On Leveraging Tests to Infer Nullable Annotations

Jens Dietrich **David J. Pearce** Mahin Chandramohan

@WhileyDave

Nullable Annotations

```
class Rectangle {
  private @NonNull Point p1;
  private @NonNull Point p2;

  Rectangle(@NonNull Point p1, @NonNull Point p2) {...}

  boolean contains(@Nullable Point p) {...}
}
```

- `@NonNull` indicates variable **cannot** hold null
- `@Nullable` indicates variable **can** hold null
- We assume nonnull by default (e.g. `@NonNullbyDefault` in Eclipse)

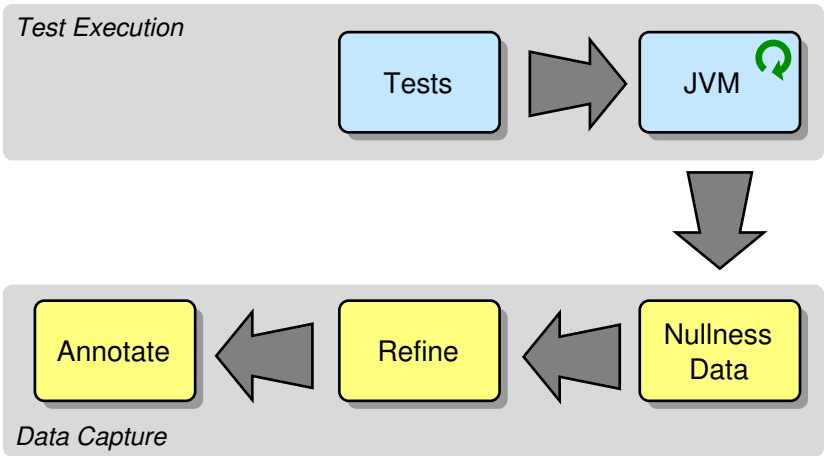
Problem Statement

> *"Agreed this idea, but it is a* **HUGE** *work if we want to add NotNull and Nullable to all public functions in commons-lang."*
>
> *– Jin Xu*[1]

[1] https://issues.apache.org/jira/browse/LANG-1598

Approach

Overview

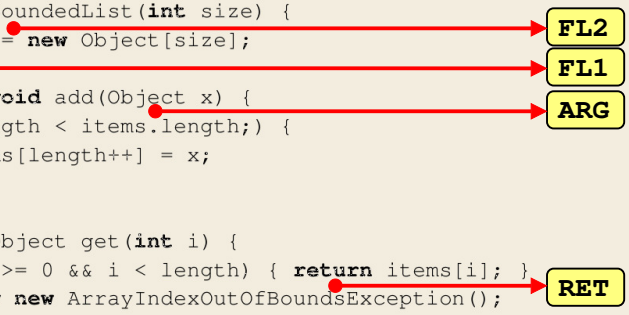

Test Execution
Tests
JVM
Nullness Data
Refine
Annotate
Data Capture

Dynamic Instrumentation

```
class BoundedList {
  private Object[] items;
  private int length;

  public BoundedList(int size) {
    items = new Object[size];
  }
  public void add(Object x) {
    if(length < items.length;) {
      items[length++] = x;
  } }

  public Object get(int i) {
     if(i >= 0 && i < length) { return items[i]; }
     throw new ArrayIndexOutOfBoundsException();
} }
```



Nullness Issues

```
{
"className":"$s.ConcurrentReferenceHashMap",
"methodName":"put",
"descriptor":"(Ljava/lang/Object;Ljava/lang/Object;Z)Ljava/lang/Object;",
"kind":"RETURN_VALUE",
"argsIndex":-1,
"stacktrace":[
  "$s.ConcurrentReferenceHashMap::put:282",
  "$s.ConcurrentReferenceHashMap::put:271",
  "$s.ConcurrentReferenceHashMapTests::shouldGetSize:331"
  ]
}
```

- **Trigger**: `shouldGetSize()`
- **Deduplication**: *duplicates reported as one issue*

Sanitisation

- **Scope**
  - Classes used only for testing may not be annotated
  - Eliminated using Maven project structure
- **Negative Tests**:
  - Exercise abnormal (but possible) behaviour
  - Identified using lightweight static analysis
- **Shaded Dependencies**:
  - Dependencies included directly as source
  - Annotations unlikely, as process is automated
- **Deprecation**:
  - Ignore issues for `@Deprecated` items

LSP Propagation

```
public class A {
   public @Nullable Object foo (Object arg) ;
}

public class B extends A {
   public @Nullable Object foo (@Nullable Object arg) ;
}

public class C extends B {
   public Object foo (@Nullable Object arg) ;
}
```

- **Limitation**: *cannot propagate across project boundaries!*

Evaluation

Research Questions

- (**RQ1**) How does observed nullability compare to existing annotations?
- (**RQ2**) Can sanitisation improve precision?
- (**RQ3**) Can propagation improve recall?
- (**RQ4**) Does repeated sanitisation / propagation reach fixpoint?

True Positives (TP)

Number of existing annotations that were inferred

False Positives (FP)

Number of non-existent annotations inferred

False Negatives (FN)

Number of existing annotations not inferred

Precision

$$\frac{TP}{(TP + FP)}$$

Recall

$$\frac{TP}{(TP + FN)}$$

Benchmarks

		main		test		
program	version	Java	Kotlin	Java	Kotlin	coverage
spring-beans	5.3.22	301	2	126	4	60%
spring-context	5.3.22	640	5	483	7	63%
spring-core	5.3.22	499	1	214	14	66%
spring-orm	5.3.22	72	0	32	0	39%
spring-oxm	5.3.22	31	0	19	0	58%
spring-web	5.3.22	653	1	268	5	18%
spring-webmvc	5.3.22	368	3	225	5	39%
guava	31.1	619	0	502	0	70%
error-prone	2.18.0	745	0	1,222	0	73%

Table: Java (and Kotlin) source files for `main` / `test` scope, and branch coverage.

(Partial) Results

Benchmark	Existing	Base			Sanitized		
		Seen	Recall	Precision	Seen	Recall	Precision
spring-beans	1,290	1,320	0.54	0.52	687	0.50	0.95
spring-context	1,435	5,945	0.49	0.12	718	0.47	0.94
spring-core	1,510	1,171	0.52	0.67	780	0.47	0.92
spring-orm	377	279	0.47	0.63	184	0.45	0.93
spring-oxm	84	64	0.54	0.70	49	0.54	0.92
spring-web	2,025	1,656	0.45	0.55	941	0.42	0.90
spring-webmvc	1,437	2,392	0.69	0.41	1,048	0.67	0.92
guava	3,993	4,923	0.48	0.39	2,464	0.48	0.77
error-prone	507	1,736	0.39	0.11	1,337	**0.39**	**0.15**

- **error-prone**: Adding missing `@Nullable` for methods returning `java.lang.Void` improves recall (0.72) and precision (0.79).

Industrial Application

- (**Spring, #29150**)[1] `SettableListenableFuture::get()` missing `@Nullable`.
- (**Spring, #29242**)[2] `CustomDateEditor::dateFormat()` missing `@Nullable`. **[Rejected — tests refined instead]**
- (**Guava, #6510**)[3] Add `@Nullable` (and `@NonNull`) and various new tests.
- (**ErrorProne, #3792**)[4] Add `@Nullable` when returning `java.lang.Void`. **[OPEN]**

[1] https://github.com/spring-projects/spring-framework/pull/29150
[2] https://github.com/spring-projects/spring-framework/pull/29242
[3] https://github.com/google/guava/issues/6510
[4] https://github.com/google/error-prone/issues/3792

@WhileyDave