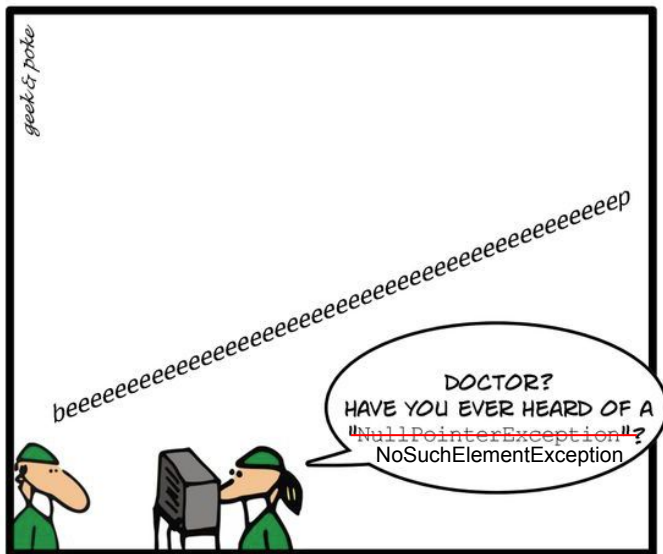


Java 8 Optional Checker Framework



RECENTLY IN THE OPERATING ROOM

Justin Kotalik (jkotalik)
Jasper Hugunin (jasperh)
Han Zhang (gzezhang)

Motivation

- People are inclined to use the Optional Class to solve NullPointerExceptions
- In general a bad idea:
 - If you modify to .get() on optional objects...
 - Transfers the error to a NoSuchElementException :(
 - Also adds an extra layer of uncertainty with introducing a new class
- Both are masking lacking null checks



Problem to solve: finding improper uses of .get() with Optional Objects.

Approach

- Use the Checker Framework:
- Adapt Nullness Checker
- Null \leftrightarrow Not present; dereference \leftrightarrow Optional.get()

Subtyping Checker	Nullness Checker	Mutation Checker	Tainting Checker	...	Your Checker		
Base Checker (enforces subtyping rules)						Type inference	Other tools
Checker Framework (enables creation of pluggable type-checkers)						Annotation File Utilities (.java ↔ .class files)	
Type Annotations syntax and classfile format ("JSR 308") (no built-in semantics)							



dereference of possibly-null reference y
y.charAt(2);

Main.java

/TestProject/src

line 6

Checker Framework Problem

Main Challenge

- We need to understand the Nullness Checker
- But, the Nullness Checker is complicated
- Luckily, one of the main developers, Michael Ernst, is our teacher
- We can ask him for help if we get stuck understanding something
- The Checker Framework has lots of simpler examples as well