The long Good-Bye
to
NullException

Stephan Herrmann
The Road Behind

1965

- The billion dollar mistake
  - 1965 Tony Hoare introduced Null references in ALGOL W

2006 – 3.2

- Bug 110030 – Provide support for null reference analysis

2011 – 3.8

- Bug 186342 – Using annotations for null checking
  - EclipseCon Europe 2011: “Bye-bye NPE”

2013 – 4.3

- Bug 383368 – syntactic null analysis for field references

2014 – 4.4

- Bug 392099 – Apply null annotations on types for null analysis
  - EclipseCon Europe 2014: “A Deep Dive into the Void”

2015 – 4.5

- Bug 331651 – Support external null annotations for libraries

Continuous improvement of analysis for:
- loops, assert, generics, modules, interfacing with “legacy” code
- injection, well-known libraries

Stephan Herrmann @EclipseCon Europe 2022 – published under the EPL
String val;
...
val = null;
...
uc = val.toUpperCase();
Null Reference Analysis

As part of flow analysis

Only local analysis

String `val`;

...

`val = x`;

...

`if (val != null)`
  `uc = val.toUpperCase();`
`else`
  `lc = val.toLowerCase();`
Using Annotations for Null Checking

- Inter-procedural analysis
- Simple “contracts”

```java
String meth(@NonNull String val1, @Nullable String val2) {
    if (someFlag)
        return val1.toUpperCase();
    else
        return val2.toLowerCase();
}

s = meth("hello", null);

s = meth(null, "hello");
```
Flow Analysis for Fields?

- **Unexpected errors**
- **3 risks of shared data**
  - concurrency
  - aliasing
  - side effects
- **Compromise**
  - accept minimal risk
  - no sophistication
  - “syntactic analysis”

```java
class Test {
    @Nullable String f;

    String meth() {
        if (this.f != null) {
            // some code here
            return this.f.toUpperCase();
        }
        return "<don't know>";
    }
}
```
Null Annotations on Types

Since Java 8

JSR 308

@Target(TYPE_USE)

Contracts?

Extended type system

```java
String meth(@NonNull List<? @Nullable Person> val)
{
    return val.get(0).getName();
}
...
```

2014 – 4.4
External Null Annotations

@NonNull Map<@NonNull String,NonNull Person> val = x;

String name = val.get("Joe").getName();

...
External Null Annotations

“Legacy” libraries

Files: *.eea

Command “Annotate”
External Null Annotations

- "Legacy" libraries
- Files: *.eea
- Command "Annotate"

```java
Map<String, Person> val = x;
String name = val.get("Joe").getName();
```

@NonNull Map<String, Person> does not permit null keys

@NonNull @NonNull

File: *.eea

@NonNull

Command "Annotate"
External Null Annotations

@NonNull Map<@NonNull String, NonNull Person> val = x;

String name = val.get("Joe").getName();

...
The Road Behind

1965

- The billion dollar mistake
  - 1965 Tony Hoare introduced Null references in ALGOL W

2006 – 3.2

- Bug 110030 – Provide support for null reference analysis

2011 – 3.8

- Bug 186342 – Using annotations for null checking
  - EclipseCon Europe 2011: “Bye-bye NPE”

2013 – 4.3

- Bug 383368 – syntactic null analysis for field references

2014 – 4.4

- Bug 392099 – Apply null annotations on types for null analysis
  - EclipseCon Europe 2014: “A Deep Dive into the Void”

2015 – 4.5

- Bug 331651 – Support external null annotations for libraries

Stephan Herrmann @EclipseCon Europe 2022 – published under the EPL
Difficulties

Adding a core concept interacts with all that’s already there

Language design
- generics, wildcards
- type inference
- records
- patterns (e.g., instanceof)

Object initialization
@NonNull fields: when does the contract start?

“Legacy code”
- interact with_
- retrofit using .eea
Generics meet Legacy

Warn when legacy code can taint null-checked values

2020 – 4.15

Stephan Herrmann @EclipseCon Europe 2022 – published under the EPL
Advances concerning **External Annotations**
What “3rd Party” Code?

- Originally: overlays for jars
- Other code that cannot be annotated:
  - Generated code (if you don't own the code generator!)
- Solution
  - *Every* classpath entry can refer to external annotations
  - EEA can be superimposed even on sources
What “3rd Party” Code?

- Originally: overlays for jars
- Other code that cannot be annotated:
  - Generated code (if you don’t own the code generator)
- Solution:
  - Every classpath entry can refer to external annotations
  - EEA can be superimposed even on sources

2021 – 4.19
What “3rd Party” Code?

Originally: overlays for jars

Other code that cannot be annotated:
- Generated code (if you don't own the code generator)!

Solution:
- Every classpath entry can refer to external annotations
- EEA can be superimposed even on sources

2021 – 4.19
How do clients see my code?

- Internally, .eea have become part of generated sources
- Clients of those classes should see the same API!
- PDE supports new directive in MANIFEST.MF
  - Ensure .eea are included in deployed jar (build.properties)
  - `Eclipse-ExportExternalAnnotations`: true
- PDE will do the rest behind the scenes
  - Resolved elements of **Plug-in Dependencies** will have proper `annotationpath`

- Useful for
  - Plug-in projects with ...
  - ... generated source ...
  - ... superimposed with .eea
How to Manage .eea?

- Text files
  - Initially expected inside each project using a legacy library

- Should each project maintain its own set of .eea?
  - Shareable as jar files / artifacts

- Brute force:
  - Search all classpath locations for .eea

- Bad impact on IDE performance

see also: lastnpe.org
addresses such issues by extending m2e
The challenge for EEA in the IDE

- JDT should precisely know where to find .eea
  - But now .eea are artifacts needing dependency management
- Dependency management is handled by your build system
- JDT doesn't know any build system
  - But JDT knows about classpath containers
    - Plug-in Dependencies
    - Maven Dependencies
    - ...
- Solution
  - Specify annotation location relative to a classpath container:
    - annotationpath=org.eclipse.pde.core.requiredPlugins/org.example.annotations
  - If annotation artifact is in your dependencies* then JDT will find it for eea lookup
IDE vs. CI Builds

IDE “knows” about .eea
- Annotation path is configured via .classpath
  - .classpath may depend on Eclipse-specifics (like classpath containers)
- Read .eea:
  - compiler
  - hover
- Write .eea:
  - Ctrl+1 Annotate

Build tools don't know about .classpath
- Add .eea artifacts to your dependencies
- Catch all: -annotationpath CLASSPATH

<plugin>
  <groupId>org.eclipse.tycho</groupId>
  <artifactId>tycho-compiler-plugin</artifactId>
  <configuration>
    <useProjectSettings>true</useProjectSettings>
    <compilerArgs>
      <compilerArg>-annotationpath</compilerArg>
      <compilerArg>CLASSPATH</compilerArg>
    </compilerArgs>
  </configuration>
</plugin>

Stephan Herrmann @EclipseCon Europe 2022 – published under the EPL
Summary

Did I promise too much in 2011?
  ▶ Yes

Is treatment of null a reason to abandon Java?
  ▶ No

Is it possible to create provably NPE-free code?
  ▶ Yes, but only in green field, clean room development.

Which TYPE_USE annotations?
  ▶ Not “JSR-305”!
  ▶ org.checkerframework.checkers.nullness
  ▶ org.eclipse.jdt.annotation_2.2.x
  ▶ org.jspecify

Stephan Herrmann @EclipseCon Europe 2022 – published under the EPL