A sneak-preview of ultimate **null-reference analysis** by the Eclipse Java compiler

---

Effortless prototyping using Object Teams

Stephan Herrmann, GK Software AG

Eclipse DemoCamps Indigo

June 21/22, 2011

Eclipse SDK 3.7 by the Number

- **Oloh says:**
  - Platform: 7.8m + 2.5m LOC, $ 130m
  - JDT: 1.1m + 0.3m LOC, $ 17m
- **> 4600 Bugs resolved for 3.7**
  - 15 blocker, 51 critical
  - 13 P1, 65 P2
- **181 Bugs involve an NPE** (~ 4%)
  - 1 blocker, 5 critical (~ 10%)
  - 2 P1, 9 P2 (~ 14%)

The Problem: we got used to NPE,
no longer see how **embarrassing** this is.

JDT: Existing Null Analysis

- What we can
  - detailed flow analysis
    - branches/conditionals, loops, try-catch, assert
    - distinguish potential/definite problems
  - follow variable assignments
  - use hints from existing null-checks
  - unboxing
  - signal missing & redundant checks
  - 491 distinct JUnit tests

[null]-Bugs Resolved for 3.7

133125 FIXED [compiler][null] need to report the null status of expressions and analyze them simultaneously

319201 FIXED [null] no warning when unboxing SingleNameReference causes NPE

320170 FIXED [compiler] [null] Whitebox issues in null analysis

198044 DUPL [compiler][null] Redundant null check gives false positive due to asserts

123399 WONT [compiler][null] missing null ref warning upon specific if/do while case

335093 FIXED [compiler][null] minimal hook for future null annotation support

333089 FIXED [compiler][null]AIOOBE while assigning variable a potentially null value in try/finally

338718 WORK [compiler][null]redundant null check not caught by warnings checker

338339 WONT [compiler][null] API for annotation based null analysis

338303 FIXED [compiler][null] Warning about Redundant assignment conflicts with definite assignment analysis

248040 INVA [compiler][null] Fake Potential Null Pointer Access triggered by check against null.

324178 FIXED [null] ConditionalExpression.nullStatus(..) doesn't take into account the analysis of condition itself

336428 FIXED [compiler][null] bogus warning "redundant null check" in condition of do {} while() loop

326950 FIXED [compiler][null]Do not optimize code generation based on static analysis (dead code)

339250 FIXED [null] Incorrect redundant null check warning on a String

342300 FIXED [null]Spurious "null pointer access" warning on unboxing

341499 FIXED [compiler][null] allocate extra bits in all methods of UnconditionalFlowInfo

332713 DUPL [compiler][null]Bogus "Null comparison always yields false"

Existing Null Analysis

- **What we can**
  - detailed flow analysis
    - branches/conditionals, loops, try-catch, assert
    - distinguish potential/definite problems
  - follow variable assignments
  - use hints from existing null-checks
  - unboxing
  - signal missing & redundant checks
  - 491 distinct JUnit tests
- **Limitations**
  - all incoming values are assumed as “unknown”
    - method arguments & method call results
  - no “common sense”
    - no correlation

Future Null Analysis

- Work in progres, not part of 3.7
  - Prototype is available
- Wiki
  - http://wiki.eclipse.org/JDT_Core/Null_Analysis

Two Perspectives

eclipse JDT

- **Type system**
  - Object ain't no type
    - Object_nonnull
      - cannot assign null
    - Object_or_null
      - cannot dereference
  - rules for “casting”, e.g.

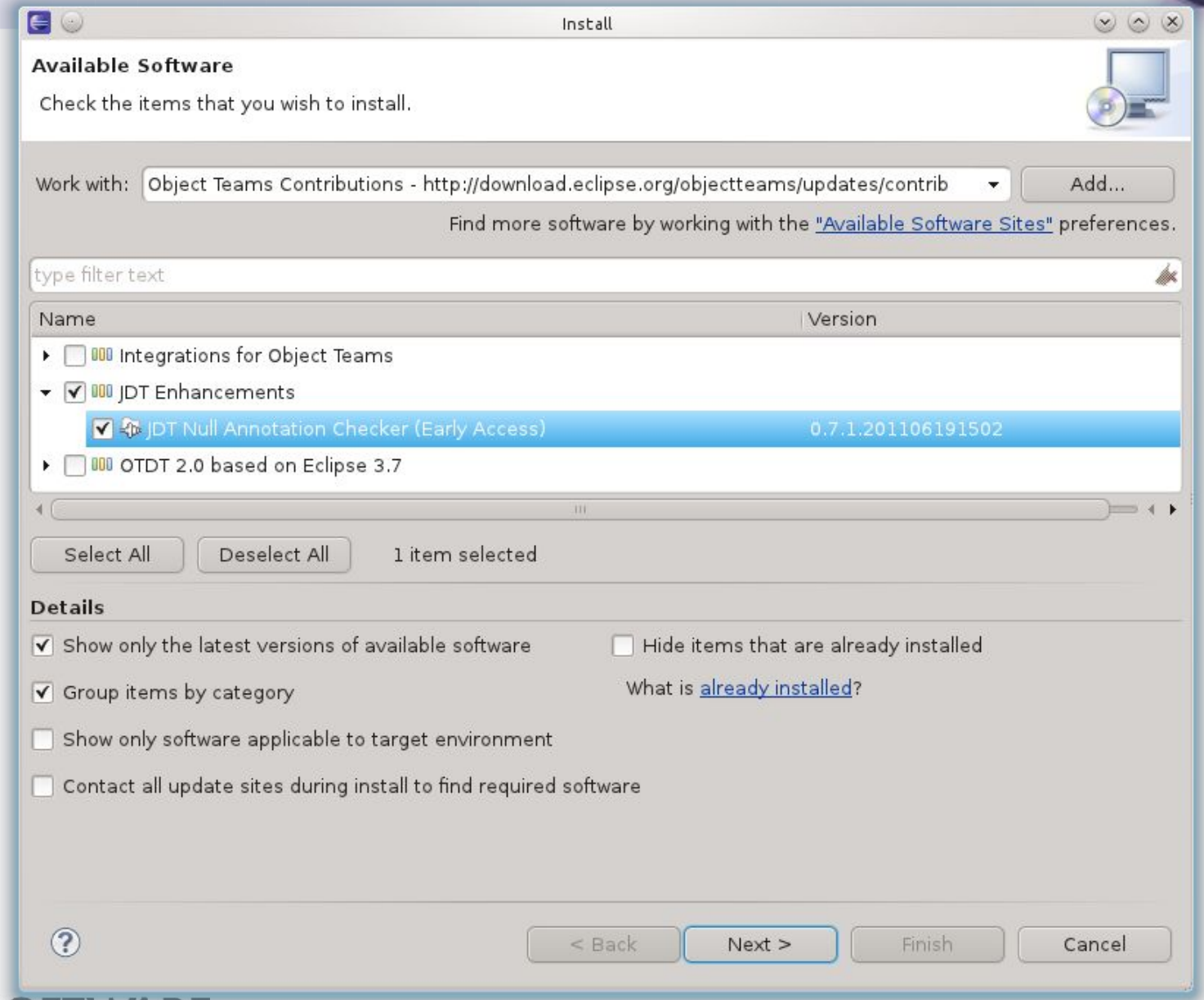
```
if (o != null) {
  Object_nonnull o2 = o;
  o2.m();
}
```

- **Contracts**
  - predicate nullable / nonnull
    - pre for parameters
    - post for return
    - inv for fields?
  - if method has contract check method call & impl. against contract
- **Locals**
  - use existing inference

One Syntax

```
@Nullable Object basicGetObject(@NonNull String key);
@NonNull Object safeGetObject(@Nullable String key);
```

Installing the Prototype



Work In Progress

- **Consolidate concepts**
  - decide on terminology (type system vs. contracts**?**)
  - evaluate migration paths for adopters
  - inherit annotations or just check compatibility**?**
- **UI**
  - add to preference page
  - more quickfixes & cleanups
- **Future**
  - fields
  - annotations for legacy libraries

Please try it! Feedback appreciated.

References

- **Wiki**
  - http://wiki.eclipse.org/JDT_Core/Null_Analysis
- **Bugzilla**
  - https://bugs.eclipse.org/186342 (master)
- **Update Site (Early Access)**
  - http://download.eclipse.org/objectteams/updates/contrib
- **Examplary Annotation Types**
  - http://download.eclipse.org/objectteams/contrib/org.eclipse.jdt.annotations.zip

Deeply Integrated Solution

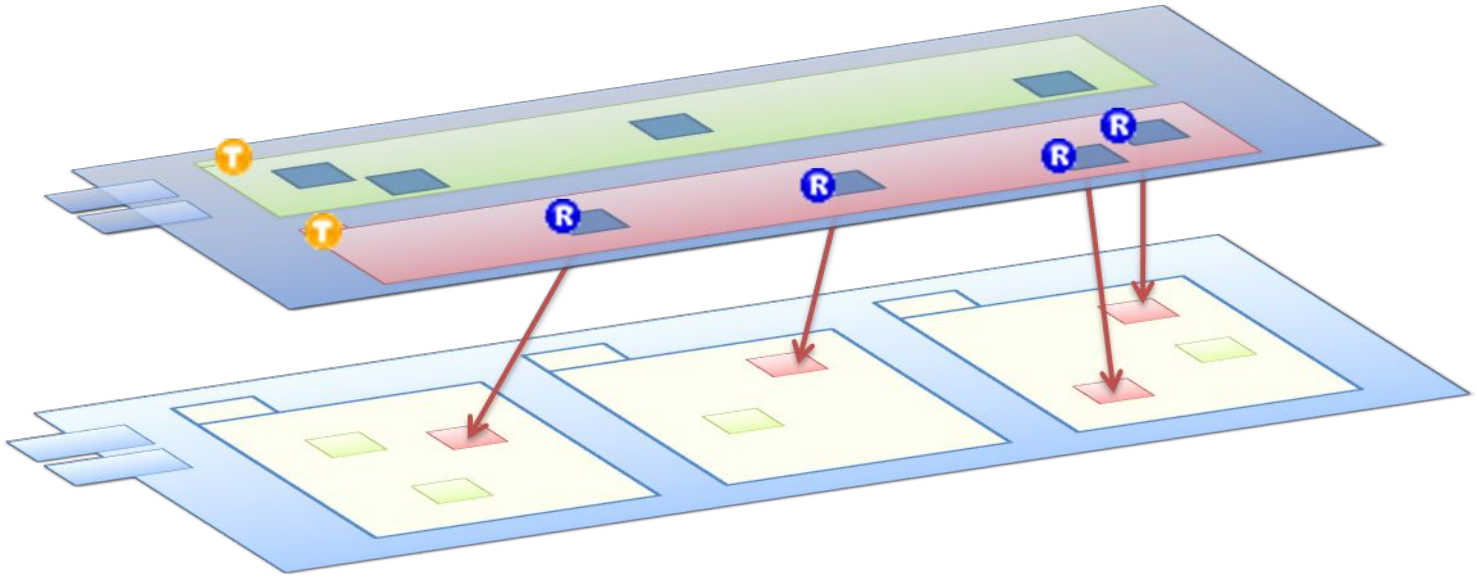
- **Want this to be intrinsic part of the JDT**
  - ready to use for everybody (no additional install)
  - performance (by hooking into existing flow analysis)
  - uniformly integrate into UI, too
    - preferences, quickfixes, refactoring
- **As of 3.7 it is not part of the JDT**
  - develop in a branch?
  - provide a patch feature?
- **Develop / deploy as an Add-on?**
  - facilitate develop / build / deploy
  - still leverage advantages of deep integration?
  - separate yet deeply integrated?

    => “extreme modularity”

Object Teams

- "Extreme Modularity"
  - better than
    - patch / branch: patch hunks? classes & methods!
    - inline: scattered impl? feature locally in one place!
    - regular add-on: copy&paste? NO!
- Robust
  - against some changes in the base
- Readability
  - explain implementation just by reading

Neue Dimensionen der Modularität



- **Extreme Modularity:**
  - optimale Struktur für das neue Feature
    - Team mit Rollen
    - Rollen kümmern sich nur um das neue Feature
  - Verbindung beider Welten mit
    - playedBy
    - callout & callin

Claims

With Object Teams

- *every* feature can be implemented as a module
- all interfaces are explicit and narrow
- maintenance is a breeze

June 22, 2011:
Object Teams Development Tooling 2.0.0
freshly graduated
1st time part of the Release Train