Consistency in OSS Libraries: Google's Approach

Open Source Summit Japan 2019

Garrett Jones and Tomohiro Suzuki

About us

Garrett Jones
Staff Software Engineer
Google Seattle

Tomohiro Suzuki
Software Engineer
Google New York

Google

Agenda

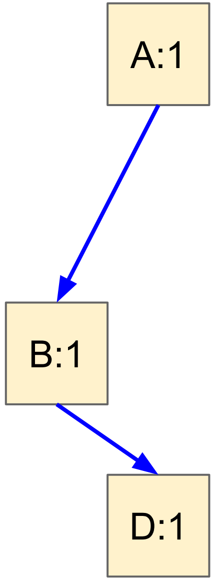
- Intro to diamond dependency conflicts
- Google's Java Library Best Practices
- Linkage Checker
- Q&A

Story

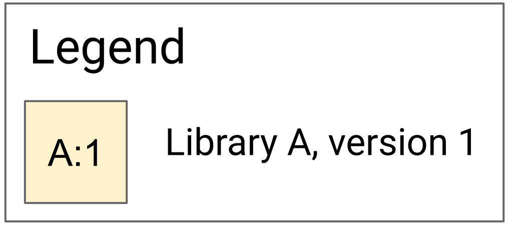
- Google monorepo vs OSS independent libraries
- Megathread prompted by user complaints
- No consensus
- Proposals, summits, proposals, hackathons
- Wrote best practices, created tools

Diamond Dependency Conflicts

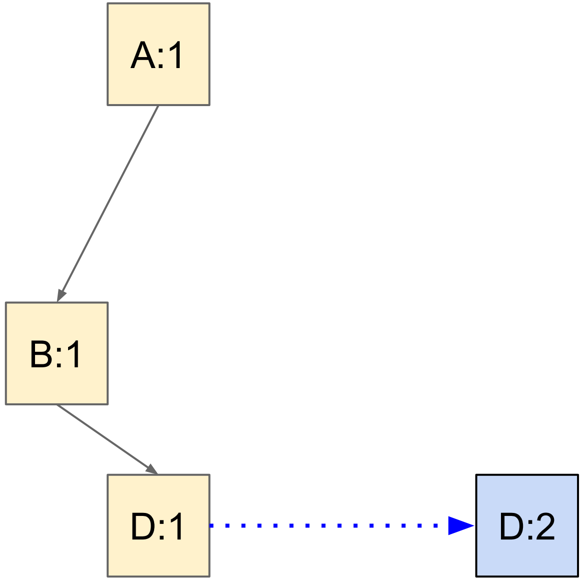
Diamond dependency conflicts: A visual representation



Linear dependency graph: everything is happy

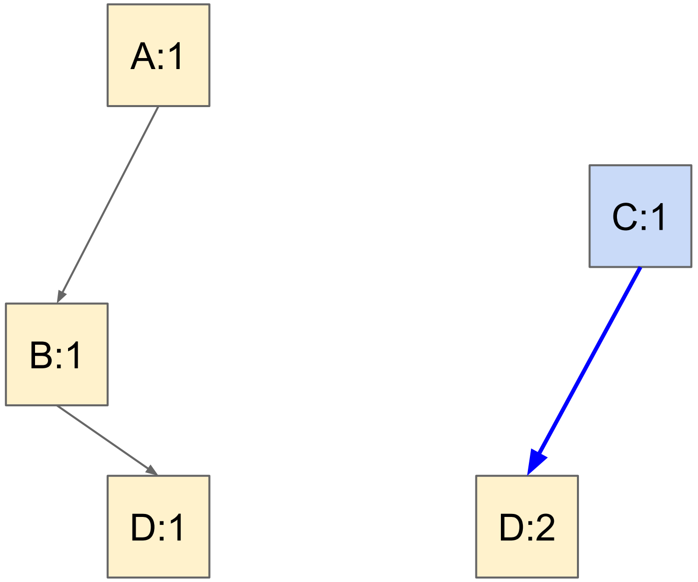


Diamond dependency conflicts: A visual representation



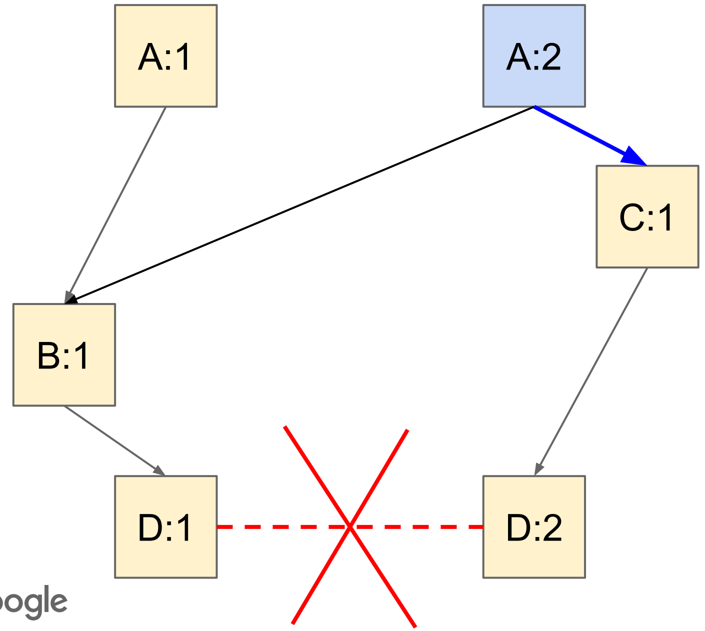
D introduces a new major version (D:2)

Diamond dependency conflicts: A visual representation



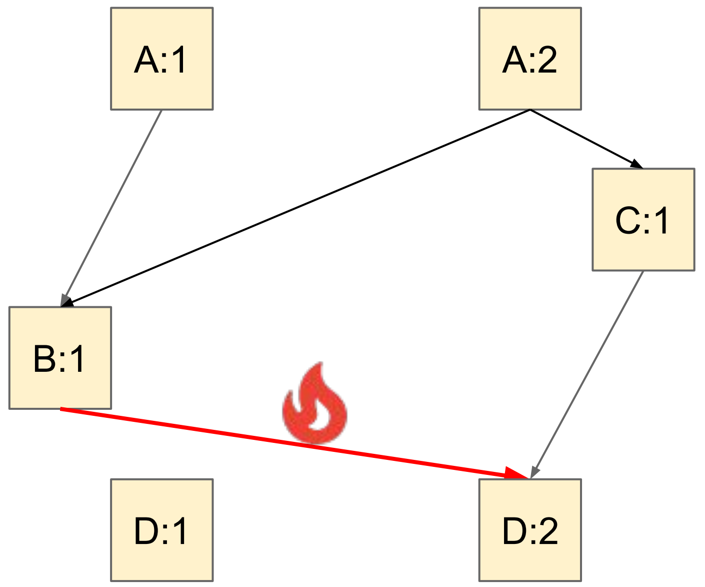
Another package, C, declares a dependency on D:2

Diamond dependency conflicts: A visual representation



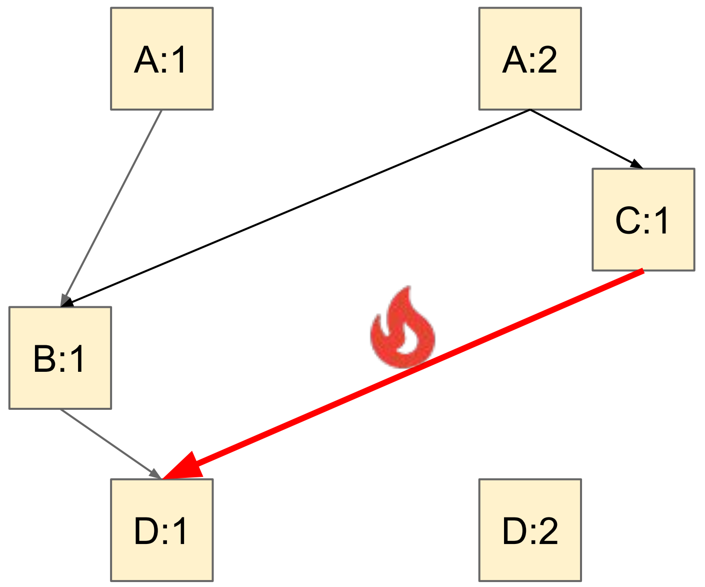
A new version of A attempts to add C as a dependency. Diamond dependency conflict! Only one of D:1 or D:2 can be chosen.

Diamond dependency conflicts: A visual representation



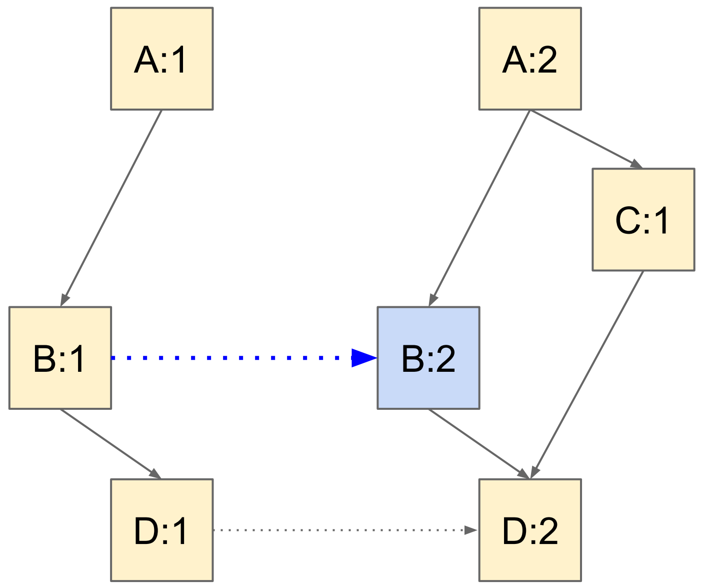
When D:2 is selected, this breaks B

Diamond dependency conflicts: A visual representation



When D:1 is selected, this breaks C

Diamond dependency conflicts: A visual representation



The only solution: A has to force B to make a new version that depends on D:2

Google's approach

1) Fix burning conflicts
2) **Establish best practices for library developers***
3) **Create tools***
4) Create BOMs (bill of materials) for library users

Java Library Best Practices

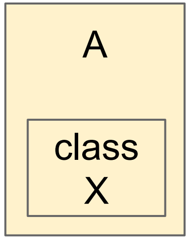
(for library developers)

Google's Java library best practices (JLBP)

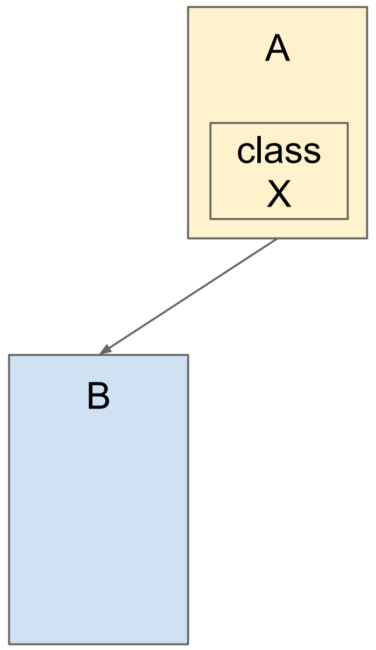
- 18 best practices, published at github.com/cloud-opensource-java/library-best-practices
- I will cover only 5 here

JLBP-1: Minimize dependencies
JLBP-2: Minimize API surface
JLBP-3: Use Semantic versioning
JLBP-4: Avoid dependencies on unstable libraries and features
JLBP-5: Avoid dependencies that overlap classes with other dependencies
JLBP-6: Rename artifacts and packages together
JLBP-7: Make breaking transitions easy
JLBP-8: Advance widely used functionality to a stable version
JLBP-9: Support the minimum Java version of your consumers
JLBP-10: Maintain API stability as long as needed for consumers
JLBP-11: Stay up to date with compatible dependencies
JLBP-12: Make level of support and API stability clear
JLBP-13: Quickly remove references to deprecated features in dependencies
JLBP-14: Do not use version ranges
JLBP-15: Produce a BOM for multi-module projects
JLBP-16: Ensure upper version alignment of dependencies for consumers
JLBP-17: Coordinate Rollout of Breaking Changes
JLBP-18: Only shade dependencies as a last resort

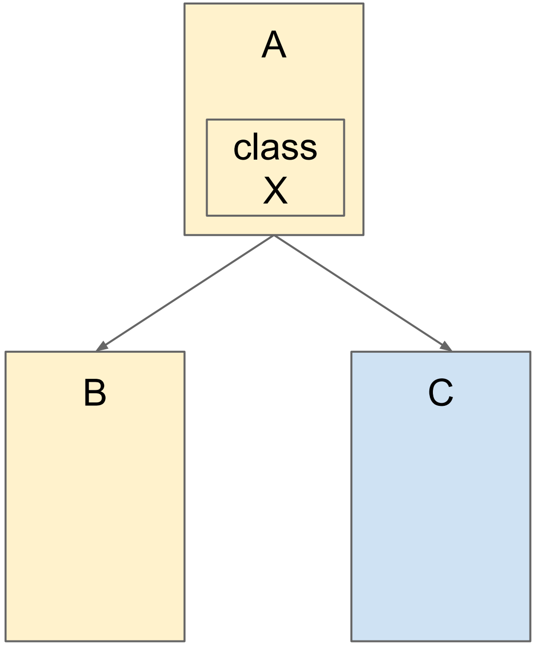
Problem 1: "Overlapping" classes



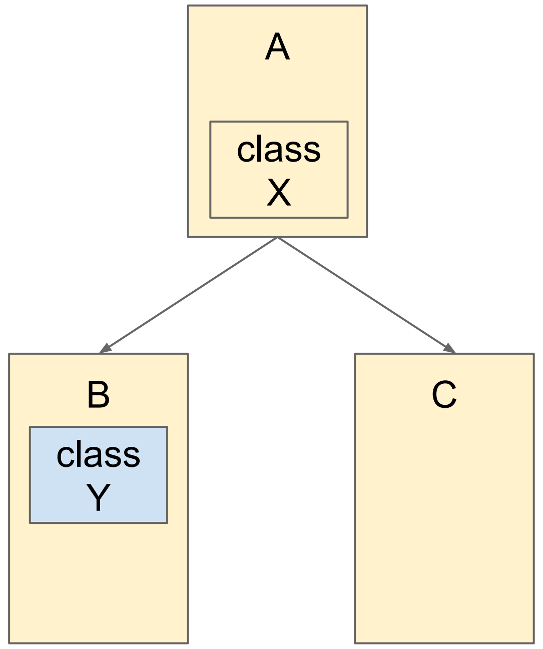
Problem 1: "Overlapping" classes



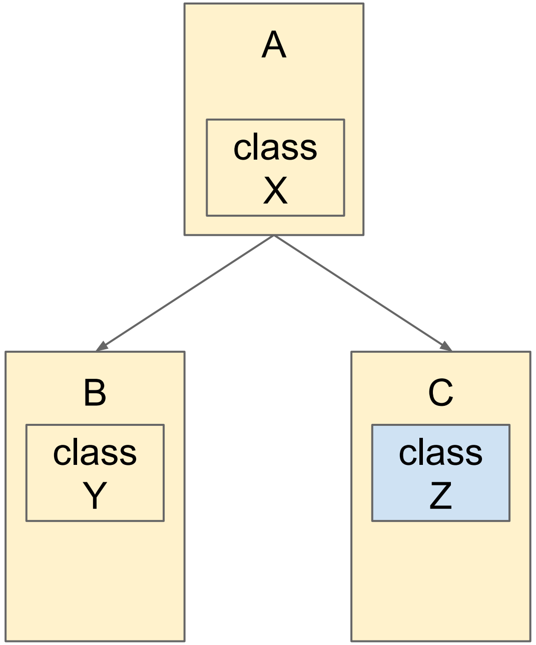
Problem 1: "Overlapping" classes



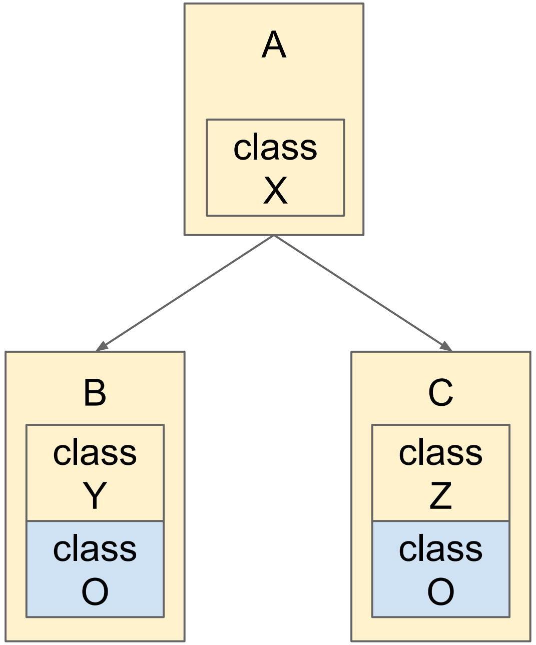
Problem 1: "Overlapping" classes



Problem 1: "Overlapping" classes

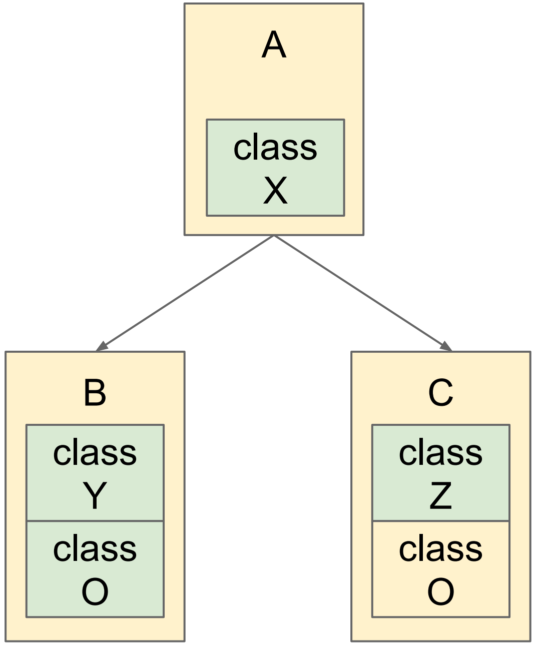


Problem 1: "Overlapping" classes



Google

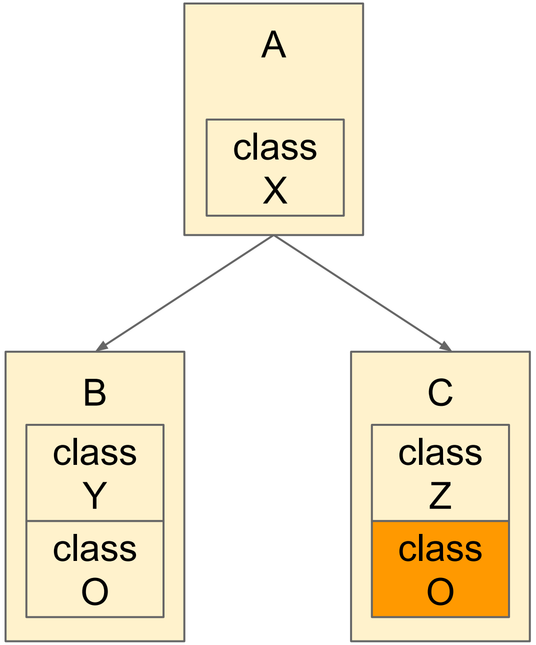
Problem 1: "Overlapping" classes



Classes loaded:

- class X [from library A]
- class Y [from library B]
- class O [from library B]
- class Z [from library C]

Problem 1: "Overlapping" classes



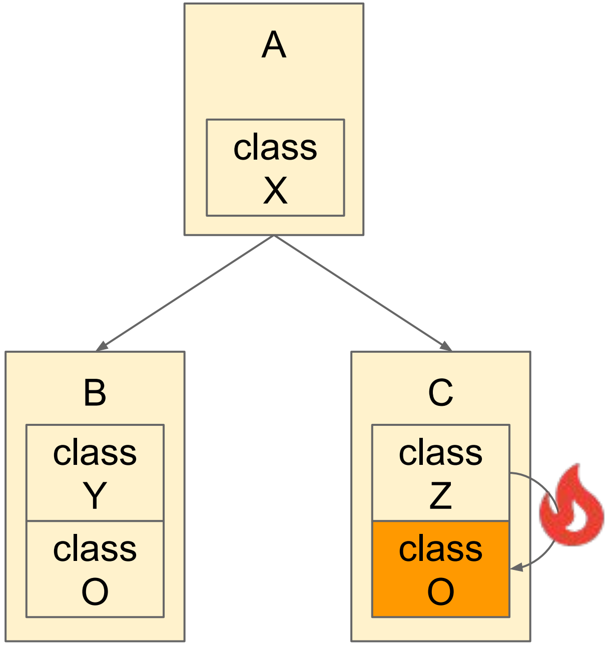
Classes loaded:

- class X [from library A]
- class Y [from library B]
- class O [from library B]
- class Z [from library C]

Classes not loaded:

- class O [from library C]

Problem 1: "Overlapping" classes



Classes loaded:

- class X [from library A]
- class Y [from library B]
- class O [from library B]
- class Z [from library C]

Classes not loaded:

- class O [from library C]

What can break?

- calls from Z to O (for things not in the version of O included in B)

What needs to be true?

Every fully-qualified class name must be present in only one library in the dependency tree

JLBP-5: Avoid dependencies that overlap classes with other dependencies

JLBP-6: Rename artifacts and packages together

Example violation of JLBP-5 & JLBP-6

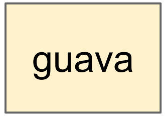
`javax.servlet:`**`javax.servlet-api`**`:3.1.0`
& `javax.servlet:`**`servlet-api`**`:2.5`

both contain classes with the same names under javax.servlet.

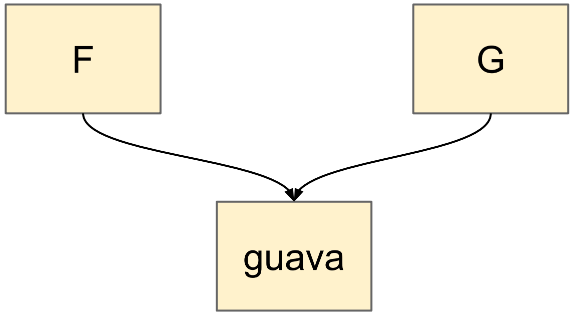
JLBP-6 corollaries

- Don't combine artifacts together while keeping the same fully-qualified names
- Don't split artifacts apart while keeping the same fully-qualified names

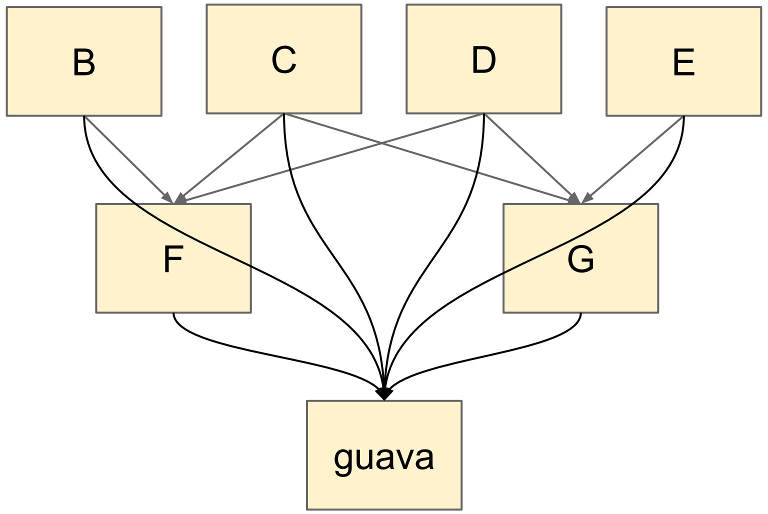
Problem 2: Wide scale breakage from changes



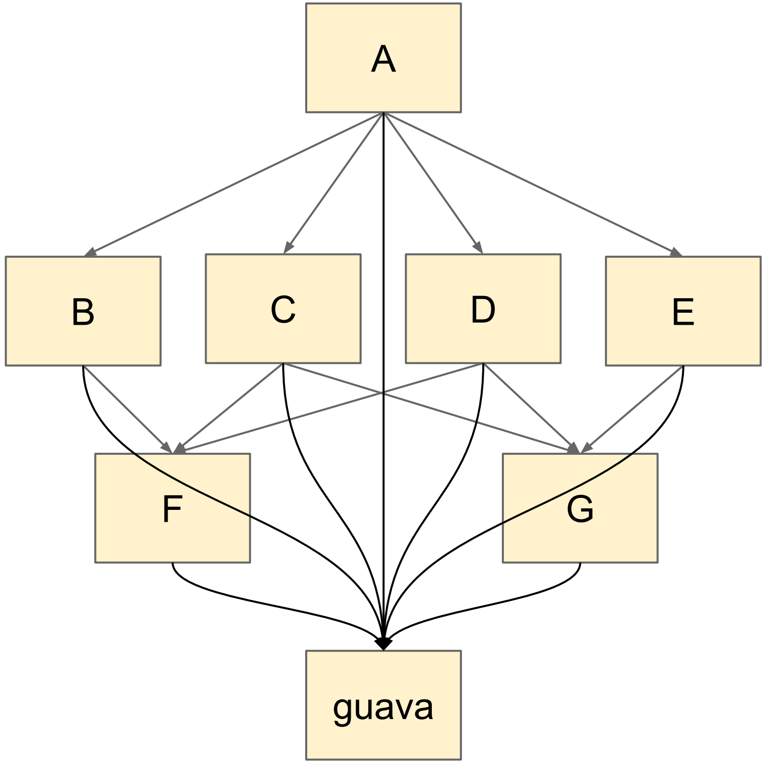
Problem 2: Wide scale breakage from changes


$$\begin{tikzcd}
F \arrow[dr] & & G \arrow[dl] \\
& guava &
\end{tikzcd}$$

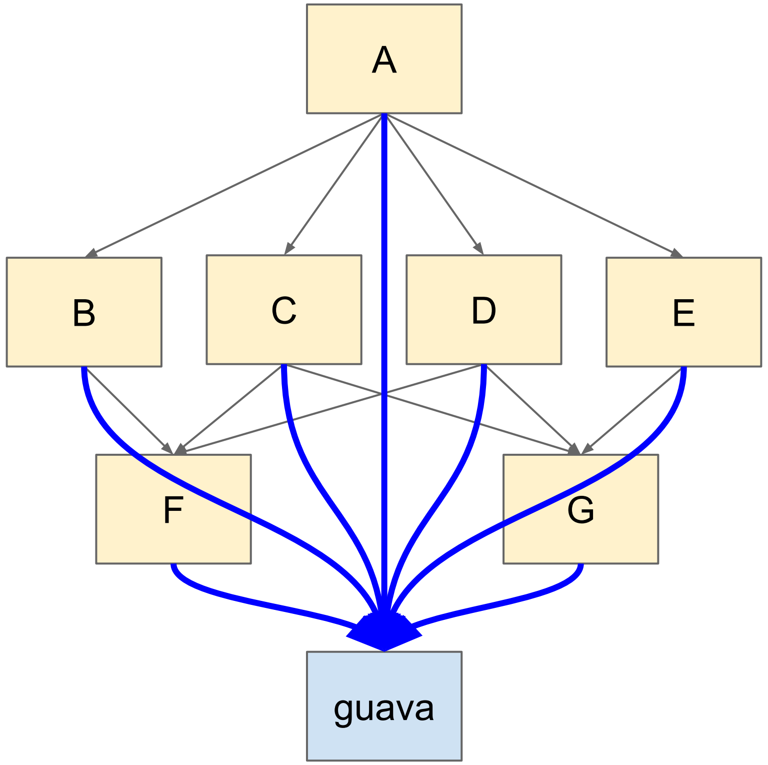
Problem 2: Wide scale breakage from changes



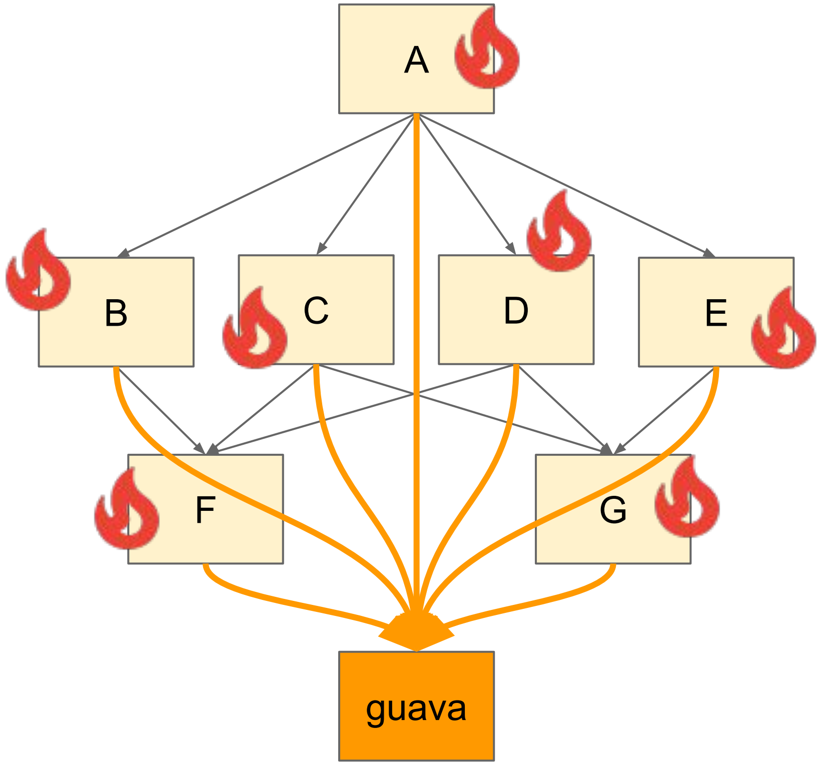
Problem 2: Wide scale breakage from changes



Problem 2: Wide scale breakage from changes



Problem 2: Wide scale breakage from changes



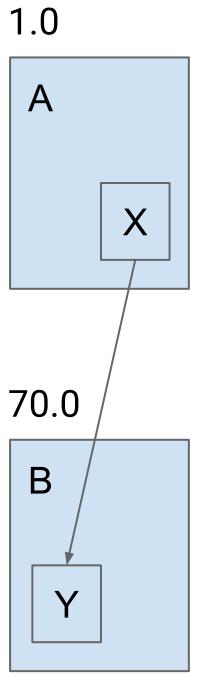
JLBP-8: Advance widely used functionality to a stable version

(also follow JLBP-3: Use Semantic Versioning -
"stable version" = 1.0.0+)

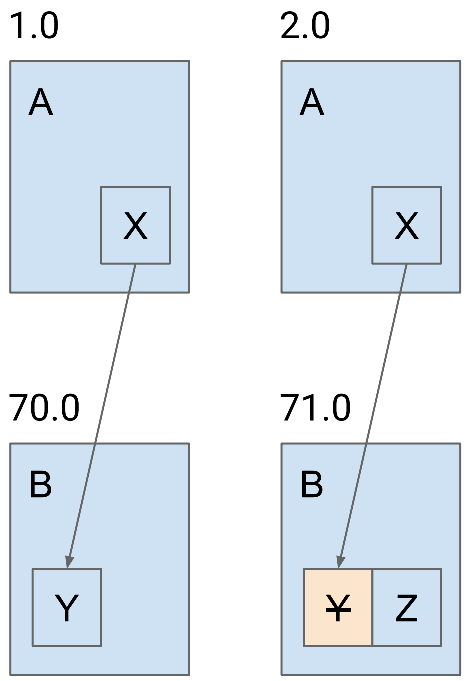
Example violation of JLBP-8

`com.google.auth:google-auth-library-java`: still uses a 0.x version

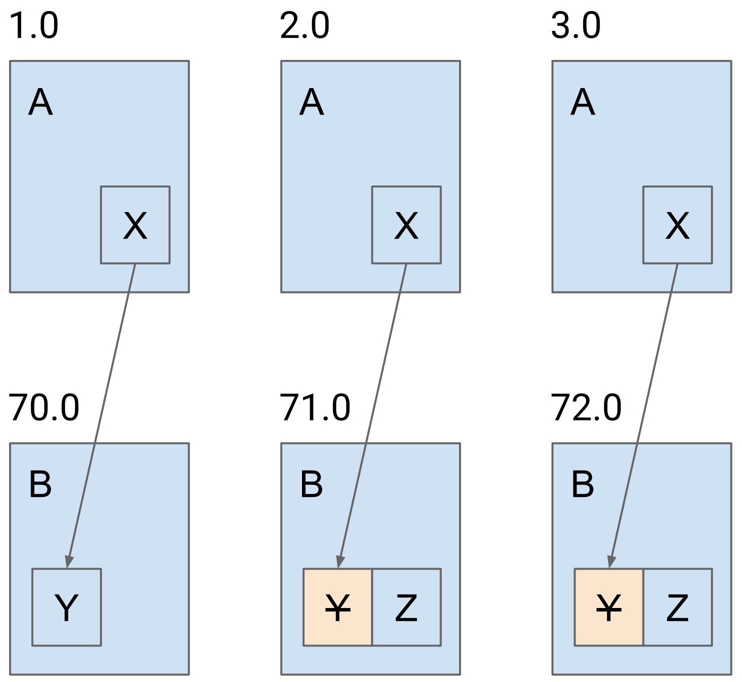
Problem 3: finding compatible dependencies



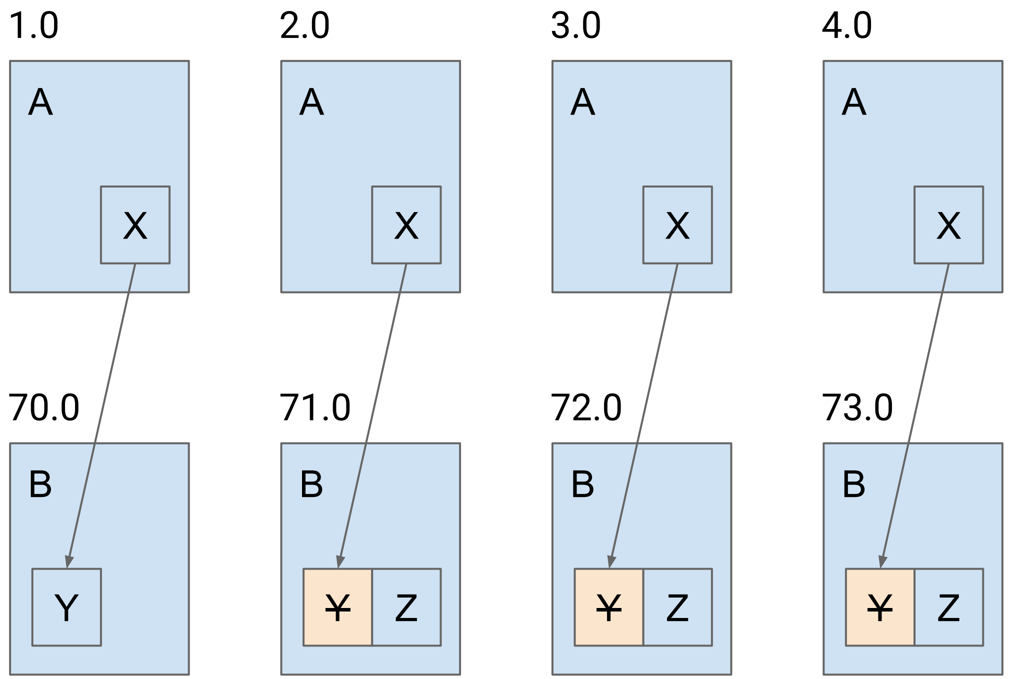
Problem 3: finding compatible dependencies



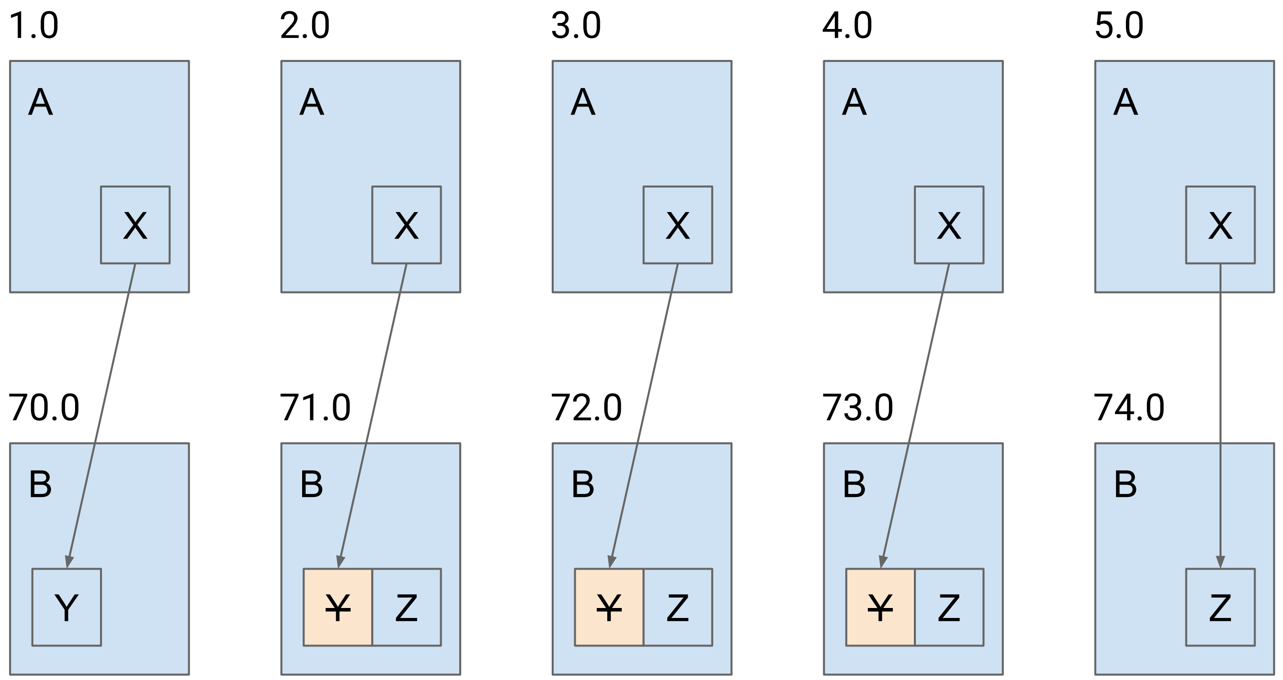
Problem 3: finding compatible dependencies



Problem 3: finding compatible dependencies

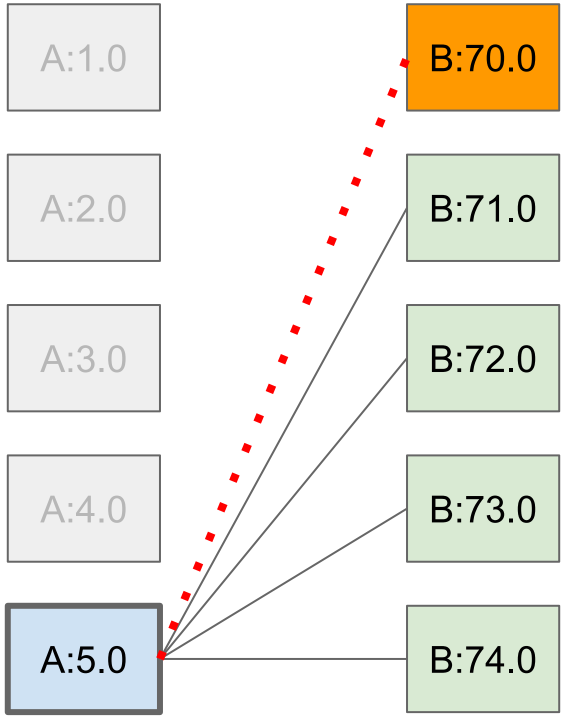


Problem 3: finding compatible dependencies



Problem 3: finding compatible dependencies

For the latest A (5.0)

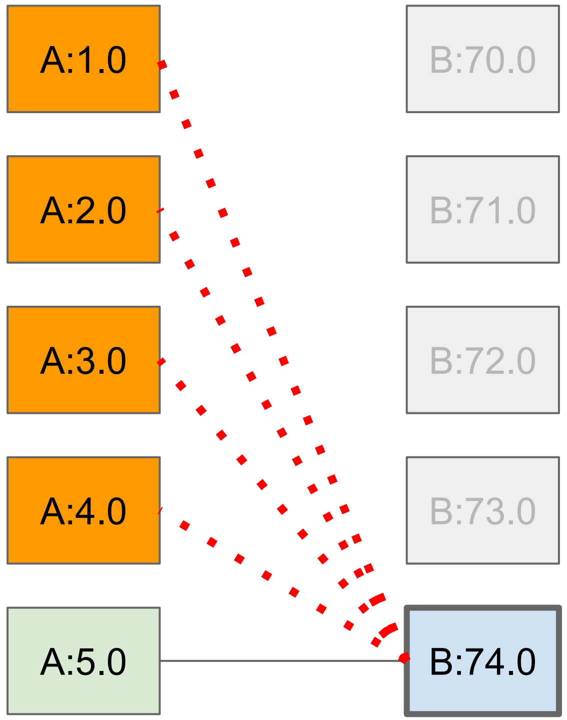


The latest A (5.0) can use the last 4 versions of B.

This is nice and flexible.

Problem 3: finding compatible dependencies

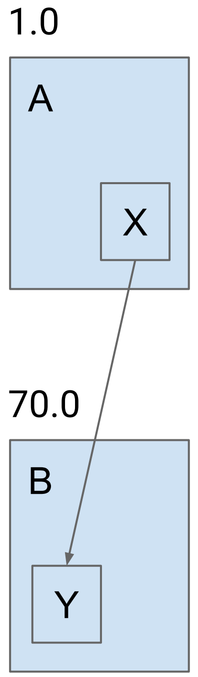
For the latest B (74.0)



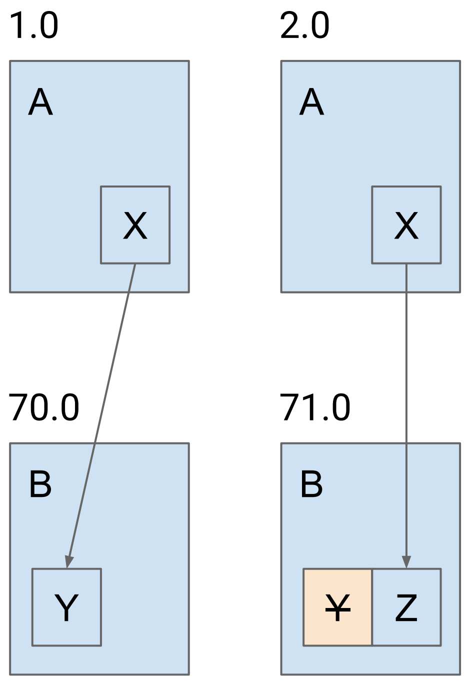
The latest B (74.0) can only use the last version of A (5.0).

This is somewhat restrictive!

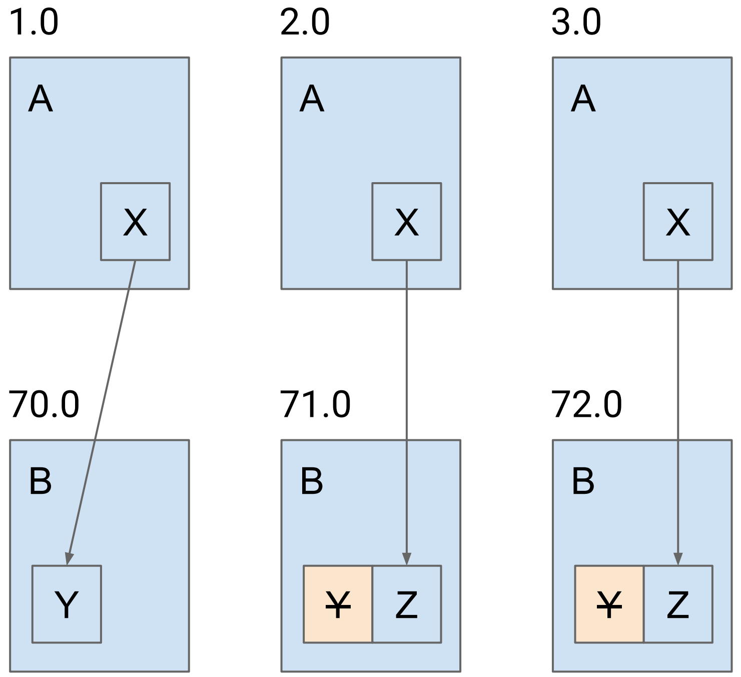
Problem 3: finding compatible dependencies



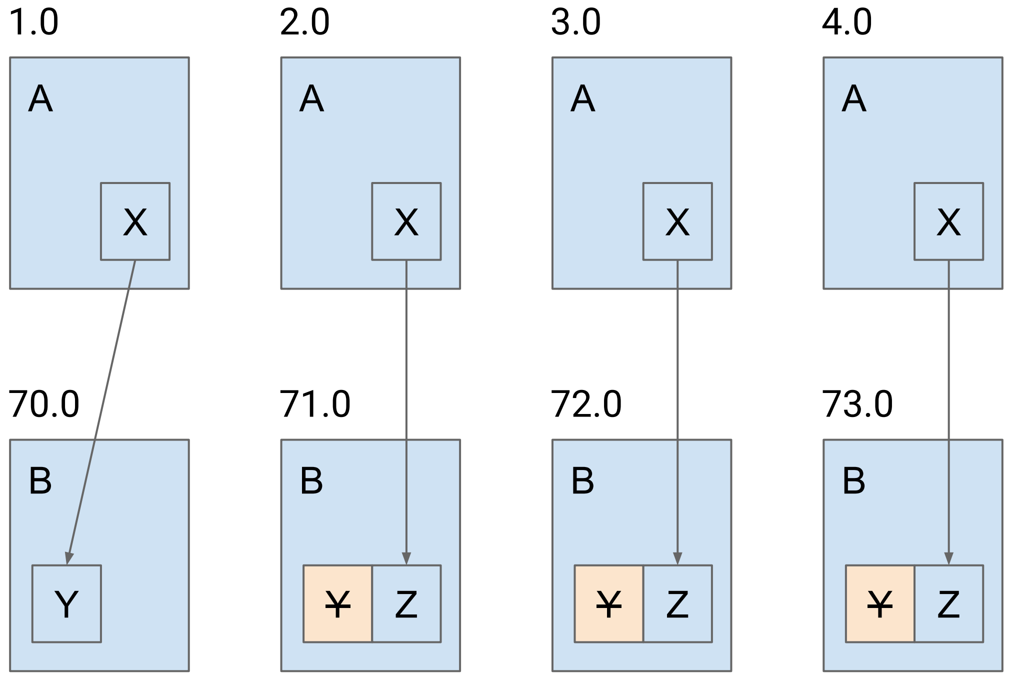
Problem 3: finding compatible dependencies



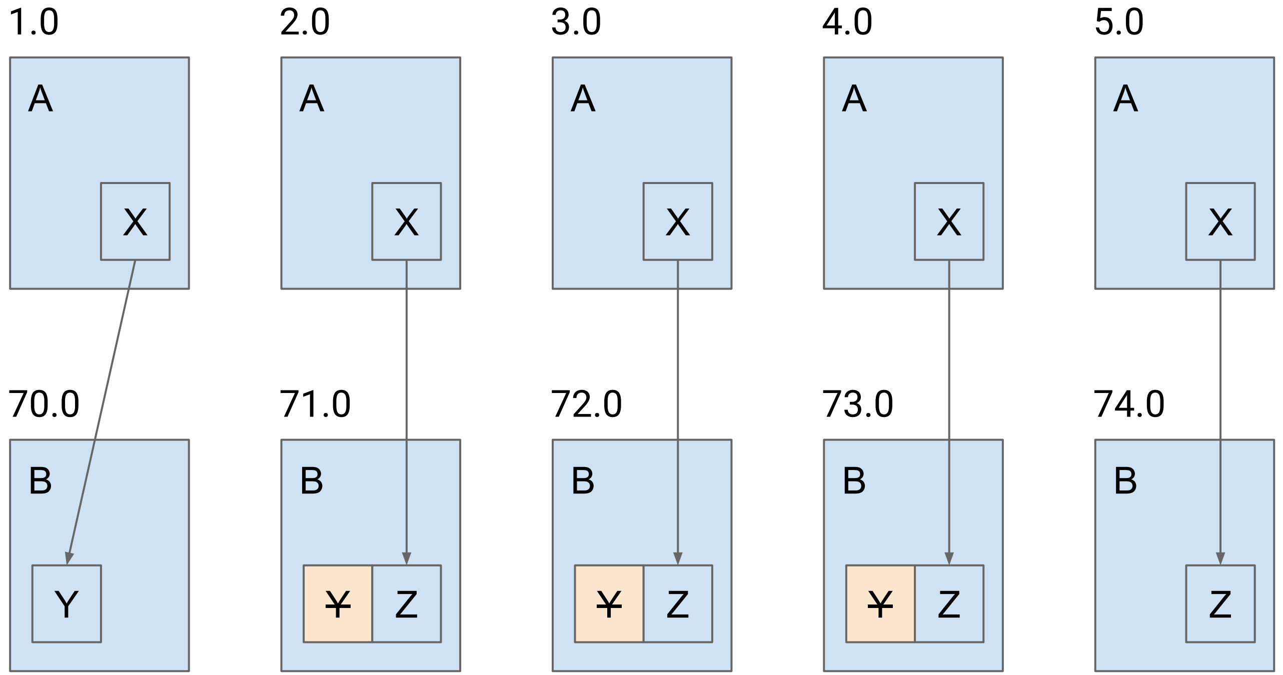
Problem 3: finding compatible dependencies



Problem 3: finding compatible dependencies

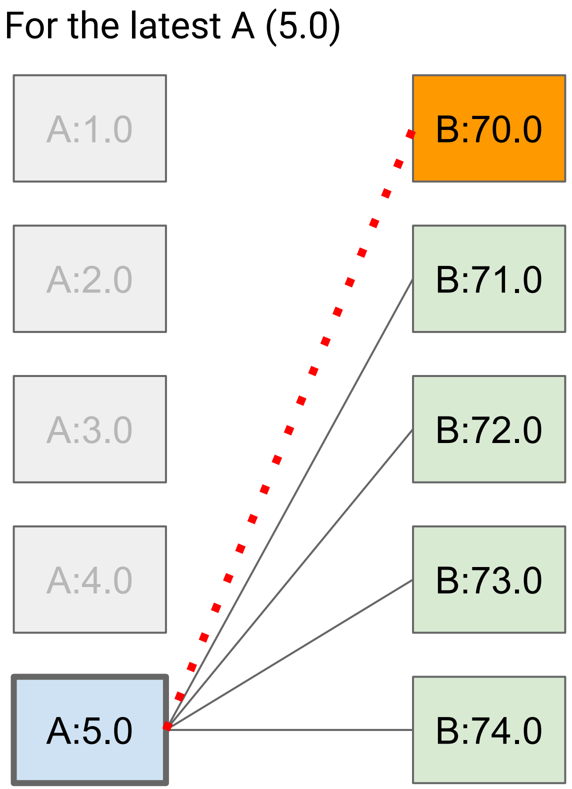


Problem 3: finding compatible dependencies



Google

Problem 3: finding compatible dependencies

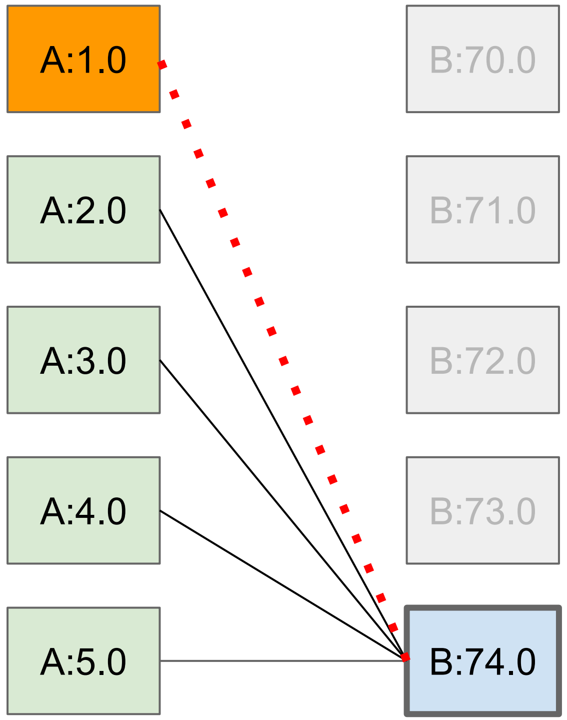


The latest A (5.0) can still use the last 4 versions of B.

No regression, things are going good.

Problem 3: finding compatible dependencies

For the latest B (74.0)



The latest B (74.0) can can used with the last 4 versions of A.

Great improvement! This is a lot more flexible.

Google

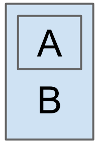
JLBP-13: Quickly remove references to deprecated features in dependencies

Example violation of JLBP-13

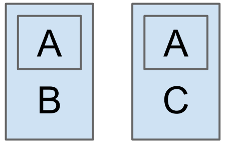
`com.google.api:api-common-java`: usage of deprecated methods in Guava removed 1 year + 3 months after deprecation (1.7.0), instead of earlier (e.g. 1.2.0)

What about shading?

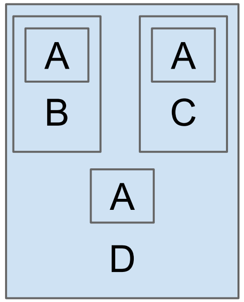
What about shading?



What about shading?

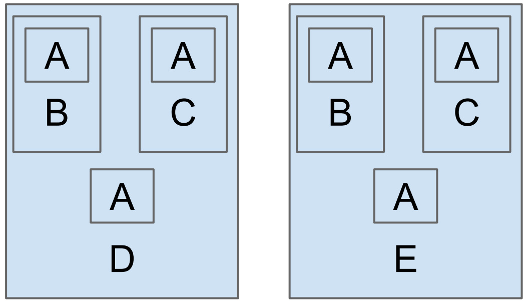


What about shading?



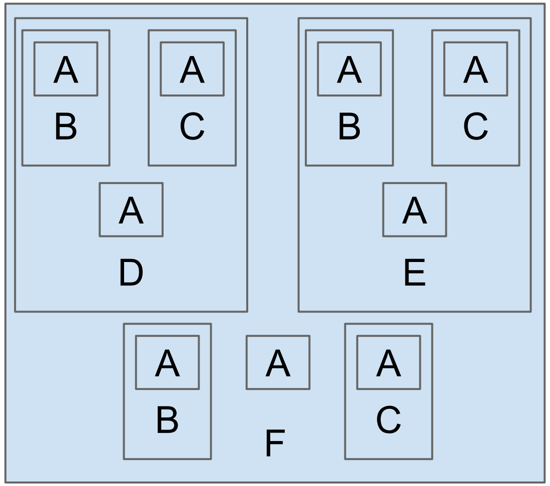
Google

What about shading?



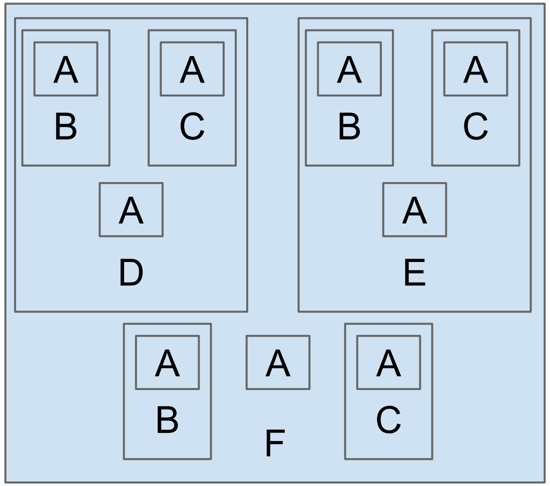
Google

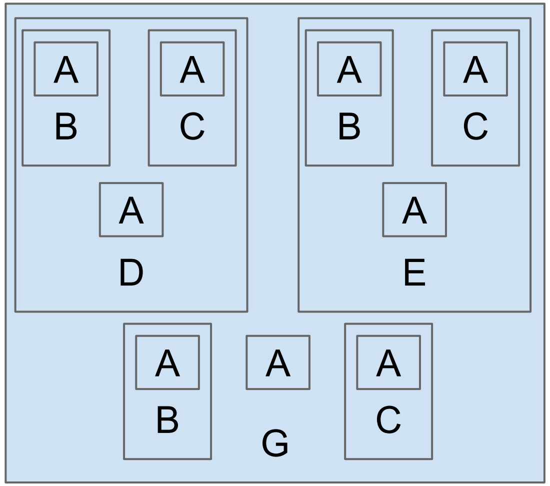
What about shading?



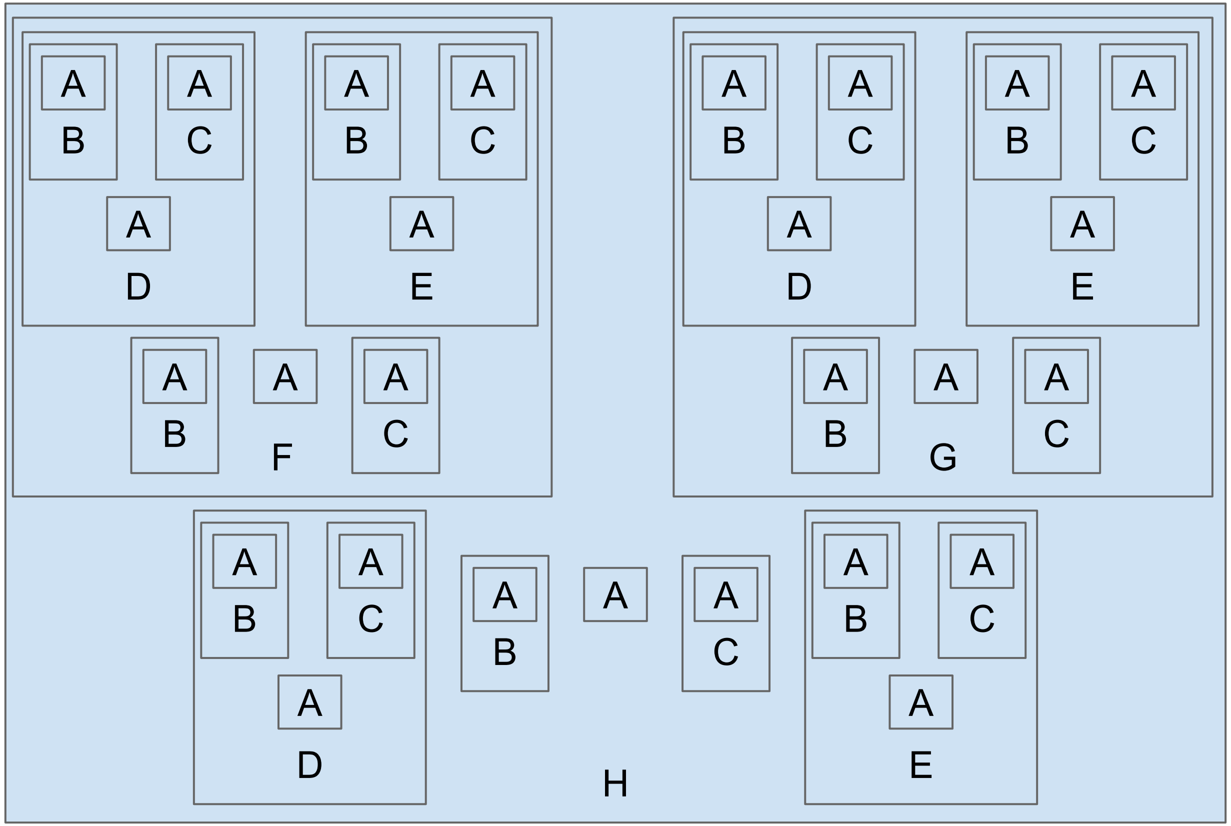
Google

What about shading?

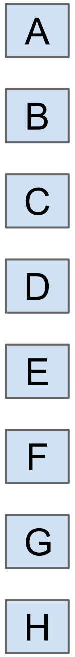




What about shading?



What about shading?



What about shading?

No shading: 8 units

With shading: 54 units (27 of which are copies of A)

54/8 = 6.75 x the size!

What about shading?

Other problems:

- Bad shading config can create overlapping classes or missing classes
- Shaded dependencies can't be overridden to roll out security fixes
- Shading doesn't work well with JNI or reflection

JLBP-18: Only shade dependencies as a last resort

This page is intentionally left blank

Consistency in OSS Libraries: Google’s Approach

Linkage Checker

github.com/GoogleCloudPlatform/cloud-opensource-java

Tomohiro Suzuki (suztomo@google.com)

Agenda: Linkage Checker

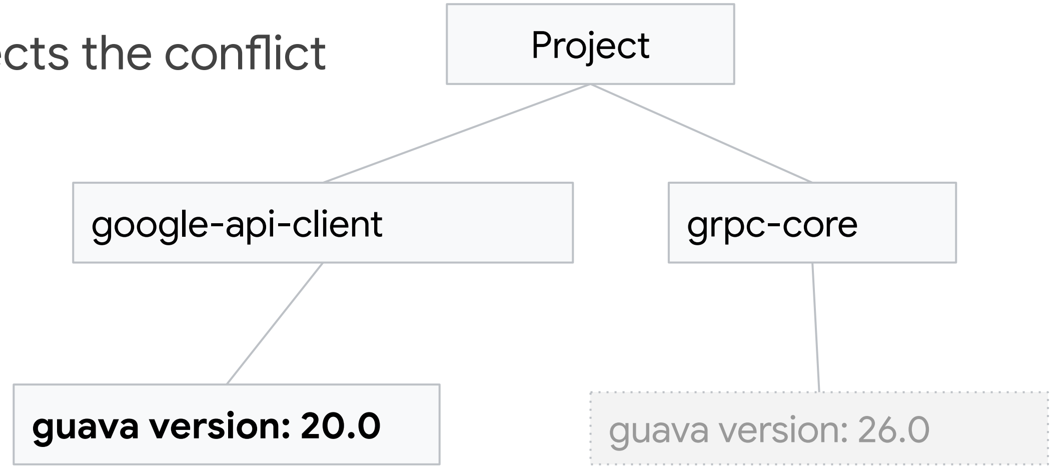
- Demo
- Implementation
- Case Study

Linkage Checker: Demo

Demo: Linkage Checker Maven Enforcer Rule

Problem: a simple project having dependency conflicts

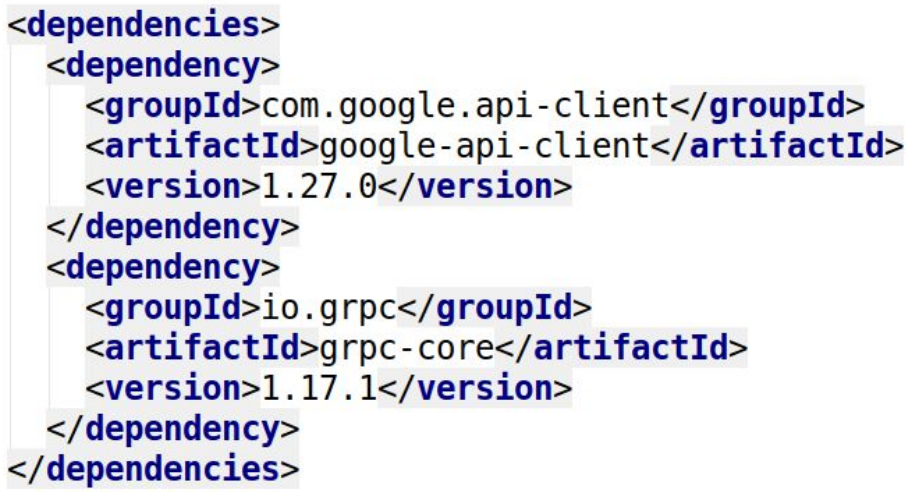
Linkage Checker Enforcer Rule detects the conflict



Google

Demo: Linkage Checker Maven Enforcer Rule

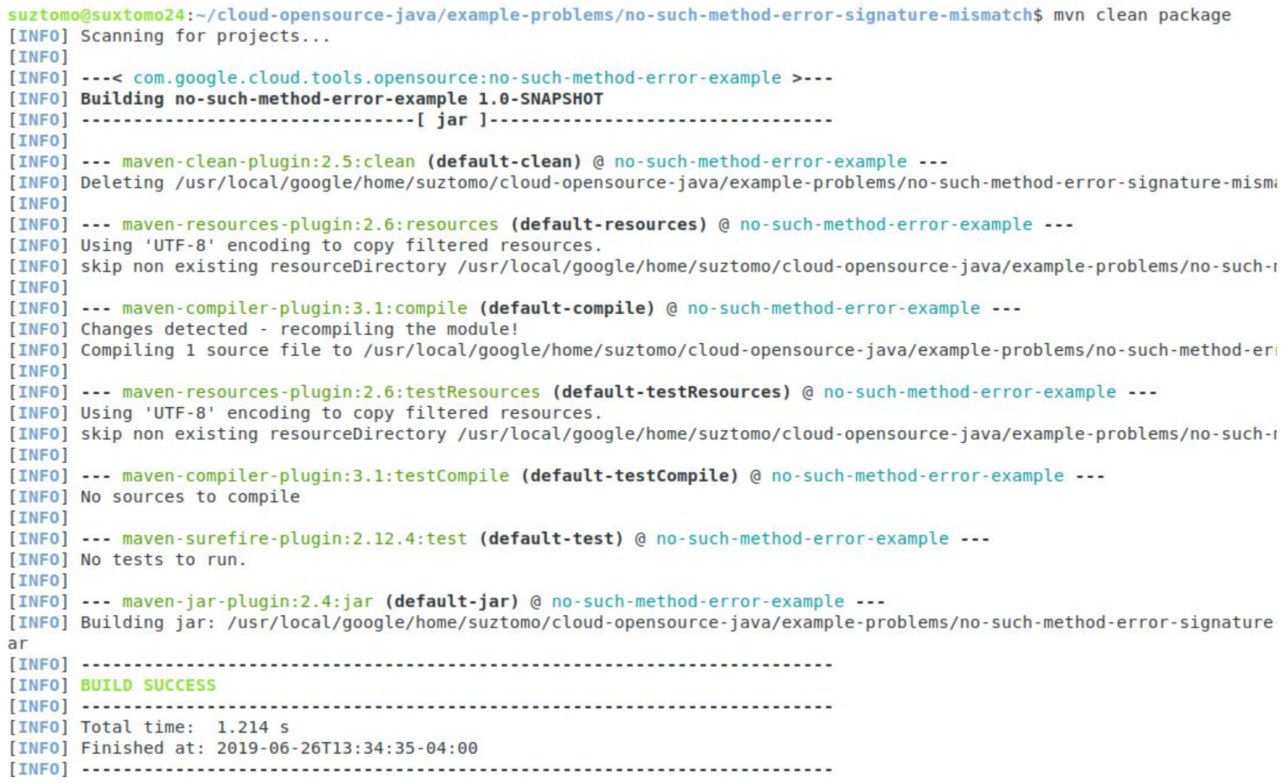
A simple project with two dependencies



```xml
<dependencies>
  <dependency>
    <groupId>com.google.api-client</groupId>
    <artifactId>google-api-client</artifactId>
    <version>1.27.0</version>
  </dependency>
  <dependency>
    <groupId>io.grpc</groupId>
    <artifactId>grpc-core</artifactId>
    <version>1.17.1</version>
  </dependency>
</dependencies>
```

Demo: Linkage Checker Maven Enforcer Rule

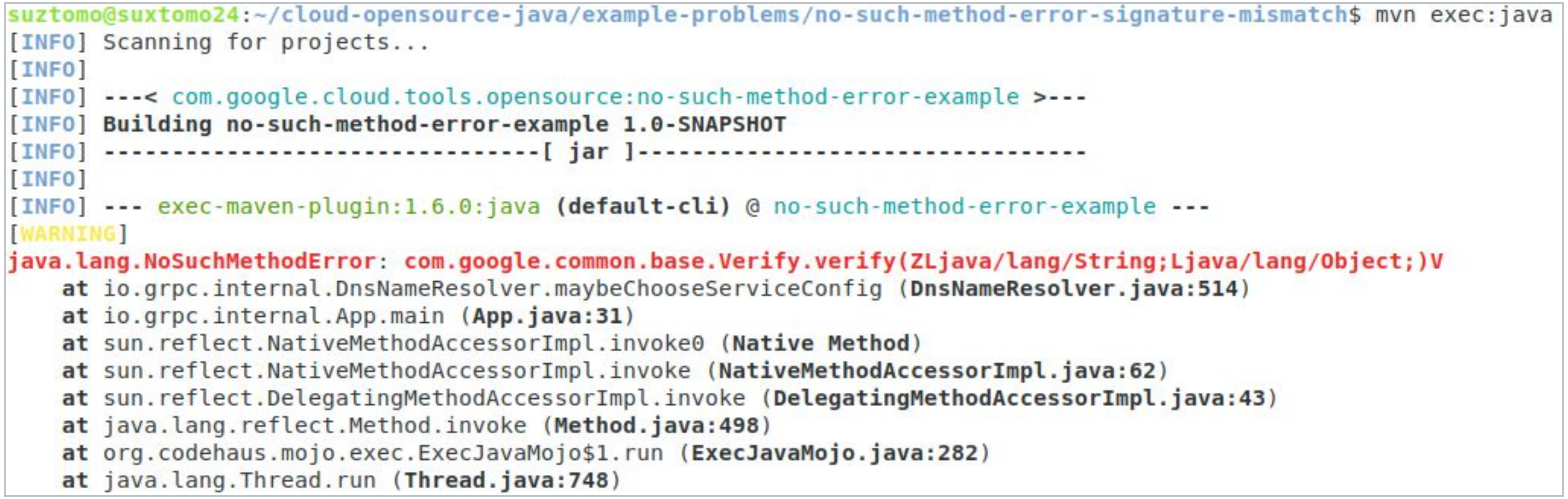
No issue in compile



```
suztomo@suxtomo24:~/cloud-opensource-java/example-problems/no-such-method-error-signature-mismatch$ mvn clean package
[INFO] Scanning for projects...
[INFO]
[INFO] ---< com.google.cloud.tools.opensource:no-such-method-error-example >---
[INFO] Building no-such-method-error-example 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ no-such-method-error-example ---
[INFO] Deleting /usr/local/google/home/suztomo/cloud-opensource-java/example-problems/no-such-method-error-signature-mism
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ no-such-method-error-example ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /usr/local/google/home/suztomo/cloud-opensource-java/example-problems/no-such-
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ no-such-method-error-example ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to /usr/local/google/home/suztomo/cloud-opensource-java/example-problems/no-such-method-er
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ no-such-method-error-example ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /usr/local/google/home/suztomo/cloud-opensource-java/example-problems/no-such-
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ no-such-method-error-example ---
[INFO] No sources to compile
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ no-such-method-error-example ---
[INFO] No tests to run.
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ no-such-method-error-example ---
[INFO] Building jar: /usr/local/google/home/suztomo/cloud-opensource-java/example-problems/no-such-method-error-signature
ar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  1.214 s
[INFO] Finished at: 2019-06-26T13:34:35-04:00
[INFO] ------------------------------------------------------------------------
```

Demo: Linkage Checker Maven Enforcer Rule

Runtime Error!

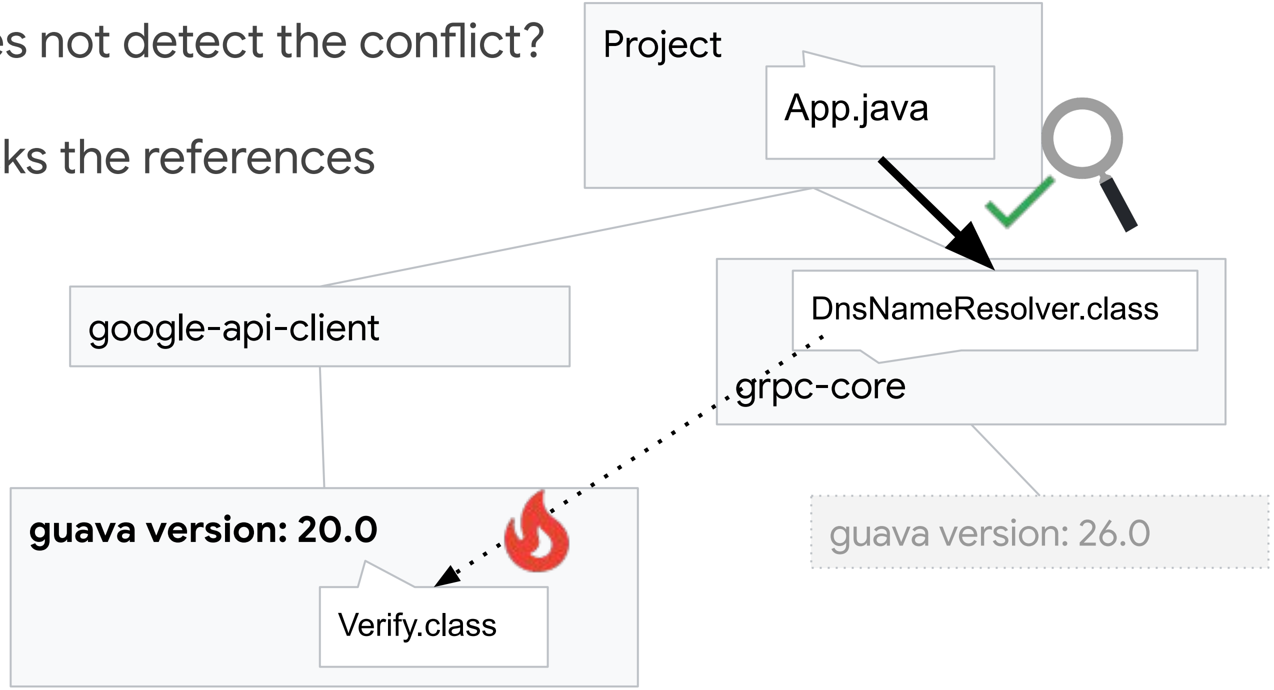


```
suztomo@suxtomo24:~/cloud-opensource-java/example-problems/no-such-method-error-signature-mismatch$ mvn exec:java
[INFO] Scanning for projects...
[INFO]
[INFO] ---< com.google.cloud.tools.opensource:no-such-method-error-example >---
[INFO] Building no-such-method-error-example 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- exec-maven-plugin:1.6.0:java (default-cli) @ no-such-method-error-example ---
[WARNING]
java.lang.NoSuchMethodError: com.google.common.base.Verify.verify(ZLjava/lang/String;Ljava/lang/Object;)V
    at io.grpc.internal.DnsNameResolver.maybeChooseServiceConfig (DnsNameResolver.java:514)
    at io.grpc.internal.App.main (App.java:31)
    at sun.reflect.NativeMethodAccessorImpl.invoke0 (Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke (NativeMethodAccessorImpl.java:62)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke (DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke (Method.java:498)
    at org.codehaus.mojo.exec.ExecJavaMojo$1.run (ExecJavaMojo.java:282)
    at java.lang.Thread.run (Thread.java:748)
```

Demo: Linkage Checker Maven Enforcer Rule

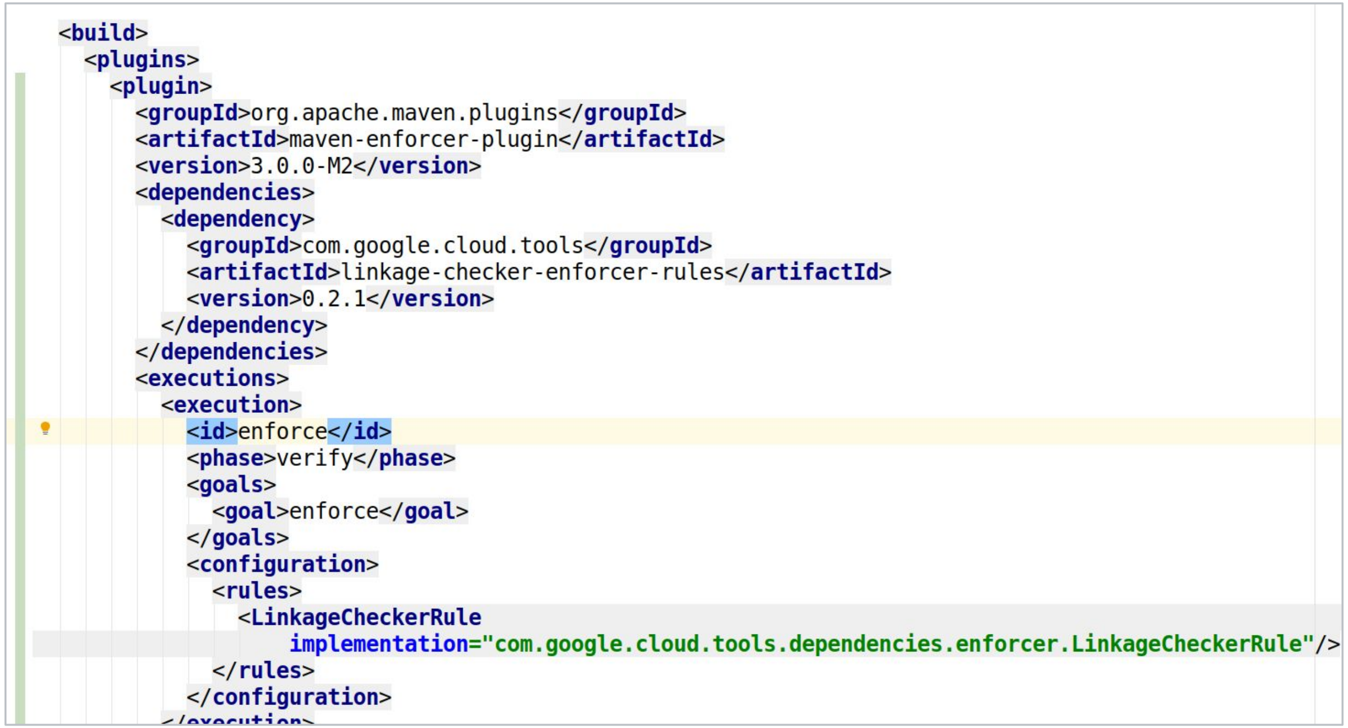
Q: Why java compiler does not detect the conflict?

A: the compiler only checks the references from the project



Demo: Linkage Checker Maven Enforcer Rule

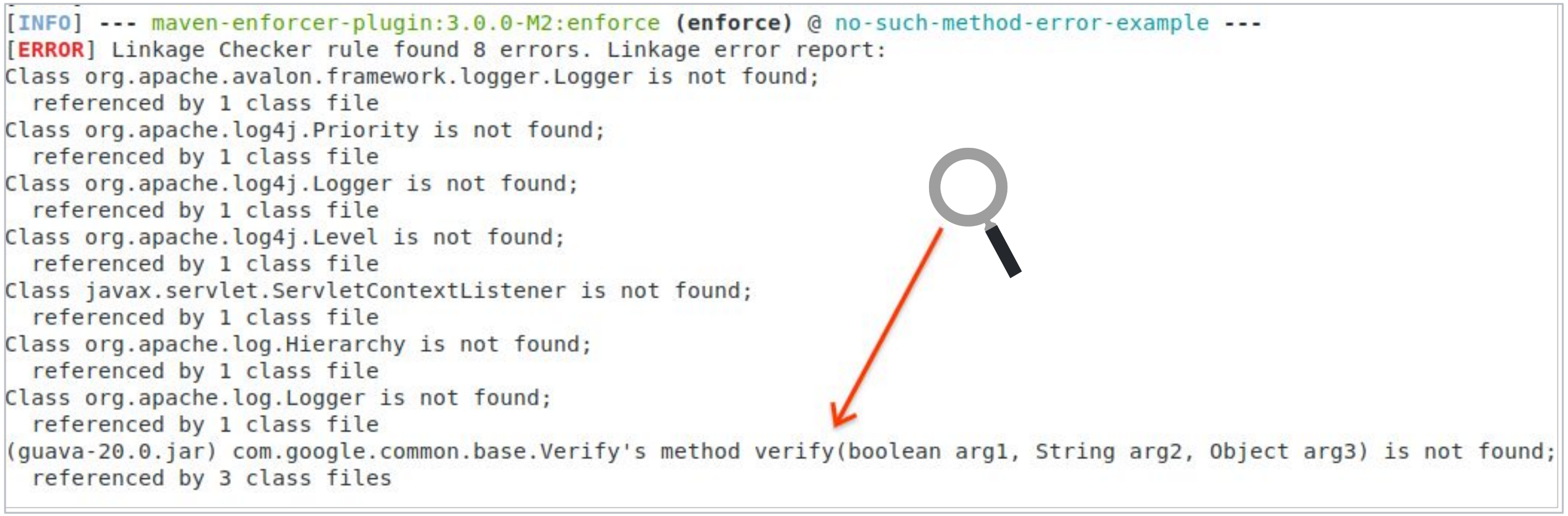
Add the enforcer rule



```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-enforcer-plugin</artifactId>
      <version>3.0.0-M2</version>
      <dependencies>
        <dependency>
          <groupId>com.google.cloud.tools</groupId>
          <artifactId>linkage-checker-enforcer-rules</artifactId>
          <version>0.2.1</version>
        </dependency>
      </dependencies>
      <executions>
        <execution>
          <id>enforce</id>
          <phase>verify</phase>
          <goals>
            <goal>enforce</goal>
          </goals>
          <configuration>
            <rules>
              <LinkageCheckerRule
                  implementation="com.google.cloud.tools.dependencies.enforcer.LinkageCheckerRule"/>
            </rules>
          </configuration>
```

Demo: Linkage Checker Maven Enforcer Rule

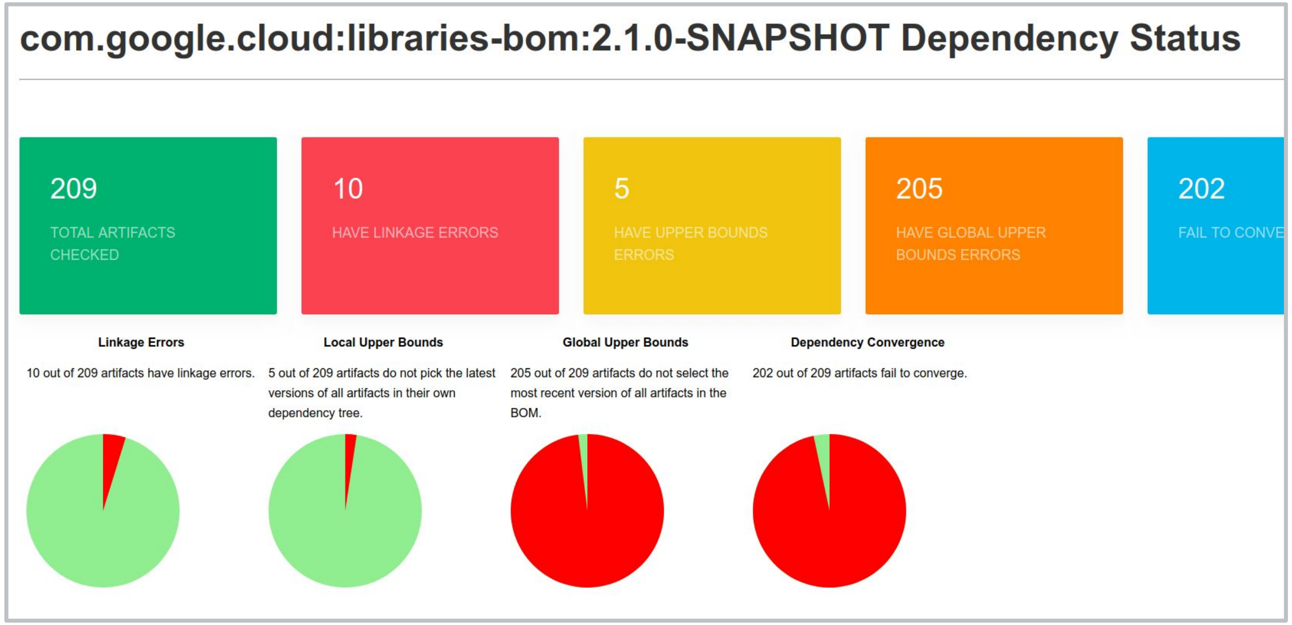
$ mvn install



```
[INFO] --- maven-enforcer-plugin:3.0.0-M2:enforce (enforce) @ no-such-method-error-example ---
[ERROR] Linkage Checker rule found 8 errors. Linkage error report:
Class org.apache.avalon.framework.logger.Logger is not found;
  referenced by 1 class file
Class org.apache.log4j.Priority is not found;
  referenced by 1 class file
Class org.apache.log4j.Logger is not found;
  referenced by 1 class file
Class org.apache.log4j.Level is not found;
  referenced by 1 class file
Class javax.servlet.ServletContextListener is not found;
  referenced by 1 class file
Class org.apache.log.Hierarchy is not found;
  referenced by 1 class file
Class org.apache.log.Logger is not found;
  referenced by 1 class file
(guava-20.0.jar) com.google.common.base.Verify's method verify(boolean arg1, String arg2, Object arg3) is not found;
  referenced by 3 class files
```

Demo: Google Libraries BOM Dashboard

Checks compatibility of artifacts in BOM (a set of libraries)



com.google.cloud:libraries-bom:2.1.0-SNAPSHOT Dependency Status

209 TOTAL ARTIFACTS CHECKED

10 HAVE LINKAGE ERRORS

5 HAVE UPPER BOUNDS ERRORS

205 HAVE GLOBAL UPPER BOUNDS ERRORS

202 FAIL TO CONVE

Linkage Errors

10 out of 209 artifacts have linkage errors.

Local Upper Bounds

5 out of 209 artifacts do not pick the latest versions of all artifacts in their own dependency tree.

Global Upper Bounds

205 out of 209 artifacts do not select the most recent version of all artifacts in the BOM.

Dependency Convergence

202 out of 209 artifacts fail to converge.

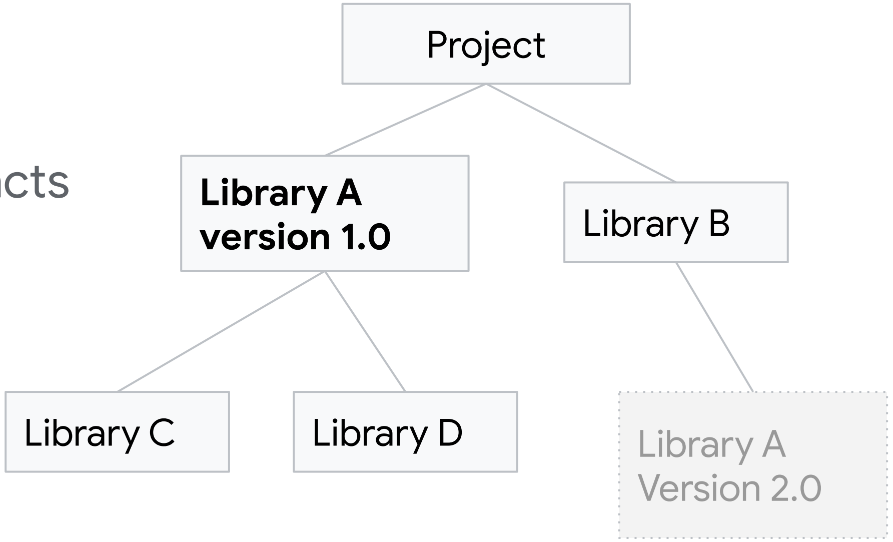
https://github.com/GoogleCloudPlatform/cloud-opensource-java#google-cloud-platform-java-dependency-dashboard

Google

Linkage Checker: Implementation

Implementation Step 1: Dependency Tree

Creates dependency tree of Maven artifacts (Maven's dependency mediation)



Implementation Step 2: JAR, Class files, and Constant Pool

Extracts references from constant pool

- JAR file
    - Class File
        - Constant Pool

(Example)
io.grpc.internal.DnsNameResolver constant pool:

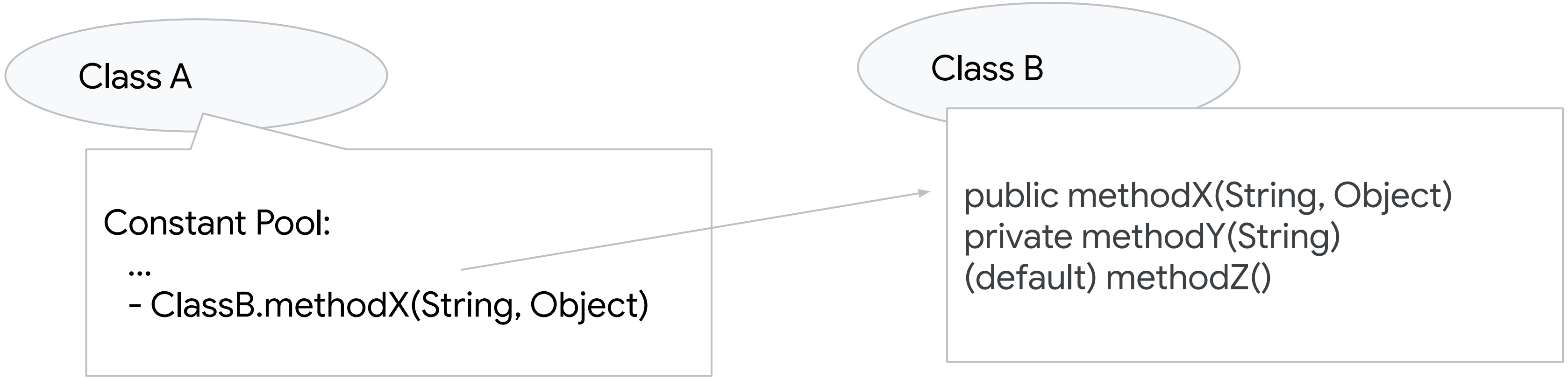
- class io.grpc.NameResolver
...
- method: “com.google.base.Verify.verify(bool, String, Object)”

Google

Implementation Step 3: Verification of Referents

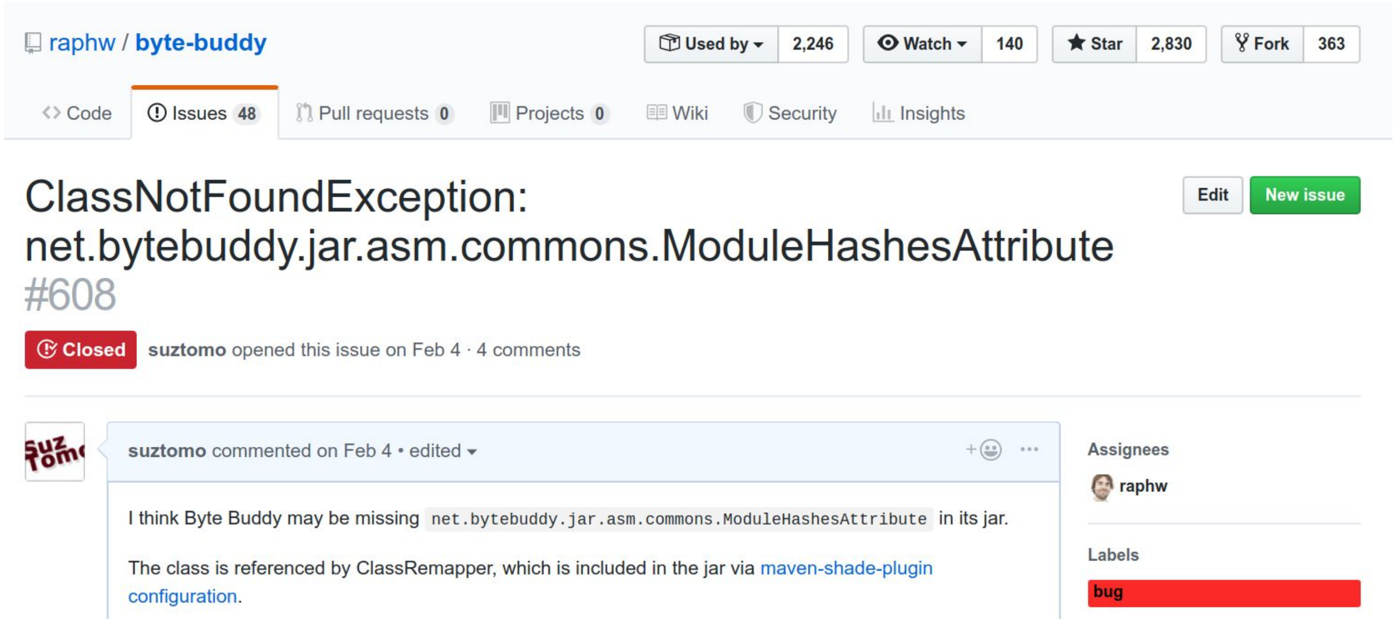
Verifies the referents of the references

Does the referent (class B) exist?
Does the referent have the method with the expected signature?
Does the method accessible from class A?
etc...



Case Study

Case 1: Missing Class



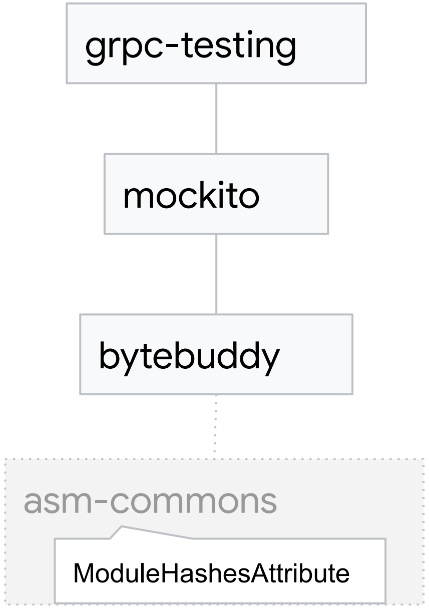
ClassNotFoundException: net.bytebuddy.jar.asm.commons.ModuleHashesAttribute #608

suztomo opened this issue on Feb 4 · 4 comments

suztomo commented on Feb 4 • edited

I think Byte Buddy may be missing `net.bytebuddy.jar.asm.commons.ModuleHashesAttribute` in its jar.

The class is referenced by ClassRemapper, which is included in the jar via maven-shade-plugin configuration.



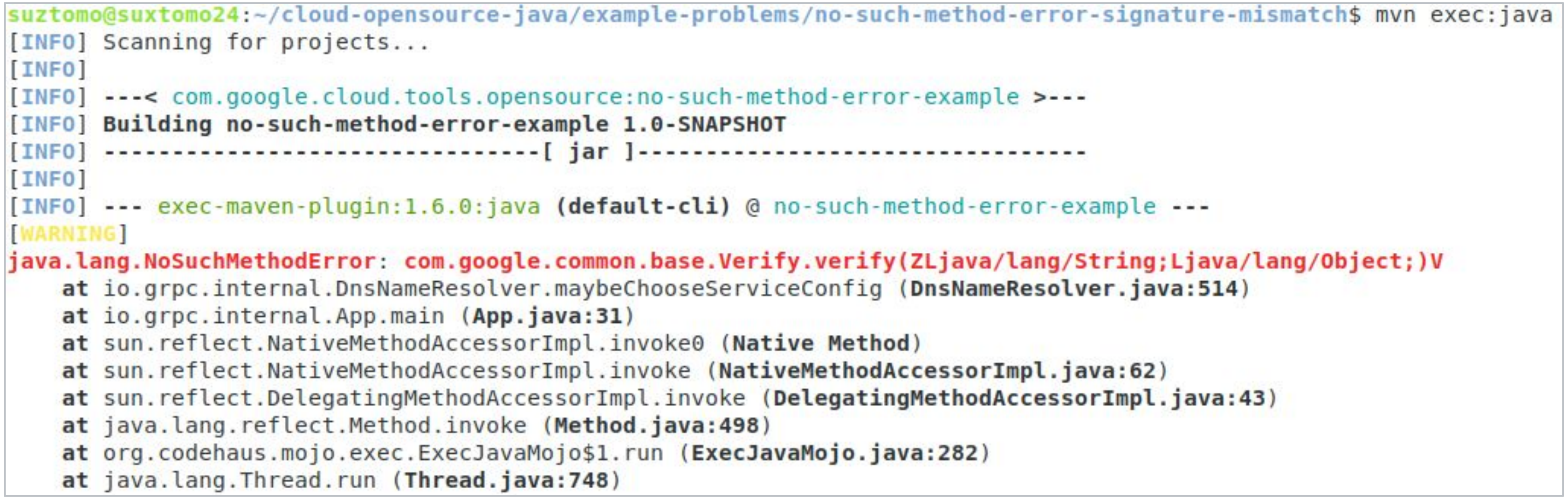
JLBP-18: Only shade dependencies as a last resort

Google

Case 2: Missing Method

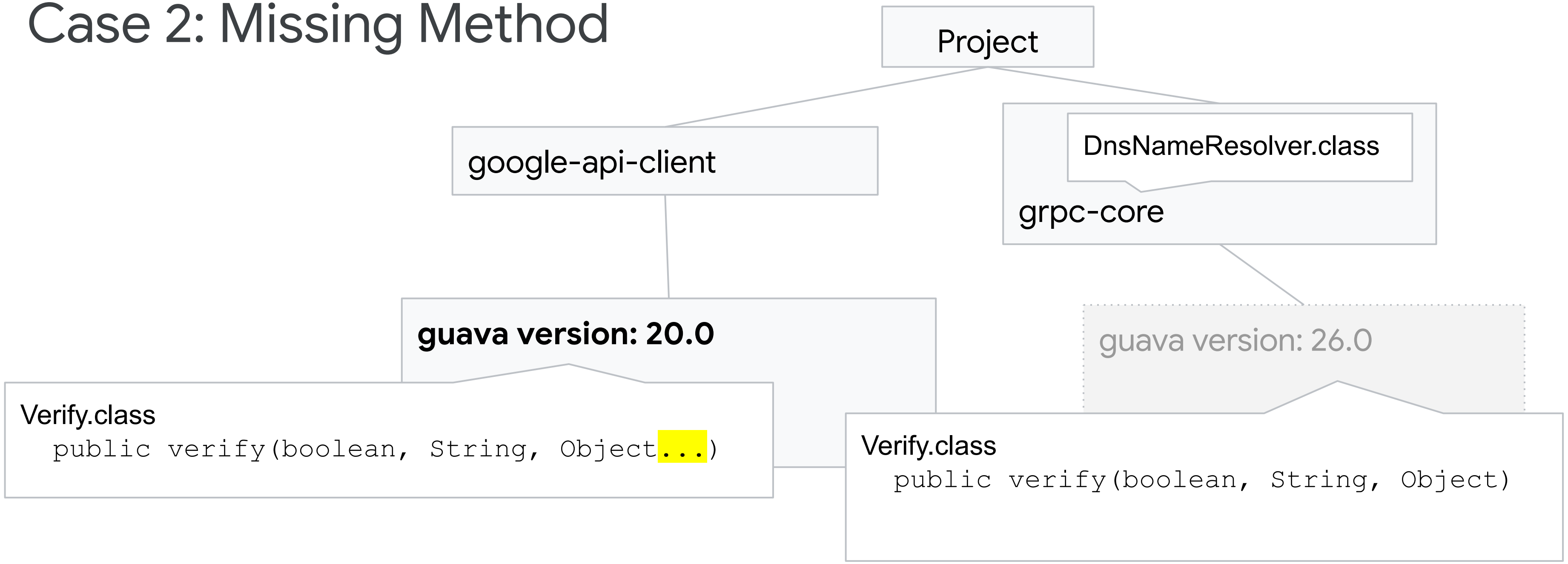
Runtime Error from the demo

```
NoSuchMethodError Verify.verify(ZLjava/lang/String;Ljava/lang/Object;)V
                      => void verify(boolean, String, Object)
```



```
suztomo@suxtomo24:~/cloud-opensource-java/example-problems/no-such-method-error-signature-mismatch$ mvn exec:java
[INFO] Scanning for projects...
[INFO]
[INFO] ---< com.google.cloud.tools.opensource:no-such-method-error-example >---
[INFO] Building no-such-method-error-example 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- exec-maven-plugin:1.6.0:java (default-cli) @ no-such-method-error-example ---
[WARNING]
java.lang.NoSuchMethodError: com.google.common.base.Verify.verify(ZLjava/lang/String;Ljava/lang/Object;)V
    at io.grpc.internal.DnsNameResolver.maybeChooseServiceConfig (DnsNameResolver.java:514)
    at io.grpc.internal.App.main (App.java:31)
    at sun.reflect.NativeMethodAccessorImpl.invoke0 (Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke (NativeMethodAccessorImpl.java:62)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke (DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke (Method.java:498)
    at org.codehaus.mojo.exec.ExecJavaMojo$1.run (ExecJavaMojo.java:282)
    at java.lang.Thread.run (Thread.java:748)
```

Case 2: Missing Method



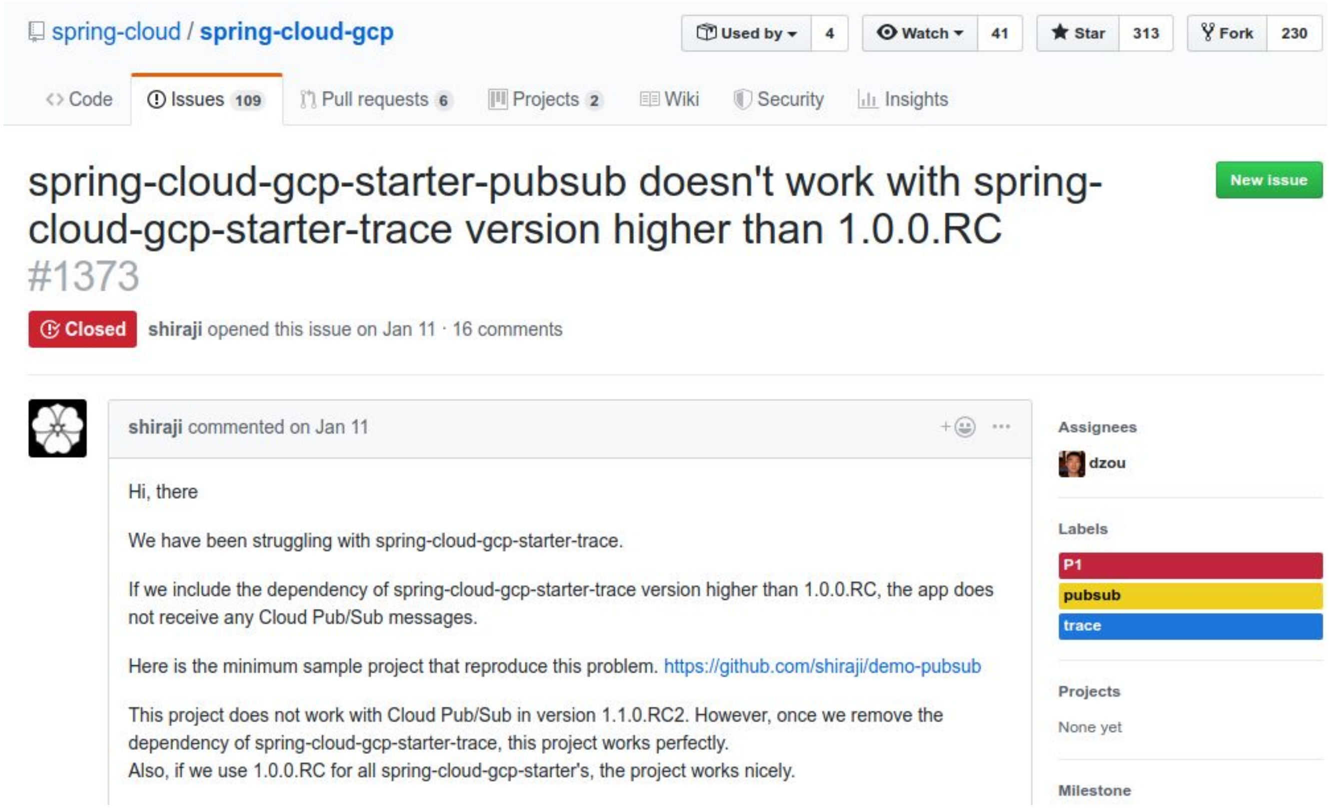
```
public verify(boolean, String, Object...)
```

```
public verify(boolean, String, Object)
```

[Java Language Specification Chapter 13: Binary Compatibility]

Google

Case 3: Missing Constructor



spring-cloud-gcp-starter-pubsub doesn't work with spring-cloud-gcp-starter-trace version higher than 1.0.0.RC #1373

Closed shiraji opened this issue on Jan 11 · 16 comments

shiraji commented on Jan 11

Hi, there

We have been struggling with spring-cloud-gcp-starter-trace.

If we include the dependency of spring-cloud-gcp-starter-trace version higher than 1.0.0.RC, the app does not receive any Cloud Pub/Sub messages.

Here is the minimum sample project that reproduce this problem. https://github.com/shiraji/demo-pubsub

This project does not work with Cloud Pub/Sub in version 1.1.0.RC2. However, once we remove the dependency of spring-cloud-gcp-starter-trace, this project works perfectly.
Also, if we use 1.0.0.RC for all spring-cloud-gcp-starter's, the project works nicely.

Assignees
dzou

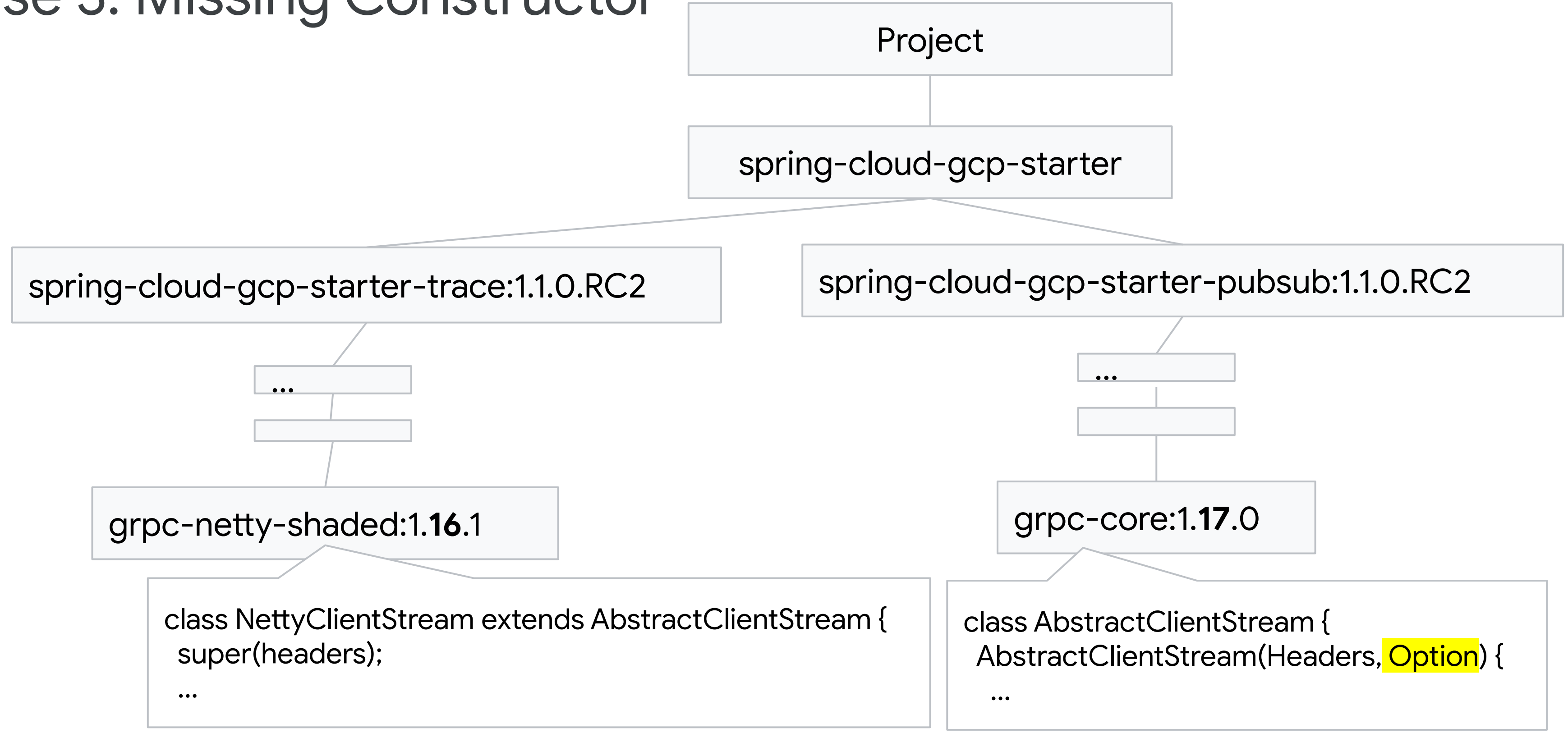
Labels
P1
pubsub
trace

Projects
None yet

Milestone

Google

Case 3: Missing Constructor


Project
spring-cloud-gcp-starter
spring-cloud-gcp-starter-trace:1.1.0.RC2
spring-cloud-gcp-starter-pubsub:1.1.0.RC2
...
...
grpc-netty-shaded:1.**16**.1
grpc-core:1.**17**.0

```
class NettyClientStream extends AbstractClientStream {
  super(headers);
  ...
```

```
class AbstractClientStream {
  AbstractClientStream(Headers, Option) {
   ...
```

Google

Conclusion

Diamond dependency issues:
  Dependency tree generated by different libraries may have conflicts.

- Java Library Best Practices
- Linkage Checker

https://github.com/GoogleCloudPlatform/cloud-opensource-java/

Google

Q&A