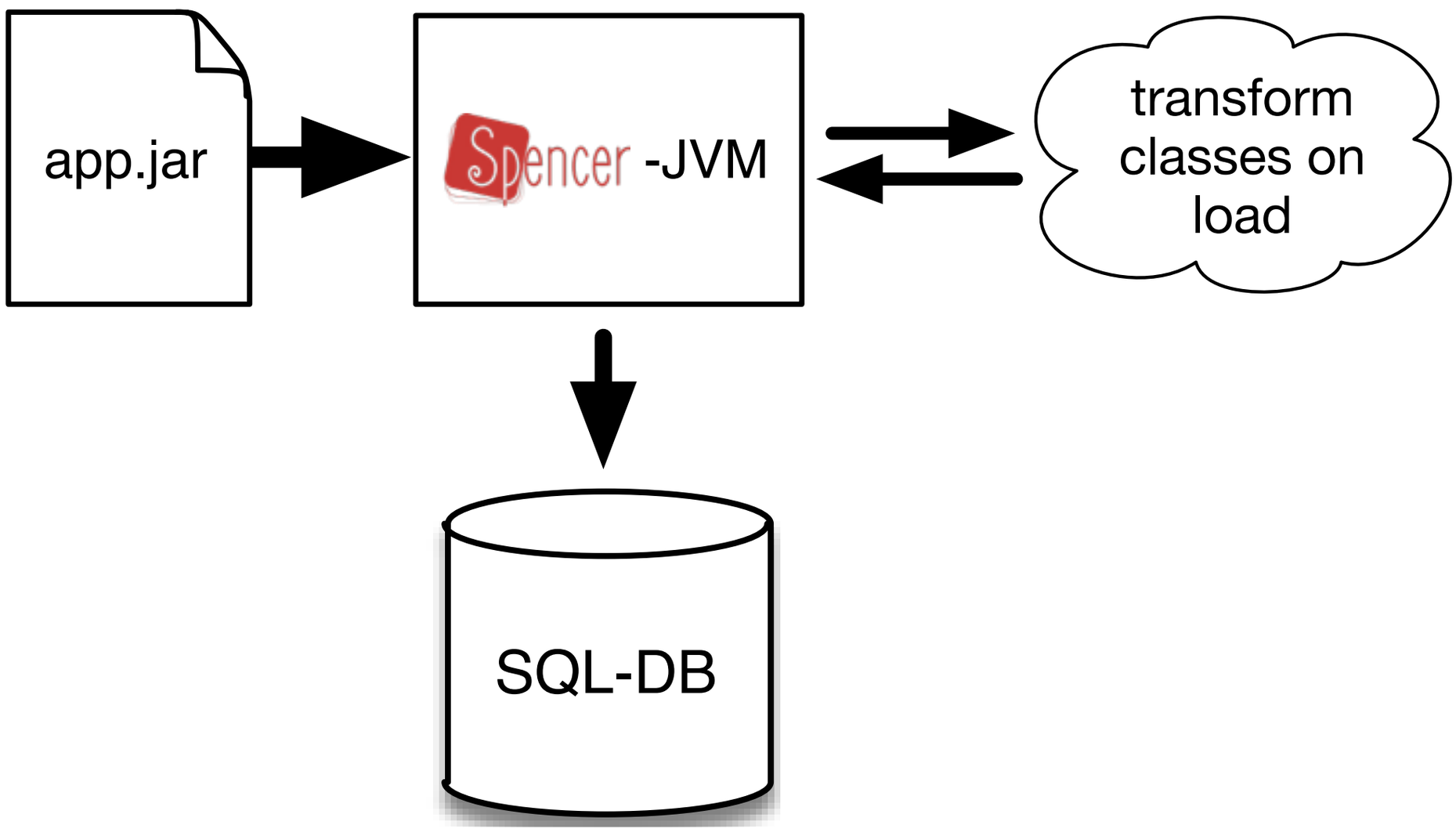
Object Variables

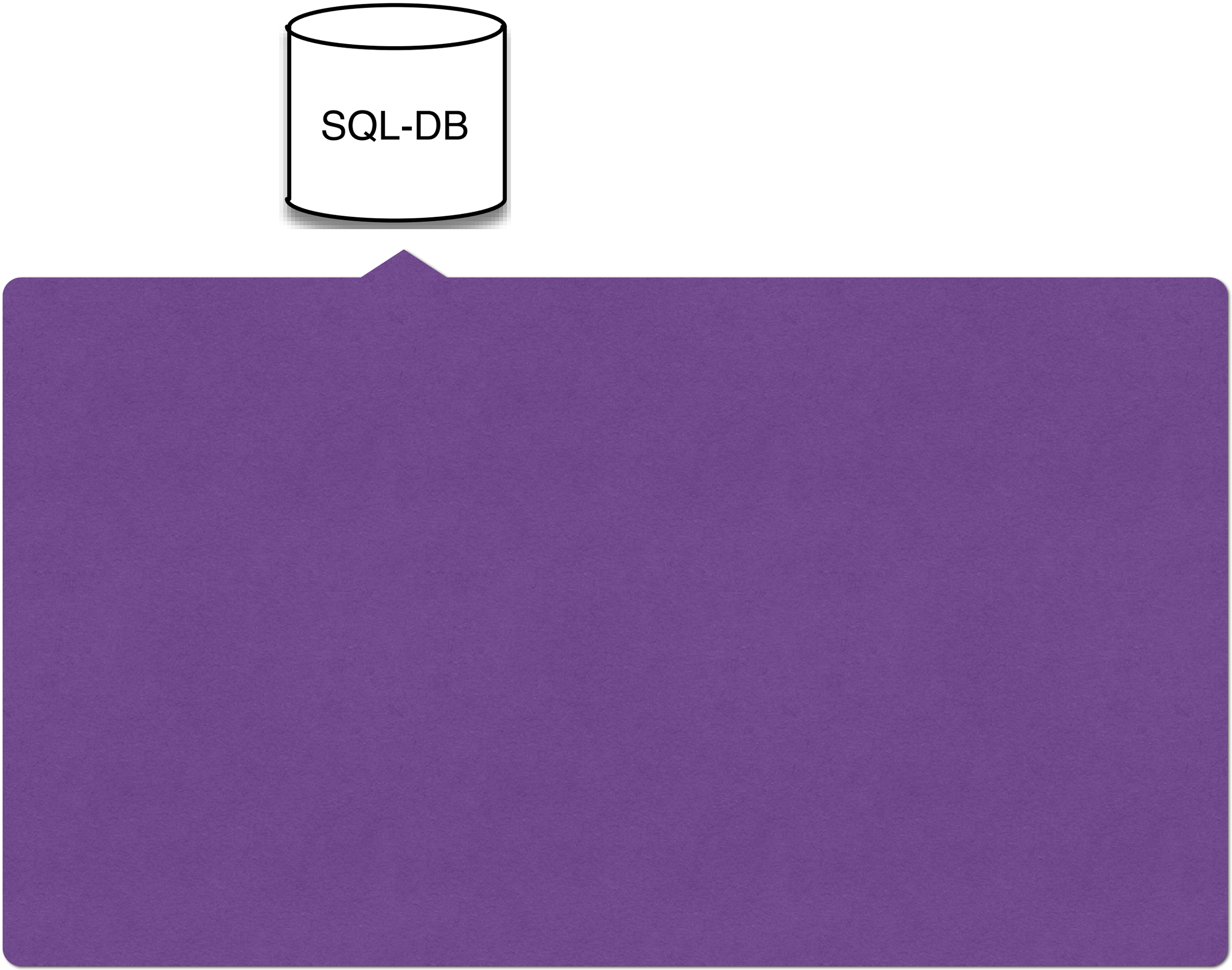
thread

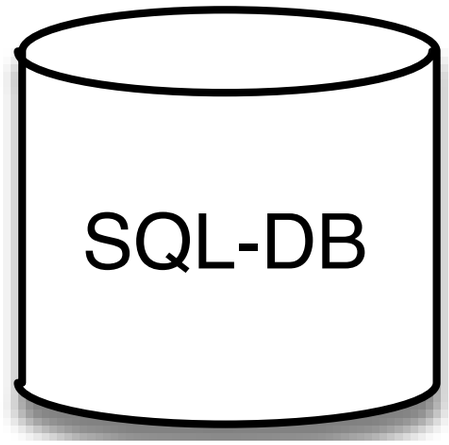
klass

allocationSite

Collecting Data

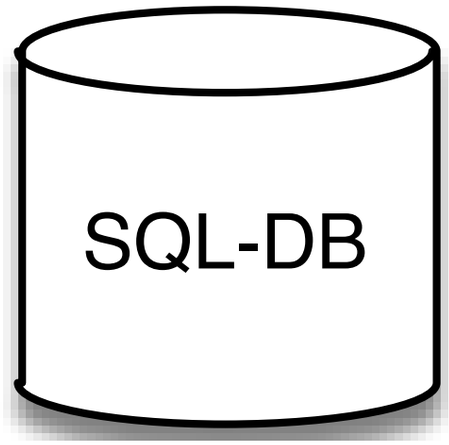



SQL-DB



calls ✓

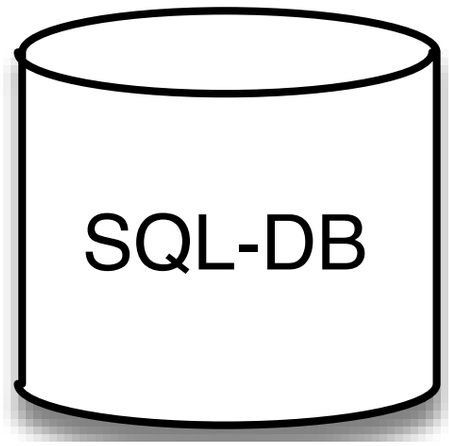
```
# SELECT * FROM calls WHERE callstart = 511073 ;
 caller | callee |    name    | callstart | callend |  callsitefile  | callsiteline | thread
--------+--------+------------+-----------+---------+----------------+--------------+--------
  10530 |  10247 | startsWith |    511073 |  511091 | MetaIndex.java |          242 | main
```



calls ✓
uses ✓

```
# SELECT * FROM calls WHERE callstart = 511073 ;
 caller | callee |    name    | callstart | callend |  callsitefile  | callsiteline | thread
--------+--------+------------+-----------+---------+----------------+--------------+--------
  10530 |  10247 | startsWith |    511073 |  511091 | MetaIndex.java |          242 | main


# SELECT * FROM uses WHERE idx ⩾ 511073 AND idx ⩽ 511091 ;
 caller | callee | name  |   method   |   kind    |  idx   | thread
--------+--------+-------+------------+-----------+--------+--------
  10247 |  10247 | var_1 | startsWith | varstore  | 511074 | main
  10247 |  10247 | var_1 | startsWith | varload   | 511075 | main
  … snip …
  10247 |  10247 | var_5 | startsWith | varload   | 511088 | main
  10247 |  10453 | _0    | startsWith | read      | 511089 | main
```

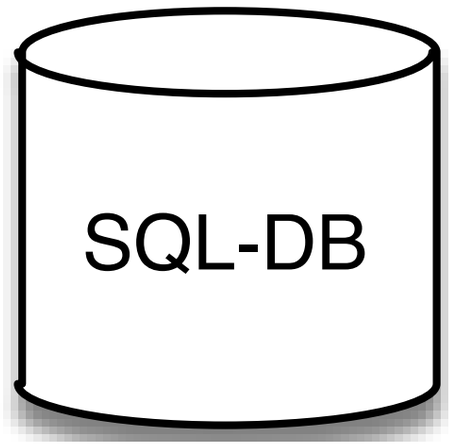


calls ✓
uses ✓
refs ✓

```
# SELECT * FROM calls WHERE callstart = 511073 ;
 caller | callee |    name    | callstart | callend |  callsitefile  | callsiteline | thread
--------+--------+------------+-----------+---------+----------------+--------------+--------
  10530 |  10247 | startsWith |    511073 |  511091 | MetaIndex.java |          242 | main


# SELECT * FROM uses WHERE idx ⩾ 511073 AND idx ⩽ 511091 ;
 caller | callee | name  |   method    |   kind    |  idx   | thread
--------+--------+-------+-------------+-----------+--------+--------
  10247 |  10247 | var_1 | startsWith | varstore  | 511074 | main
  10247 |  10247 | var_1 | startsWith | varload   | 511075 | main
  … snip …
  10247 |  10247 | var_5 | startsWith | varload   | 511088 | main
  10247 |  10453 | _0    | startsWith | read      | 511089 | main


# SELECT * FROM refs WHERE caller = 10247 AND kind = 'field' ;
 caller | callee | kind  | name  | refstart | refend | thread
--------+--------+-------+-------+----------+--------+--------
  10247 |  10248 | field | value |   421877 |        | main
```



calls ✓
uses ✓
refs ✓
… ✓

```
# SELECT * FROM calls WHERE callstart = 511073 ;
 caller | callee |    name    | callstart | callend |  callsitefile  | callsiteline | thread
--------+--------+------------+-----------+---------+----------------+--------------+--------
  10530 |  10247 | startsWith |    511073 |  511091 | MetaIndex.java |          242 | main


# SELECT * FROM uses WHERE idx ⩾ 511073 AND idx ⩽ 511091 ;
 caller | callee | name  |   method    |   kind    |  idx   | thread
--------+--------+-------+-------------+-----------+--------+--------
  10247 |  10247 | var_1 | startsWith | varstore  | 511074 | main
  10247 |  10247 | var_1 | startsWith | varload   | 511075 | main
  … snip …
  10247 |  10247 | var_5 | startsWith | varload   | 511088 | main
  10247 |  10453 | _0    | startsWith | read      | 511089 | main


# SELECT * FROM refs WHERE caller = 10247 AND kind = 'field' ;
 caller | callee | kind  | name  | refstart | refend | thread
--------+--------+-------+-------+----------+--------+--------
  10247 |  10248 | field | value |   421877 |        | main
```

Spencer DSL

- Object selections as single expressions
- Compiled to SQL queries
- Simplicity > Expressivity

Spencer DSL

- Query combinators combine queries for more powerful analysis

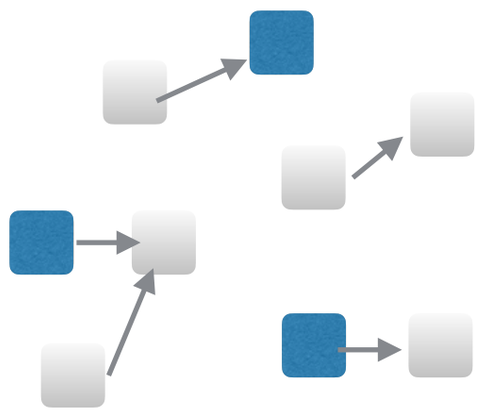
- Query results are cached

Spencer DSL
— Immutable Aggregates —

```
MutableObj()
```

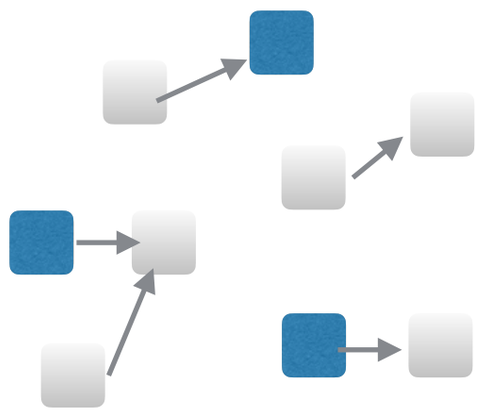
Spencer DSL
— Immutable Aggregates —

```
MutableObj()
```



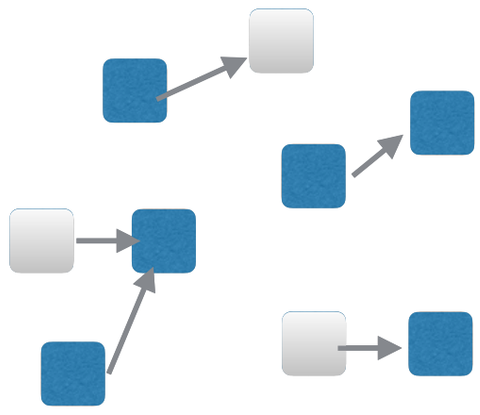
Spencer DSL
— Immutable Aggregates —

```
Not(MutableObj())
```



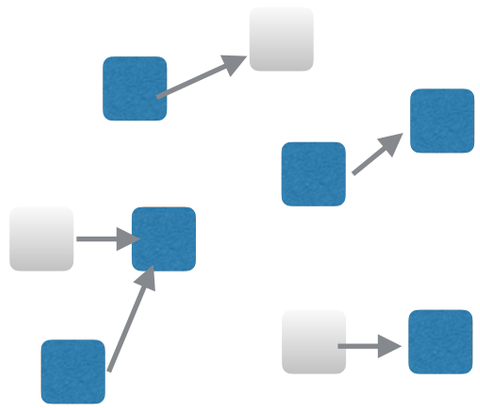
Spencer DSL
— Immutable Aggregates —

```
Not(MutableObj())
```



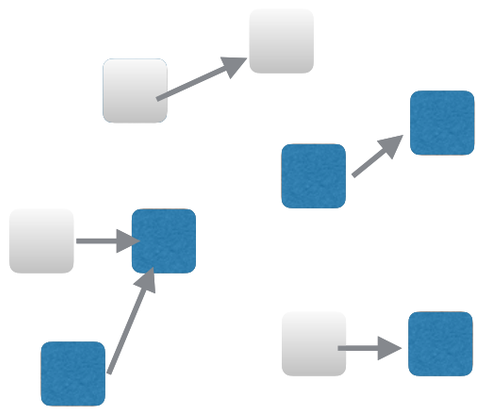
Spencer DSL
— Immutable Aggregates —

```
HeapDeeply(Not(MutableObj()))
```



Spencer DSL
— Immutable Aggregates —

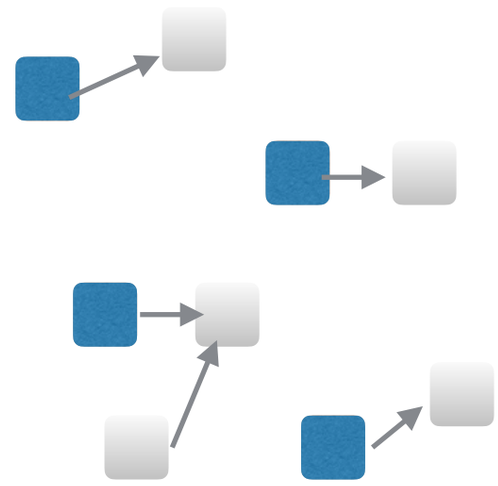
```
HeapDeeply(Not(MutableObj()))
```



Spencer DSL

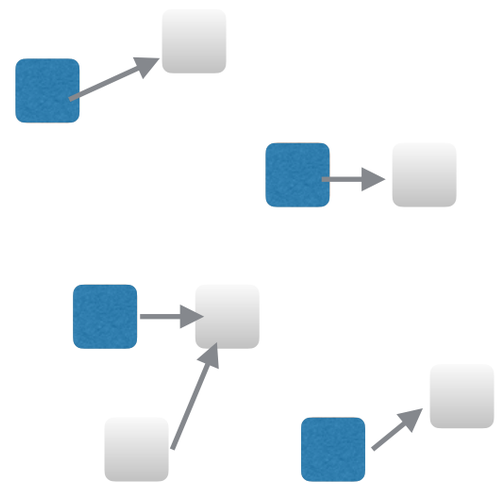
— All Strings —

```
InstanceOf(java.lang.String)
```



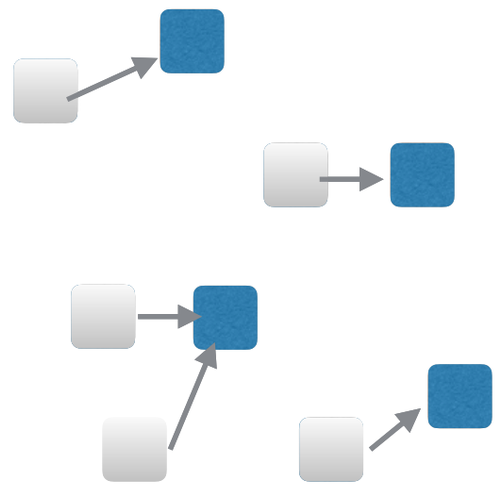
Spencer DSL
— All Data of Strings —

```
HeapReferredFrom(
  InstanceOf(java.lang.String)
)
```



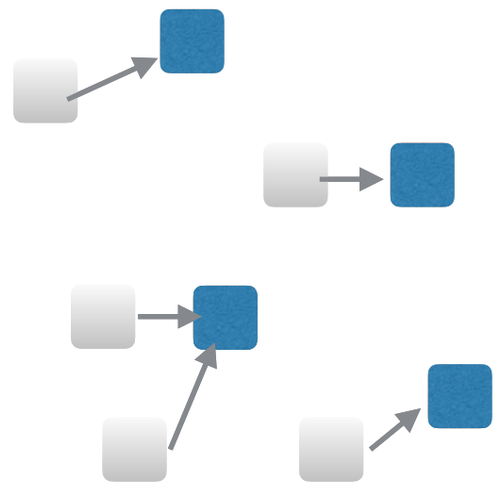
Spencer DSL
— All Data of Strings —

```
HeapReferredFrom(
  InstanceOf(java.lang.String)
)
```



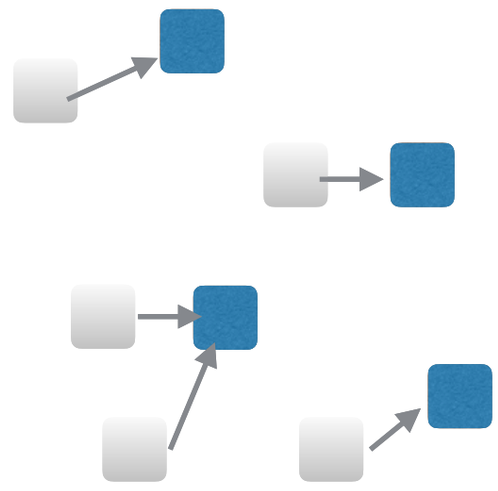
Spencer DSL
— All Shared Data of Strings —

```
And(
  HeapReferredFrom(
    InstanceOf(java.lang.String)
  )
)
```



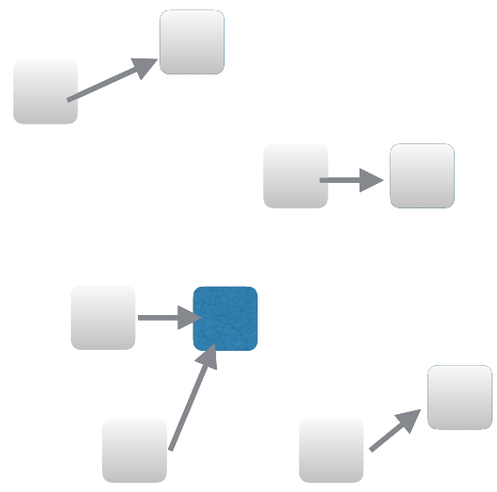
Spencer DSL
— All Shared Data of Strings —

```
And(
  HeapReferredFrom(
    InstanceOf(java.lang.String)
  )
  Not(HeapUniqueObj())
)
```



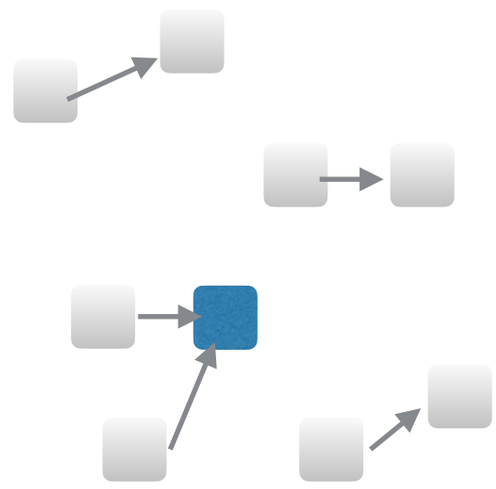
Spencer DSL
— All Shared Data of Strings —

```
And(
  HeapReferredFrom(
    InstanceOf(java.lang.String)
  )
  Not(HeapUniqueObj())
)
```



Spencer DSL
— All Objects Sharing Data With Strings —

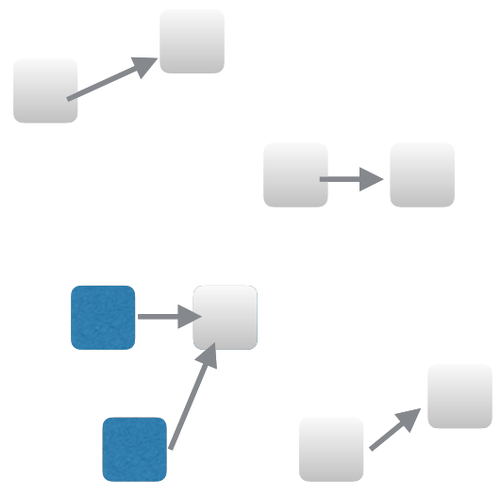
```
HeapRefersTo(
 And(
  HeapReferredFrom(
   InstanceOf(java.lang.String)
  )
  Not(HeapUniqueObj())
 )
)
```



Spencer DSL

— All Objects Sharing Data With Strings —

```
HeapRefersTo(
 And(
  HeapReferredFrom(
   InstanceOf(java.lang.String)
  )
  Not(HeapUniqueObj())
 )
)
```



Spencer DSL: Primitive Queries

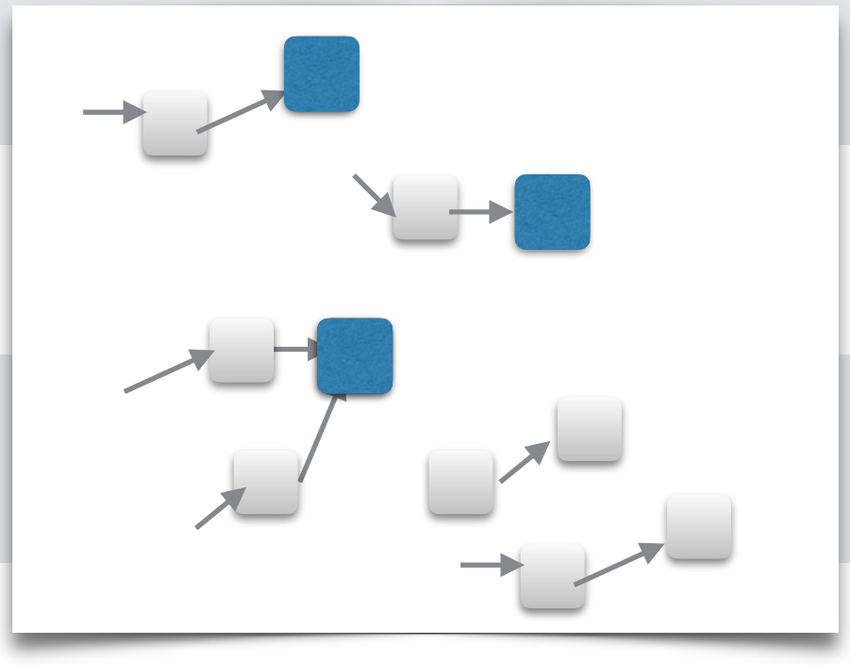
Query	Meaning
MutableObj()	Objects that are mutated after being constructed.
InstanceOf(`java.foo.Bar`)	Instances of a given class.
StationaryObj()	Objects that are never written after being read for the first time — “lazily initialised immutability”.
HeapUniqueObj()	Objects that have one, not more aliases from fields of other objects.
TinyObj()	Objects that have no field references to other objects.
…	

Spencer DSL: Query Combinators
— Walking the Reference Graph —

Query	Meaning
RefersTo(Q)	Objects that have references to objects selected by Q.
CanReach(Q)	
ReferredFrom(Q)	
ReachableFrom(Q)	
Deeply(Q)	

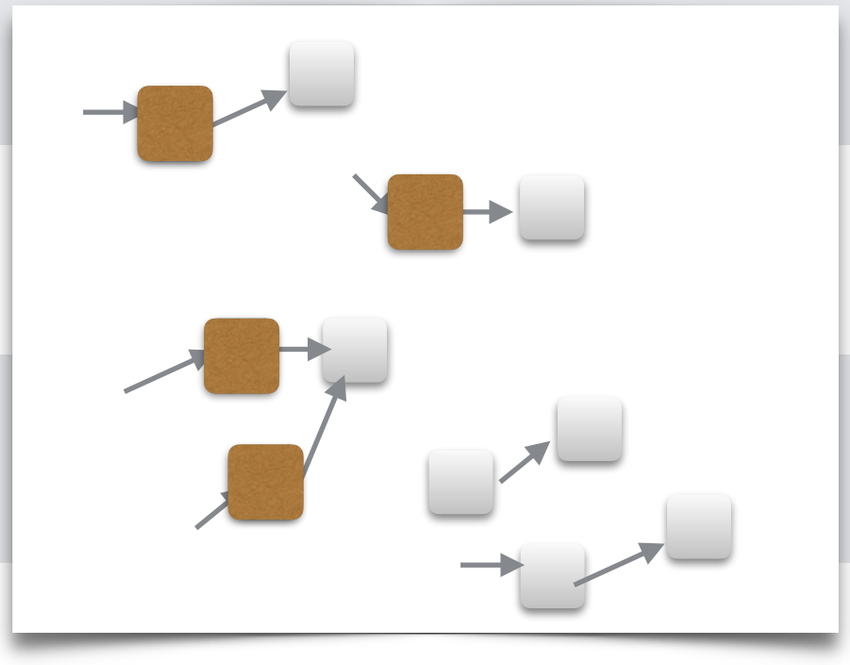
Spencer DSL: Query Combinators
— Walking the Reference Graph —

Query	Meaning
RefersTo(Q)	Objects that have references to objects selected by Q.
CanReach(Q)	
ReferredFrom(Q)	
ReachableFrom(Q)	
Deeply(Q)	



Spencer DSL: Query Combinators
— Walking the Reference Graph —

Query	Meaning
RefersTo(Q)	Objects that have references to objects selected by Q.
CanReach(Q)	
ReferredFrom(Q)	
ReachableFrom(Q)	
Deeply(Q)	

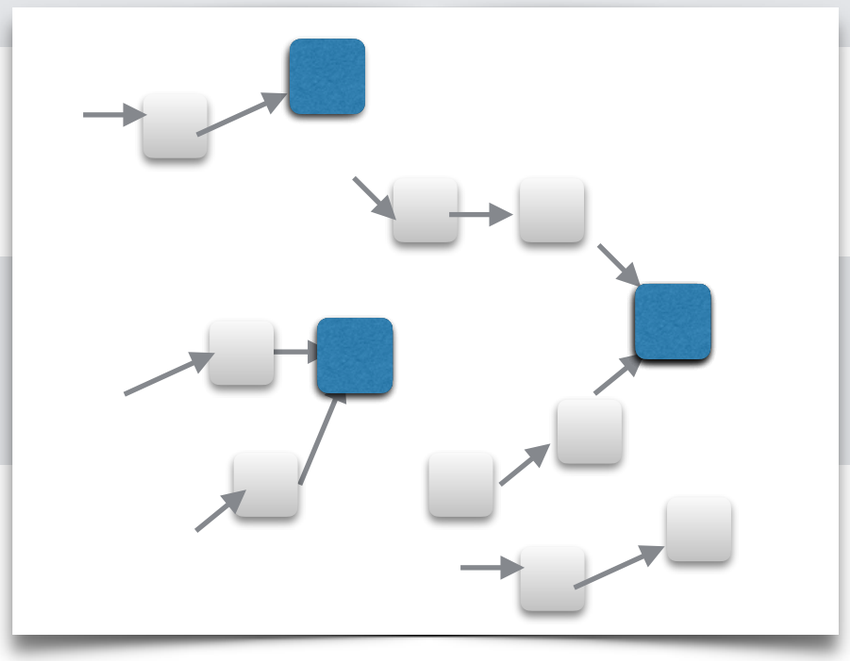


Spencer DSL: Query Combinators
— Walking the Reference Graph —

Query	Meaning
RefersTo(Q)	Objects that have references to objects selected by Q.
CanReach(Q)	Objs that have **transitive** references to objs selected by Q.
ReferredFrom(Q)	
ReachableFrom(Q)	
Deeply(Q)	

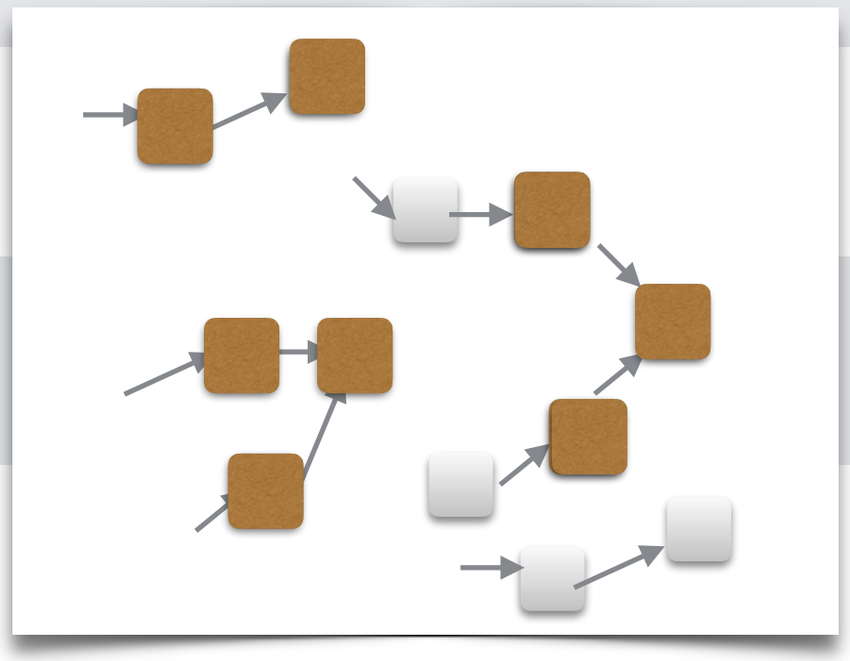
Spencer DSL: Query Combinators
— Walking the Reference Graph —

Query	Meaning
RefersTo(Q)	Objects that have references to objects selected by Q.
CanReach(Q)	Objs that have **transitive** references to objs selected by Q.
ReferredFrom(Q)	
ReachableFrom(Q)	
Deeply(Q)	



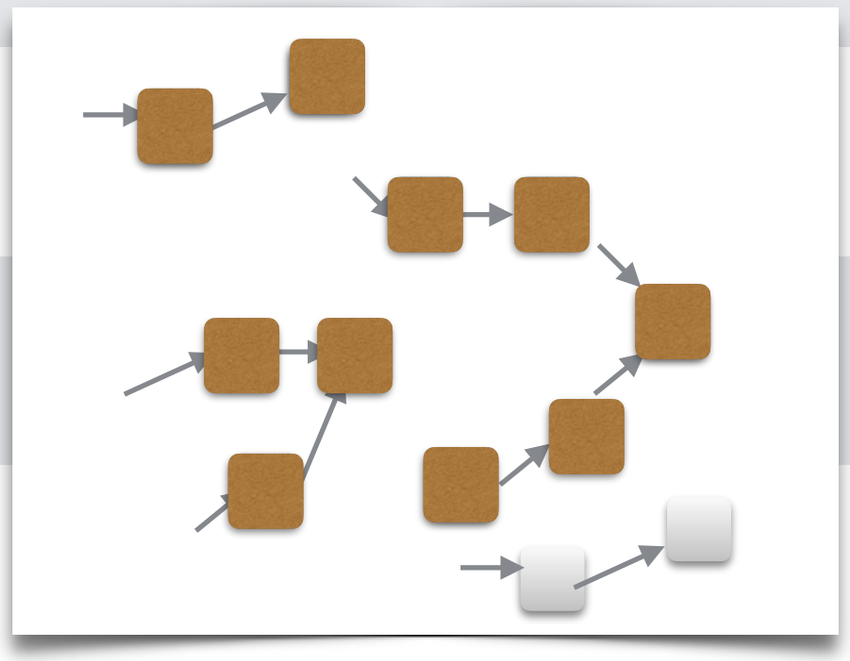
Spencer DSL: Query Combinators
— Walking the Reference Graph —

Query	Meaning
RefersTo(Q)	Objects that have references to objects selected by Q.
CanReach(Q)	Objs that have **transitive** references to objs selected by Q.
ReferredFrom(Q)	
ReachableFrom(Q)	
Deeply(Q)	



Spencer DSL: Query Combinators
— Walking the Reference Graph —

Query	Meaning
RefersTo(Q)	Objects that have references to objects selected by Q.
CanReach(Q)	Objs that have **transitive** references to objs selected by Q.
ReferredFrom(Q)	
ReachableFrom(Q)	
Deeply(Q)	



Spencer DSL: Query Combinators
— Walking the Reference Graph —

Query	Meaning
RefersTo(Q)	Objects that have references to objects selected by Q.
CanReach(Q)	Objs that have **transitive** references to objs selected by Q.
ReferredFrom(Q)	Objects that are referenced from objects selected by Q.
ReachableFrom(Q)	
Deeply(Q)	

Spencer DSL: Query Combinators
— Walking the Reference Graph —

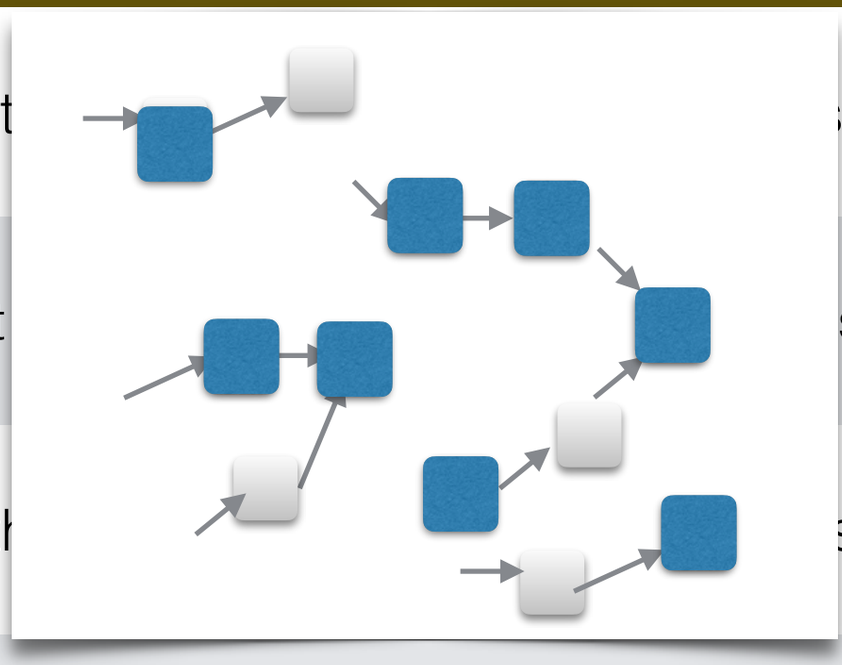
Query	Meaning
RefersTo(Q)	Objects that have references to objects selected by Q.
CanReach(Q)	Objs that have **transitive** references to objs selected by Q.
ReferredFrom(Q)	Objects that are referenced from objects selected by Q.
ReachableFrom(Q)	Objs that are **transitively** referenced from objs selected by Q.
Deeply(Q)	

Spencer DSL: Query Combinators
— Walking the Reference Graph —

Query	Meaning
RefersTo(Q)	Objects that have references to objects selected by Q.
CanReach(Q)	Objs that have **transitive** references to objs selected by Q.
ReferredFrom(Q)	Objects that are referenced from objects selected by Q.
ReachableFrom(Q)	Objs that are **transitively** referenced from objs selected by Q.
Deeply(Q)	Objects selected by Q that can only transitively reach objects selected by Q.

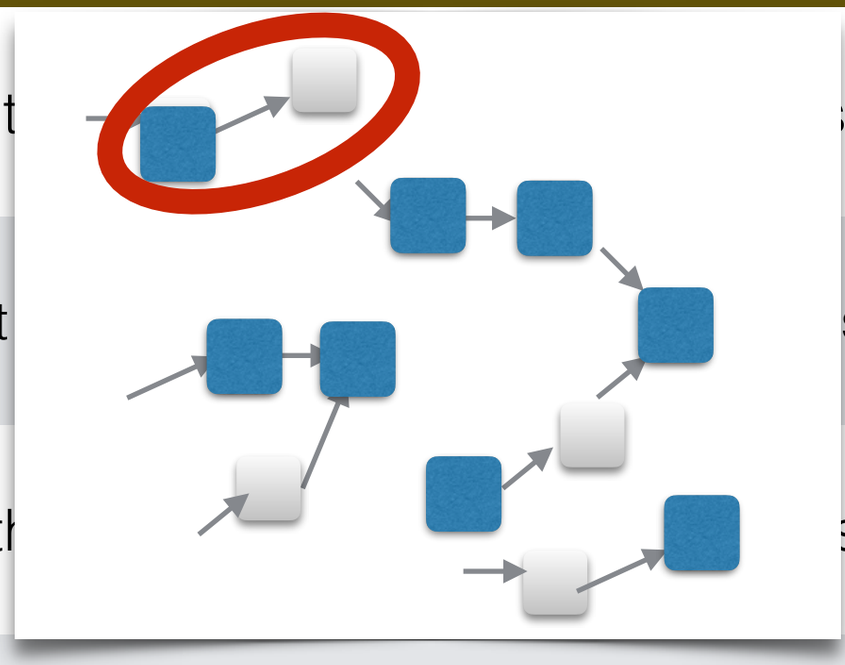
Spencer DSL: Query Combinators
— Walking the Reference Graph —

Query	Meaning
RefersTo(Q)	Objects t…elected by Q.
CanReach(Q)	Objs that …selected by Q.
ReferredFrom(Q)	Objects th…selected by Q.
ReachableFrom(Q)	Objs that are **transitively** referenced from objs selected by Q.
Deeply(Q)	Objects selected by Q that can only transitively reach objects selected by Q.



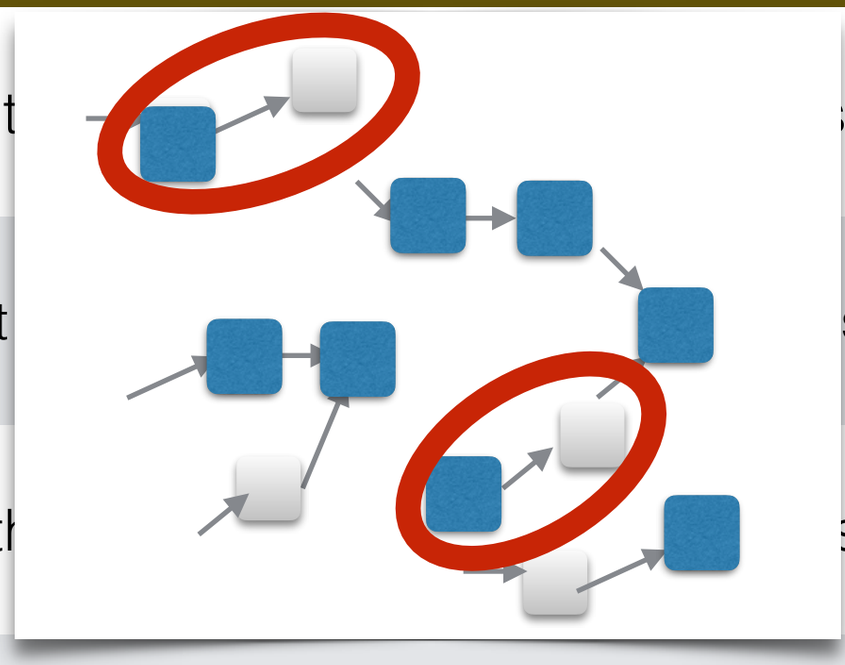
Spencer DSL: Query Combinators
— Walking the Reference Graph —

Query	Meaning
RefersTo(Q)	Objects t…elected by Q.
CanReach(Q)	Objs that …selected by Q.
ReferredFrom(Q)	Objects th…selected by Q.
ReachableFrom(Q)	Objs that are **transitively** referenced from objs selected by Q.
Deeply(Q)	Objects selected by Q that can only transitively reach objects selected by Q.



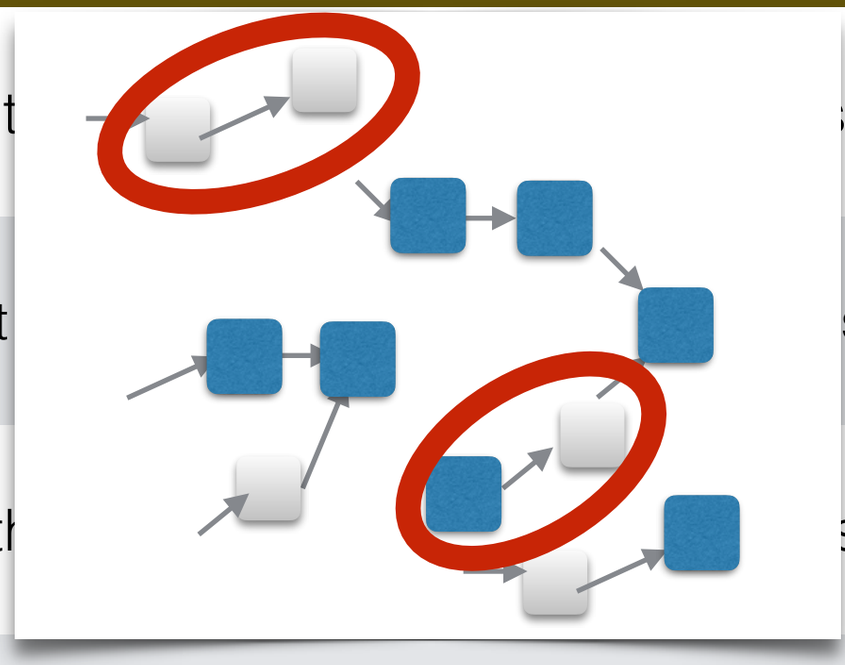
Spencer DSL: Query Combinators
— Walking the Reference Graph —

Query	Meaning
RefersTo(Q)	Objects t…elected by Q.
CanReach(Q)	Objs that …selected by Q.
ReferredFrom(Q)	Objects th…selected by Q.
ReachableFrom(Q)	Objs that are **transitively** referenced from objs selected by Q.
Deeply(Q)	Objects selected by Q that can only transitively reach objects selected by Q.



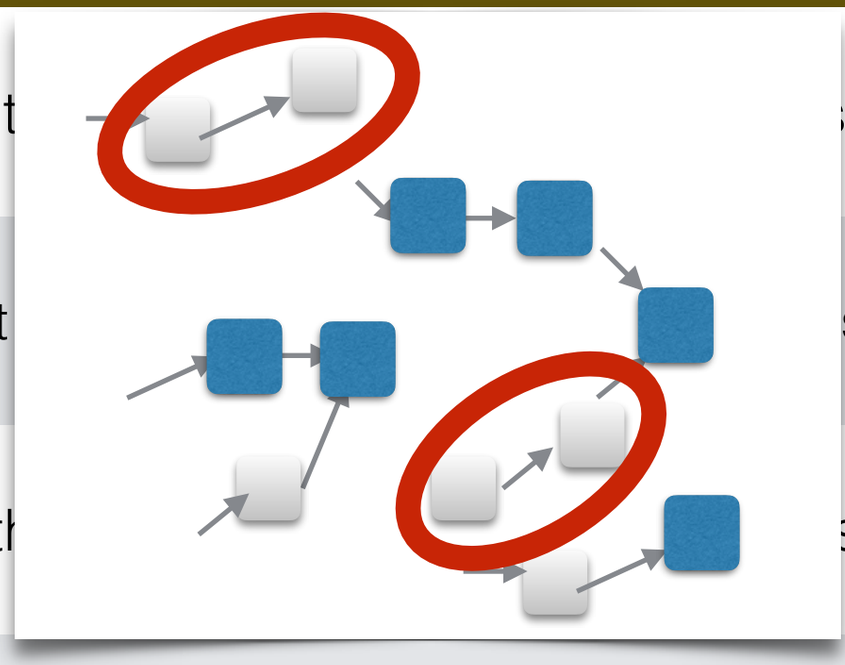
Spencer DSL: Query Combinators
— Walking the Reference Graph —

Query	Meaning
RefersTo(Q)	Objects t…elected by Q.
CanReach(Q)	Objs that …selected by Q.
ReferredFrom(Q)	Objects th…selected by Q.
ReachableFrom(Q)	Objs that are **transitively** referenced from objs selected by Q.
Deeply(Q)	Objects selected by Q that can only transitively reach objects selected by Q.



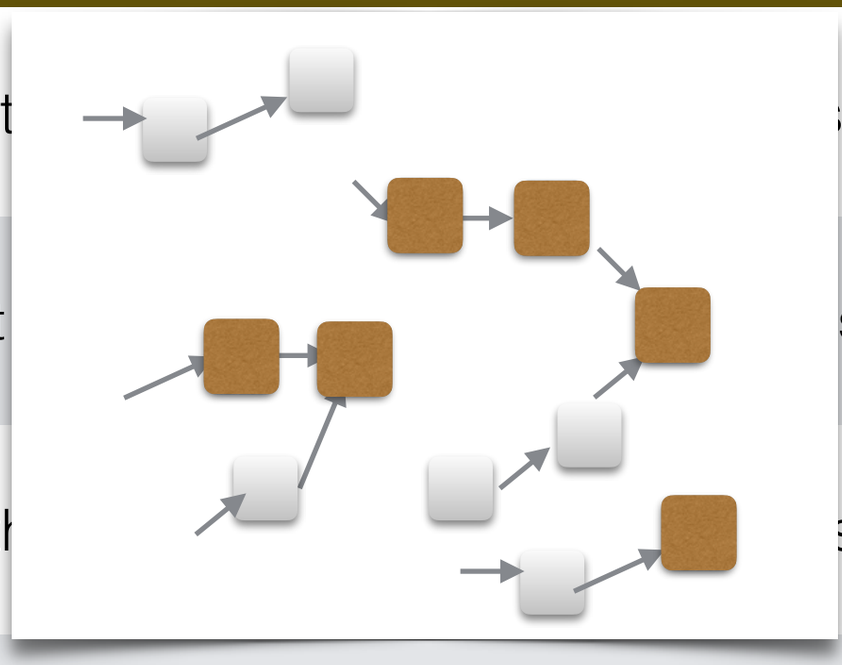
Spencer DSL: Query Combinators
— Walking the Reference Graph —

Query	Meaning
RefersTo(Q)	Objects t…elected by Q.
CanReach(Q)	Objs that …selected by Q.
ReferredFrom(Q)	Objects th…selected by Q.
ReachableFrom(Q)	Objs that are **transitively** referenced from objs selected by Q.
Deeply(Q)	Objects selected by Q that can only transitively reach objects selected by Q.



Spencer DSL: Query Combinators
— Walking the Reference Graph —

Query	Meaning
RefersTo(Q)	Objects t…elected by Q.
CanReach(Q)	Objs that …selected by Q.
ReferredFrom(Q)	Objects th…selected by Q.
ReachableFrom(Q)	Objs that are **transitively** referenced from objs selected by Q.
Deeply(Q)	Objects selected by Q that can only transitively reach objects selected by Q.



Spencer DSL: Query Combinators

— Walking the Heap —

Query	Meaning
HeapRefersTo(Q)	Objects that have references to objects selected by Q.
CanHeapReach(Q)	Objs th… Q.
HeapReferredFrom(Q)	Objects… Q.
HeapReachableFrom(Q)	Objs tha… Q.
HeapDeeply(Q)	Objects selected by Q that can only transitively reach objects selected by Q.

Like before — but only considering fields, rather than fields and stack variables

Spencer DSL: Query Combinators
— Logical Connectives —

Query	Meaning
And(*Q Q' ..*)	Objects that are selected by all inner queries — set intersection.
Or(*Q Q' ..*)	
Not(*Q*)	

Spencer DSL: Query Combinators
— Logical Connectives —

Query	Meaning
And(Q Q' ..**)**	Objects that are selected by all inner queries — set intersection.
Or(Q Q' ..**)**	Objects that are selected by at least one inner queries — set union.
Not(Q**)**	

Spencer DSL: Query Combinators
— Logical Connectives —

Query	Meaning
And(Q Q' ..**)**	Objects that are selected by all inner queries — set intersection.
Or(Q Q' ..**)**	Objects that are selected by at least one inner queries — set union.
Not(Q**)**	Objects that are not selected by the inner query.

Spencer: Tracing as a Service

Spencer hosts trace data for you to analyse.

We built a DSL for object queries
that lets you explore a data set iteratively.

http://spencer-t.racing

- “Mining for Safety using Interactive Trace Analysis”, *Workshop on Quantitative Aspects of Programming Languages and Systems (QAPL) 2017*

- *“Spencer: Interactive Heap Analysis for the Masses”, 14th International Conference on Mining Software Repositories (MSR) 2017*

Stephan Brandauer, Tobias Wrigstad
http://stbr.me/spencer
sbrandauer

Demo

http://spencer.it.uu.se/query/test/InstanceOf(java.lang.String)

Demo Failure Slides

HeapRefersTo(And(HeapRefer ×

www.spencer-t.racing/query/test/HeapRefersTo(And(HeapReferredFrom(InstanceOf(java.lang.String))%20Not(Hea...

Spencer

Available Data Sets

Query

✔ HeapRefersTo(And(HeapReferredFrom(InstanceOf(java.lang.String)) Not(HeapUniqueObj())))

Objects that are field-referring to objects that are heap-referred to from objects that are instances of cl java.lang.String, and not are never aliased.

refine query ›

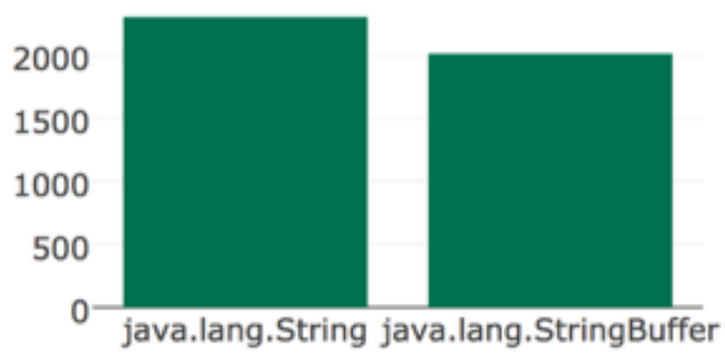
dFrom(InstanceOf(java.lang.String)) Not(HeapUniqueObj()))) 4%

Object Variables

thread

klass

# of objs p. klass



allocationSite

HeapRefersTo(And(HeapRefer ×

www.spencer-t.racing/query/test/HeapRefersTo(And(HeapReferredFrom(InstanceOf(java.lang.String))%20Not(Hea...

Spencer

Available Data Sets

Query

✔ HeapRefersTo(And(HeapReferredFrom(InstanceOf(java.lang.String)) Not(HeapUniqueObj())))

Objects that are field-referring to objects that are heap-referred to from objects that are instances of cl java.lang.String, and not are never aliased.

refine query ›

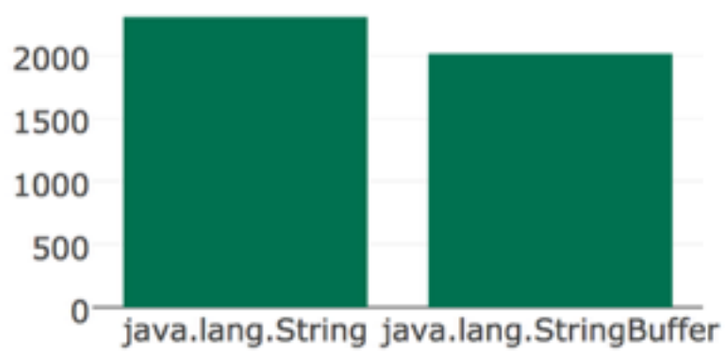
dFrom(InstanceOf(java.lang.String)) Not(HeapUniqueObj()))) 4%

Object Variables

thread

klass

# of objs p. klass



allocationSite

HeapRefersTo(And(HeapRefer ×

www.spencer-t.racing/query/test/HeapRefersTo(And(HeapReferredFrom(InstanceOf(java.lang.String))%20Not(Hea...

Spencer

Available Data Sets

Query

✔ HeapRefersTo(And(HeapReferredFrom(InstanceOf(java.lang.String)) Not(HeapUniqueObj())))

Objects that are field-referring to objects that are heap-referred to from objects that are instances of cl java.lang.String, and not are never aliased.

refine query ›

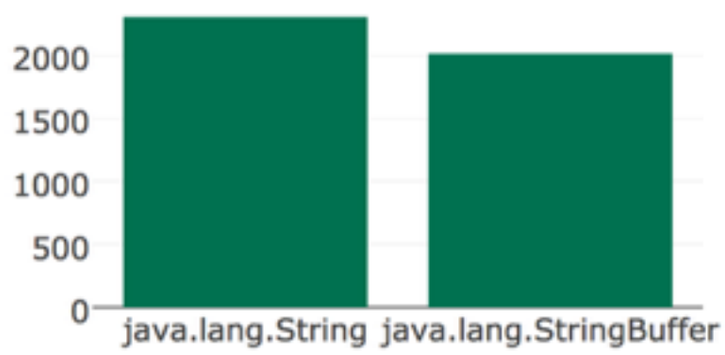
dFrom(InstanceOf(java.lang.String)) Not(HeapUniqueObj()))) 4%

Object Variables

thread

klass

# of objs p. klass



allocationSite

HeapRefersTo(And(HeapRefer ×

www.spencer-t.racing/query/test/HeapRefersTo(And(HeapReferredFrom(InstanceOf(java.lang.String))%20Not(Hea...

Spencer

Available Data Sets

Query

HeapRefersTo(And(HeapReferredFrom(InstanceOf(java.lang.String)) Not(HeapUniqueObj())))

Objects that are field-referring to objects that are heap-referred to from objects that are instances of c java.lang.String, and not are never aliased.

refine query >

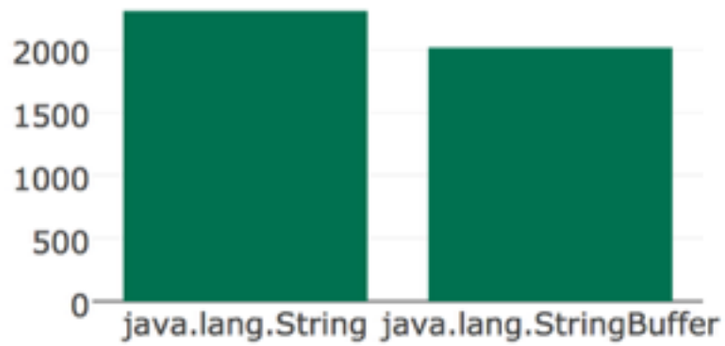
dFrom(InstanceOf(java.lang.String)) Not(HeapUniqueObj()))) 4%

Object Variables

thread

klass

# of objs p. klass



allocationSite

Query

✔ HeapRefersTo(And(HeapReferredFrom(InstanceOf(java.lang.String)) Not(HeapUniqueObj())))

Objects that are field-referring to objects that are heap-referred to from objects that are instances of cl java.lang.String, and not are never aliased.

refine query ›

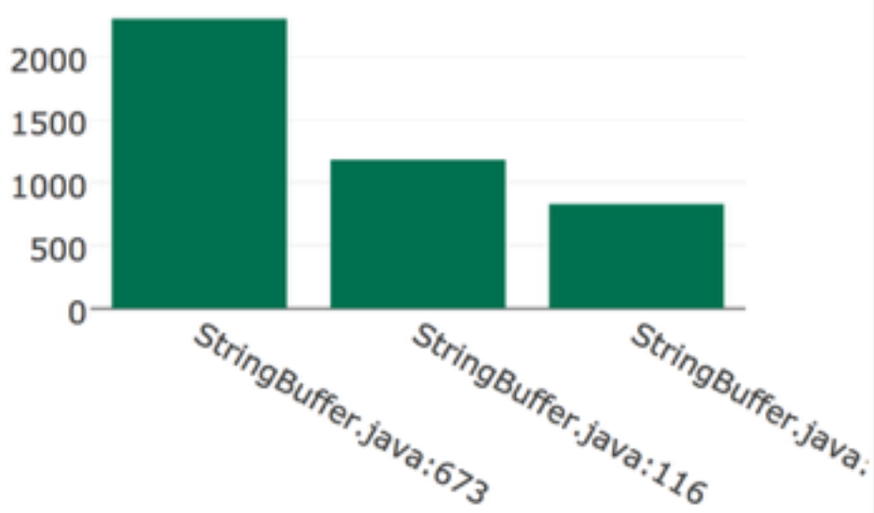
dFrom(InstanceOf(java.lang.String)) Not(HeapUniqueObj())))

4%

Object Variables

thread | klass | allocationSite

# of objs p. allocationSite

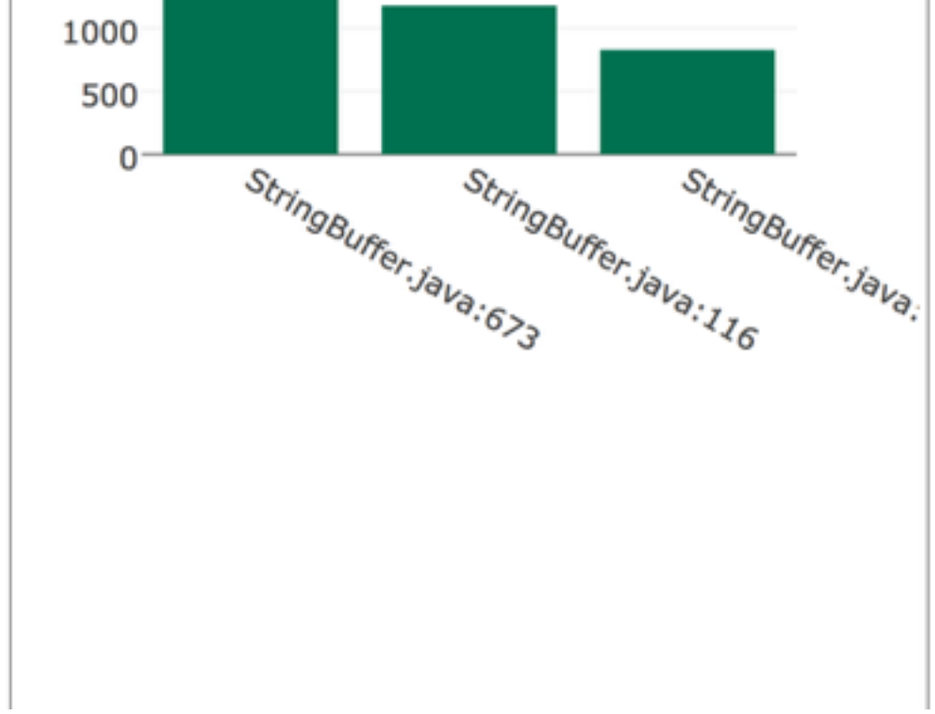


Per Field Statistics

Percentage of objects referred to from a field that was selected by the query.

HeapUniqueObj()

Field Name	Selected [%]
net.sourceforge.pmd.rules.basic.BooleanInstantiation::description	100
net.sourceforge.pmd.rules.basic.BrokenNullCheck::properties	100
net.sourceforge.pmd.rules.basic.BrokenNullCheck::examples	100
net.sourceforge.pmd.rules.basic.BrokenNullCheck::description	100
net.sourceforge.pmd.rules.basic.BooleanInstantiation::properties	100
net.sourceforge.pmd.rules.basic.BooleanInstantiation::ruleChainVisits	100
sun.util.locale.provider.LocaleServiceProviderPool::providers	100
java.io.BufferedWriter::cb	100
sun.util.locale.provider.LocaleResources::cache	100
sun.util.locale.provider.JRELocaleProviderAdapter::numberFormatProvider	100
sun.util.locale.provider.JRELocaleProviderAdapter::localeResourcesMap	100

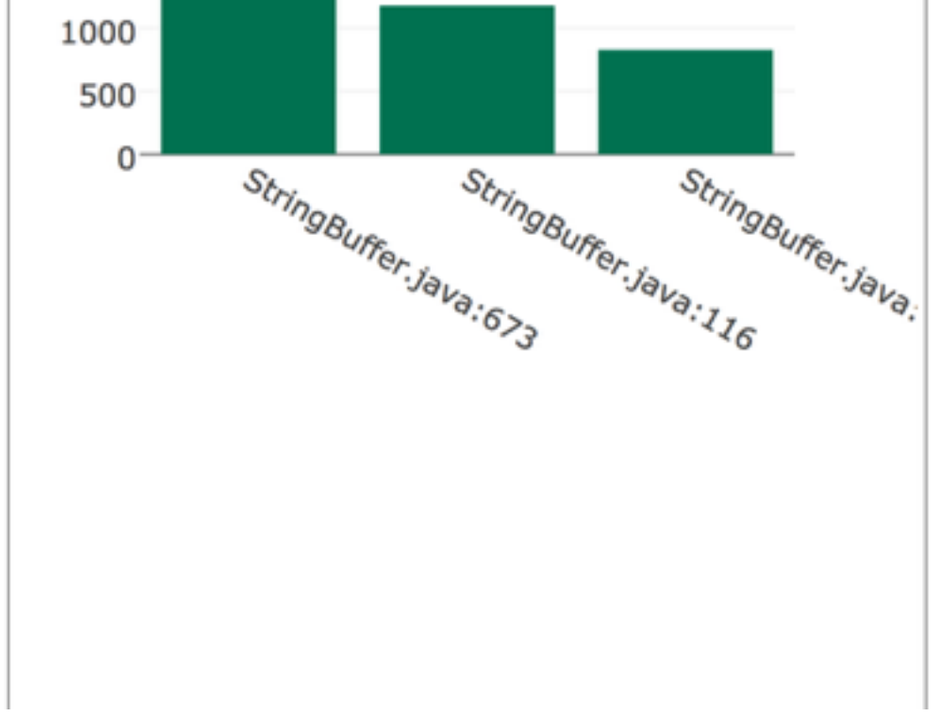


Per Field Statistics

Percentage of objects referred to from a field that was selected by the query.

HeapUniqueObj()

Field Name	Selected [%]
net.sourceforge.pmd.rules.basic.BooleanInstantiation::description	100
net.sourceforge.pmd.rules.basic.BrokenNullCheck::properties	100
net.sourceforge.pmd.rules.basic.BrokenNullCheck::examples	100
net.sourceforge.pmd.rules.basic.BrokenNullCheck::description	100
net.sourceforge.pmd.rules.basic.BooleanInstantiation::properties	100
net.sourceforge.pmd.rules.basic.BooleanInstantiation::ruleChainVisits	100
sun.util.locale.provider.LocaleServiceProviderPool::providers	100
java.io.BufferedWriter::cb	100
sun.util.locale.provider.LocaleResources::cache	100
sun.util.locale.provider.JRELocaleProviderAdapter::numberFormatProvider	100
sun.util.locale.provider.JRELocaleProviderAdapter::localeResourcesMap	100

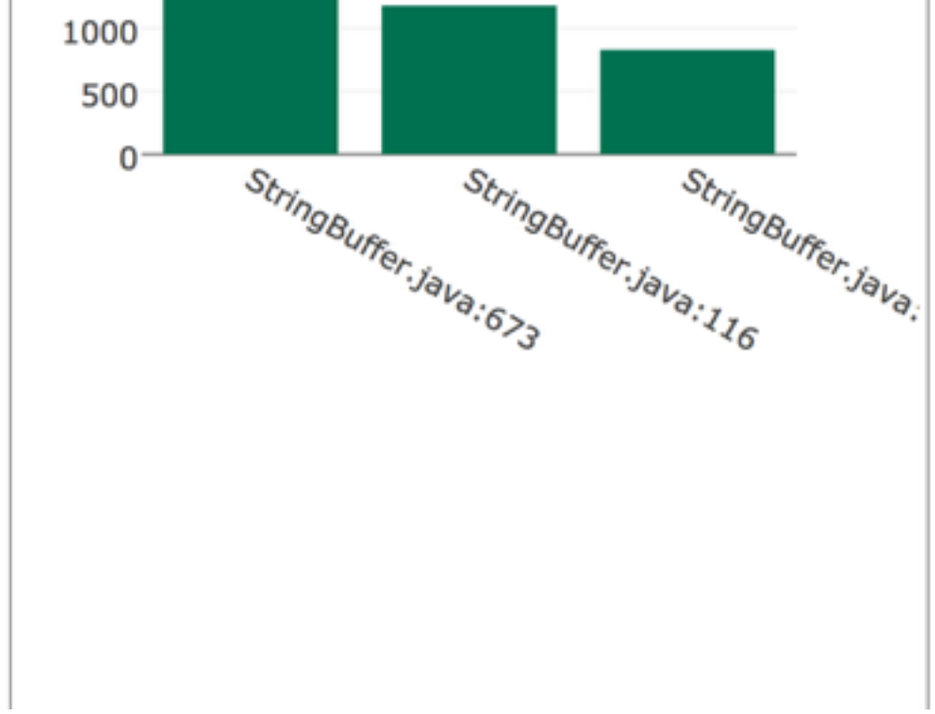


Per Field Statistics

Percentage of objects referred to from a field that was selected by the query.

HeapUniqueObj()

Field Name	Selected [%]
net.sourceforge.pmd.rules.basic.BooleanInstantiation::description	100
net.sourceforge.pmd.rules.basic.BrokenNullCheck::properties	100
net.sourceforge.pmd.rules.basic.BrokenNullCheck::examples	100
net.sourceforge.pmd.rules.basic.BrokenNullCheck::description	100
net.sourceforge.pmd.rules.basic.BooleanInstantiation::properties	100
net.sourceforge.pmd.rules.basic.BooleanInstantiation::ruleChainVisits	100
sun.util.locale.provider.LocaleServiceProviderPool::providers	100
java.io.BufferedWriter::cb	100
sun.util.locale.provider.LocaleResources::cache	100
sun.util.locale.provider.JRELocaleProviderAdapter::numberFormatProvider	100
sun.util.locale.provider.JRELocaleProviderAdapter::localeResourcesMap	100



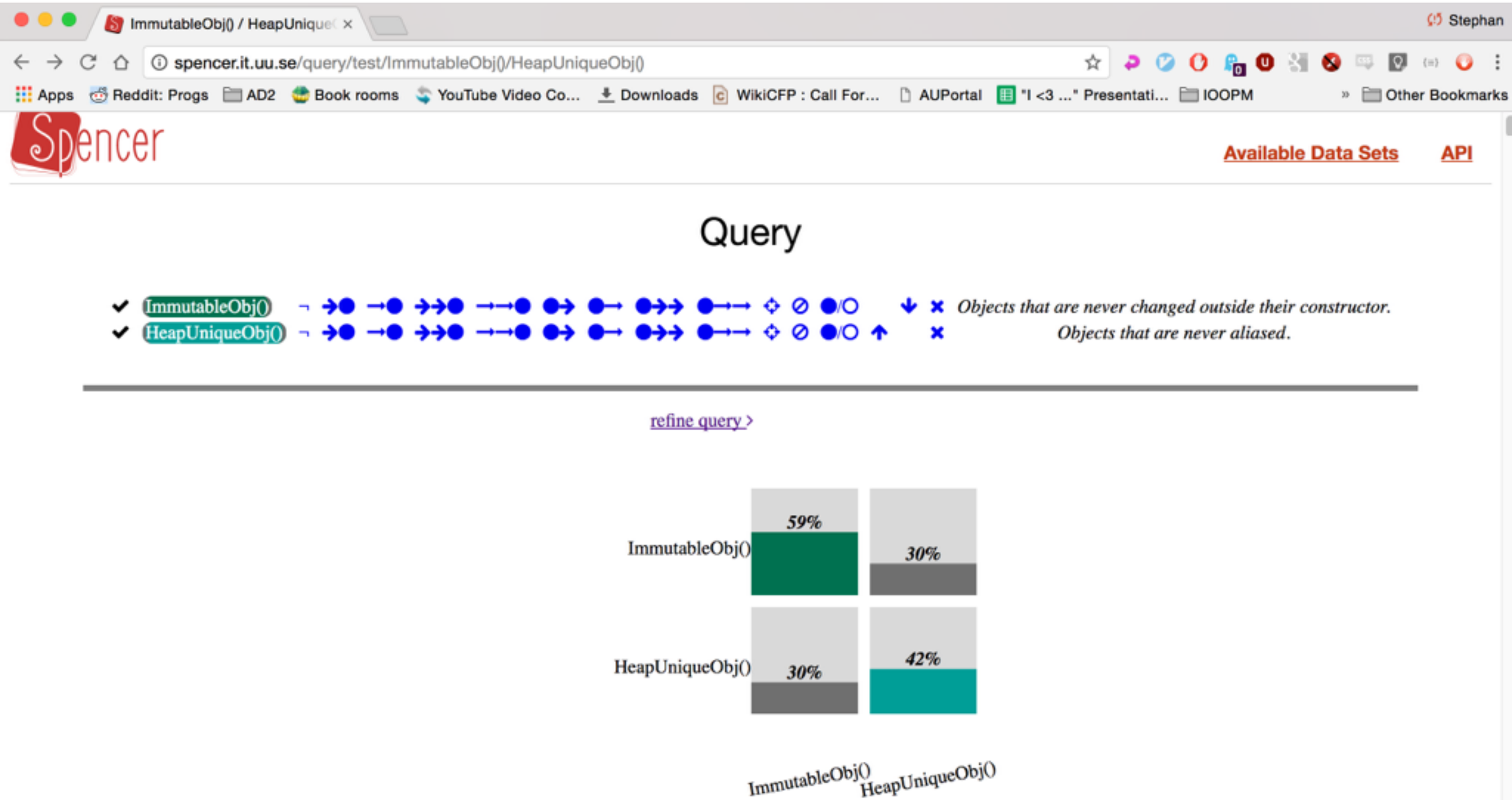
Per Field Statistics

Percentage of objects referred to from a field that was selected by the query.

HeapUniqueObj()

Field Name	Selected [%]
net.sourceforge.pmd.rules.basic.BooleanInstantiation::description	100
net.sourceforge.pmd.rules.basic.BrokenNullCheck::properties	100
net.sourceforge.pmd.rules.basic.BrokenNullCheck::examples	100
net.sourceforge.pmd.rules.basic.BrokenNullCheck::description	100
net.sourceforge.pmd.rules.basic.BooleanInstantiation::properties	100
net.sourceforge.pmd.rules.basic.BooleanInstantiation::ruleChainVisits	100
sun.util.locale.provider.LocaleServiceProviderPool::providers	100
java.io.BufferedWriter::cb	100
sun.util.locale.provider.LocaleResources::cache	100
sun.util.locale.provider.JRELocaleProviderAdapter::numberFormatProvider	100
sun.util.locale.provider.JRELocaleProviderAdapter::localeResourcesMap	100

ImmutableObj() / HeapUniqueObj()



```
$ curl http://spencer-t.racing/json/select/test/HeapRefersTo(..
  ..And(HeapReferredFrom(InstanceOf(java.lang.String))..
  ..%20Not(HeapUniqueObj())))
```

```
$ curl http://spencer-t.racing/json/select/test/HeapRefersTo(..
  ..And(HeapReferredFrom(InstanceOf(java.lang.String))..
  ..%20Not(HeapUniqueObj())))
{
  “query”:”...”,
  “objects":
    [42171,42174,42259, …]
}
```

```
$ curl http://spencer-t.racing/json/select/test/HeapRefersTo(
  ..And(HeapReferredFrom(Inst
```

```
In [35]: meta = meta_info('ImmutableObj()')
         meta.sample(10)
```

```
selected 484576 bytes
meta: got 10167219 bytes
```

Out[35]:

	allocationSite	firstusage	id	klass	lastusage	numCalls	numFieldReads	numFieldWrite
16265	ZipCoder.java:78	3910628	38553	[C	3910847	1	0	0
2501	StringBuilder.java:89	0	11435	[C	0	1	0	0
48147	ZipCoder.java:89	0	87599	[B	0	1	0	0
31971	String.java:207	7320085	62982	[C	7320298	1	0	0
31436	ZipCoder.java:89	0	61918	[B	0	1	0	0
56210	String.java:2032	12765482	100566	[C	12765509	1	0	0
33053	String.java:207	0	64394	[C	0	1	0	0
60881	StringBuffer.java:671	0	112662	[C	0	1	0	0
51189	ZipCoder.java:89	0	92728	[B	0	1	0	0
52143	String.java:1933	12257310	94185	java.lang.String	12257718	12	14	1

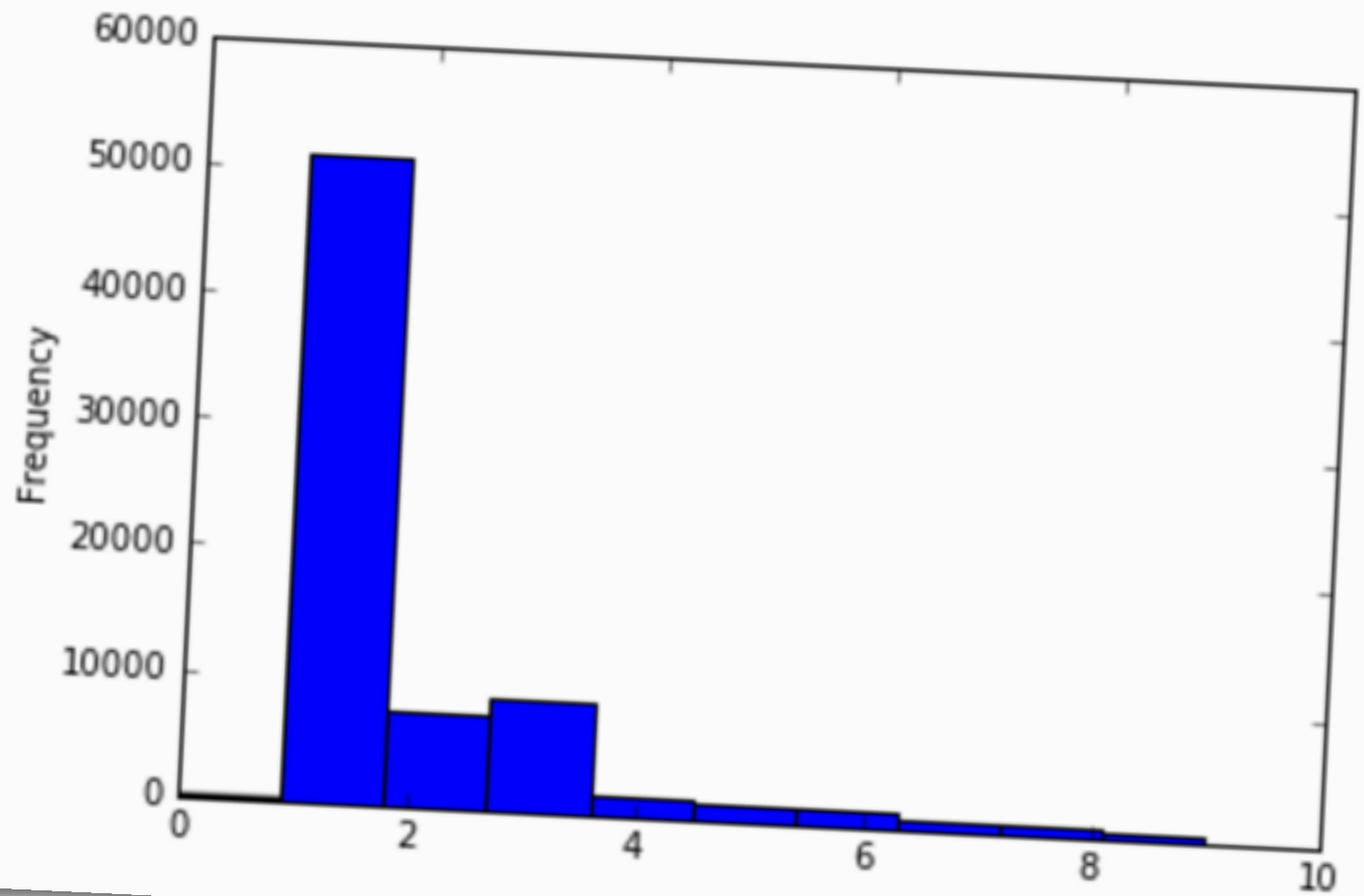
$ curl http://spencer-t.racing/json/select/test/HeapRefersTo(

Number of Calls

To show a histogram of the calls, we can do this:

In [17]:
```
meta['numCalls'].where(meta['numCalls'] < 10).plot(kind = 'hist', bins=10)
```

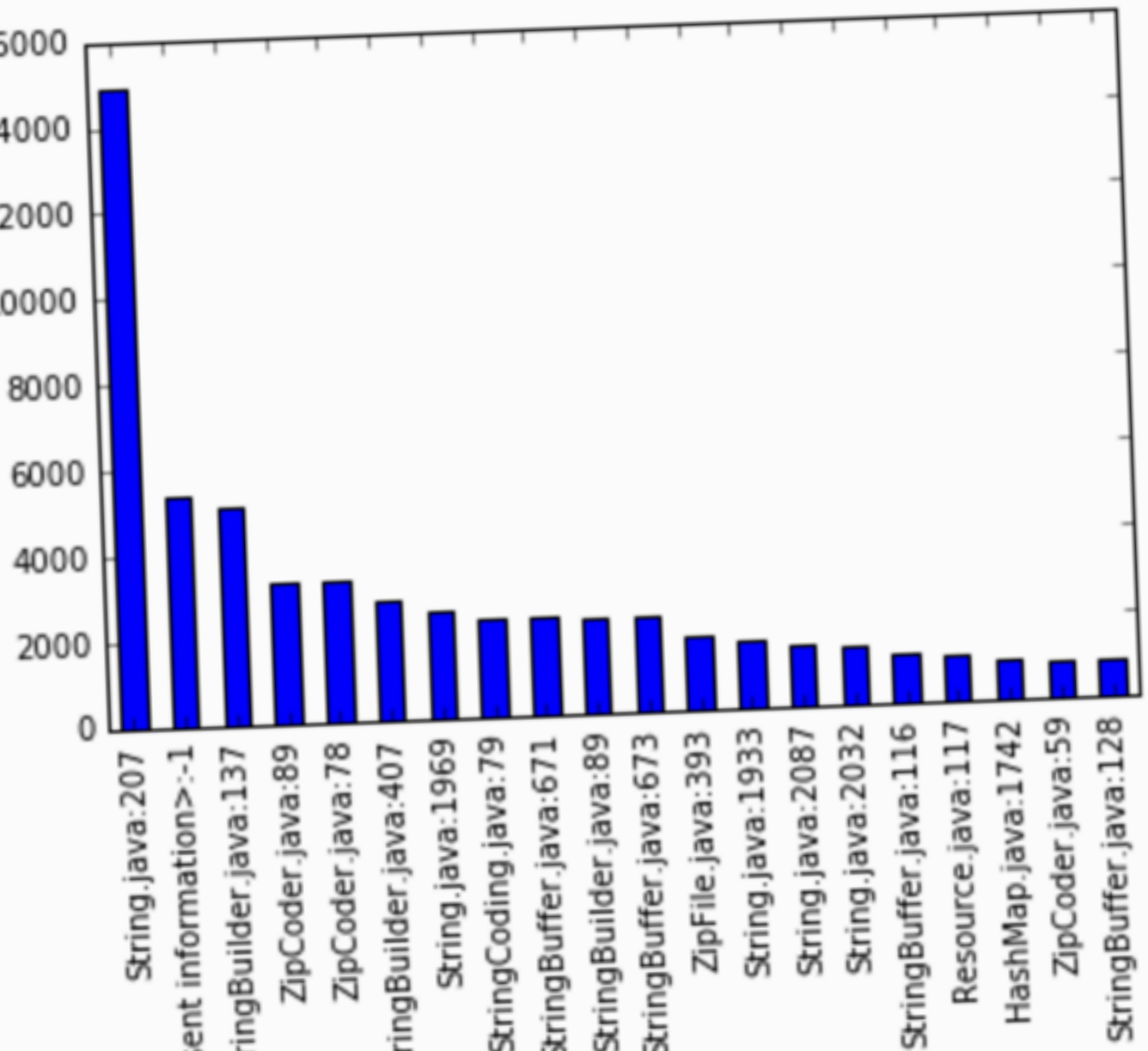
Out[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x11750bed0>
```



Number of Allocations per Allocation Site (Top 20)

```
In [18]: meta['allocationSite'].value_counts()[:20].plot(kind='bar')
```

```
Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x11731ea50>
```



Backup Slides

Dynamic Analysis	Static Analysis
false positives (“upper bound”)	false negatives (“lower bound”)
often-used code weighed stronger	all code weighed equally
easily deals with runtime code generation, dynamic code loading	easily can produce sound claims

“Safety”

unique	*at most one variable/field refers to object at a time*
stack bound	*no field ever refers to the object*
heap-unique	*at most one field refers to object at a time*
deeply immutable	*shallow immutable + can only reach (via fields) other shallow immutable objects*
shallow immutable	*object never changed outside of constructor*
safe	*at least one of the above*

Dynamic Analysis

Static Analysis

"What proportion of objects are safe?"

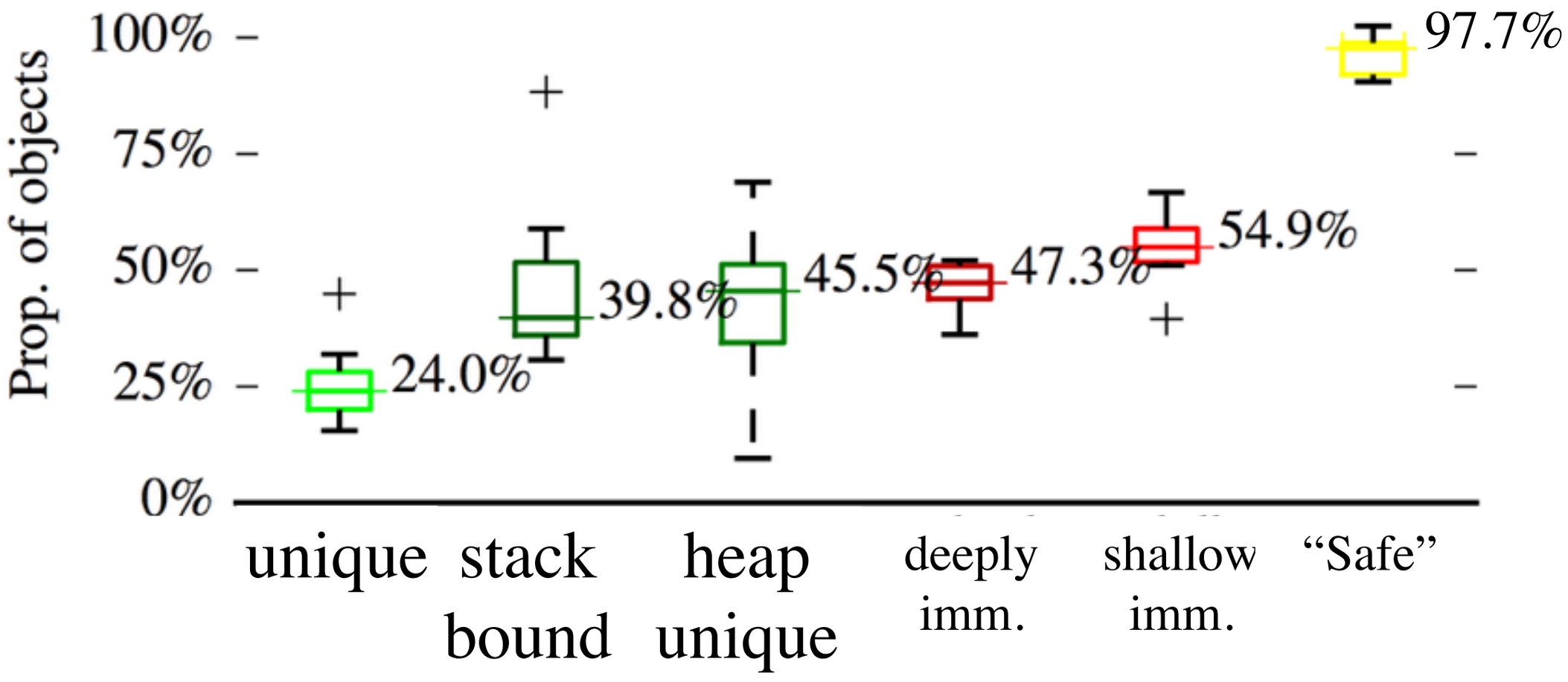
Dynamic Analysis

Static Analysis

"What proportion of classes only produce safe instances?"

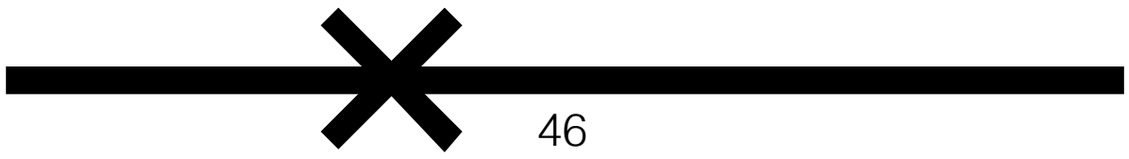
"What proportion of fields only contain safe instances?"

Per Object Analysis



Per Class Analysis

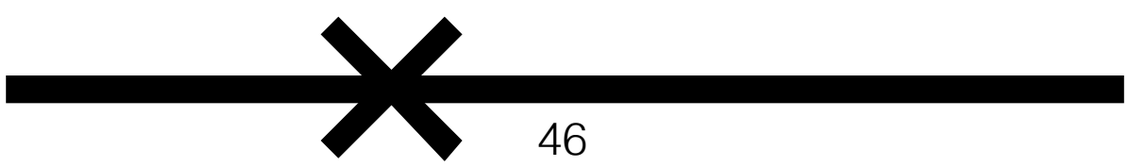
Out of all classes with more than 10 instances,
how many classes…



Per Class Analysis

Out of all classes with more than 10 instances,
how many classes…

1) … have ONLY instances that fulfil a safety property?

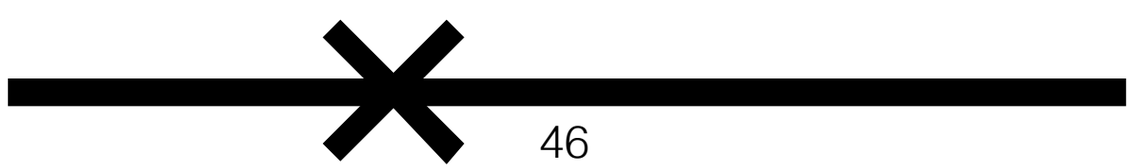


Per Class Analysis

Out of all classes with more than 10 instances,
how many classes…

1) … have ONLY instances that fulfil a safety property?

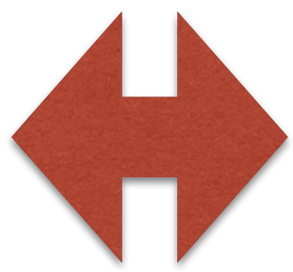
2) … have NO instances that fulfil a safety property?



Spencer DSL

— Compiling to SQL —

InstanceOf(java.lang.String)

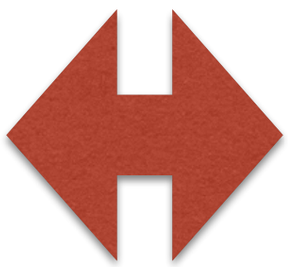


```
SELECT id FROM objects WHERE klass = 'java.lang.String'
```

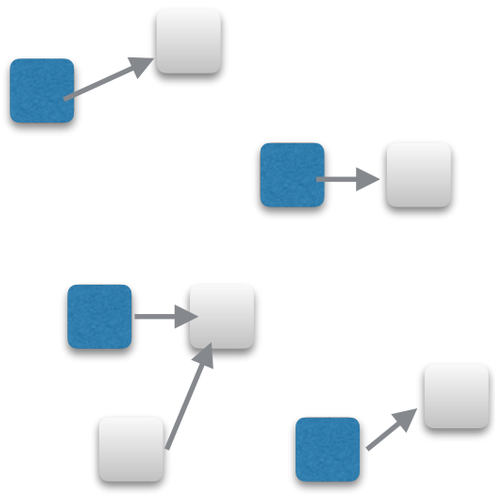
Spencer DSL

— Compiling to SQL —

InstanceOf(java.lang.String)

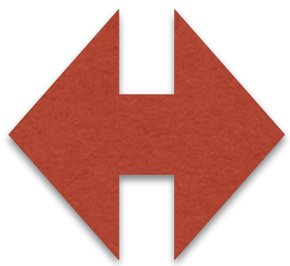


```
SELECT id FROM objects WHERE klass = 'java.lang.String'
```

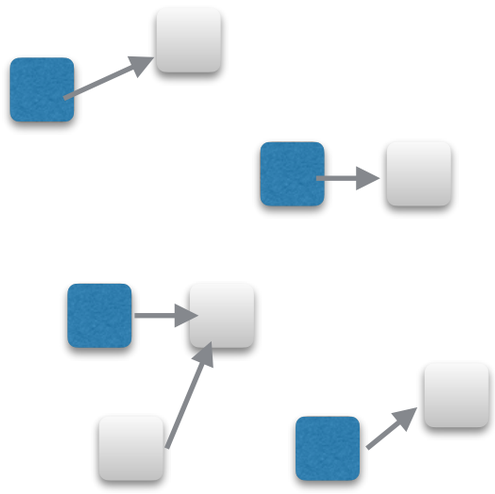


Spencer DSL

HeapReferredFrom(
  InstanceOf(java.lang.String))

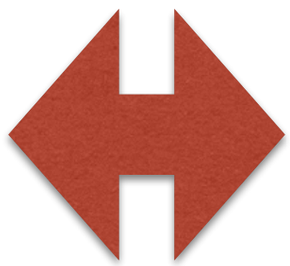


```
SELECT callee AS id
FROM   refs
WHERE  kind = 'field'
AND    caller IN (
  SELECT id FROM objects WHERE klass = 'java.lang.String'
)
```

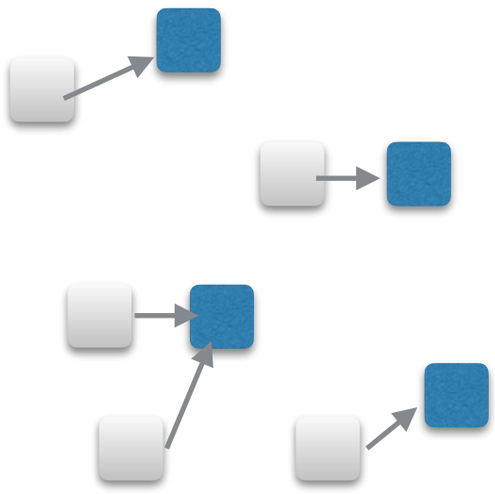


Spencer DSL

HeapReferredFrom(
  InstanceOf(java.lang.String))

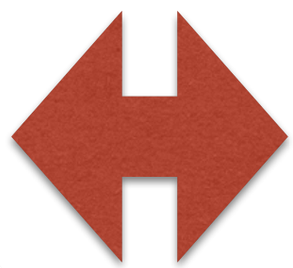


```
SELECT callee AS id
FROM   refs
WHERE  kind = 'field'
AND    caller IN (
  SELECT id FROM objects WHERE klass = 'java.lang.String'
)
```

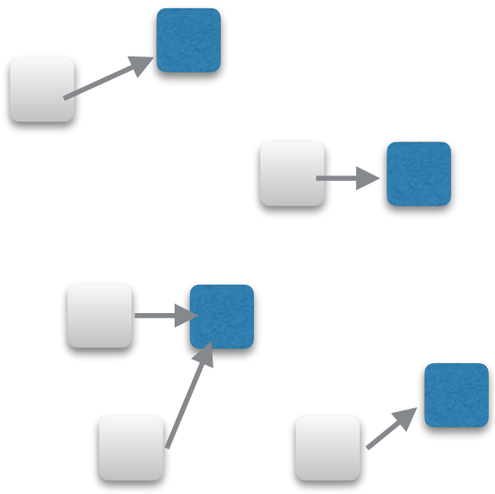


Spencer DSL

```
And(
  HeapReferredFrom(
    InstanceOf(java.lang.String))
?)
```

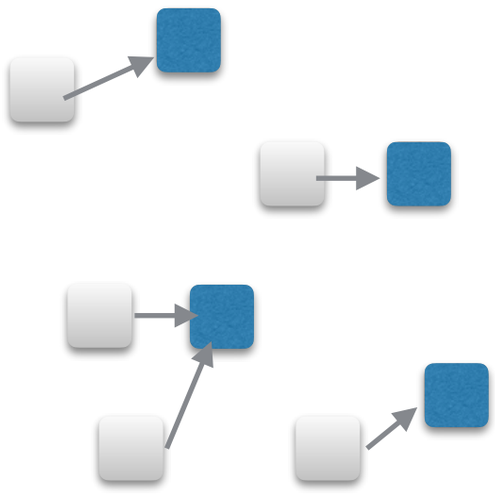


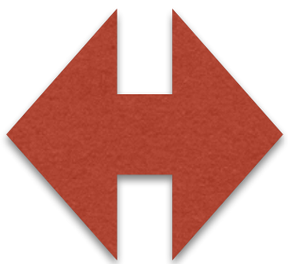
```
(
  SELECT callee AS id
  FROM   refs
  WHERE  kind = 'field'
  AND    caller IN (
    SELECT id FROM objects WHERE klass = 'java.lang.String'
  )
) INTERSECT (
?
)
)
```



Spencer DSL

```
And(
  HeapReferredFrom(
    InstanceOf(java.lang.String))
Not(HeapUniqueObj()))
```



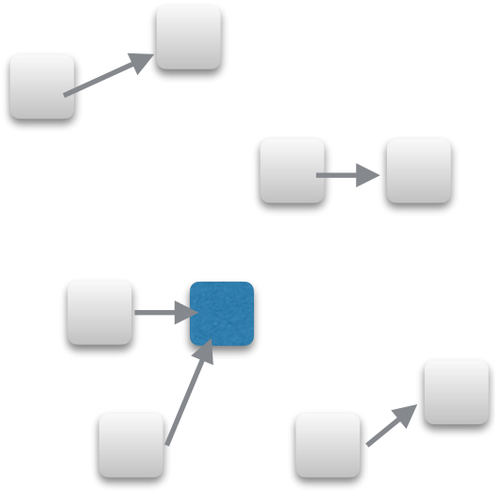


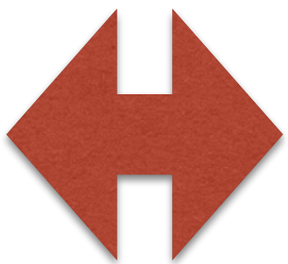
```
(
  SELECT callee AS id
  FROM   refs
  WHERE  kind = 'field'
  AND    caller IN (
    SELECT id FROM objects WHERE klass = 'java.lang.String'
  )
) INTERSECT (
  SELECT id FROM objects WHERE id > 4
  EXCEPT
    (SELECT callee AS id FROM
     (SELECT callee, time, SUM(delta) OVER(PARTITION BY callee ORDER BY time) AS sum_at_time
      FROM (
        (SELECT
           callee, refstart AS time, 1 AS delta
         FROM refs
         WHERE callee > 4 AND kind = 'field') UNION ALL (SELECT
           callee, refend AS time, -1 AS delta
         FROM refs
         WHERE callee > 4 AND kind = 'field')
      ) AS steps) AS integrated_steps
     GROUP BY callee
     HAVING MAX(sum_at_time) = 1)

  )
)
```

Spencer DSL

```
And(
  HeapReferredFrom(
    InstanceOf(java.lang.String))
Not(HeapUniqueObj()))
```



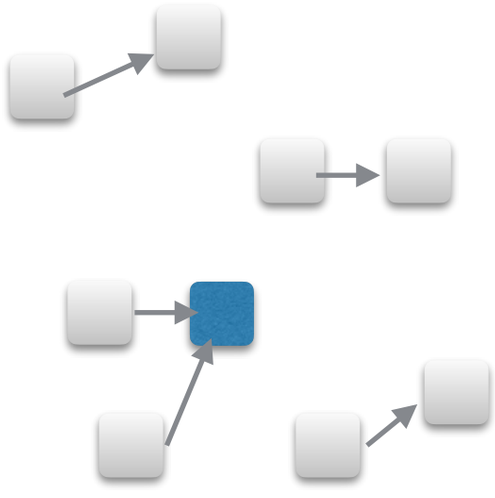


```
(
  SELECT callee AS id
  FROM   refs
  WHERE  kind = 'field'
  AND    caller IN (
    SELECT id FROM objects WHERE klass = 'java.lang.String'
  )
) INTERSECT (
  SELECT id FROM objects WHERE id > 4
  EXCEPT
    (SELECT callee AS id FROM
     (SELECT callee, time, SUM(delta) OVER(PARTITION BY callee ORDER BY time) AS sum_at_time
      FROM (
        (SELECT
           callee, refstart AS time, 1 AS delta
         FROM refs
         WHERE callee > 4 AND kind = 'field') UNION ALL (SELECT
           callee, refend AS time, -1 AS delta
         FROM refs
         WHERE callee > 4 AND kind = 'field')
      ) AS steps) AS integrated_steps
     GROUP BY callee
     HAVING MAX(sum_at_time) = 1)

  )
)
```

Spencer DSL

```
HeapRefersTo(
  And(
    HeapReferredFrom(
      InstanceOf(java.lang.String))
  Not(HeapUniqueObj()))))
```

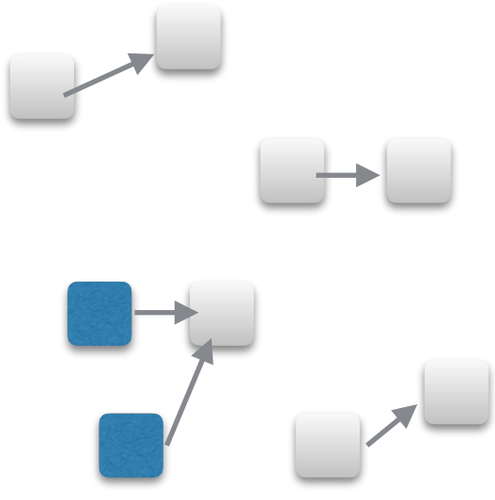


```
SELECT caller AS id
FROM   refs
WHERE  kind = 'field'
AND    callee IN (
  (
    SELECT callee AS id
    FROM   refs
    WHERE  kind = 'field'
    AND    caller IN (
      SELECT id FROM objects WHERE klass = 'java.lang.String'
    )
  ) INTERSECT (
    SELECT id FROM objects WHERE id > 4
    EXCEPT
      (SELECT callee AS id FROM
       (SELECT callee, time, SUM(delta) OVER(PARTITION BY callee ORDER BY time) AS sum_at_time
        FROM (
          (SELECT
             callee, refstart AS time, 1 AS delta
           FROM refs
           WHERE callee > 4 AND kind = 'field') UNION ALL (SELECT
             callee, refend AS time, -1 AS delta
           FROM refs
           WHERE callee > 4 AND kind = 'field')
        ) AS steps) AS integrated_steps
       GROUP BY callee
       HAVING MAX(sum_at_time) = 1)

  )
)
```

Spencer DSL

```
HeapRefersTo(
  And(
    HeapReferredFrom(
      InstanceOf(java.lang.String))
  Not(HeapUniqueObj()))))
```

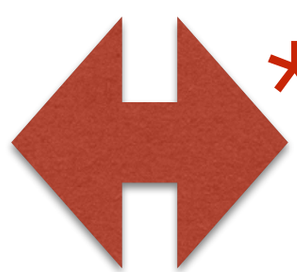


```
SELECT caller AS id
FROM   refs
WHERE  kind = 'field'
AND    callee IN (
  (
    SELECT callee AS id
    FROM   refs
    WHERE  kind = 'field'
    AND    caller IN (
      SELECT id FROM objects WHERE klass = 'java.lang.String'
    )
  ) INTERSECT (
    SELECT id FROM objects WHERE id > 4
    EXCEPT
      (SELECT callee AS id FROM
       (SELECT callee, time, SUM(delta) OVER(PARTITION BY callee ORDER BY time) AS sum_at_time
        FROM (
          (SELECT
             callee, refstart AS time, 1 AS delta
           FROM refs
           WHERE callee > 4 AND kind = 'field') UNION ALL (SELECT
             callee, refend AS time, -1 AS delta
           FROM refs
           WHERE callee > 4 AND kind = 'field')
        ) AS steps) AS integrated_steps
       GROUP BY callee
       HAVING MAX(sum_at_time) = 1)

  )
)
```

Spencer DSL

```
HeapRefersTo(
  And(
    HeapReferredFrom(
      InstanceOf(java.lang.String))
  Not(HeapUniqueObj())))
```



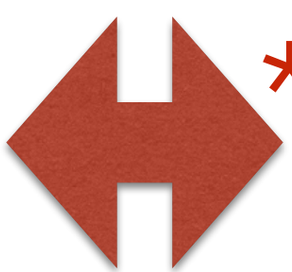
```
SELECT caller AS id
FROM   refs
WHERE  kind = 'field'
AND    callee IN (
 (
    SELECT callee AS id
    FROM   refs
    WHERE  kind = 'field'
    AND    caller IN (
      SELECT id FROM objects WHERE klass = 'java.lang.String'
    )
  ) INTERSECT (
    SELECT id FROM objects WHERE id > 4
    EXCEPT
      (SELECT callee AS id FROM
       (SELECT callee, time, SUM(delta) OVER(PARTITION BY callee ORDER BY time) AS sum_at_time
        FROM (
          (SELECT
             callee, refstart AS time, 1 AS delta
           FROM refs
           WHERE callee > 4 AND kind = 'field') UNION ALL (SELECT
             callee, refend AS time, -1 AS delta
           FROM refs
           WHERE callee > 4 AND kind = 'field')
        ) AS steps) AS integrated_steps
       GROUP BY callee
       HAVING MAX(sum_at_time) = 1)

  )
)
```

* and caching of subexpressions

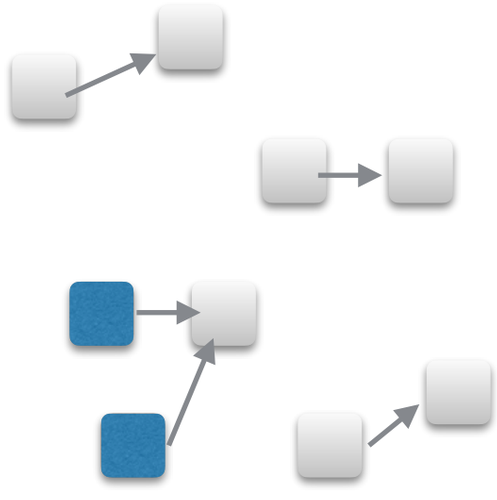
Spencer DSL

```
HeapRefersTo(
  And(
    HeapReferredFrom(
      InstanceOf(java.lang.String))
  Not(HeapUniqueObj()))))
```



```
SELECT caller AS id
FROM   refs
WHERE  kind = 'field'
AND    callee IN (
 (
    SELECT callee AS id
    FROM   refs
    WHERE  kind = 'field'
    AND    caller IN (
      SELECT id FROM objects WHERE klass = 'java.lang.String'
    )
  ) INTERSECT (
    SELECT id FROM objects WHERE id > 4
    EXCEPT
      (SELECT callee AS id FROM
       (SELECT callee, time, SUM(delta) OVER(PARTITION BY callee ORDER BY time) AS sum_at_time
        FROM (
          (SELECT
             callee, refstart AS time, 1 AS delta
           FROM refs
           WHERE callee > 4 AND kind = 'field') UNION ALL (SELECT
             callee, refend AS time, -1 AS delta
           FROM refs
           WHERE callee > 4 AND kind = 'field')
        ) AS steps) AS integrated_steps
       GROUP BY callee
       HAVING MAX(sum_at_time) = 1)

  )
)
```



* and caching of subexpressions

Per Class Analysis

Per Class Analysis

heap unique

Classes with NO heap-unique instances

Per Class Analysis

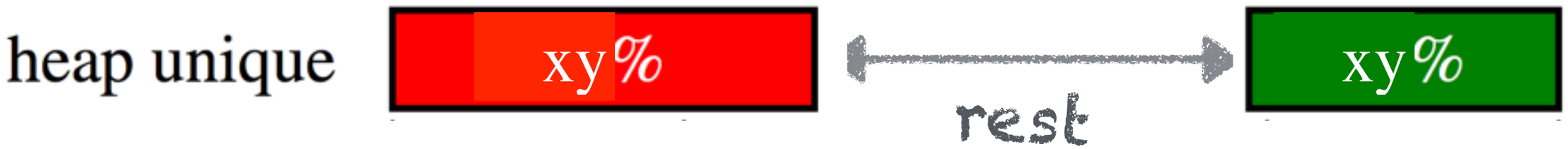
Classes with ONLY heap-unique instances

heap unique

Classes with NO heap-unique instances

Per Class Analysis

Classes with ONLY heap-unique instances



Classes with NO heap-unique instances

Per Class Analysis

heap unique

Per Class Analysis

heap unique

Hypothesis: could annotate class with "heap-shared" keyword

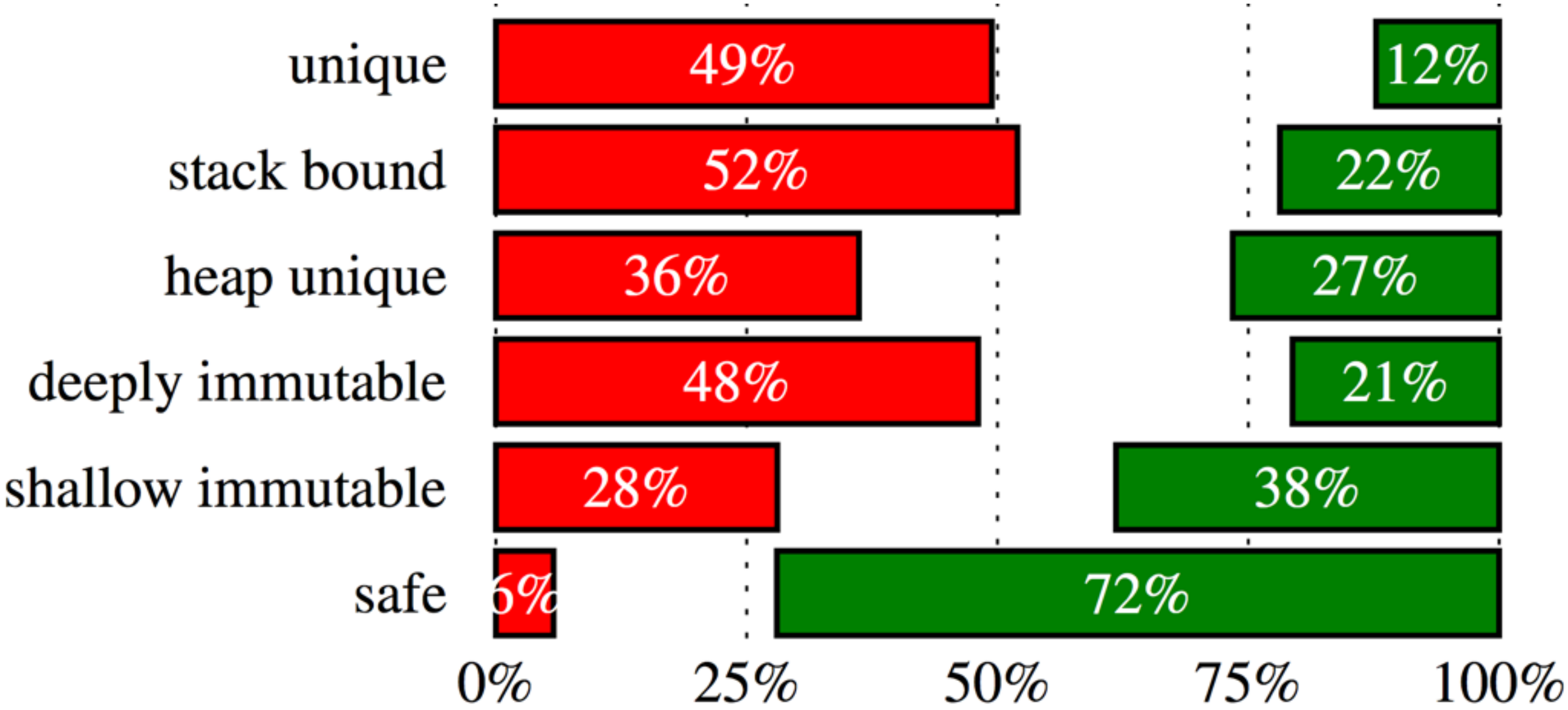
Per Class Analysis

Hypothesis: could annotate class with "heap-unique" keyword

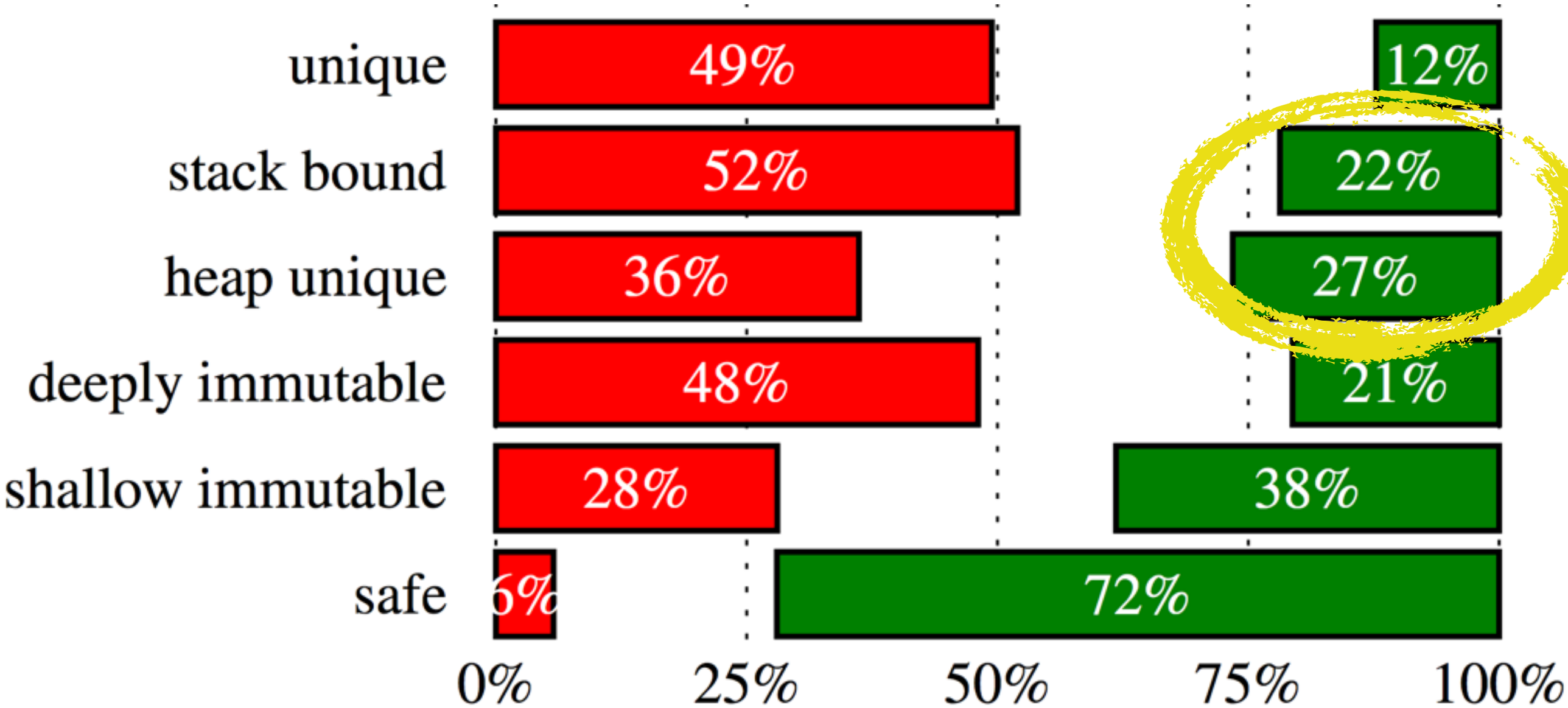
heap unique

Hypothesis: could annotate class with "heap-shared" keyword

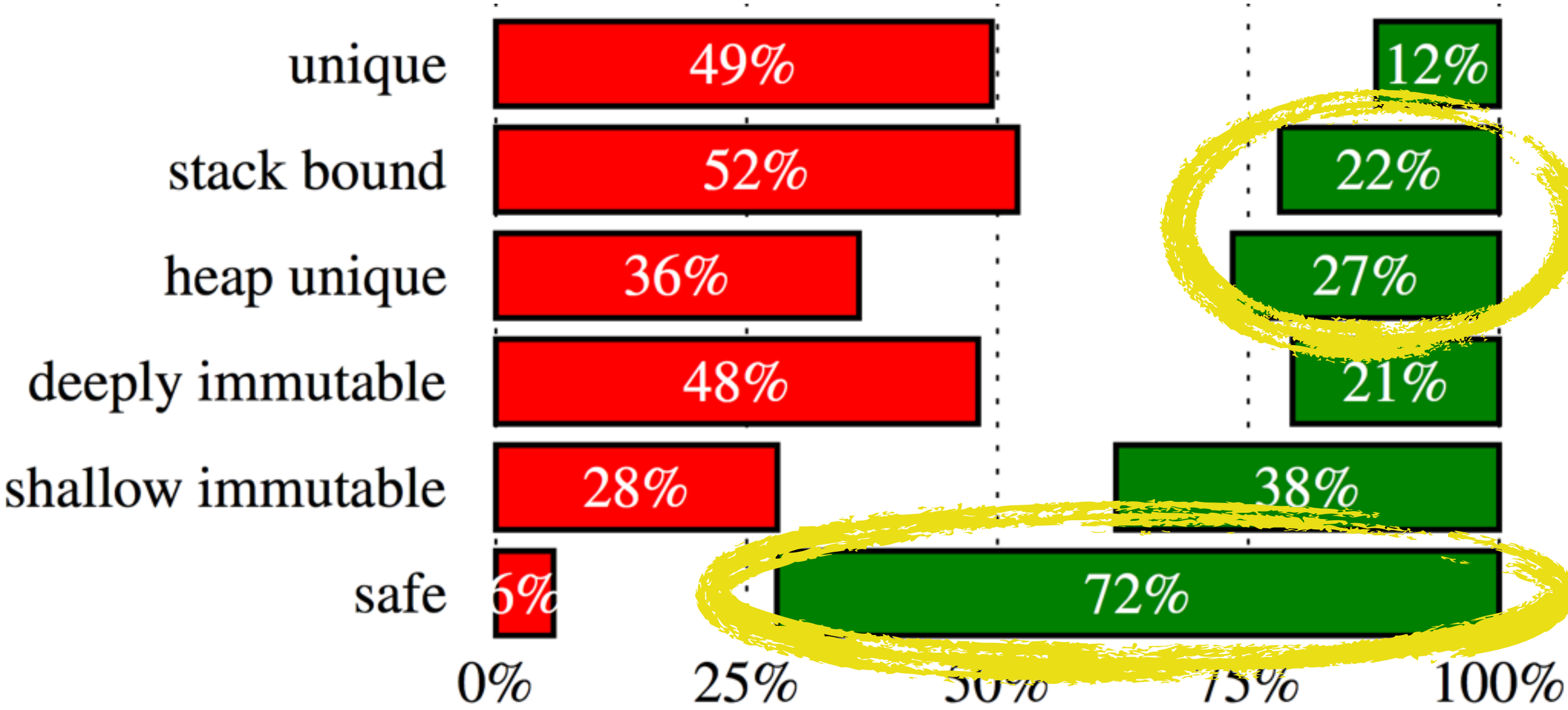
Per Class Analysis



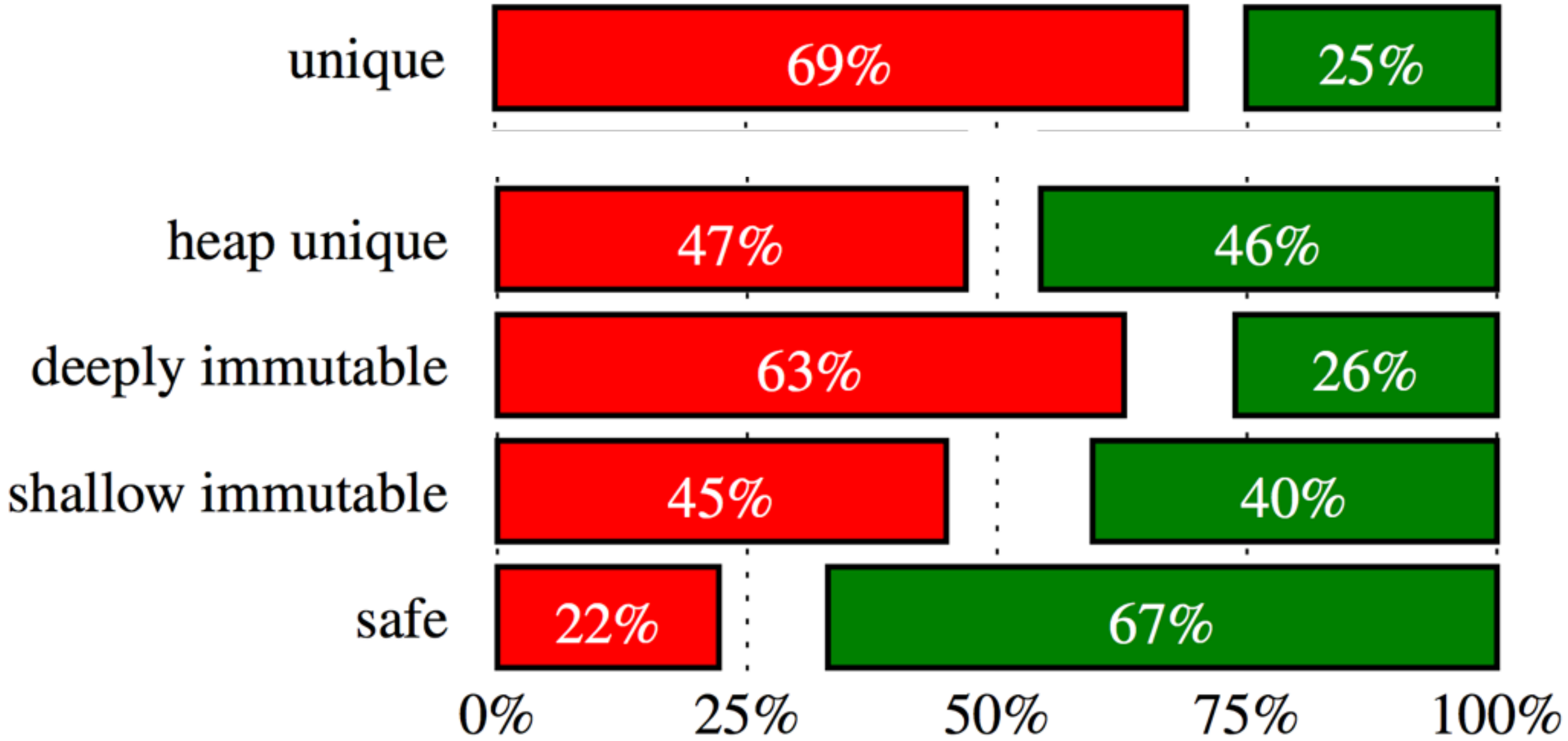
Per Class Analysis



Per Class Analysis



Per Field Analysis



Per Field Analysis