

# Marre des null en java, Découvrez nullaway

**Alexandre Navarro**



BNP PARIBAS





# Mon exception préférée : Null Pointer Exception (NPE)



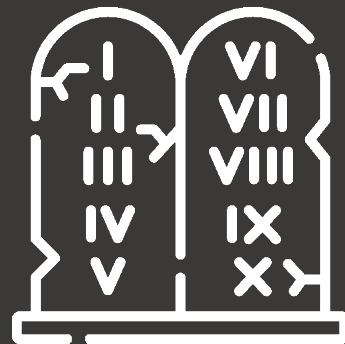
I call it my billion-dollar mistake. It  
was the invention of the null  
reference in 1965.

— *Tony Hoare* —

# Comment éviter les NPE ?

Tu appliqueras quelques commandements

- des bonnes pratiques
- un vérificateur à la compilation



# I. Tu utiliseras du pattern Optional.ofNullable

Avec un objet null

```
String driverLicenseId = Optional.ofNullable(person) Optional<Person>
    .map(Person::getDriverLicense) Optional<DriverLicense>
    .map(DriverLicense::getId) Optional<String>
    .orElse( other: "NoDriverLicenseId");

String driverOrParentLicenseId = Optional.ofNullable(person) Optional<Person>
    .map(Person::getDriverLicense) Optional<DriverLicense>
    .map(DriverLicense::getId) Optional<String>
    .orElseGet(() -> Optional.ofNullable(parent) Optional<Person>
        .map(Person::getDriverLicense) Optional<DriverLicense>
        .map(DriverLicense::getId) Optional<String>
        .orElse( other: "NoDriverLicenseId"));
```

Avec une liste null

```
List<String> driverLicenseIdList = Optional.ofNullable(personList) Optional<List<...>>
    .orElse(Collections.emptyList()) List<Person>
    .stream() Stream<Person>
    .map( Person aPerson -> Optional.ofNullable(aPerson) Optional<Person>
        .map(Person::getDriverLicense) Optional<DriverLicense>
        .map(DriverLicense::getId) Optional<String>
        .orElse( other: null)) Stream<String>
    .toList();
```



## II. Tu utiliseras des instances d'objets empty / élément neutre

```
// Use empty container or neutral element when it is possible  
List<Object> emptyList = Collections.emptyList();  
Map<Object, Object> emptyMap = Collections.emptyMap();  
Optional<Object> emptyOptional = Optional.empty();  
Integer validIntegerOrZero = Try.of(() -> Integer.parseInt(s: "12"))  
    .getOrElse( other: 0);
```

# III. Tu initialiseras entièrement tes objets, tu utiliseras des objets immuables

- Initialisation entière à la création de l'objet
- Création d'objet immuable en java via “malheureusement” des StagedBuilder Jilt

```
// Use static of method of create an object if less than 2 arguments
Price oneEuro = Price.of(BigDecimal.ONE, unit: "EUR");

// Or Use a StagedBuilder to always return a valid Immutable Person
Person johnDoe = Person.builder() FirstName
    .firstName("John") LastName
    .lastName("Doe") Optionals
    .build();
```

skinny85/jilt

Java annotation processor library for auto-generating Builder (including Staged Builder) pattern classes

# IV. Tu valideras tes objets au plus tôt




















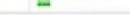






- Via `Objects.requireNonNull`



# V. Tu testeras unitairement ton code

Si couverture de test important > 85-90 voire 100 %

Alors moins de NPE statistiquement

JaCoCo												
Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
 <a href="#">org.jacoco.examples</a>		58%		64%	24	53	97	193	19	38	6	12
 <a href="#">org.jacoco.core</a>		97%		92%	137	1,507	125	3,555	19	740	2	147
 <a href="#">org.jacoco.agent.rt</a>		75%		83%	32	130	75	344	21	80	7	22
 <a href="#">jacoco-maven-plugin</a>		90%		82%	35	194	49	466	8	117	0	23
 <a href="#">org.jacoco.cli</a>		97%		100%	4	109	10	275	4	74	0	20
 <a href="#">org.jacoco.report</a>		99%		99%	4	572	2	1,345	1	371	0	64
 <a href="#">org.jacoco.ant</a>		98%		99%	4	163	8	429	3	111	0	19
 <a href="#">org.jacoco.agent</a>		86%		75%	2	10	3	27	0	6	0	1
Total	1,429 of 28,544	94%	177 of 2,338	92%	242	2,738	369	6,634	75	1,537	15	308





# Comment sont évités les NPE dans des langages récents?



- Pas de null, pas d'exception, Option/Result un objet empty, vérification à la compilation



- Type : String (not null), String? (nullable), String! (unspecified)



- Rien par défaut, vérification à la compilation possible comme kotlin?

# VI. Tu utiliseras un vérificateur de potentiels NPE à la compilation

uber/**NullAway**



CHECKER  
framework

SpotBugs



# NullAway

- Annotation Processor basé `google/error-prone`

- Annoter @Nullable `JSpecify`

@NonNull / @Nullable / @NullUnmarked

- Paramètres des méthodes / constructeurs

- Objet de retour de méthodes / champs



# NullAway Configuration

- XepOpt:NullAway:AnnotatedPackages=...
- XepOpt:NullAway:OnlyNullMarked=true
- XepOpt:NullAway:JSpecifyMode
- XepOpt:NullAway:ExcludedClassAnnotations=...
- XepOpt:NullAway:ExcludedFieldAnnotations=...



# NullAway : Configuration Maven

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-compiler-plugin</artifactId>
  <configuration>
    <compilerArgs>
      <arg>-XDcompilePolicy=simple</arg>
      <arg>--should-stop=ifError=FLOW</arg>
      <arg>-Xplugin:ErrorProne
        <!-- NullAway Configuration -->
        -Xep:NullAway:ERROR
        -XepOpt:NullAway:AnnotatedPackages=com.github.alexandrenavarro.marredesnullenjavadecouvrenullaway
        -XepOpt:NullAway:ExcludedClassAnnotations=org.jilt.JiltGenerated
      </arg>
    </compilerArgs>
    <annotationProcessorPaths>
      <path>
        <groupId>com.google.errorprone</groupId>
        <artifactId>error_prone_core</artifactId>
      </path>
      <path>
        <groupId>com.uber.nullaway</groupId>
        <artifactId>nullaway</artifactId>
      </path>
      <path>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
      </path>
      <path>
        <groupId>cc.jilt</groupId>
        <artifactId>jilt</artifactId>
      </path>
    </annotationProcessorPaths>
  </configuration>
</plugin>
```

# Démo







# Retour d'expérience

## Avantages

- Vérification à la compilation NPE
- Activable par package, très configurable
- Peu de faux positifs
- Mise en place facile sur un nouveau projet

## Inconvénients

- Long (pas dur) à migrer sur du legacy
- Builder : Il faut utiliser StagedBuilder jilt
- Rajoute un peu de temps de compilation
- Vérification seulement du code annoté (dépendances incluses), peu de check jdk
- Pas de langage, @Nullable dans jspecify pas jdk

# Et si le futur était plus radieux?



Organization	Projects
EISOP Team	EISOP
Google	Android, Error Prone, Guava
JetBrains	Kotlin, IntelliJ IDEA
Meta	Infer
Microsoft	Azure SDK for Java
Oracle	OpenJDK
PMD Team	PMD
Sonar	SonarQube, SonarCloud, SonarLint
Square	(various)
Uber	NullAway
Broadcom	Spring

The slogan for Valhalla is:

“Codes like a class, works like an int”

**JEP draft: Null-Restricted and Nullable Types (Preview)**

```
String! notNullString;    // not null
String? nullableString;   // nullable
String unspecifiedString; // unspecified
```



The background of the slide is a dark grey gradient. On the left side, there is a vertical strip showing a lush green jungle scene with a tall, cylindrical stone temple tower. Overlaid on this is a large, faint, grey Mayan stone mask with intricate carvings.

# Conclusion

- Utilisez les bonnes pratiques venant du monde fonctionnel
- Appliquez des fonctions via `Optional.ofNullable`
- Utilisez un objet empty/neutre
- Initialisez complètement / validez vos objets immuables



# Call for Action

- Essayez `uber/NullAway` comme vérificateur de NPE
- Sur les nouveaux projets
- Sur les projets legacy
- en commençant par résoudre les violations `*Nullable*`
- en activant sur 1 ou n packages puis en élargissant



# Questions

Votez



Liens



<https://github.com/alexandrenavarro/marre-des-null-en-java-decouvrez-nullaway-devovx-france-2025>



alexandrenavarro



alexandre-navarro-tech



alexandrejnavarro.bsky.social



alex\_j\_navarro