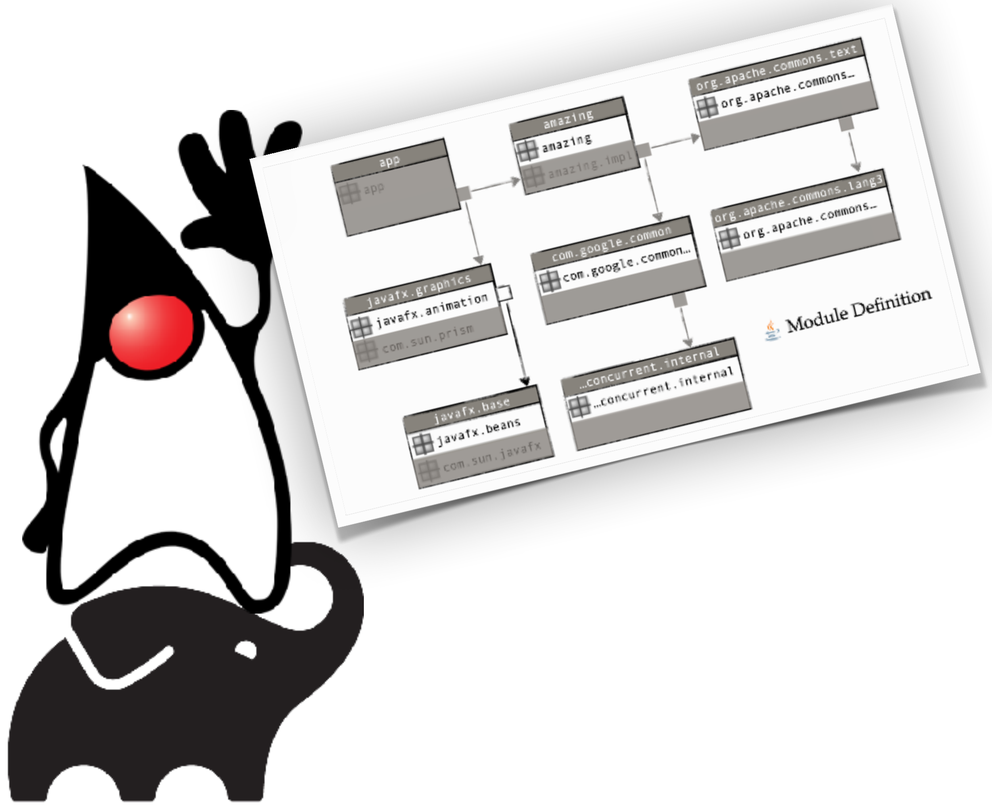
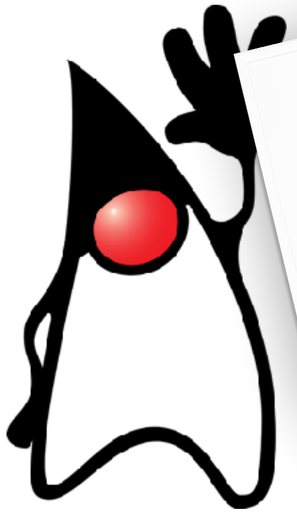
Das Java-Modul-System klug mit Gradle kombinieren

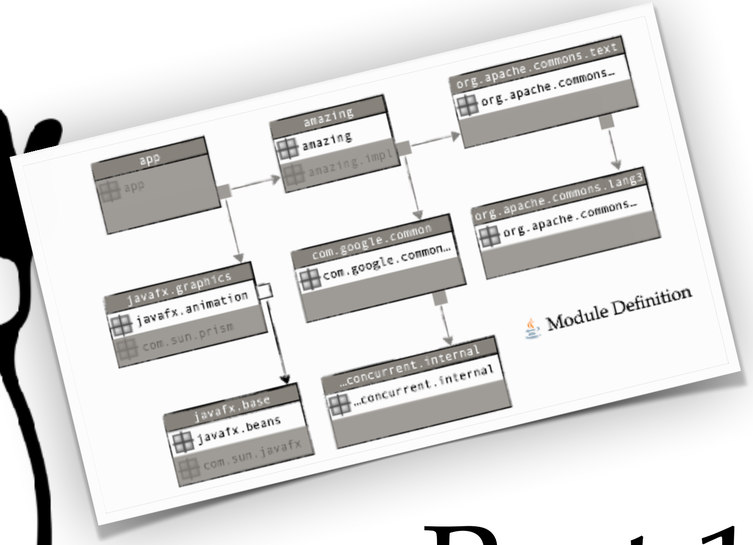


Jendrik Johannes

github.com/jjohannes

Java Forum Stuttgart 2024





Part 1: The Problem

Accidental Complexity in **Module Definition**

Software Architecture

Jar Files (uncleaned)

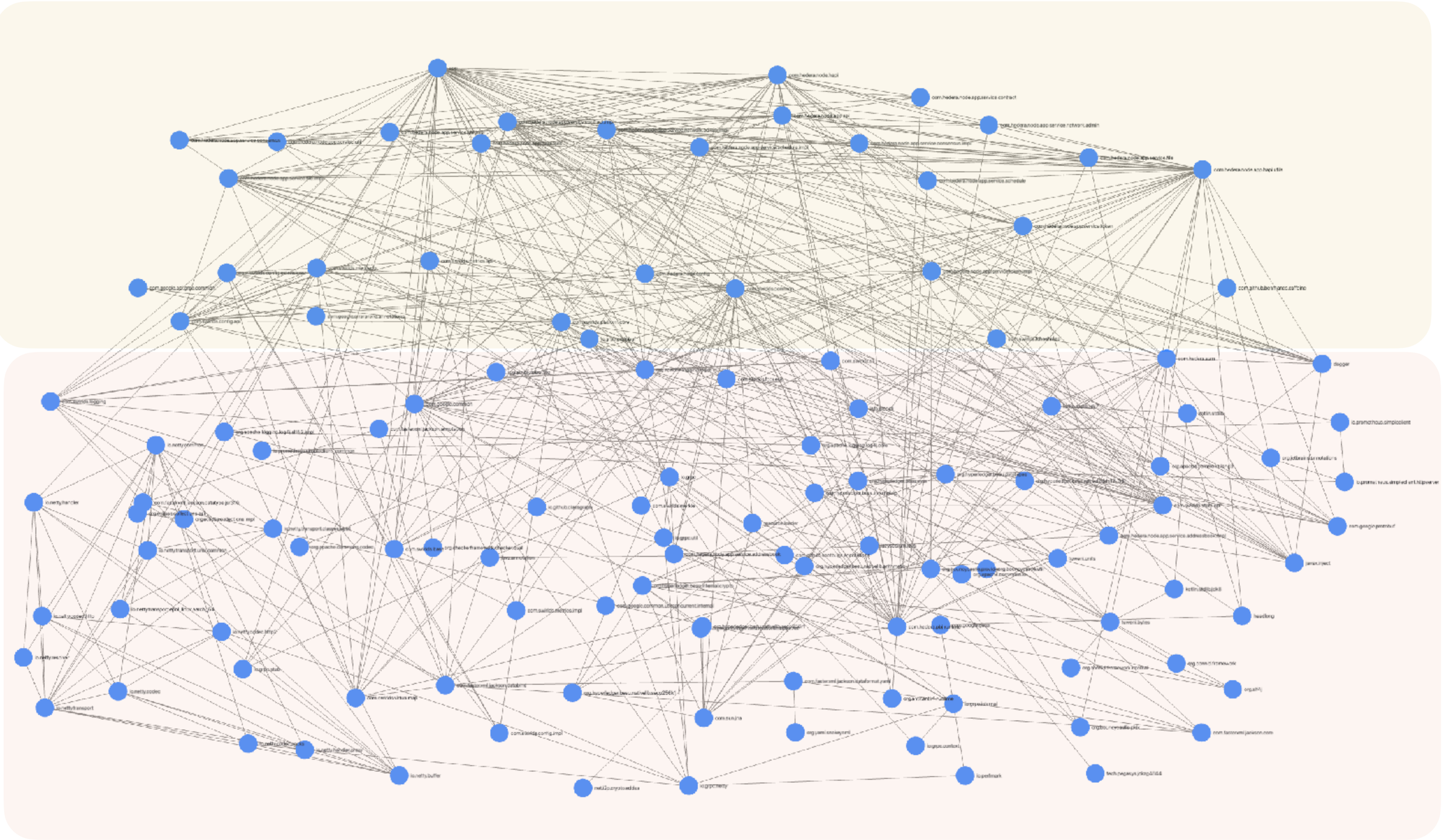
Jar Files

JVM Runtime Classic

JVM Runtime Module System

JVM Compile Time Module app

JVM Compile Time Module amazing



github.com/hashgraph/hedera-services

poloclub.github.io/argo-graph-lite/#d5b95db8-a363-46a5-8757-a846e116fa2b

Software Architecture

Jar Files (uncleaned)

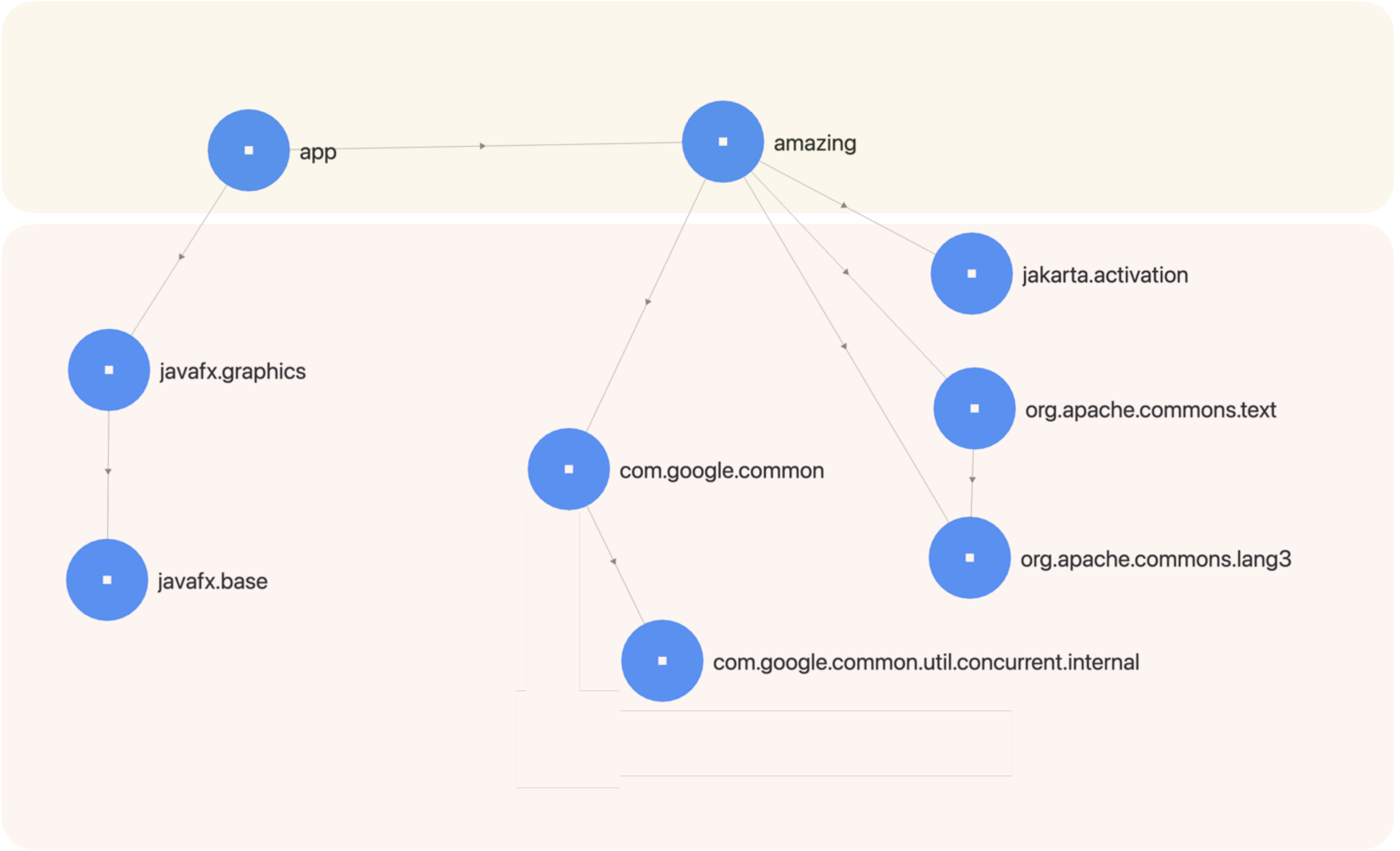
Jar Files

JVM Runtime Classic

JVM Runtime Module System

JVM Compile Time Module app

JVM Compile Time Module amazing



github.com/jjohannes/java-module-system

poloclub.github.io/argo-graph-lite/#66b5f3c7-656b-495d-842f-3675fc63216d

Software Architecture

Jar Files (uncleaned)

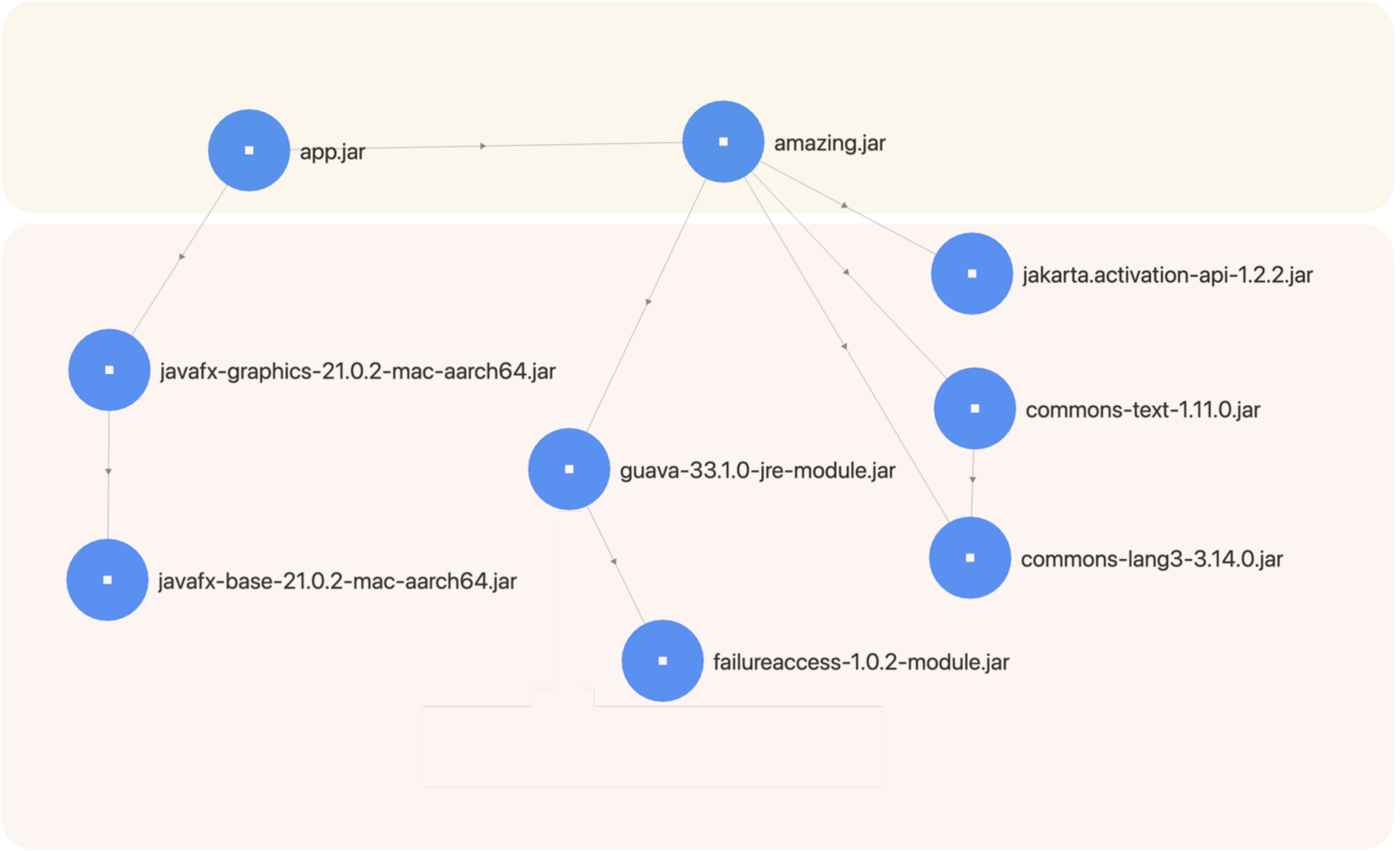
Jar Files

JVM Runtime Classic

JVM Runtime Module System

JVM Compile Time Module app

JVM Compile Time Module amazing



Software Architecture

Jar Files (uncleaned)

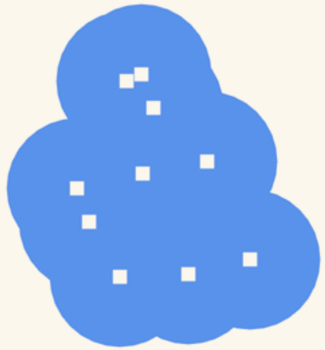
Jar Files

JVM Runtime Classic

JVM Runtime Module System

JVM Compile Time Module app

JVM Compile Time Module amazing



Software Architecture

Jar Files (uncleaned)

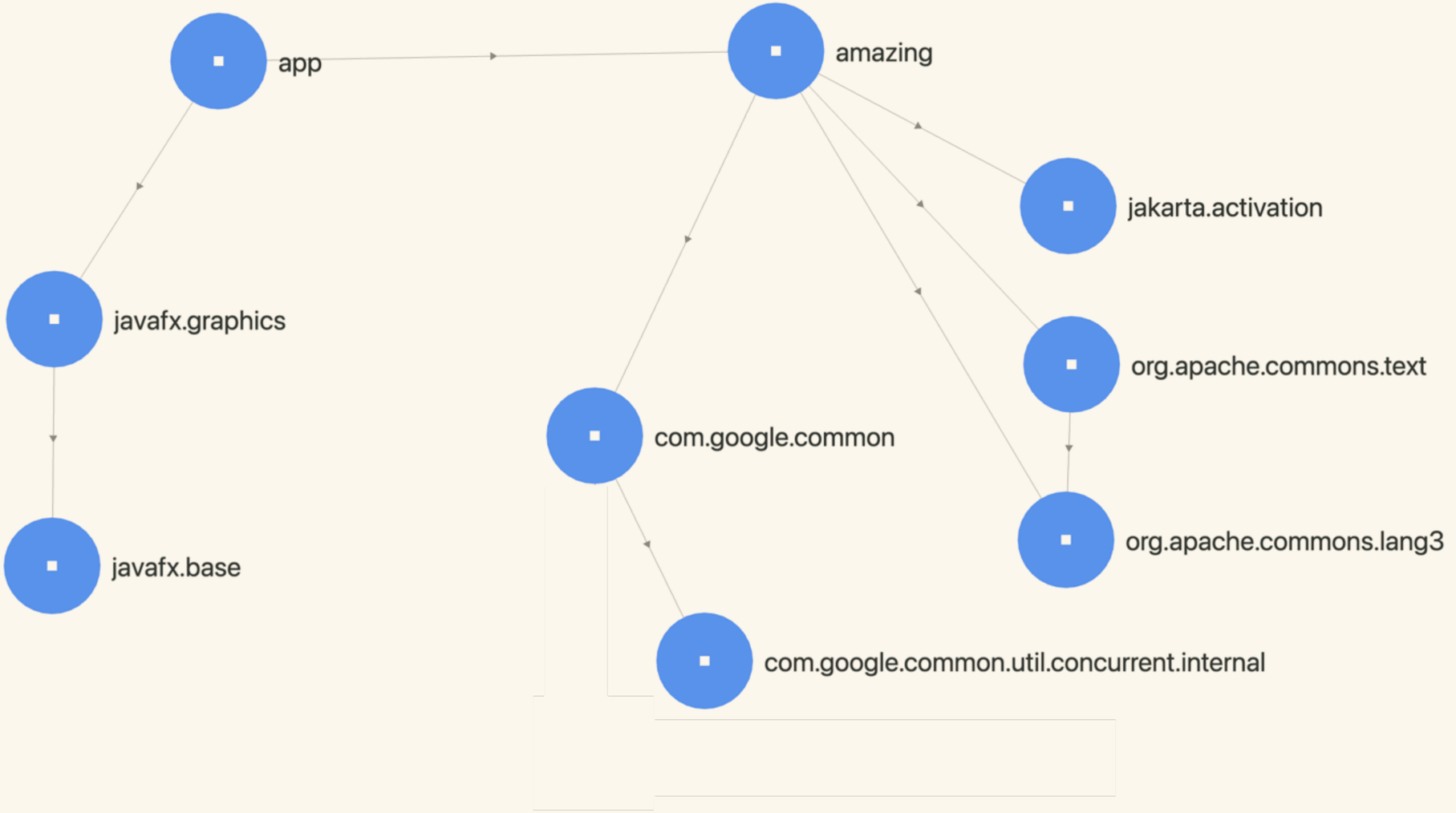
Jar Files

JVM Runtime Classic

JVM Runtime Module System

JVM Compile Time Module app

JVM Compile Time Module amazing



Software Architecture

Jar Files (uncleaned)

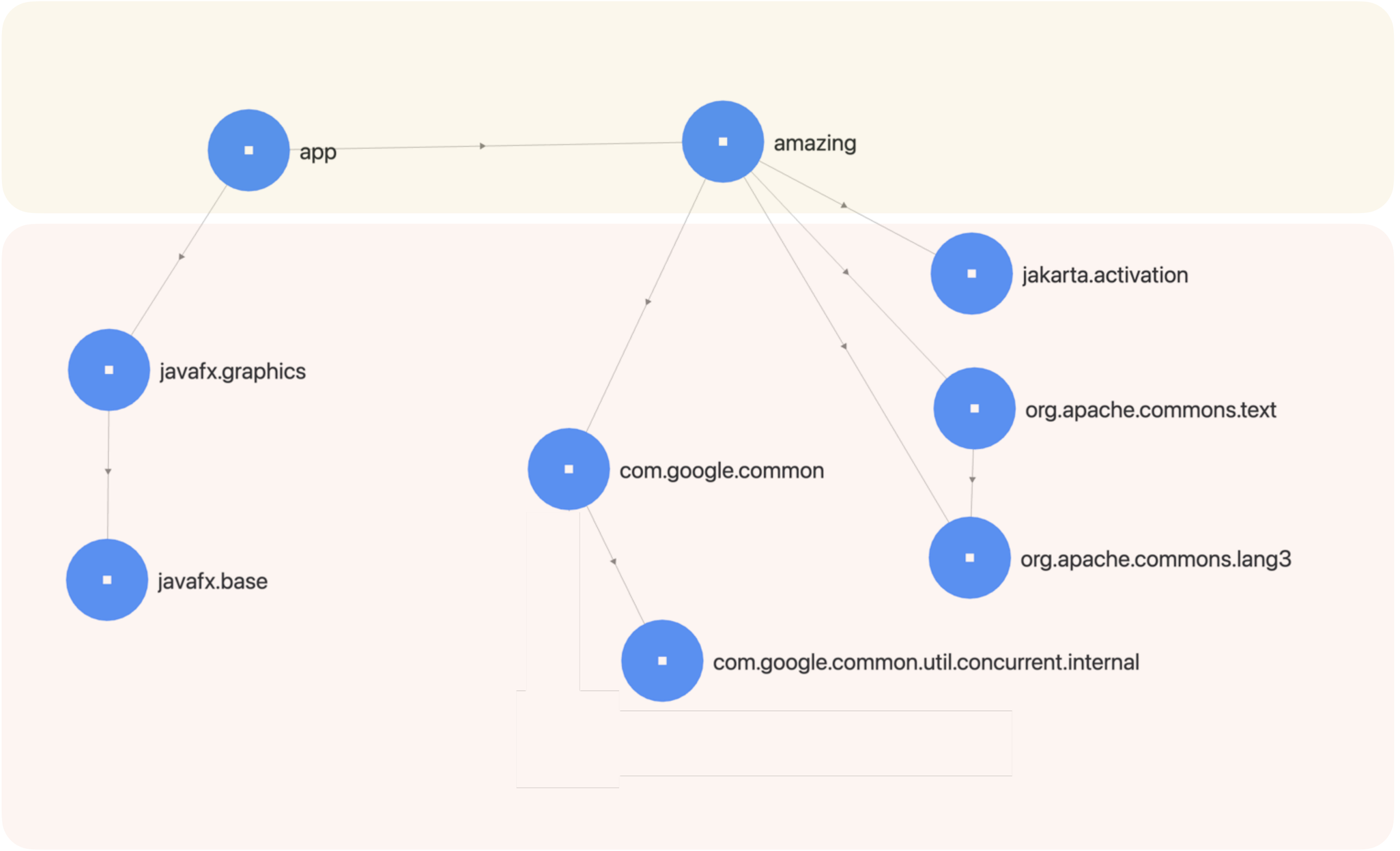
Jar Files

JVM Runtime Classic

JVM Runtime Module System

JVM Compile Time Module app

JVM Compile Time Module amazing



Software Architecture

Jar Files (uncleaned)

Jar Files

JVM Runtime Classic

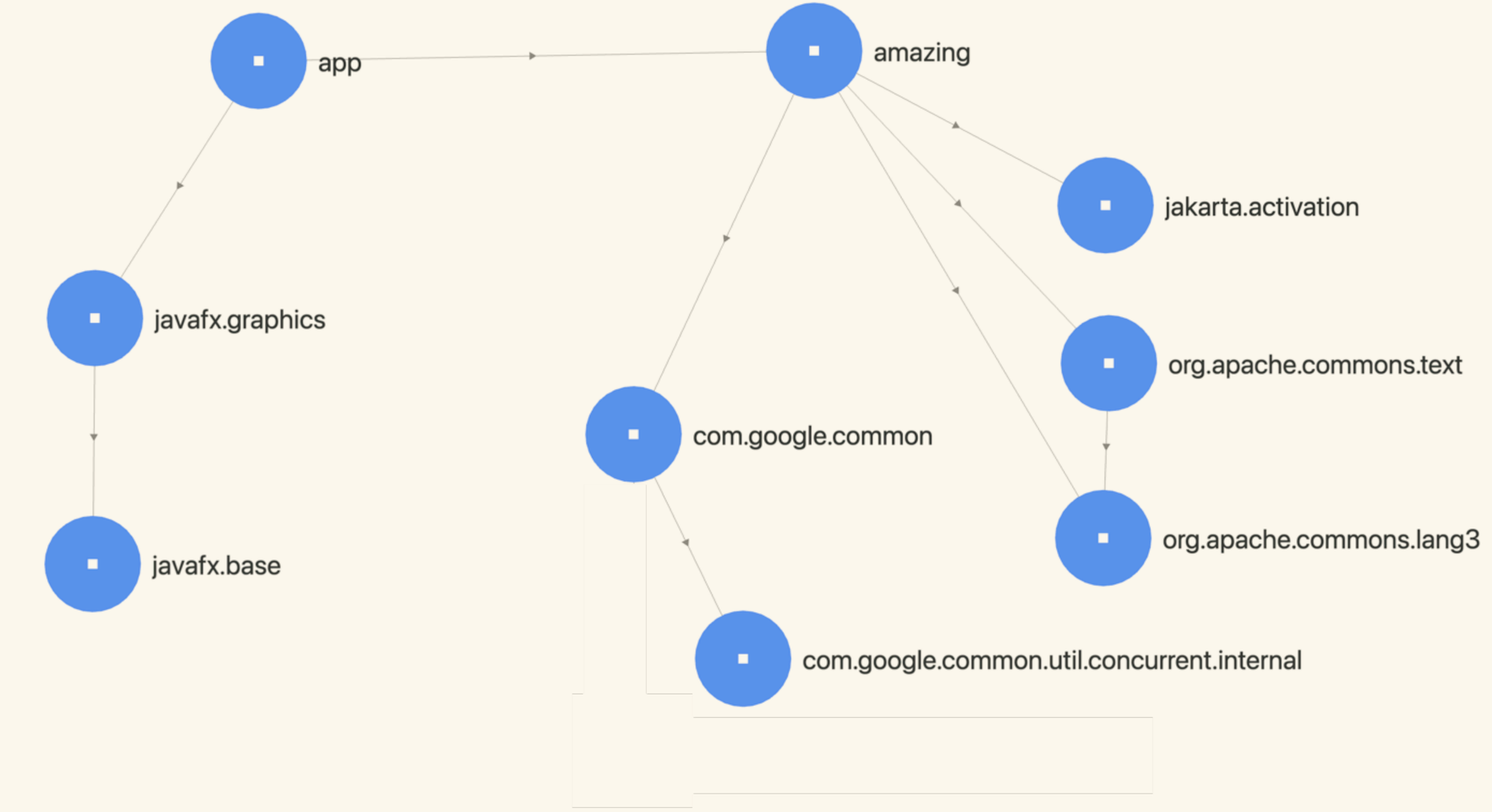
JVM Runtime Module System

JVM Compile Time Module app

JVM Compile Time Module amazing

Essential Complexity in Java's Modularity #1

Scopes are important



Software Architecture

Jar Files (uncleaned)

Jar Files

JVM Runtime Classic

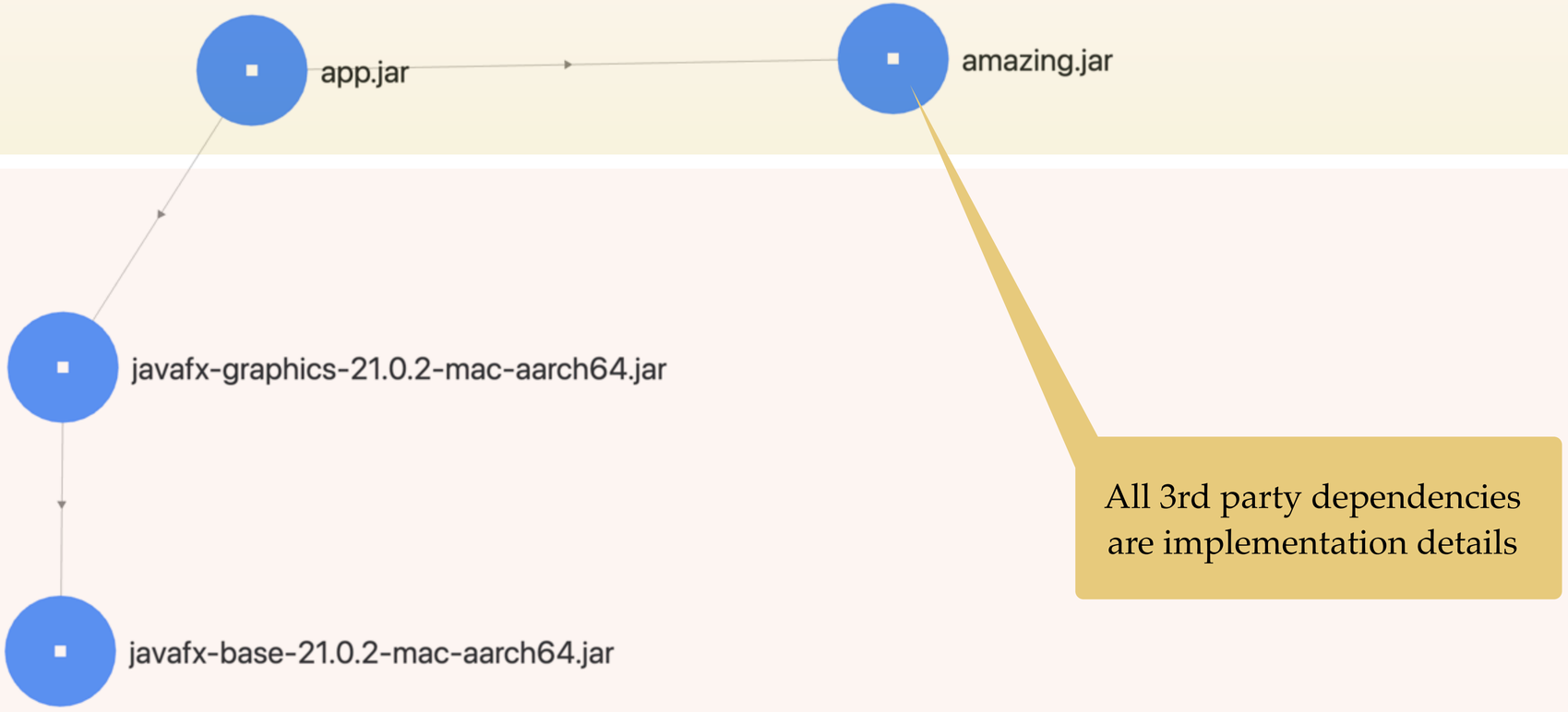
JVM Runtime Module System

JVM Compile Time
Module app

JVM Compile Time
Module amazing

Essential Complexity in Java's Modularity #1

Scopes are important



Software Architecture

Jar Files (uncleaned)

Jar Files

JVM Runtime Classic

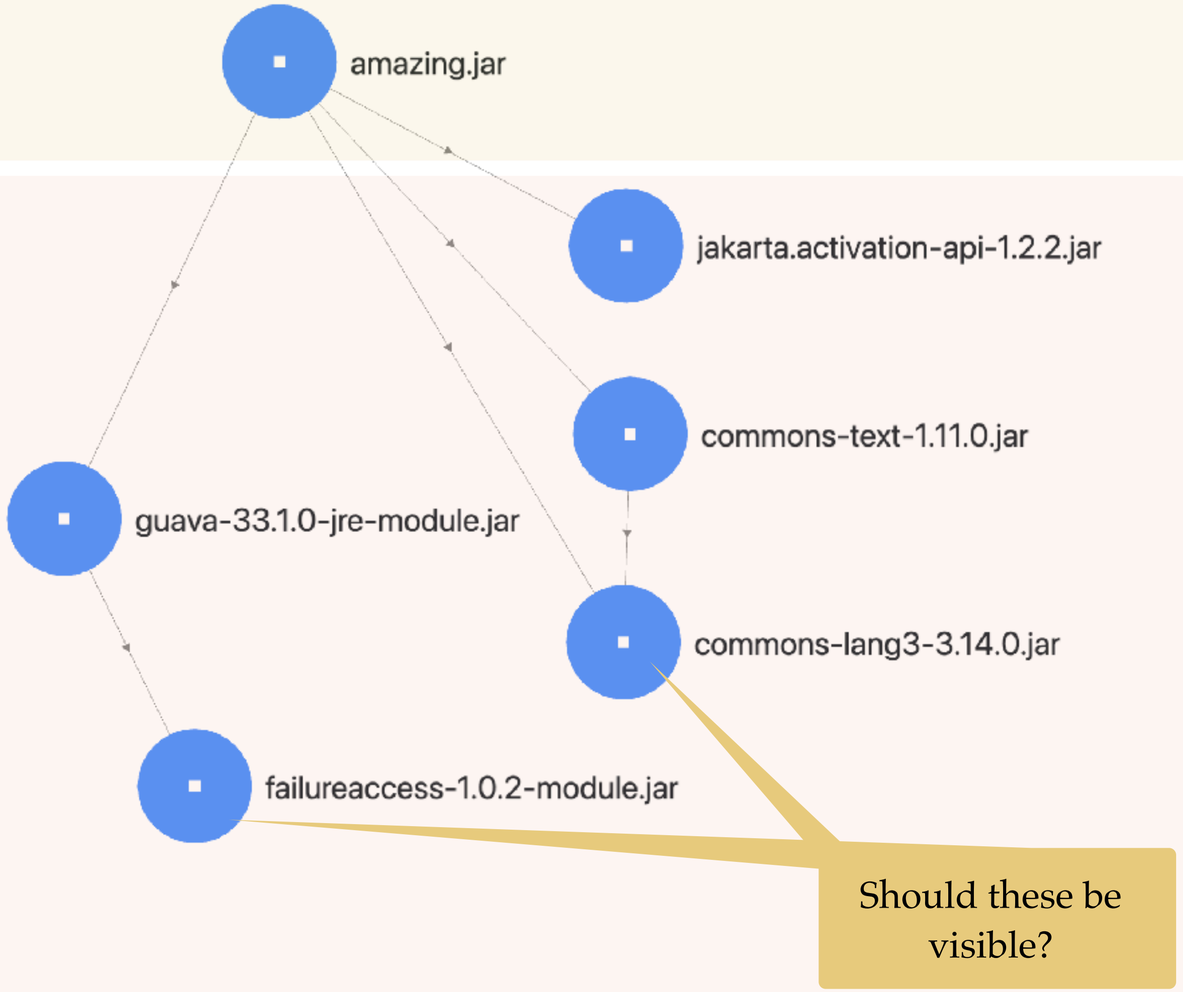
JVM Runtime Module System

JVM Compile Time Module app

JVM Compile Time Module amazing

Essential Complexity in Java's Modularity #1

Scopes are important



Software Architecture

Jar Files (uncleaned)

Jar Files

JVM Runtime Classic

JVM Runtime Module System

JVM Compile Time Module app

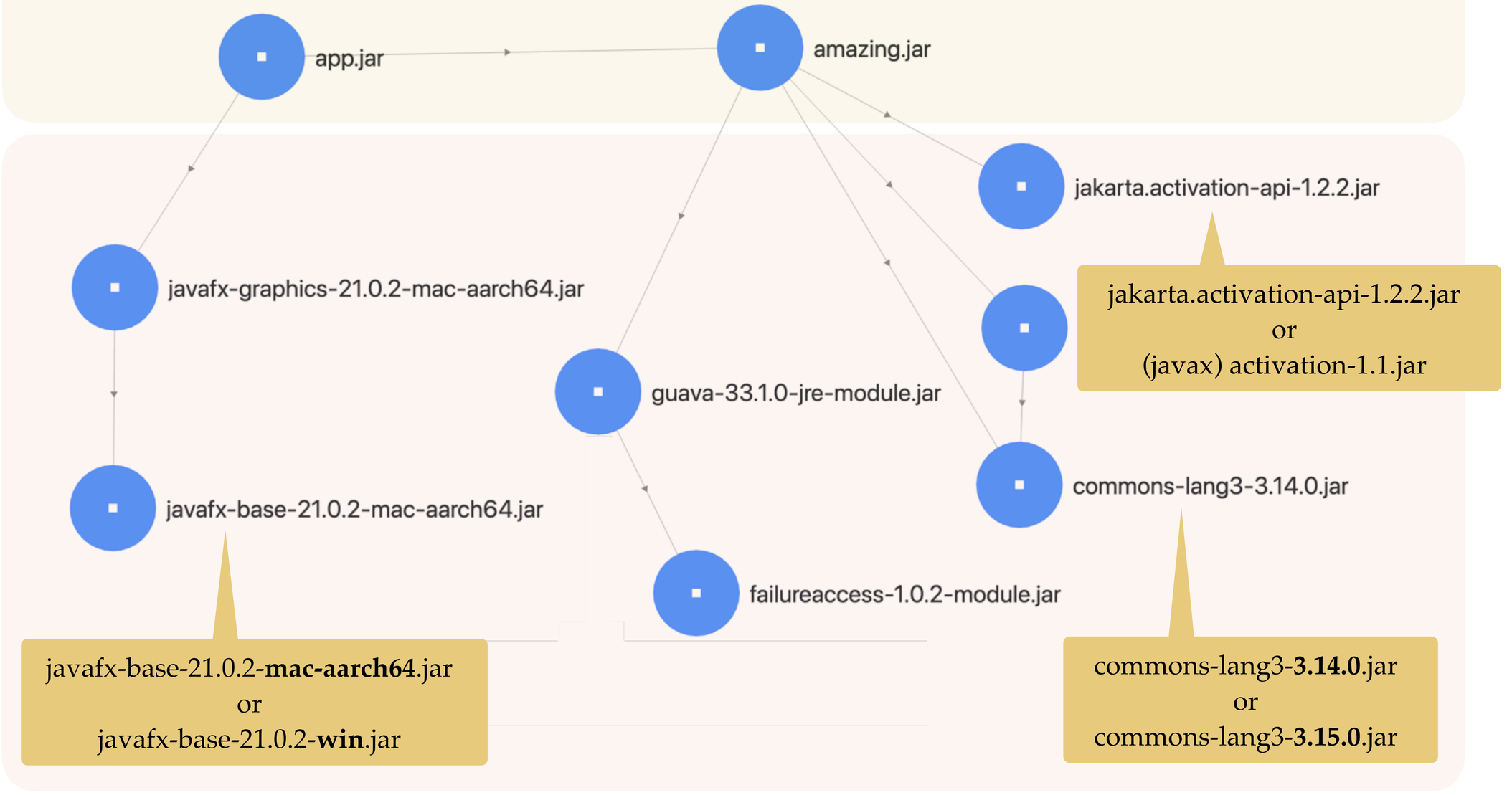
JVM Compile Time Module amazing

Essential Complexity in Java's Modularity #1

Scopes are important

Essential Complexity in Java's Modularity #2

Multiple **Versions/Variants** of 3rd party Jars



Software Architecture

Jar Files (uncleaned)

Jar Files

JVM Runtime Classic

JVM Runtime Module System

JVM Compile Time Module app

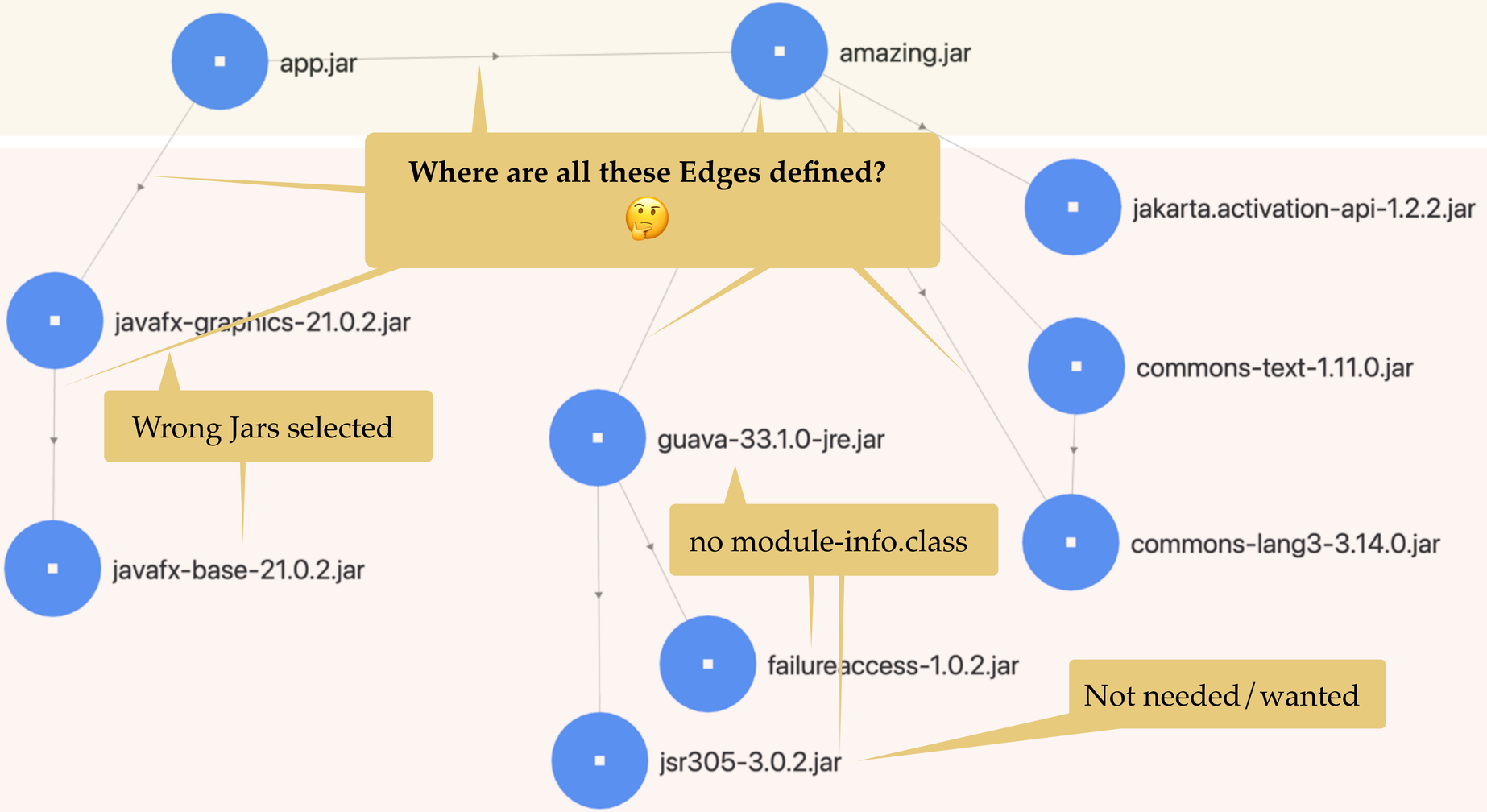
JVM Compile Time Module amazing

Essential Complexity in Java's Modularity #1

Scopes are important

Essential Complexity in Java's Modularity #2

Multiple **Versions/Variants** of 3rd party Jars



Essentia

Software Architecture

Jar Files (uncleaned)

Jar Files

JVM Runtime Classic

JVM Runtime Module System

JVM Compile Time Module app

JVM Compile Time Module amazing

Maven Repository: org.apach

mvnrepository.com/artifact/org.apache.commons/commons-text/1.12.0

Home » org.apache.commons » commons-text » 1.12.0

Apache Commons Text » 1.12.0

The Commons Text library provides additions to the standard JDK text handling. It includes algorithms for string similarity and for calculating the distance between strings.

License	Apache 2.0
Categories	String Utilities
Tags	text string apache commons
HomePage	https://commons.apache.org/proper/commons-text
Date	Apr 16, 2024
Files	pom (19 KB) jar (245 KB) View All
Repositories	Central
Ranking	#144 in MvnRepository (See Top Artifacts) #1 in String Utilities
Used By	3,599 artifacts

Maven | Gradle | Gradle (Short) | Gradle (Kotlin) | SBT | Ivy | Grape | Leiningen | Buildr

```
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-text</artifactId>
    <version>1.12.0</version>
</dependency>
```

Scope?

Accidental Complexity #1

Scopes not cared about (enough)

Software Architecture

Jar Files (uncleaned)

Jar Files

JVM Runtime Classic

JVM Runtime Module System

JVM Compile Time Module app

JVM Compile Time Module amazing

Essentia

Maven Repository: org.apach

mvnrepository.com/artifact/org.apache.commons/commons-text/1.12.0

Home » org.apache.commons » commons-text » 1.12.0

Apache Commons Text » 1.12.0

The Commons Text library provides additions to the standard JDK text handling. It includes algorithms for string similarity and for calculating the distance between strings.

License	Apache 2.0
Categories	String Utilities
Tags	text string apache commons
HomePage	https://commons.apache.org/proper/commons-text
Date	Apr 16, 2024
Files	pom (19 KB) jar (245 KB) View All
Repositories	Central
Ranking	#144 in MvnRepositor #1 in String Util
Used By	3,599 a acts

Maybe better take this one? 🤓

…or this? 🙈

Maven | Gradle | Gradle (Short) | Gradle (Kotlin) | SBT | Ivy | Grape | Leiningen | Buildr

```
implementation("org.apache.commons:commons-text:1.12.0")
```

api? runtimeOnly?

Accidental Complexity #1

Scopes not cared about (enough)

github.com/hibernate/hibernate-orm/blob/main/hibernate-core/hibernate-core.gradle

~300 Lines total
~20 Lines Module Definition

```
30    dependencies {
31        api jakartaLibs.jpa
32        api jakartaLibs.jta
33
34        implementation libs.hcann
35        implementation libs.jandex
36        implementation libs.classmate
37        implementation libs.byteBuddy
38
39        implementation jakartaLibs.jaxbApi
40        implementation jakartaLibs.jaxb
41        implementation jakartaLibs.inject
42
43        implementation libs.antlrRuntime
```

```
30      <dependencies>
31
32        <dependency>
33          <groupId>org.slf4j</groupId>
34          <artifactId>slf4j-api</artifactId>
35        </dependency>
36
37        <dependency>
38          <groupId>org.apache.hadoop</groupId>
39          <artifactId>hadoop-auth</artifactId>
40        </dependency>
41
42        <dependency>
43          <groupId>org.apache.hadoop</groupId>
44          <artifactId>hadoop-annotations</artifactId>
45        </dependency>
```

~300 Lines total
~100 Line Module Definition

github.com/apache/hadoop/blob/trunk/hadoop-common-project/hadoop-registry/pom.xml

jsr305-3.0.2.jar

Accidental Complexity #1
Scopes
not cared about (enough)

Accidental Complexity #2
Module definition mixed
with other build concerns

Software Architecture

Jar Files (uncleaned)

Jar Files

JVM Runtime Classic

JVM Runtime Module System

JVM Compile Time Module app

JVM Compile Time Module amazing

Essential Complexity in Java's Modularity #1

Scopes are important

Essential Complexity in Java's Modularity #2

Multiple **Versions/Variants** of 3rd party Jars

repo1.maven.org/maven2/org/openjfx/javafx-graphics/21.0.2/

```
org/openjfx/javafx-graphics/21.0.2/
├── javafx-graphics-21.0.2.pom
├── javafx-graphics-21.0.2.module
├── javafx-graphics-21.0.2.jar
├── javafx-graphics-21.0.2-linux.jar
├── javafx-graphics-21.0.2-mac.jar
└── javafx-graphics-21.0.2-win.jar
```

No rich metadata beyond POM 4.0.0

Empty Jar to get Maven to do a OS-based selection
-> such hacks shouldn't be needed
-> does not work for Gradle

jsr305-3.0.2.jar

Accidental Complexity #1

Scopes not cared about (enough)

Accidental Complexity #2

Module definition mixed with other build concerns

Accidental Complexity #3

Multiple Version/Variants not cared about (enough)

Software Architecture

Jar Files (uncleaned)

Jar Files

JVM Runtime Classic

JVM Runtime Module System

JVM Compile Time Module app

JVM Compile Time Module amazing

Essential Complexity in Java's Modularity #1

Scopes are important

Essential Complexity in Java's Modularity #2

Multiple **Versions/Variants** of 3rd party Jars

repo1.maven.org/maven2/org/openjfx/javafx-graphics/21.0.2/

```
org/openjfx/javafx-graphics/21.0.2/
├── javafx-graphics-21.0.2.pom
├── javafx-graphics-21.0.2.module
├── javafx-graphics-21.0.2.jar
├── javafx-graphics-21.0.2-linux.jar
├── javafx-graphics-21.0.2-mac.jar
└── javafx-graphics-21.0.2-win.jar
```

No rich metadata beyond POM 4.0.0

Empty Jar to get Maven to do a OS-based selection
-> such hacks shouldn't be needed
-> does not work for Gradle

jsr305-3.0.2.jar

Accidental Complexity #1

Scopes not cared about (enough)

Accidental Complexity #2

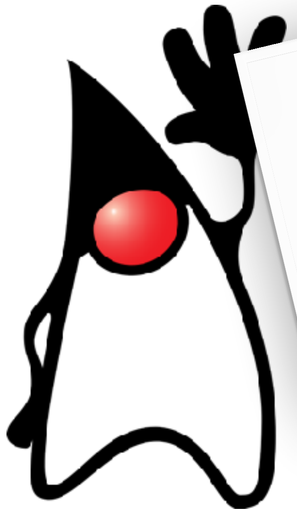
Module definition mixed with other build concerns

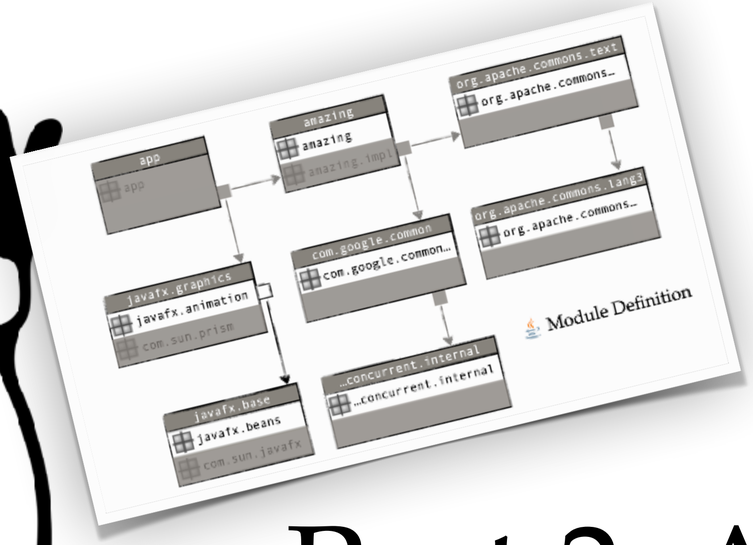
Accidental Complexity #3

Multiple Version/Variants not cared about (enough)

Accidental Complexity #4

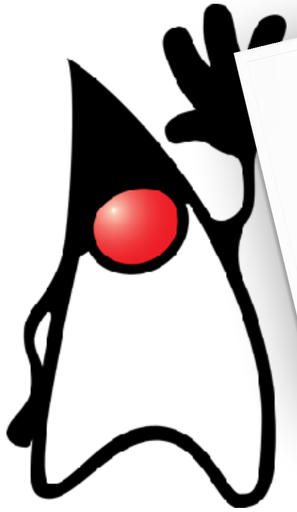
Metadata of 3rd party modules is incomplete or wrong

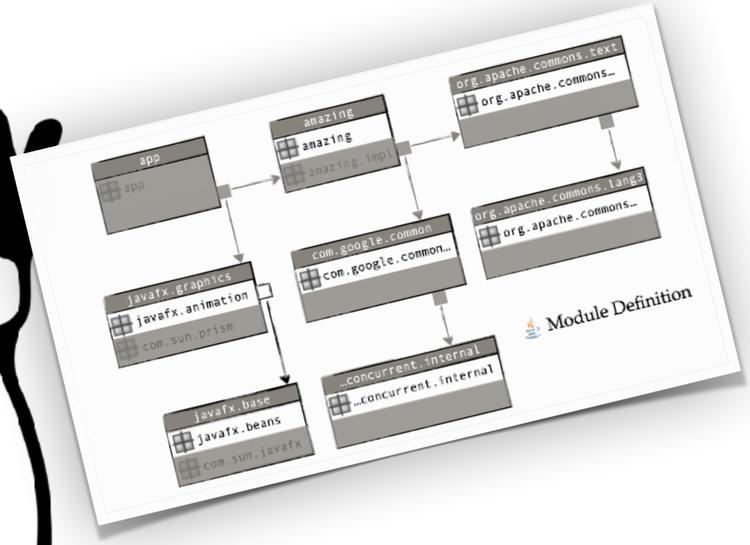


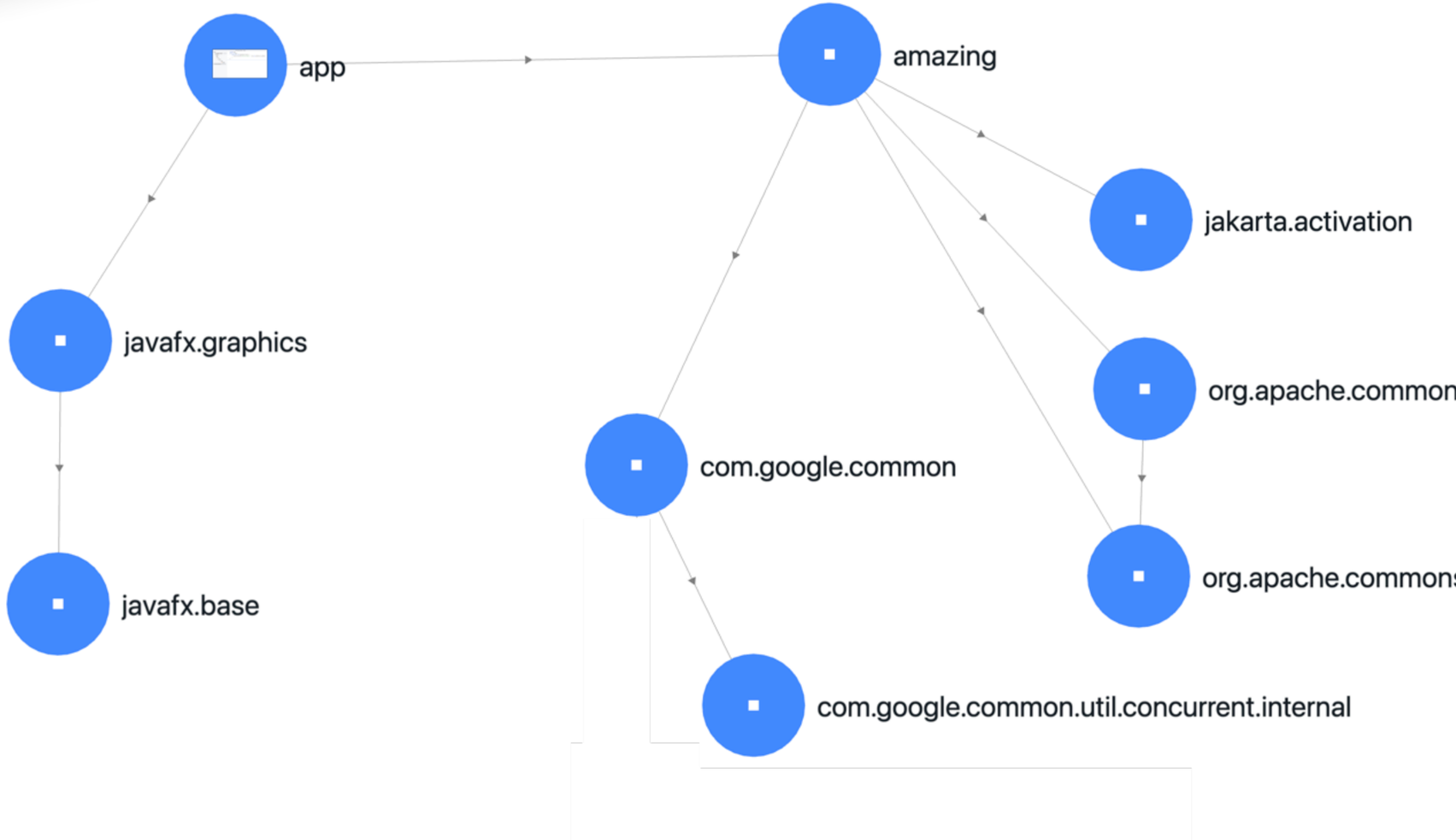


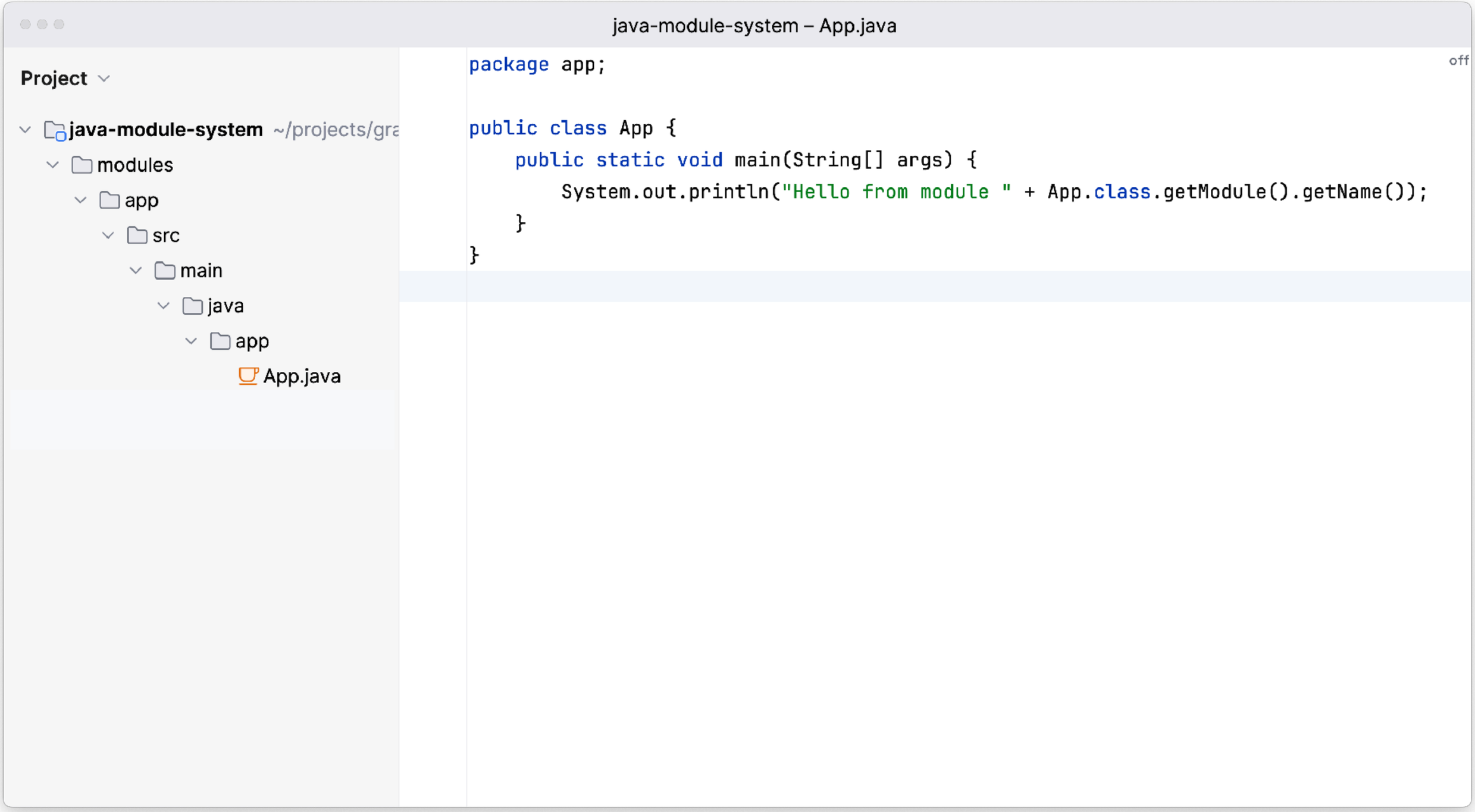
Part 2: A New Approach

Using Java for **Module Definition**




app
app
amazing
amazing
amazing.impl
org.apache.commons.text
org.apache.commons…
javafx.graphics
javafx.animation
com.sun.prism
com.google.common
com.google.common…
org.apache.commons.lang3
org.apache.commons…
javafx.base
javafx.beans
com.sun.javafx
…concurrent.internal
…concurrent.internal
Module Definition


app
amazing
jakarta.activation
javafx.graphics
org.apache.common
com.google.common
javafx.base
org.apache.commons
com.google.common.util.concurrent.internal


java-module-system – App.java
Project
java-module-system ~/projects/gra
modules
app
src
main
java
app
App.java
package app;
public class App {
public static void main(String[] args) {
System.out.println("Hello from module " + App.class.getModule().getName());
}
}

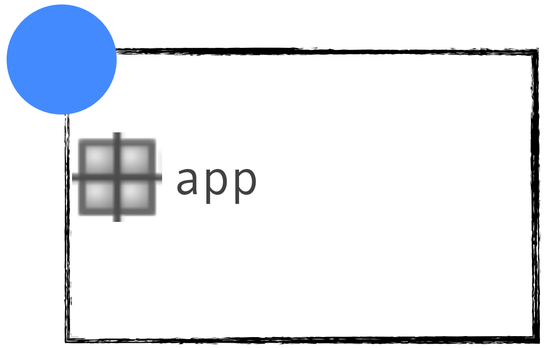
java-module-system – App.java

Project

- java-module-system ~/projects/gra
  - modules
    - app
      - src
        - main
          - java
            - app
              - App.java

```java
package app;

public class App {
    public static void main(String[] args) {
        System.out.println("Hello from module " + App.class.getModule().getName());
    }
}
```



java-module-system – DefaultAmazingService.java

Project

- java-module-system ~/projects/gradle/howto/java-m
  - modules
    - amazing
      - src
        - main
          - java
            - amazing
              - impl
                - DefaultAmazingService.java
              - AmazingService.java
    - app
      - src
        - main
          - java
            - app
              - App.java

```java
package amazing.impl;

import amazing.AmazingService;

public class DefaultAmazingService implements AmazingService {
    @Override
    public void compute() {
        System.out.println("Hello from module " +
                getClass().getModule().getName());
    }
}
```

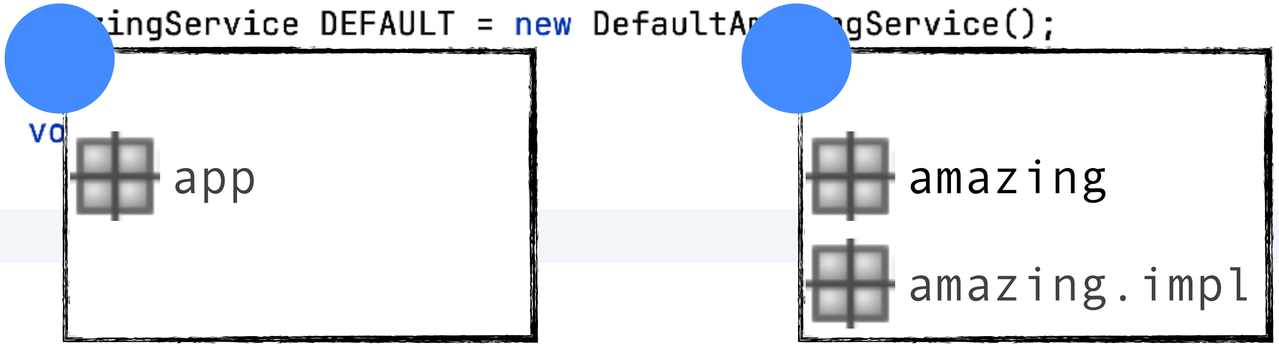
```java
package amazing;

import amazing.impl.DefaultAmazingService;

public interface AmazingService {
    ingService DEFAULT = new DefaultA gService();

    vo

}
```



java-module-system – App.java

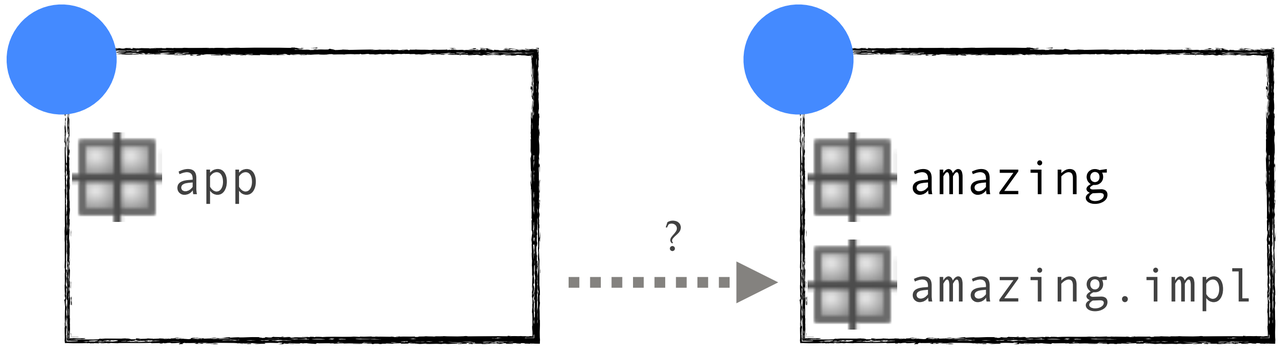
Project

- java-module-system ~/projects/gradle/howto/java-m
  - modules
    - amazing
      - src
        - main
          - java
            - amazing
              - impl
                - DefaultAmazingService.java
              - AmazingService.java
    - app
      - src
        - main
          - java
            - app
              - App.java

```java
package app;

import amazing.AmazingService;

public class App {
    public static void main(String[] args) {
        System.out.println("Hello from module " +
                App.class.getModule().getName());
        AmazingService.DEFAULT.compute();
    }
}
```

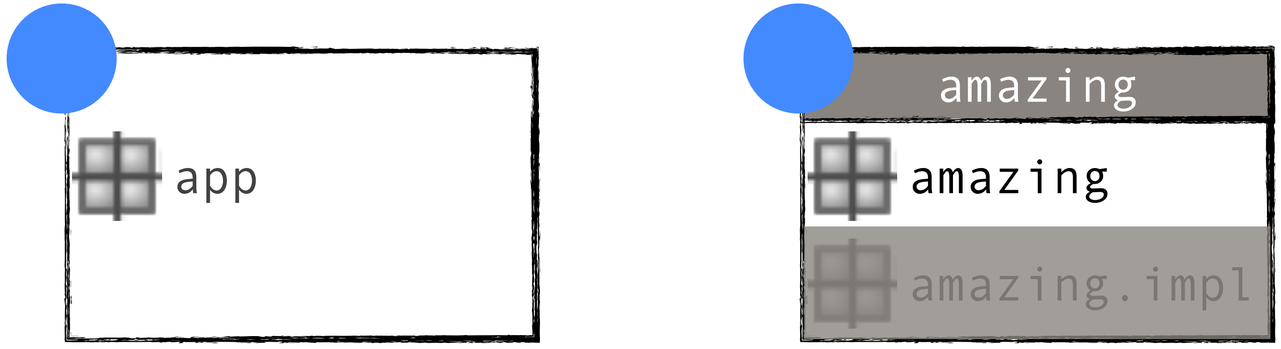


java-module-system – module-info.java (amazing)

Project

- java-module-system ~/projects/gradle/howto/java-m
  - modules
    - amazing
      - src
        - main
          - java
            - amazing
              - impl
                - DefaultAmazingService.java
              - AmazingService.java
            - module-info.java
    - app
      - src
        - main
          - java
            - app
              - App.java

```
module amazing {
    exports amazing;
}
```



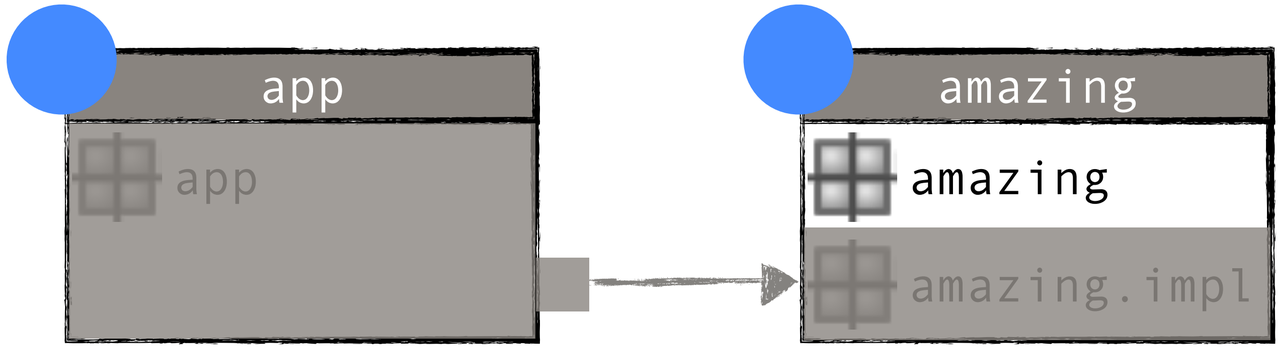
java-module-system – module-info.java (amazing)

Project

- java-module-system ~/projects/gradle/howto/java-m
  - modules
    - amazing
      - src
        - main
          - java
            - amazing
              - impl
                - DefaultAmazingService.java
              - AmazingService.java
            - module-info.java
    - app
      - src
        - main
          - java
            - app
              - App.java
            - module-info.java

```
module amazing {
    exports amazing;
}
```

```
module app {
    requires amazing;
}
```



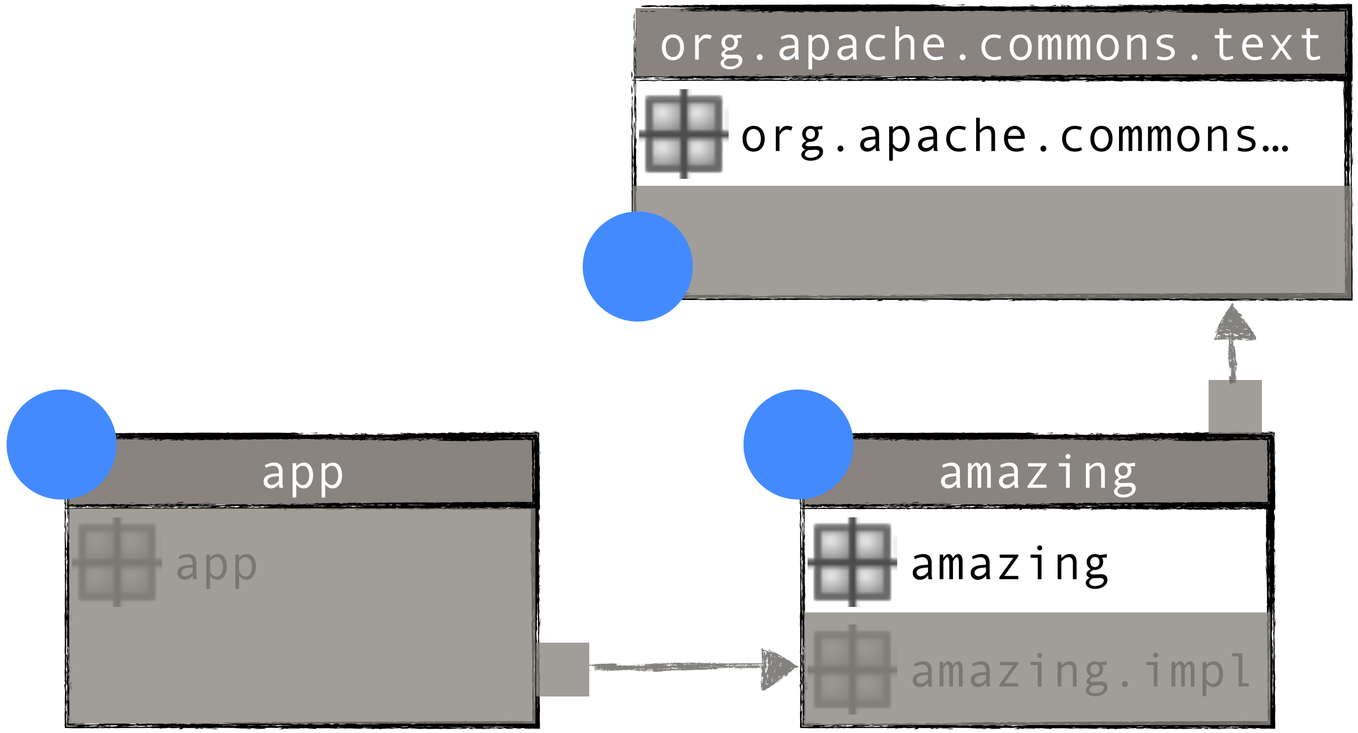
java-module-system – module-info.java (app)

Project

- java-module-system ~/projects/gradle/howto/java-m
  - modules
    - amazing
      - src
        - main
          - java
            - amazing
              - impl
                - DefaultAmazingService.java
              - AmazingService.java
              - module-info.java
    - app
      - src
        - main
          - java
            - app
              - App.java
              - module-info.java

```
module amazing {
    requires org.apache.commons.text;
    exports amazing;
}
```

```
module app {
    requires amazing;
}
```



java-module-system – module-info.java (app)

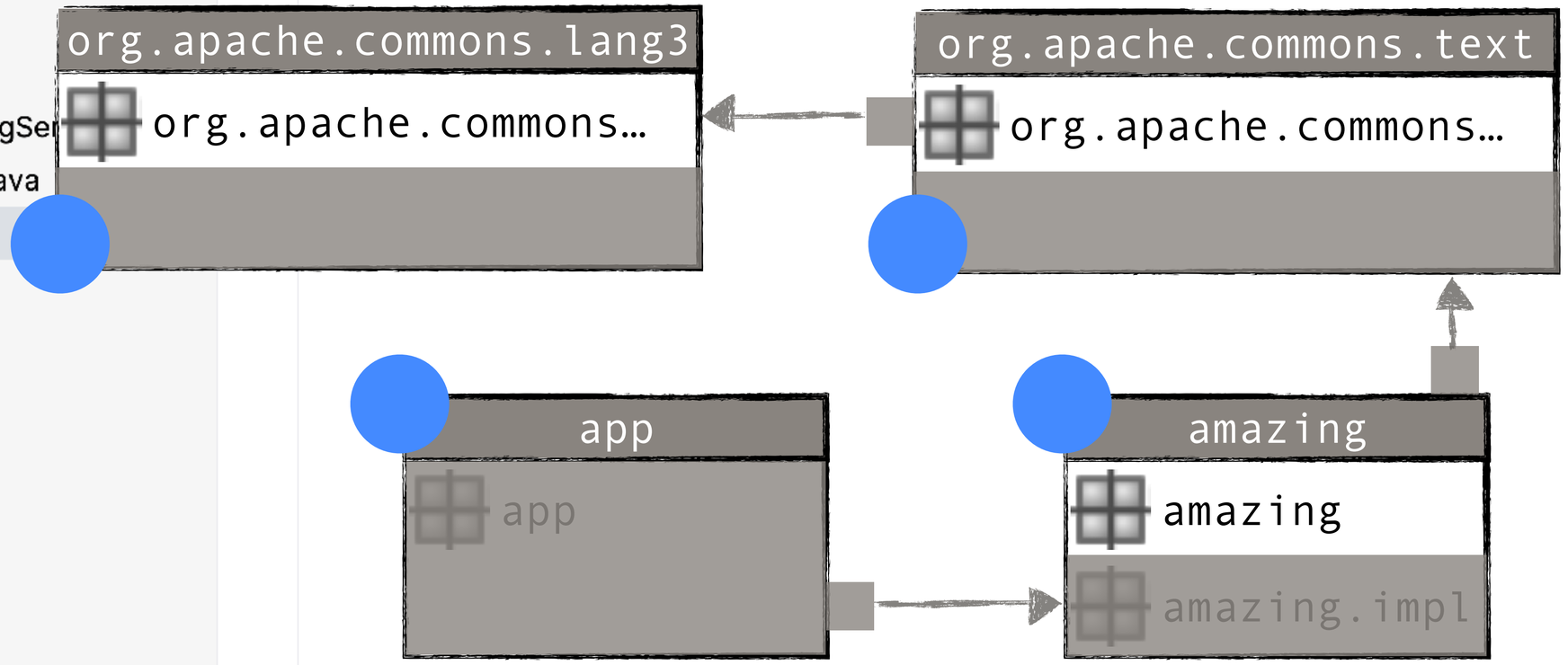
Project

- java-module-system ~/projects/gradle/howto/java-m
  - modules
    - amazing
      - src
        - main
          - java
            - amazing
              - impl
                - DefaultAmazingSer
              - AmazingService.java
            - module-info.java
    - app
      - src
        - main
          - java
            - app
              - App.java
            - module-info.java

```
module amazing {
    requires org.apache.commons.text;
    exports amazing;
}
```

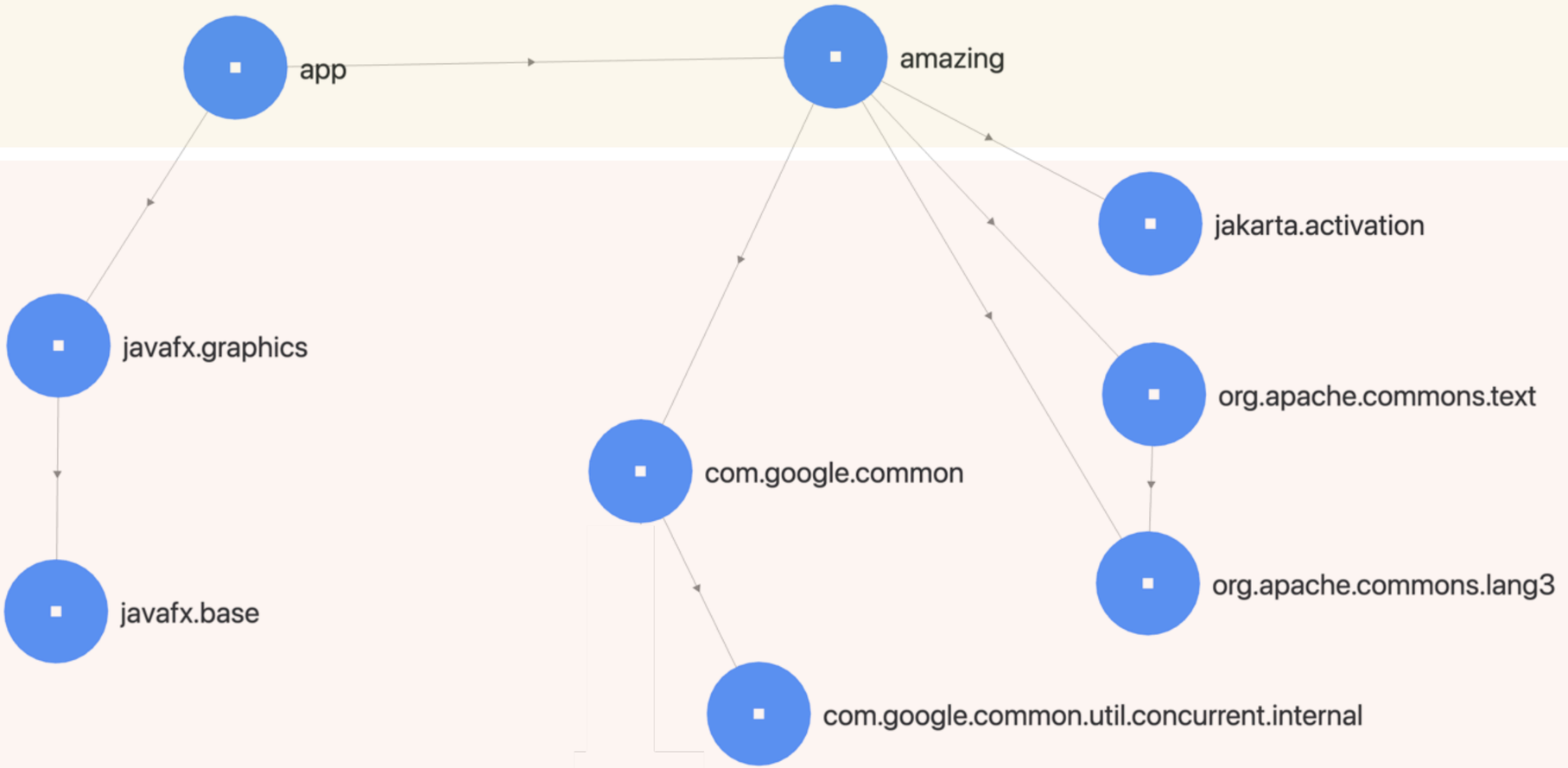
```
module app {
    requires amazing;
}
```

Scopes
- requires
- requires transitive
- requires static



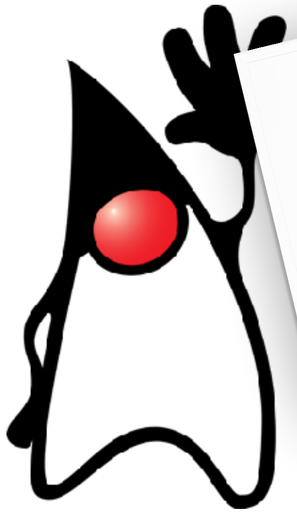
Essential Complexity in Java's Modularity #1

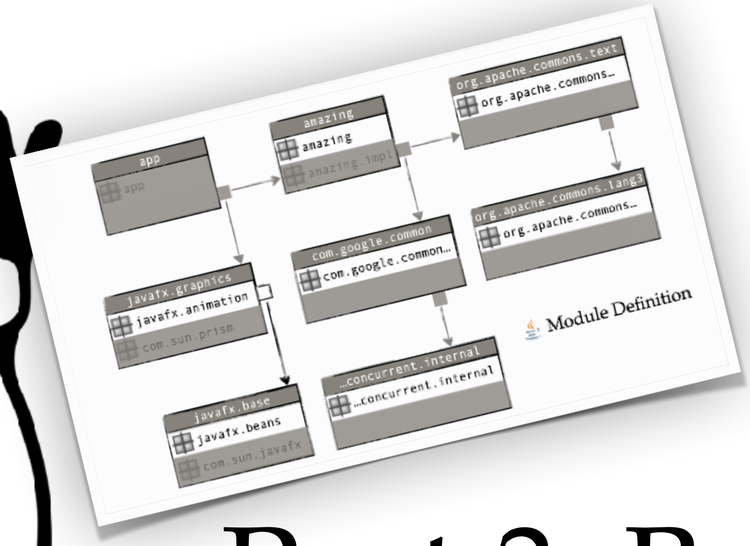
Scopes are important



module-info.java
- strict standard Syntax
- javac an java enforce it

Module Definition
only (!) in **module-info.java**





Part 3: Best of Both Worlds

Using Gradle **not (!)** for

Module Definition

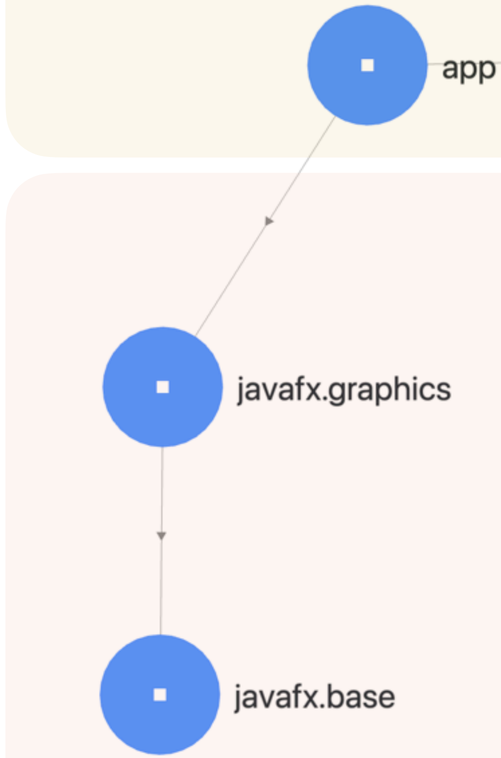
but for

Module Discovery

Essential Complexity in Java's Modularity #1

Scopes are important

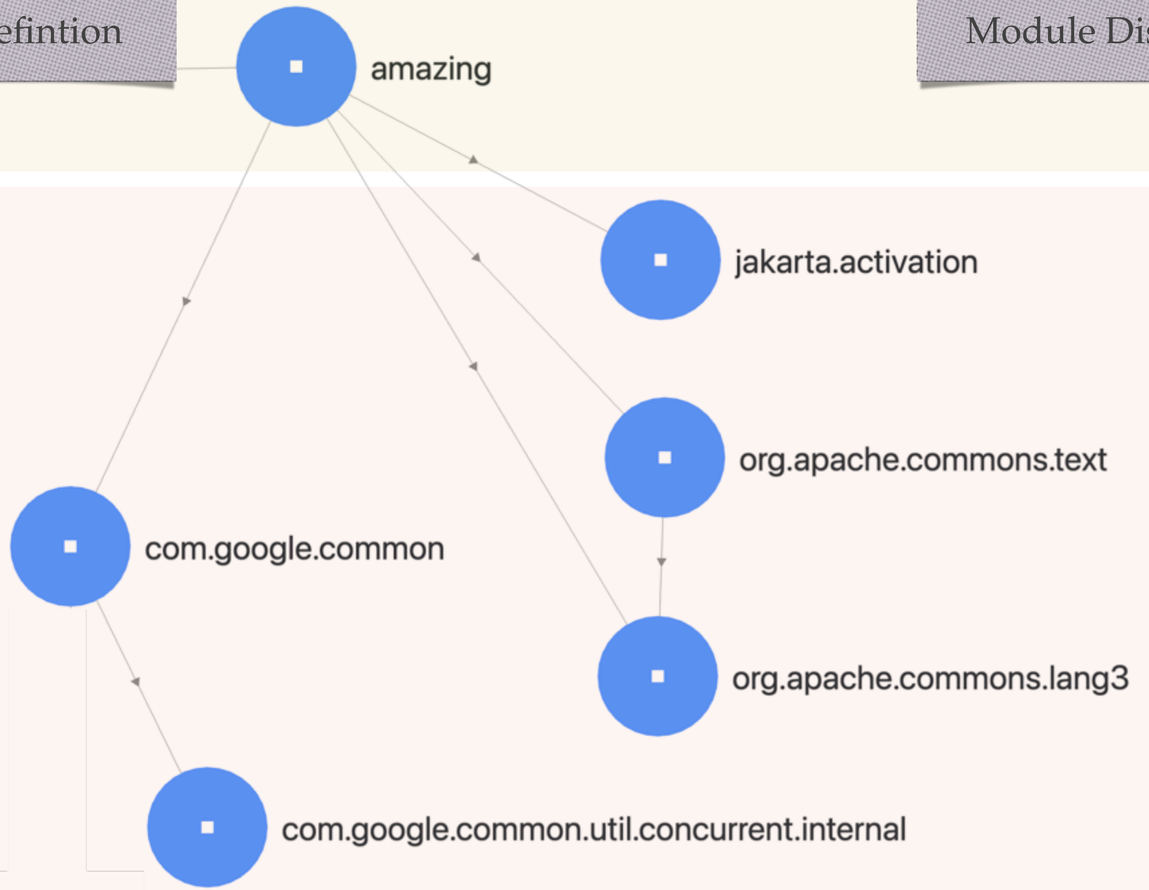
Module Defintion



Essential Complexity in Java's Modularity #2

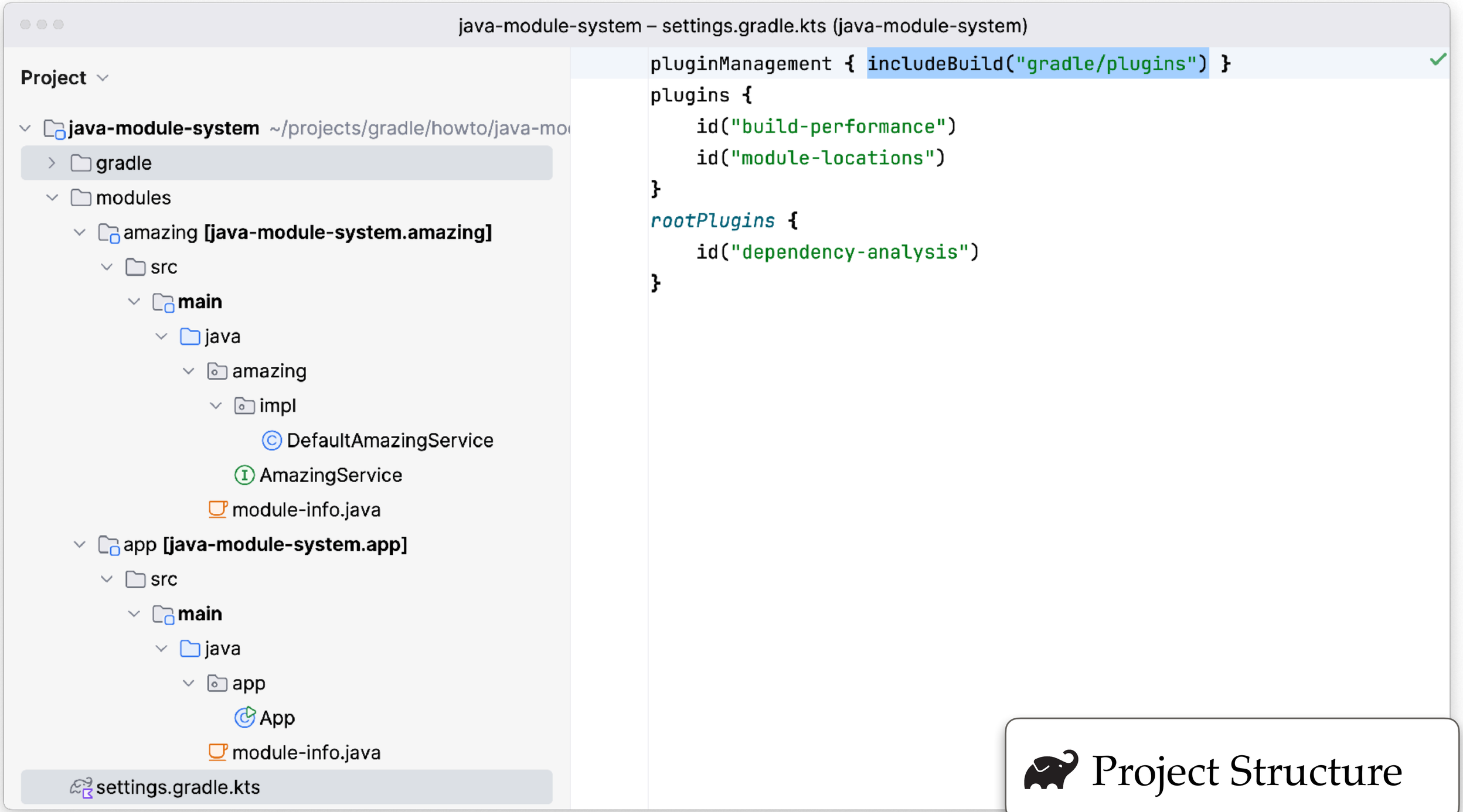
Multiple **Versions/Variants** of 3rd party Jars

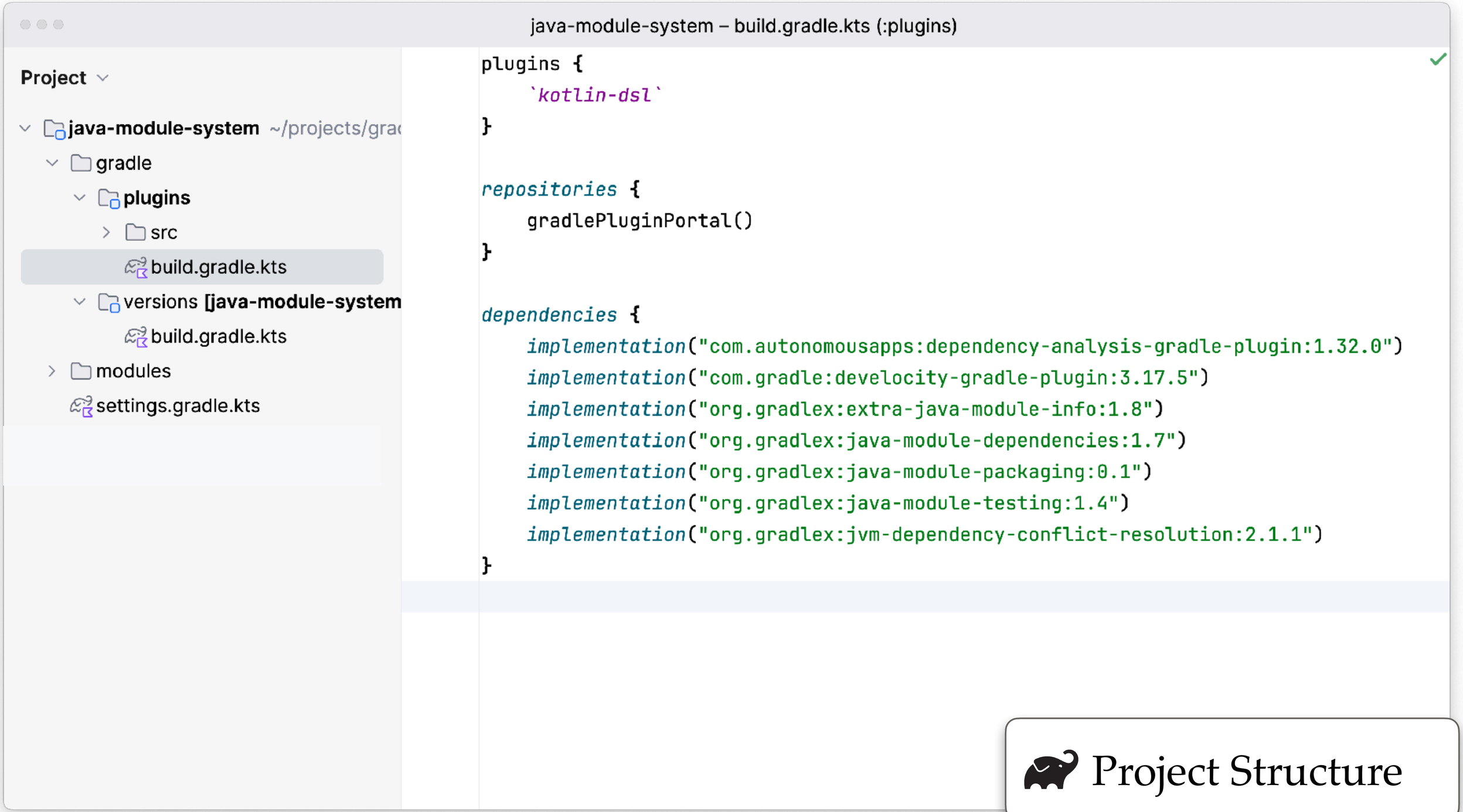
Module Discovery

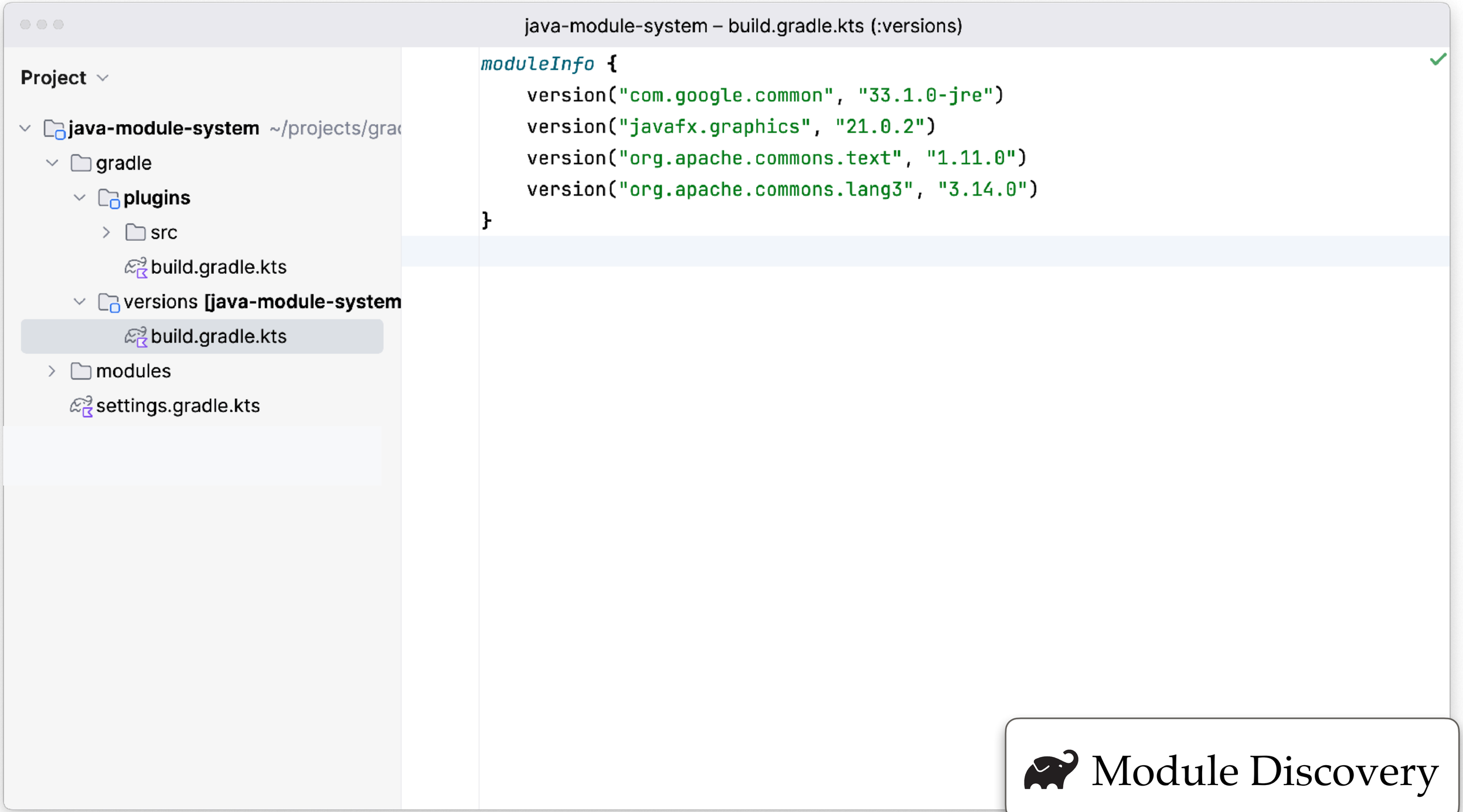


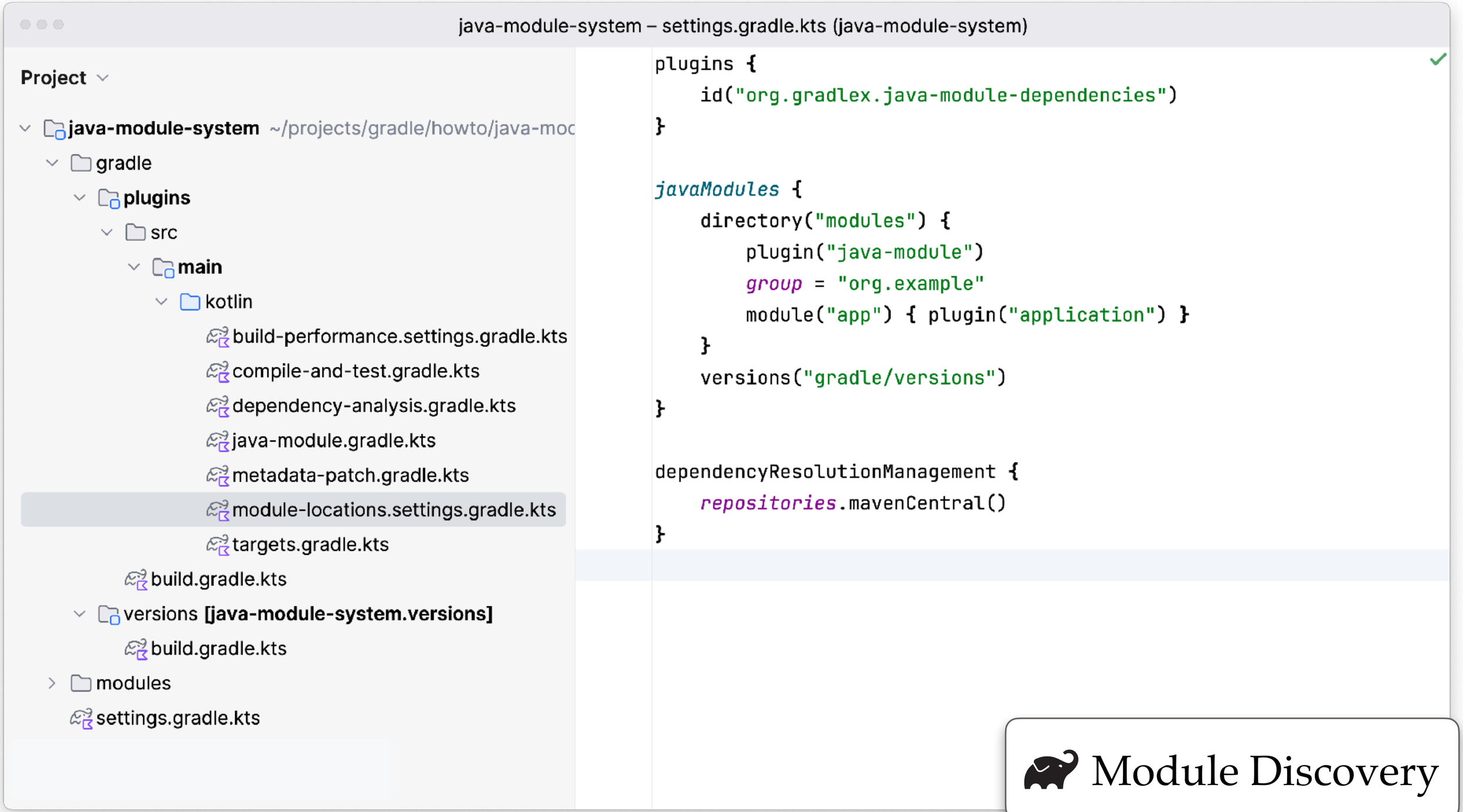
module-info.java
- strict standard Syntax
- javac an java enforce it

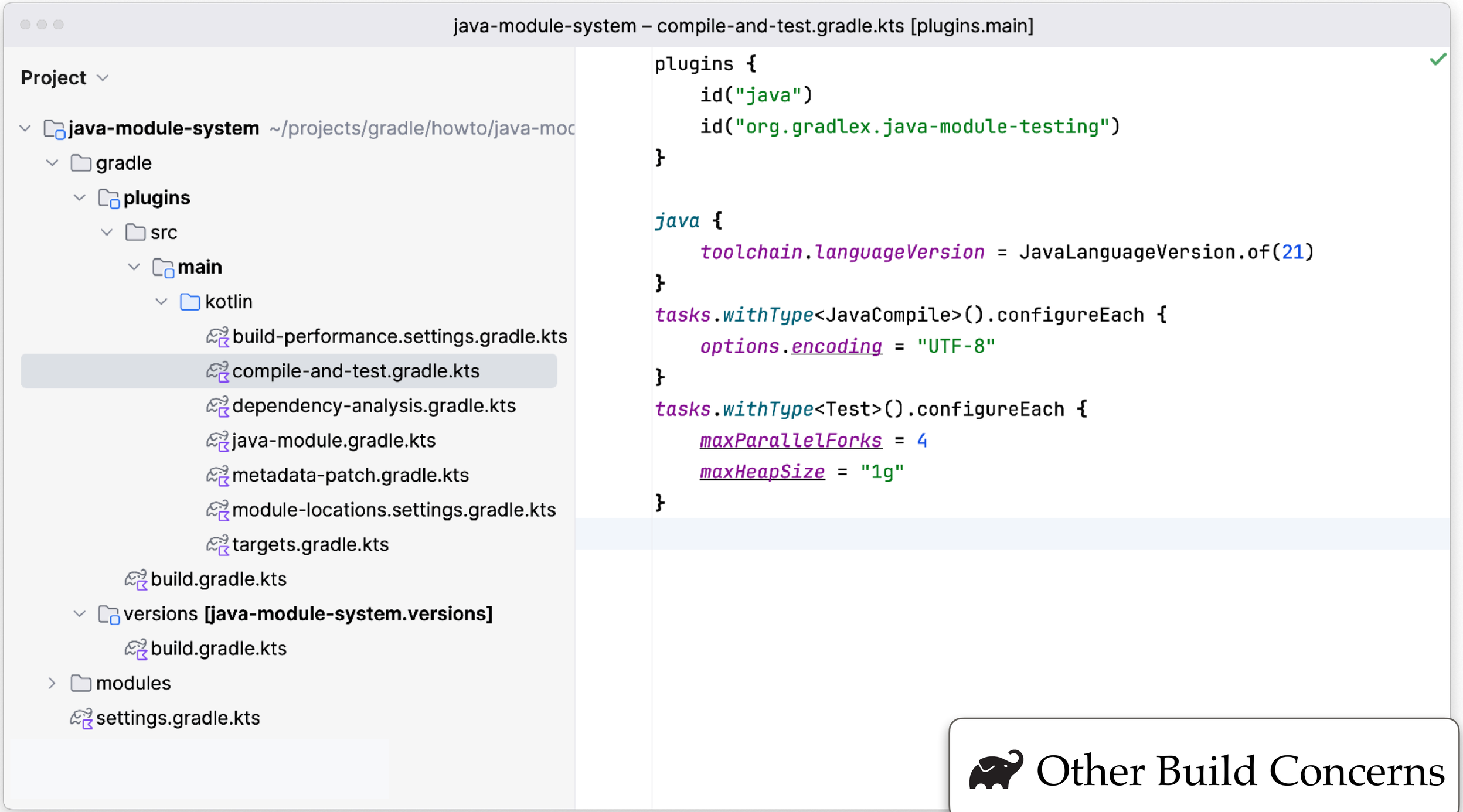
Module Definition
only (!) in **module-info.java**


java-module-system – settings.gradle.kts (java-module-system)
Project
java-module-system ~/projects/gradle/howto/java-mo
gradle
modules
amazing [java-module-system.amazing]
src
main
java
amazing
impl
DefaultAmazingService
AmazingService
module-info.java
app [java-module-system.app]
src
main
java
app
App
module-info.java
settings.gradle.kts
pluginManagement { includeBuild("gradle/plugins") }
plugins {
id("build-performance")
id("module-locations")
}
rootPlugins {
id("dependency-analysis")
}
Project Structure


java-module-system – build.gradle.kts (:plugins)
Project
java-module-system ~/projects/grad
gradle
plugins
src
build.gradle.kts
versions [java-module-system
build.gradle.kts
modules
settings.gradle.kts
plugins {
`kotlin-dsl`
}
repositories {
gradlePluginPortal()
}
dependencies {
implementation("com.autonomousapps:dependency-analysis-gradle-plugin:1.32.0")
implementation("com.gradle:develocity-gradle-plugin:3.17.5")
implementation("org.gradlex:extra-java-module-info:1.8")
implementation("org.gradlex:java-module-dependencies:1.7")
implementation("org.gradlex:java-module-packaging:0.1")
implementation("org.gradlex:java-module-testing:1.4")
implementation("org.gradlex:jvm-dependency-conflict-resolution:2.1.1")
}
Project Structure

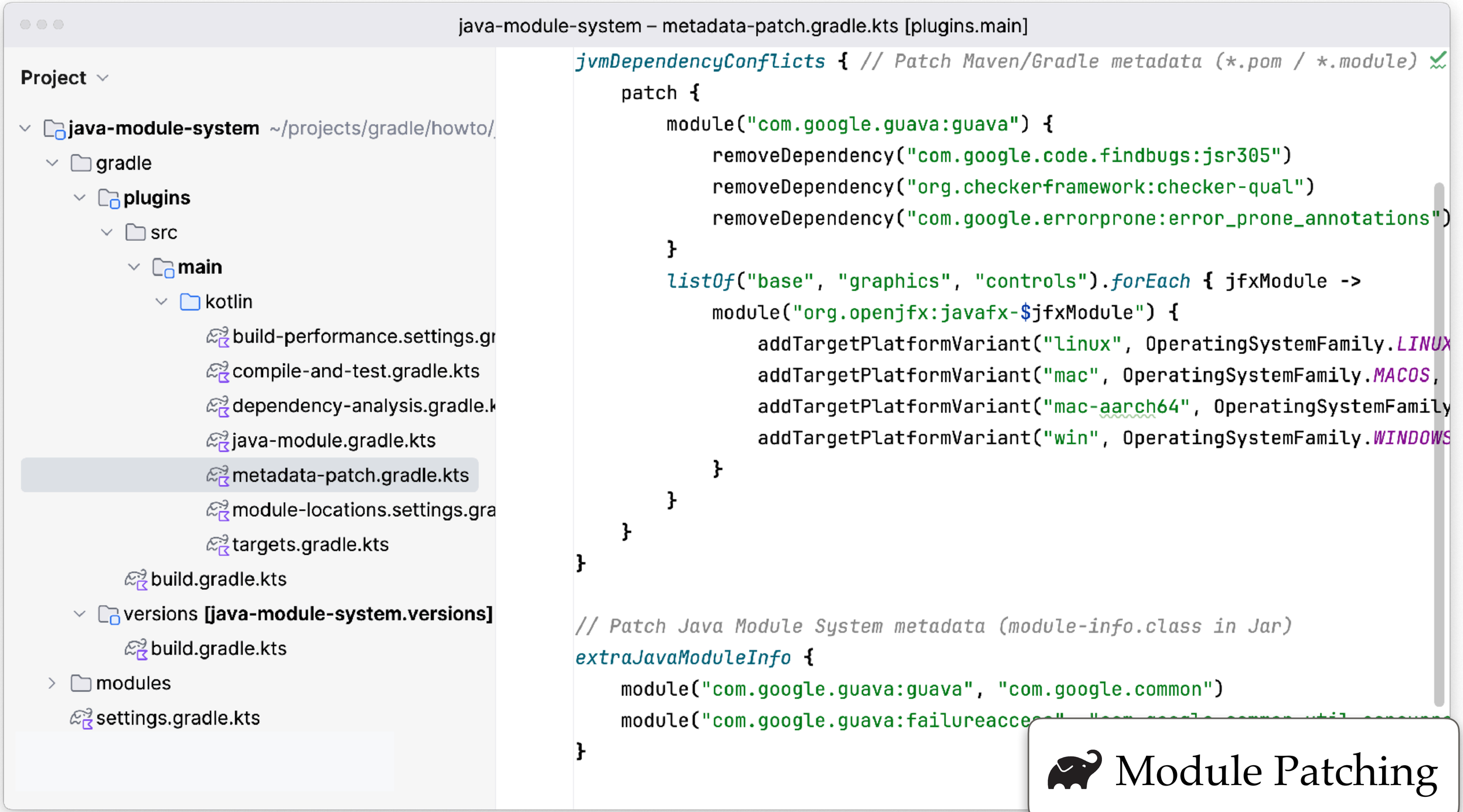

java-module-system – build.gradle.kts (:versions)
Project
java-module-system ~/projects/grad
gradle
plugins
src
build.gradle.kts
versions [java-module-system
build.gradle.kts
modules
settings.gradle.kts
moduleInfo {
version("com.google.common", "33.1.0-jre")
version("javafx.graphics", "21.0.2")
version("org.apache.commons.text", "1.11.0")
version("org.apache.commons.lang3", "3.14.0")
}
Module Discovery


java-module-system – settings.gradle.kts (java-module-system)
Project
java-module-system ~/projects/gradle/howto/java-mod
gradle
plugins
src
main
kotlin
build-performance.settings.gradle.kts
compile-and-test.gradle.kts
dependency-analysis.gradle.kts
java-module.gradle.kts
metadata-patch.gradle.kts
module-locations.settings.gradle.kts
targets.gradle.kts
build.gradle.kts
versions [java-module-system.versions]
build.gradle.kts
modules
settings.gradle.kts
plugins {
id("org.gradlex.java-module-dependencies")
}
javaModules {
directory("modules") {
plugin("java-module")
group = "org.example"
module("app") { plugin("application") }
}
versions("gradle/versions")
}
dependencyResolutionManagement {
repositories.mavenCentral()
}
Module Discovery



Other Build Concerns


java-module-system – targets.gradle.kts [plugins.main]
Project
java-module-system ~/projects/gradle/howto/java-mod
gradle
plugins
src
main
kotlin
build-performance.settings.gradle.kts
compile-and-test.gradle.kts
dependency-analysis.gradle.kts
java-module.gradle.kts
metadata-patch.gradle.kts
module-locations.settings.gradle.kts
targets.gradle.kts
build.gradle.kts
versions [java-module-system.versions]
build.gradle.kts
modules
settings.gradle.kts
plugins {
id("org.gradlex.java-module-packaging")
}
javaModulePackaging {
target("ubuntu-22.04") {
operatingSystem = OperatingSystemFamily.LINUX
architecture = MachineArchitecture.X86_64
packageTypes = listOf("deb")
}
target("macos-14") {
operatingSystem = OperatingSystemFamily.MACOS
architecture = MachineArchitecture.ARM64
packageTypes = listOf("dmg")
}
target("windows-2022") {
operatingSystem = OperatingSystemFamily.WINDOWS
architecture = MachineArchitecture.X86_64
packageTypes = listOf("exe")
}
primaryTarget(target("macos
}
OS / Arch Specifics


java-module-system – metadata-patch.gradle.kts [plugins.main]
Project
java-module-system ~/projects/gradle/howto/
gradle
plugins
src
main
kotlin
build-performance.settings.gr
compile-and-test.gradle.kts
dependency-analysis.gradle.k
java-module.gradle.kts
metadata-patch.gradle.kts
module-locations.settings.gra
targets.gradle.kts
build.gradle.kts
versions [java-module-system.versions]
build.gradle.kts
modules
settings.gradle.kts
jvmDependencyConflicts { // Patch Maven/Gradle metadata (*.pom / *.module)
patch {
module("com.google.guava:guava") {
removeDependency("com.google.code.findbugs:jsr305")
removeDependency("org.checkerframework:checker-qual")
removeDependency("com.google.errorprone:error_prone_annotations")
}
listOf("base", "graphics", "controls").forEach { jfxModule ->
module("org.openjfx:javafx-$jfxModule") {
addTargetPlatformVariant("linux", OperatingSystemFamily.LINUX
addTargetPlatformVariant("mac", OperatingSystemFamily.MACOS,
addTargetPlatformVariant("mac-aarch64", OperatingSystemFamily
addTargetPlatformVariant("win", OperatingSystemFamily.WINDOWS
}
}
}
}
// Patch Java Module System metadata (module-info.class in Jar)
extraJavaModuleInfo {
module("com.google.guava:guava", "com.google.common")
module("com.google.guava:failureacces
}
Module Patching

Essential Complexity in Java's Modularity #1

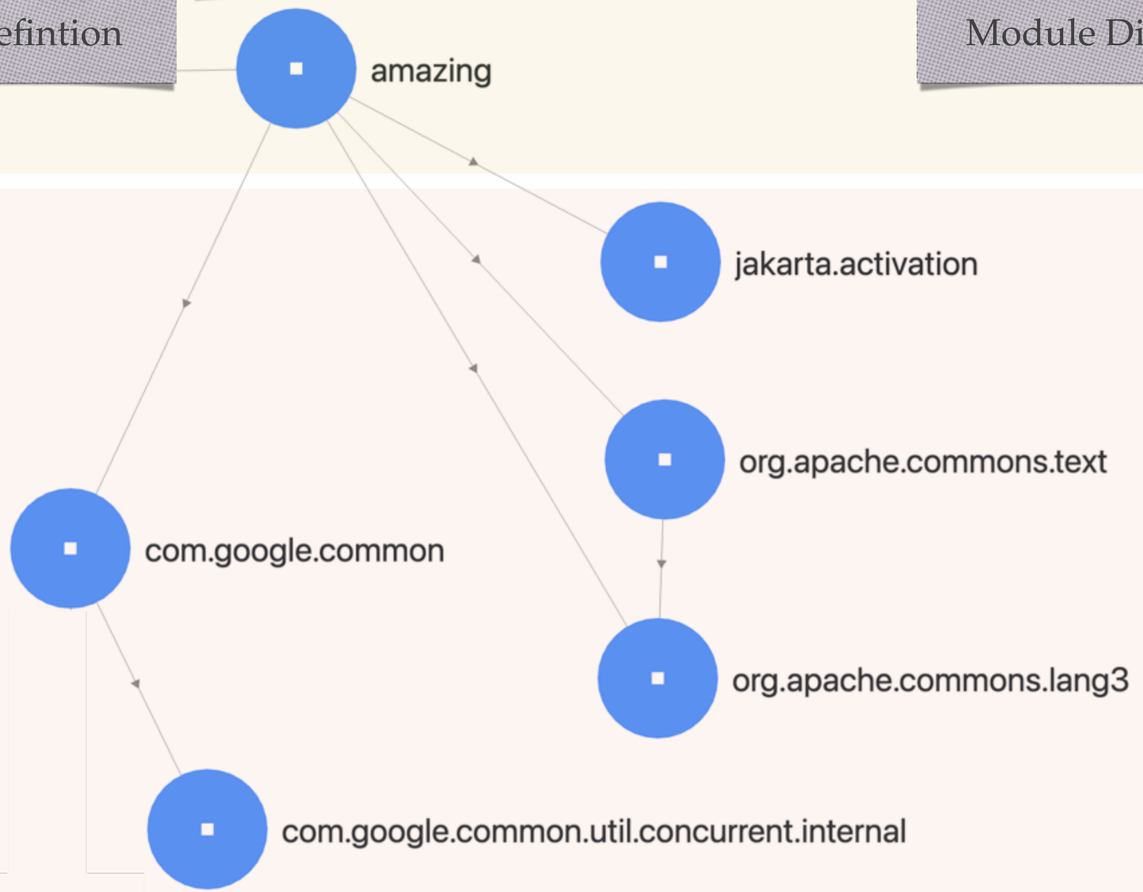
Scopes are important

Module Defintion

Essential Complexity in Java's Modularity #2

Multiple **Versions/Variants** of 3rd party Jars

Module Discovery

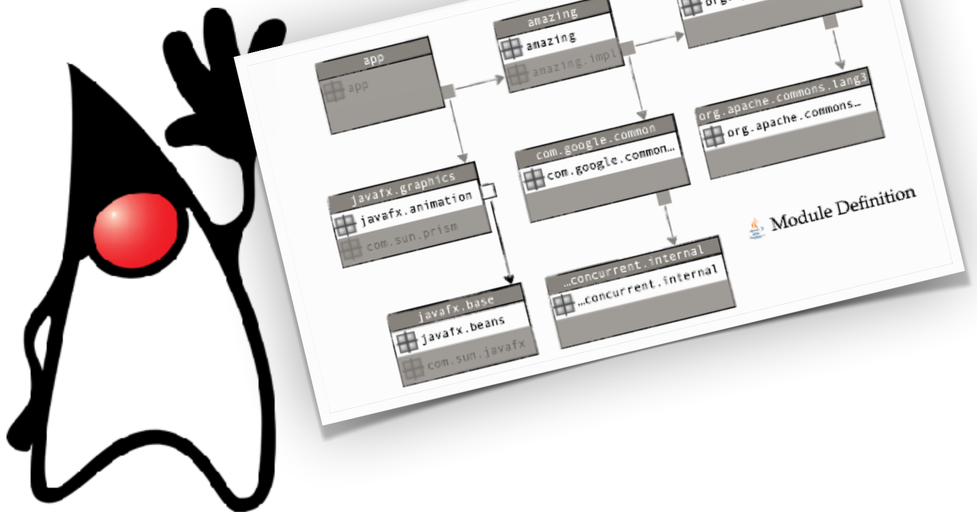


module-info.java
- strict standard Syntax
- javac an java enforce it

Module Definition
only (!) in **module-info.java**

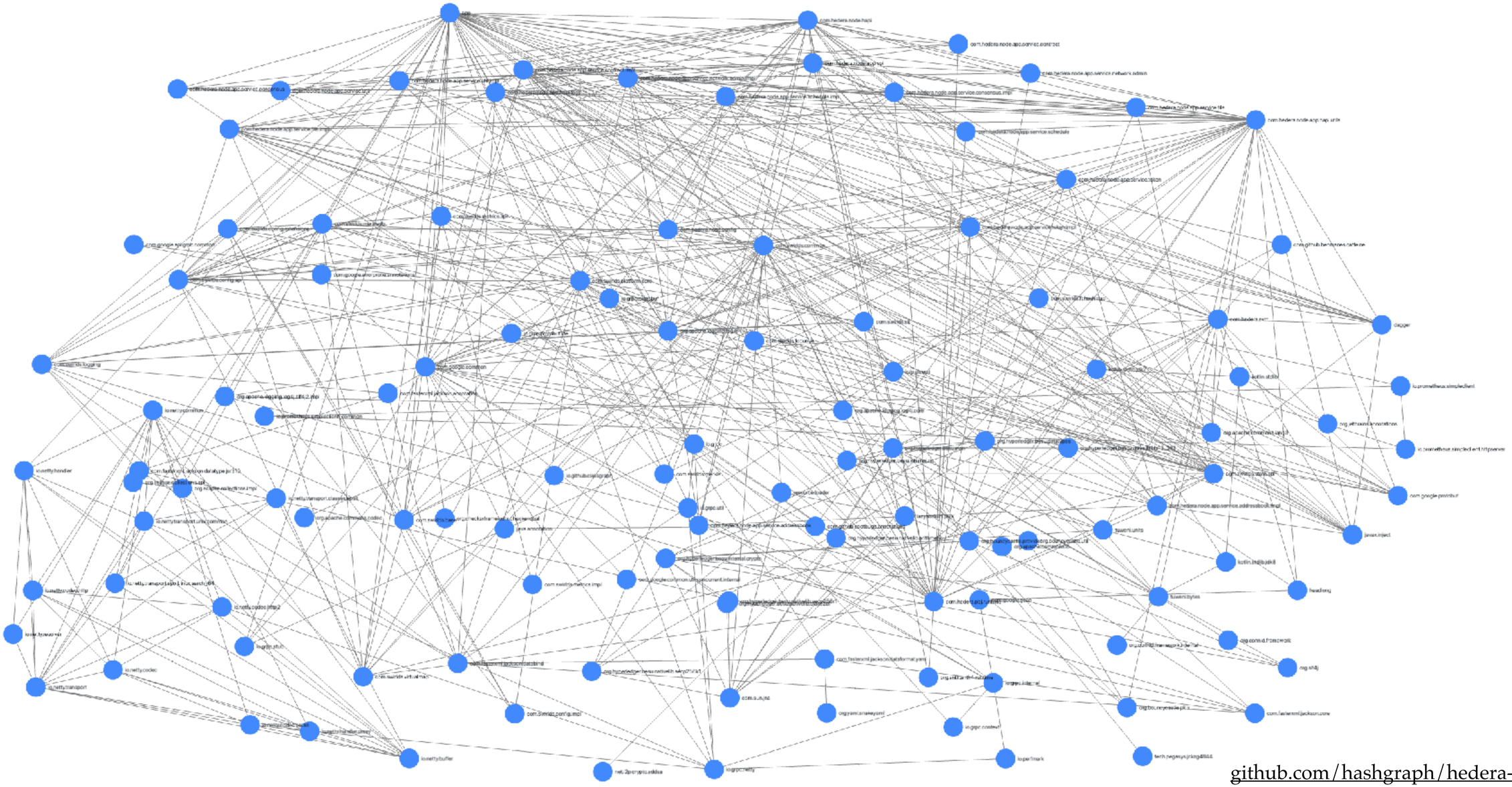
Use Gradle's **variant-aware** conflict detection and resolution

Central patch file(s) to enrich / correct metadata


app
app
amazing
amazing
amazing.impl
org.apache.commons.text
org.apache.commons…
org.apache.commons.lang3
org.apache.commons…
com.google.common
com.google.common…
javafx.graphics
javafx.animation
com.sun.prism
javafx.base
javafx.beans
com.sun.javafx
…concurrent.internal
…concurrent.internal
Module Definition

Summary

Real world example: Hedera Services



github.com/hashgraph/hedera-services

poloclub.github.io/argo-graph-lite/#d5b95db8-a363-46a5-8757-a846e116fa2b

Real world example: Hedera Services

github.com/hashgraph/hedera-services/blob/develop/hedera-node/hedera-consensus-service-impl/src/main/java/module-info.java

a module-info.java

github.com/hashgraph/hedera-services/blob/develop/hedera-node/hedera-consensus-service-impl/build.gradle.kts

a minimalistic build.gradle

github.com/hashgraph/hedera-services/blob/develop/hedera-dependency-versions/build.gradle.kts

versions

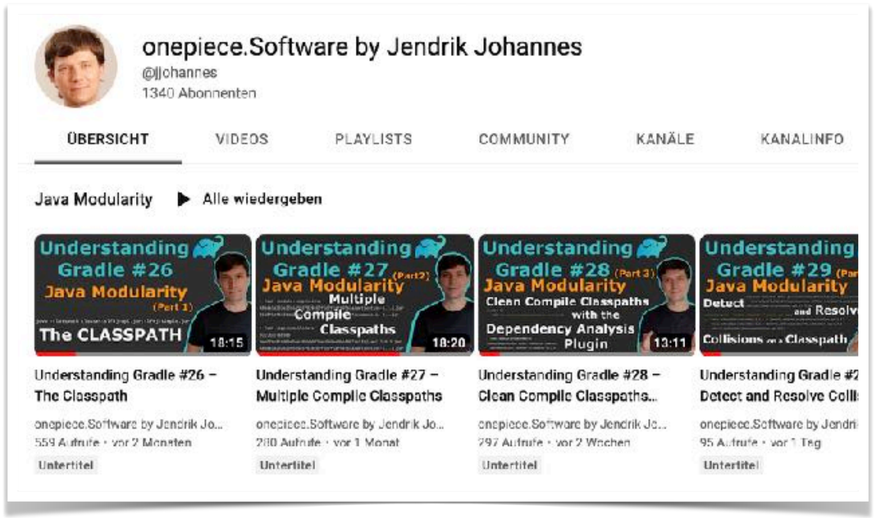
github.com/hashgraph/hedera-services/blob/develop/gradle/plugins/src/main/kotlin/com.hedera.gradle.jpms-modules.gradle.kts

3rd party patching

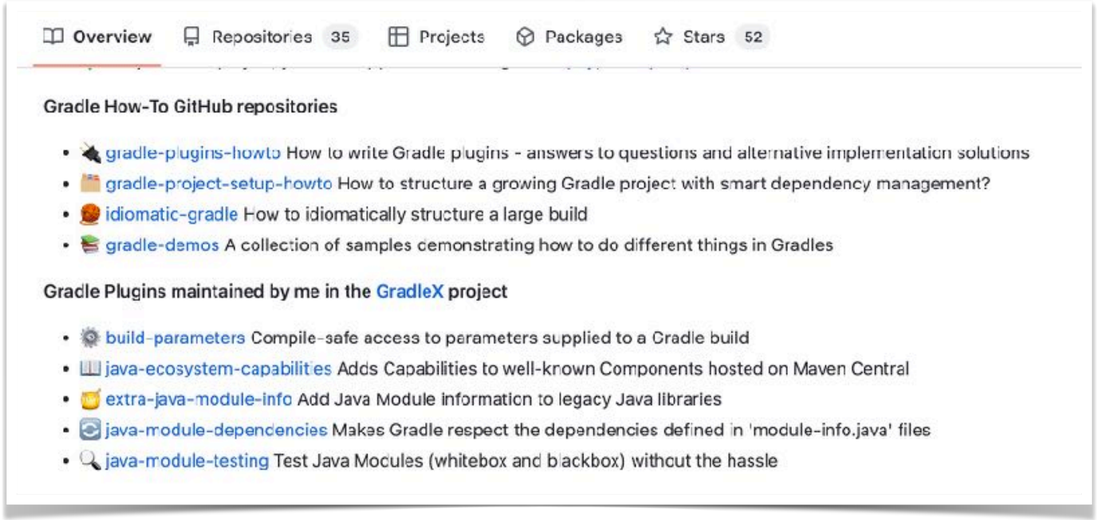
github.com/hashgraph/hedera-services/tree/develop/gradle/plugins/src/main/kotlin

other build concerns

github.com/jjohannes/java-module-system (slides / example)
github.com/jjohannes/gradle-project-setup-howto
github.com/hashgraph/hedera-services



youtube.com/@jjohannes



github.com/jjohannes

stay in touch
mastodon.social/@jendrik
linkedin.com/in/jendrikjohannes
jendrik@onepiece.software

training and consulting
onepiece.software

GradleX Project
gradlex.org