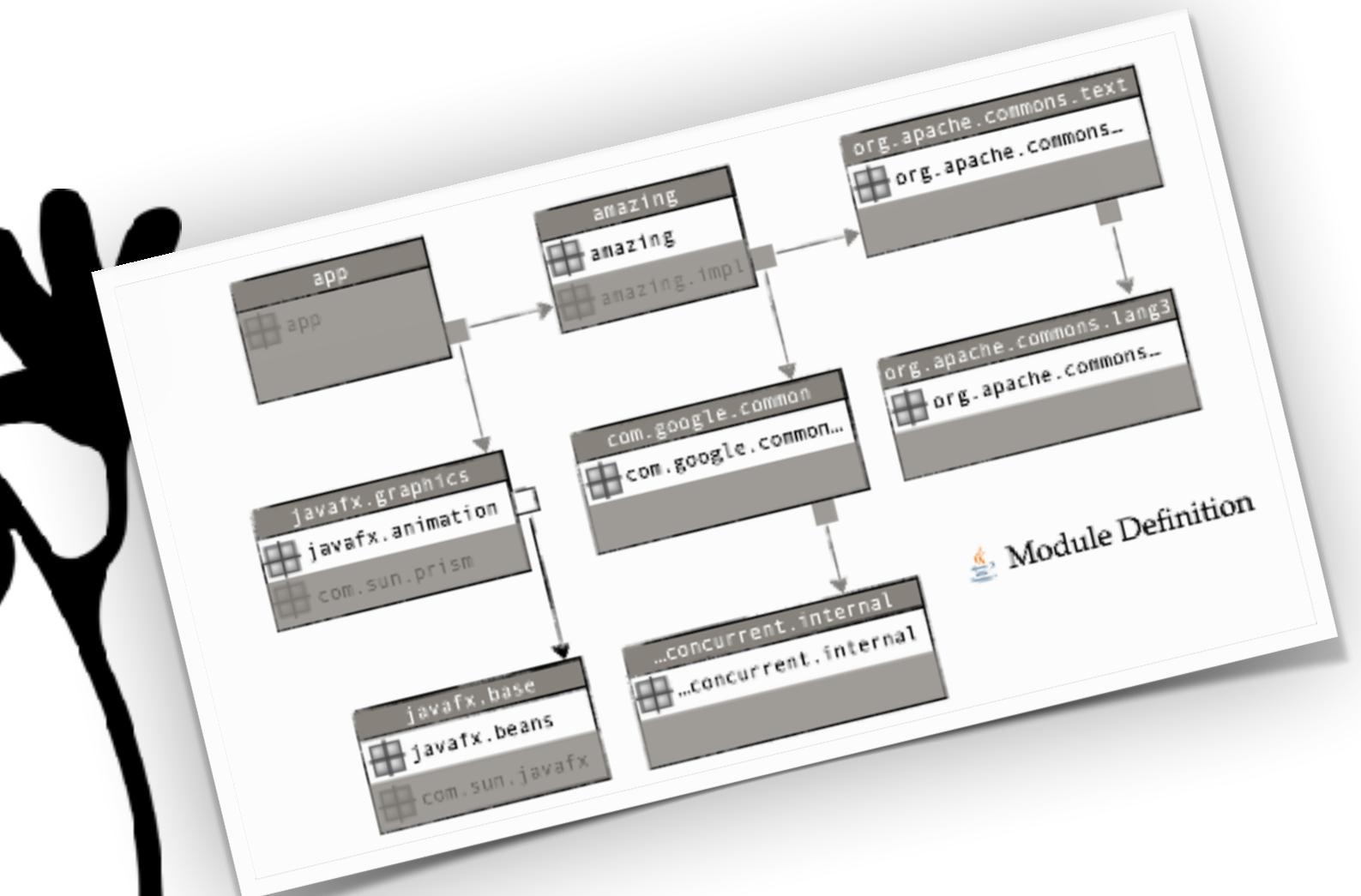
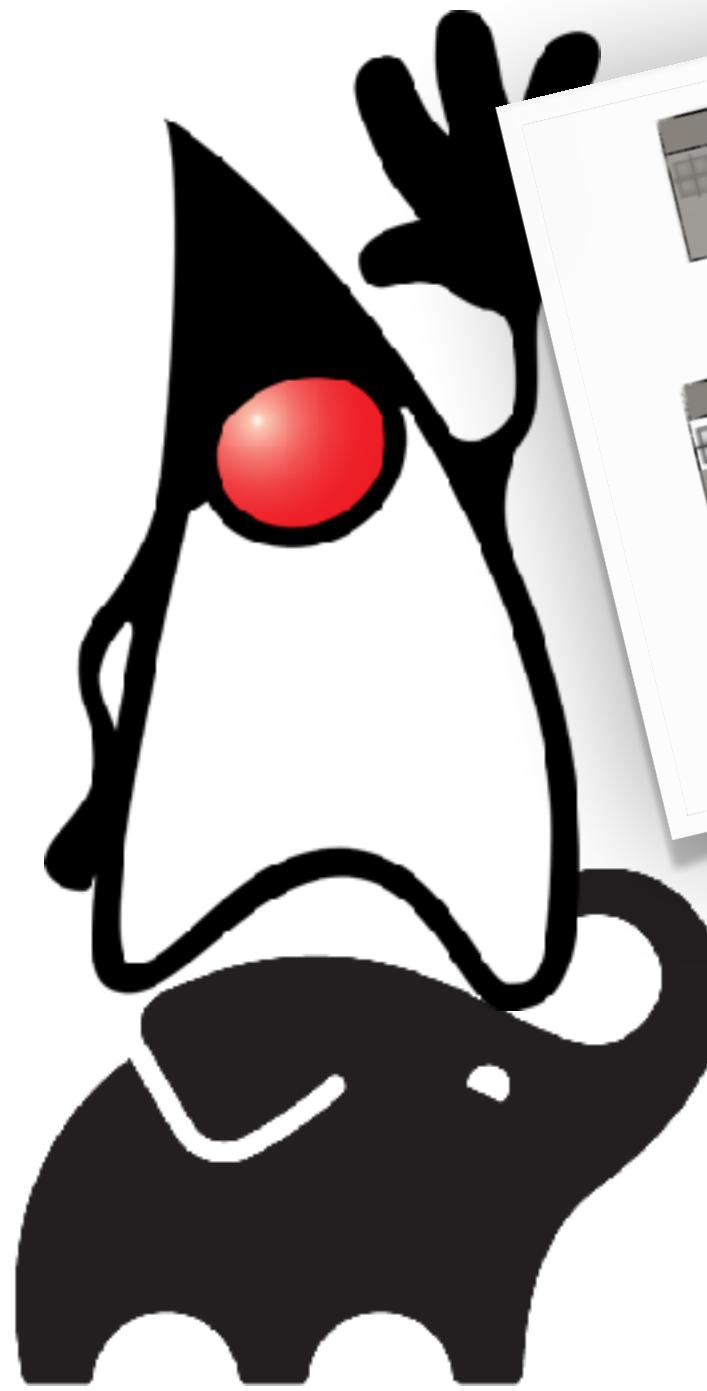


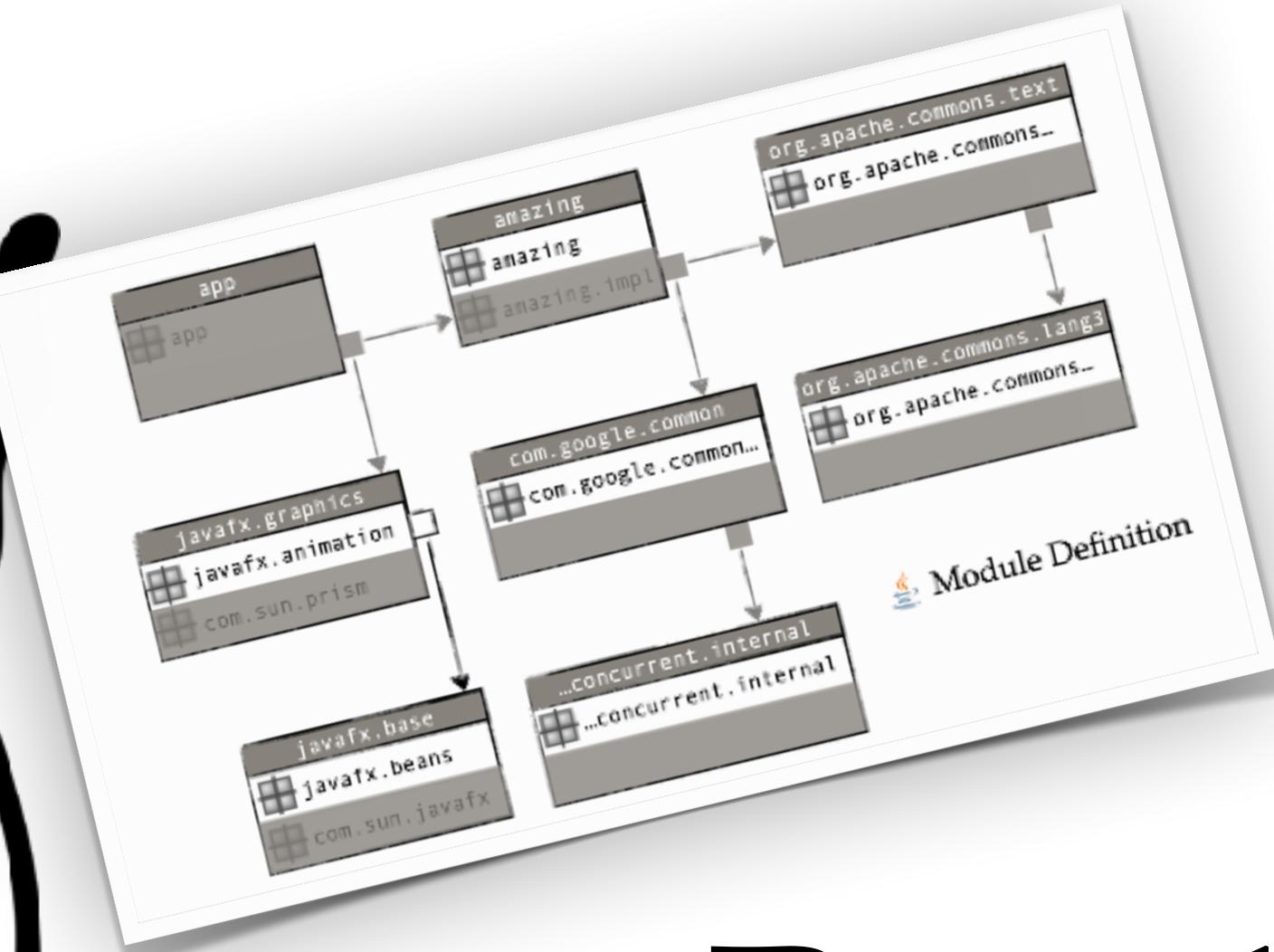
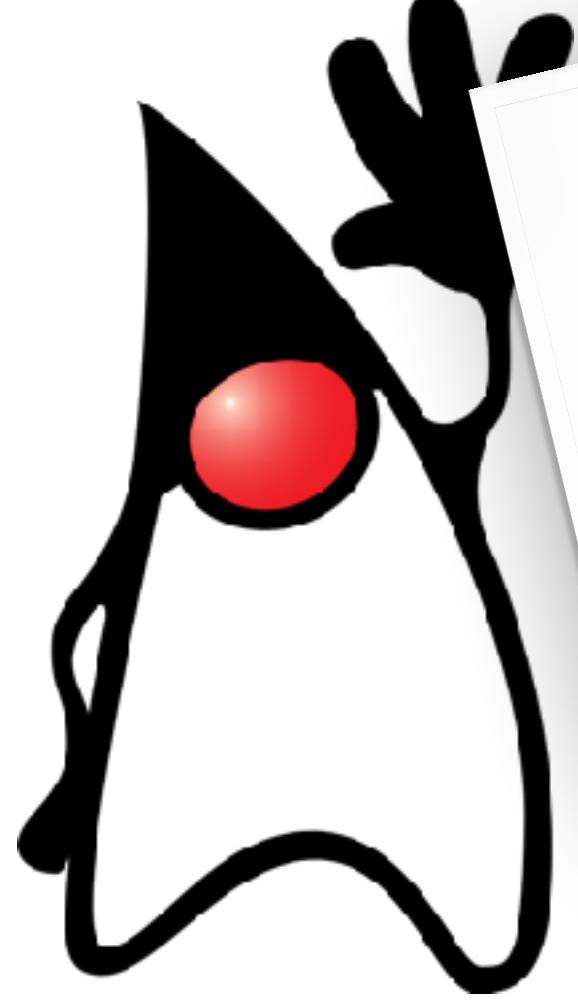
Das Java-Modul-System

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)

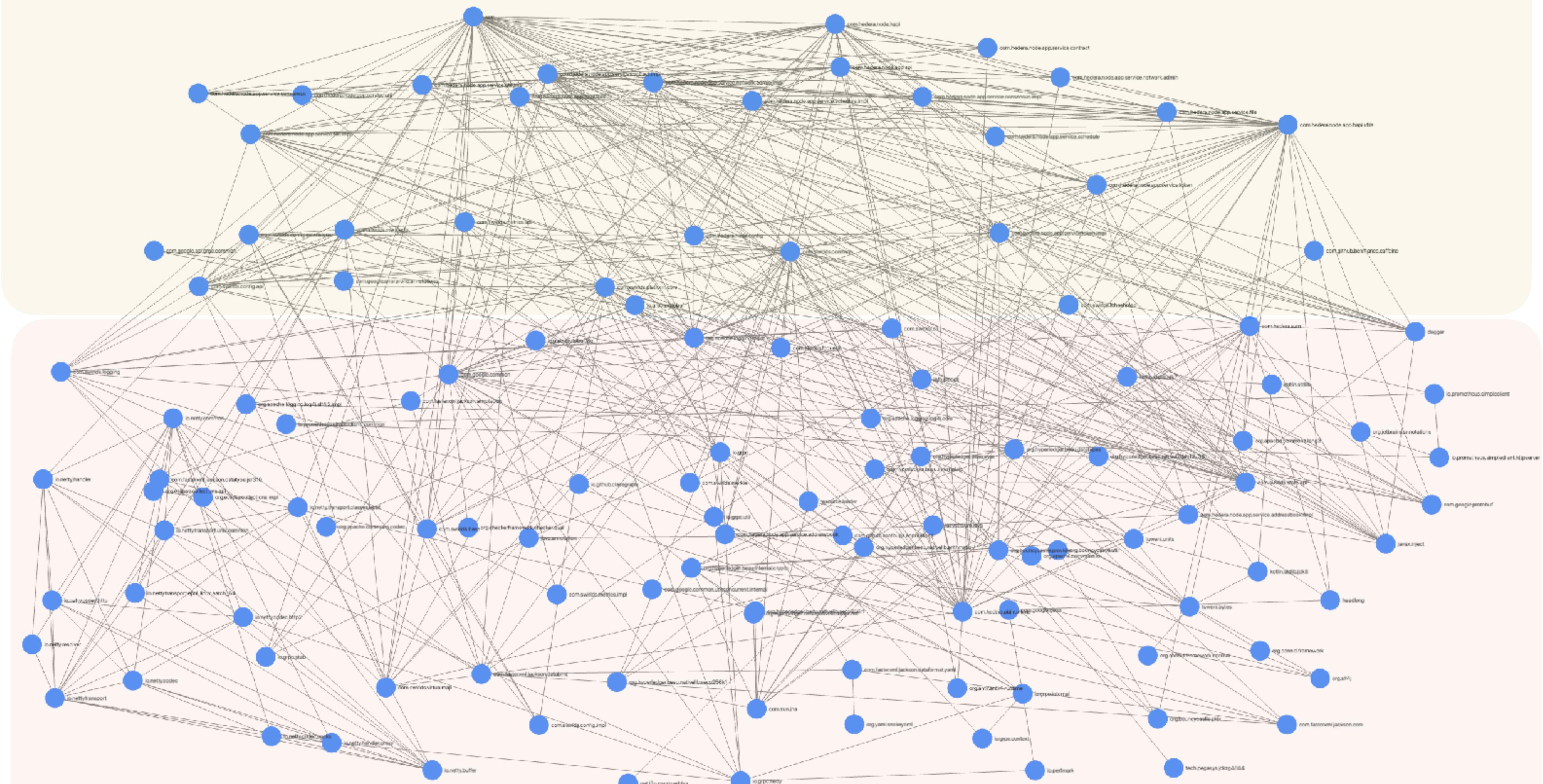
Jar Files

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)

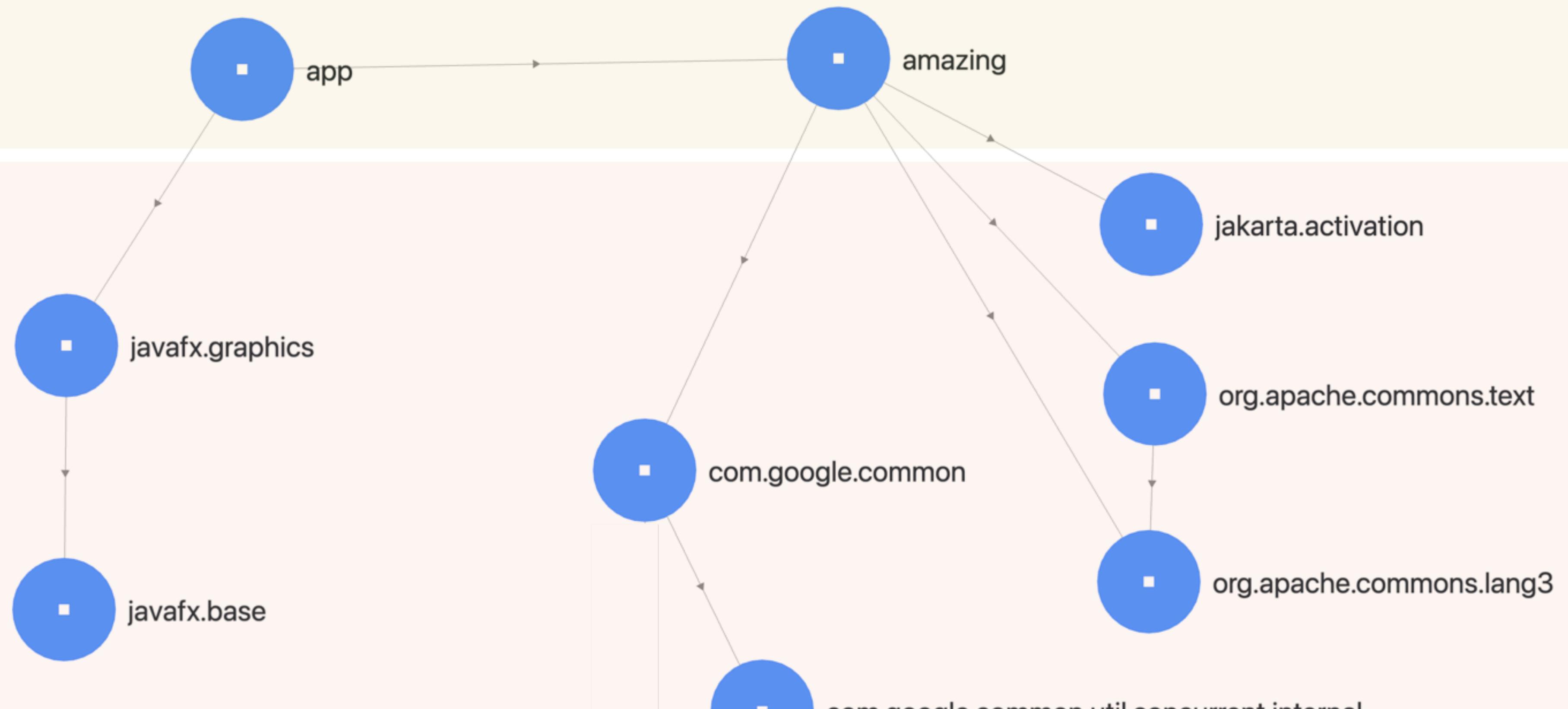
Jar Files

JVM Runtime
Classic

JVM Runtime
Module System

JVM Compile Time
Module app

JVM Compile Time
Module amazing



Software
Architecture

Jar Files
(uncleaned)

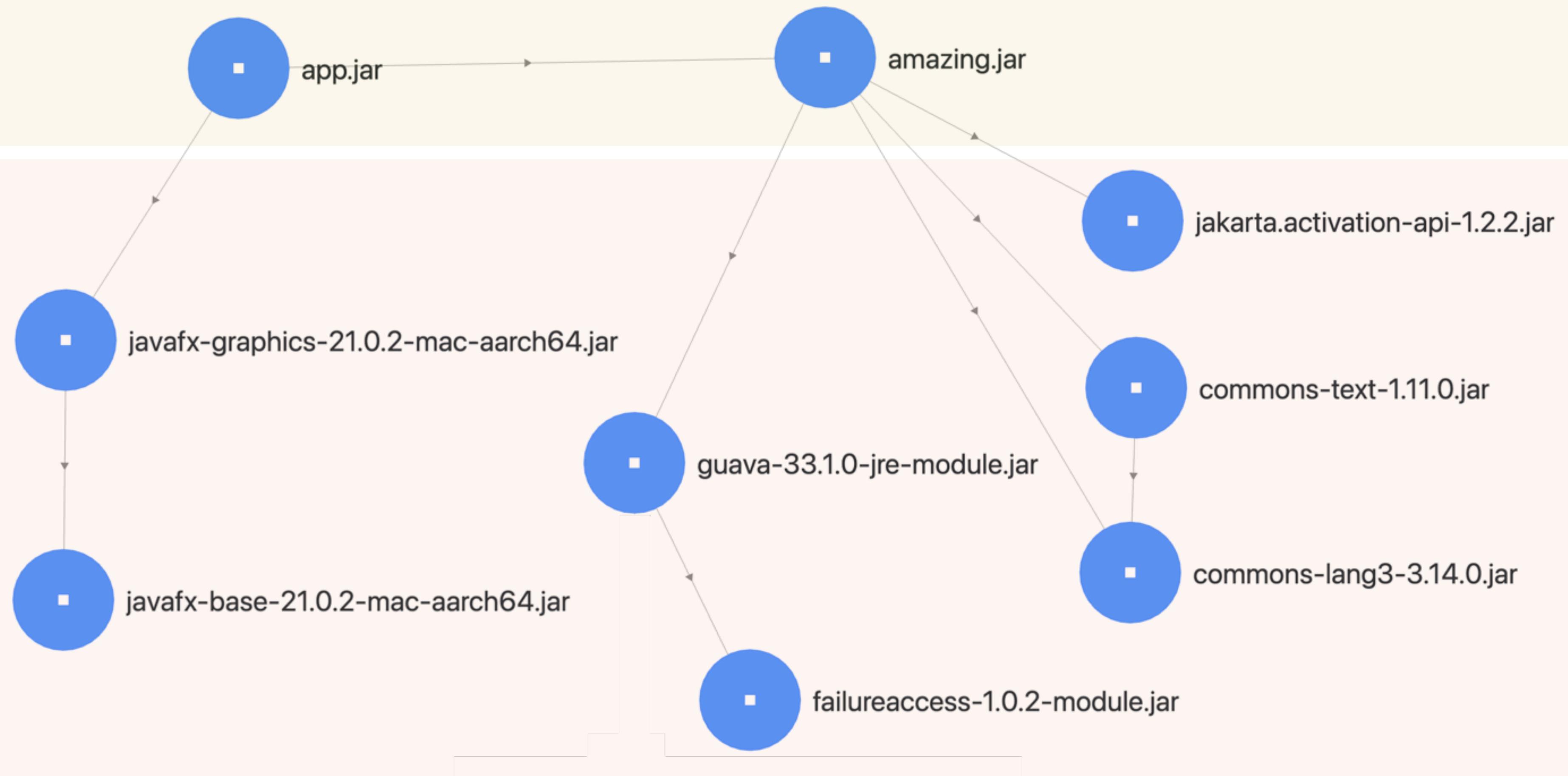
Jar Files

JVM Runtime
Classic

JVM Runtime
Module System

JVM Compile Time
Module app

JVM Compile Time
Module amazing



Software
Architecture

Jar Files
(uncleaned)

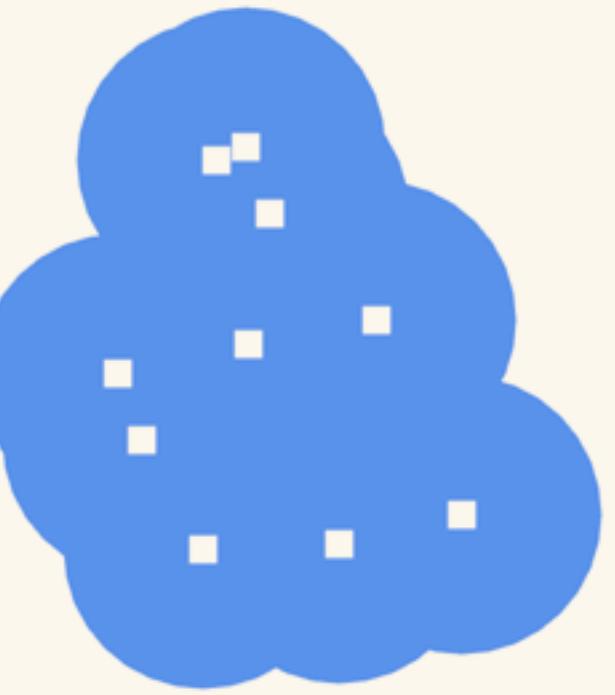
Jar Files

JVM Runtime
Classic

JVM Runtime
Module System

JVM Compile Time
Module app

JVM Compile Time
Module amazing



Software
Architecture

Jar Files
(uncleaned)

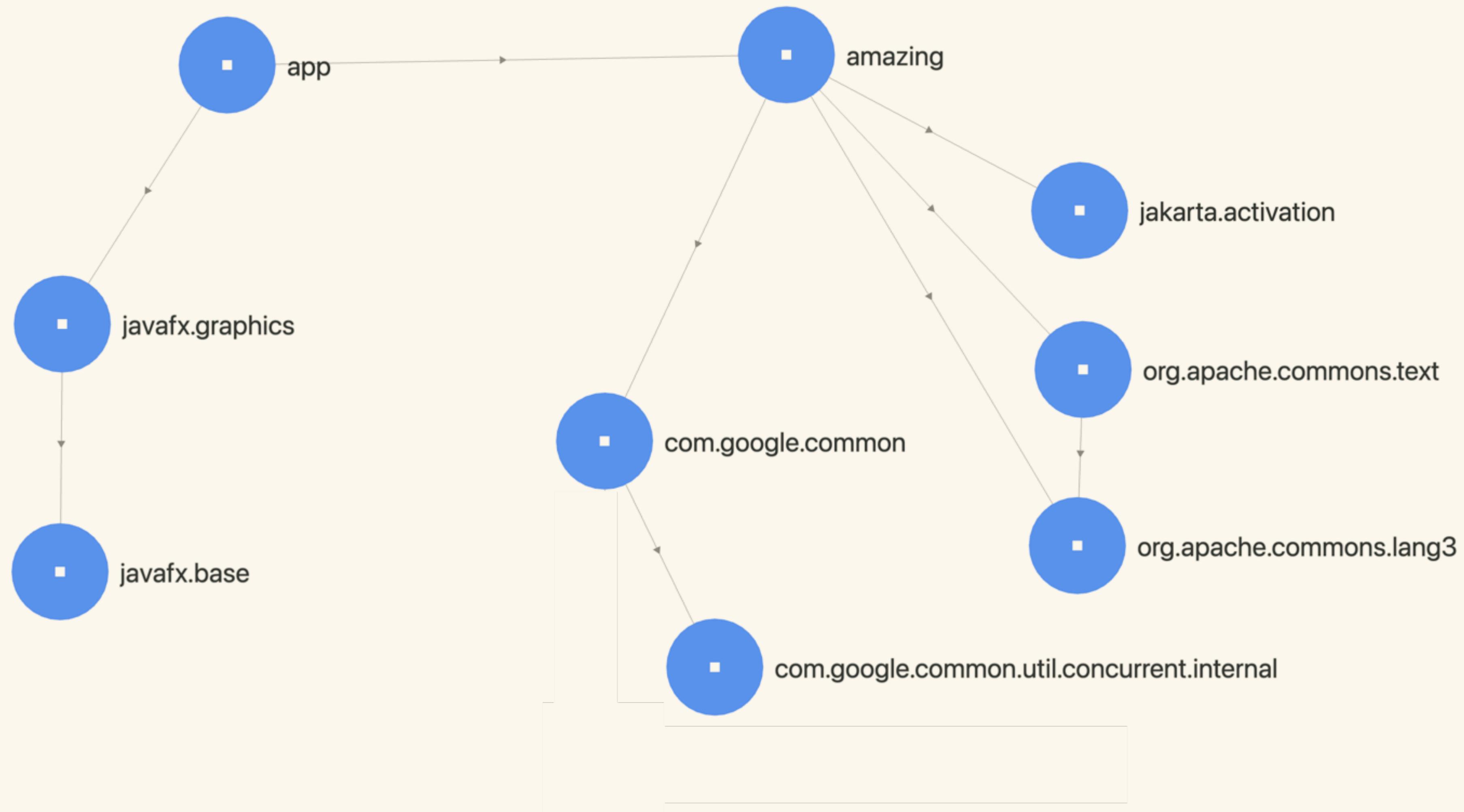
Jar Files

JVM Runtime
Classic

JVM Runtime
Module System

JVM Compile Time
Module app

JVM Compile Time
Module amazing



Software Architecture

Jar Files
(uncleaned)

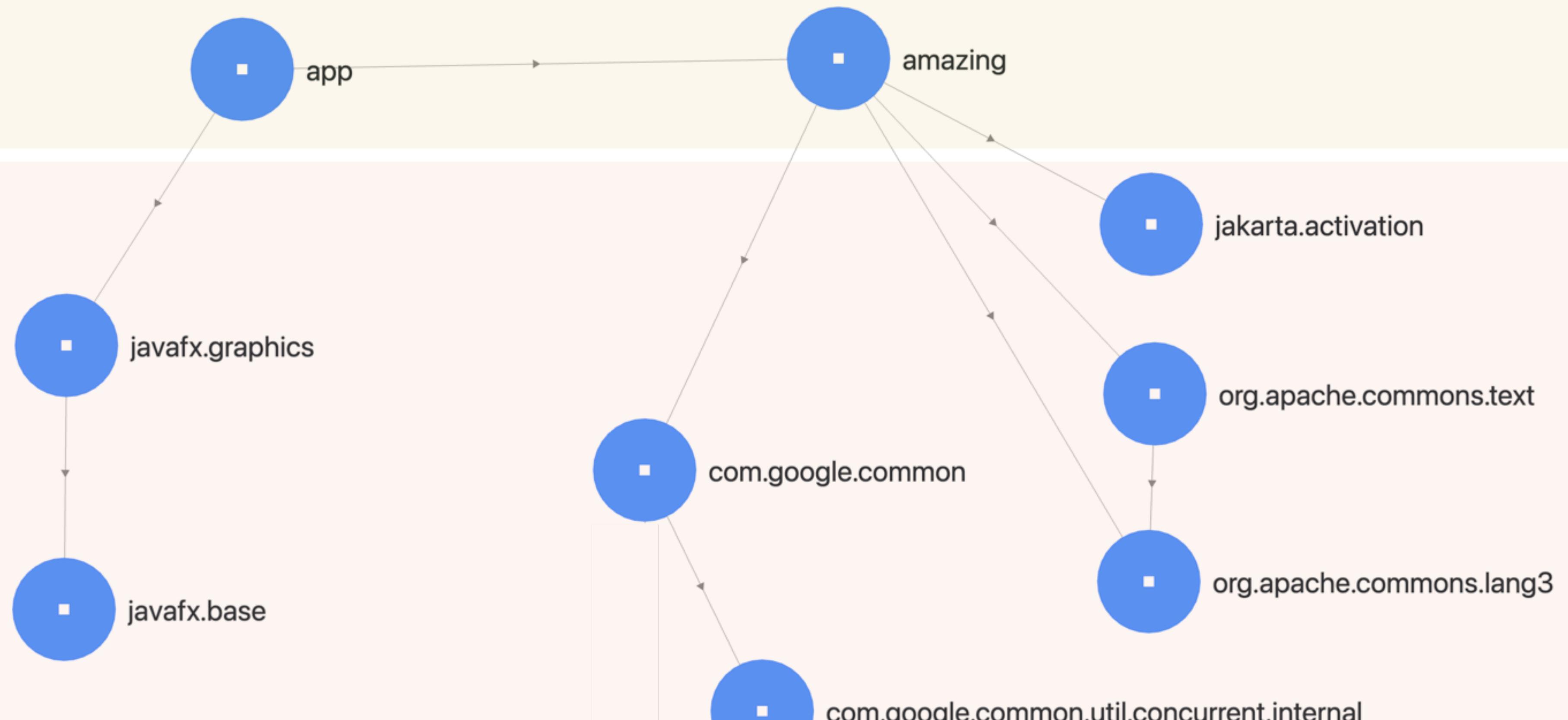
Jar Files

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

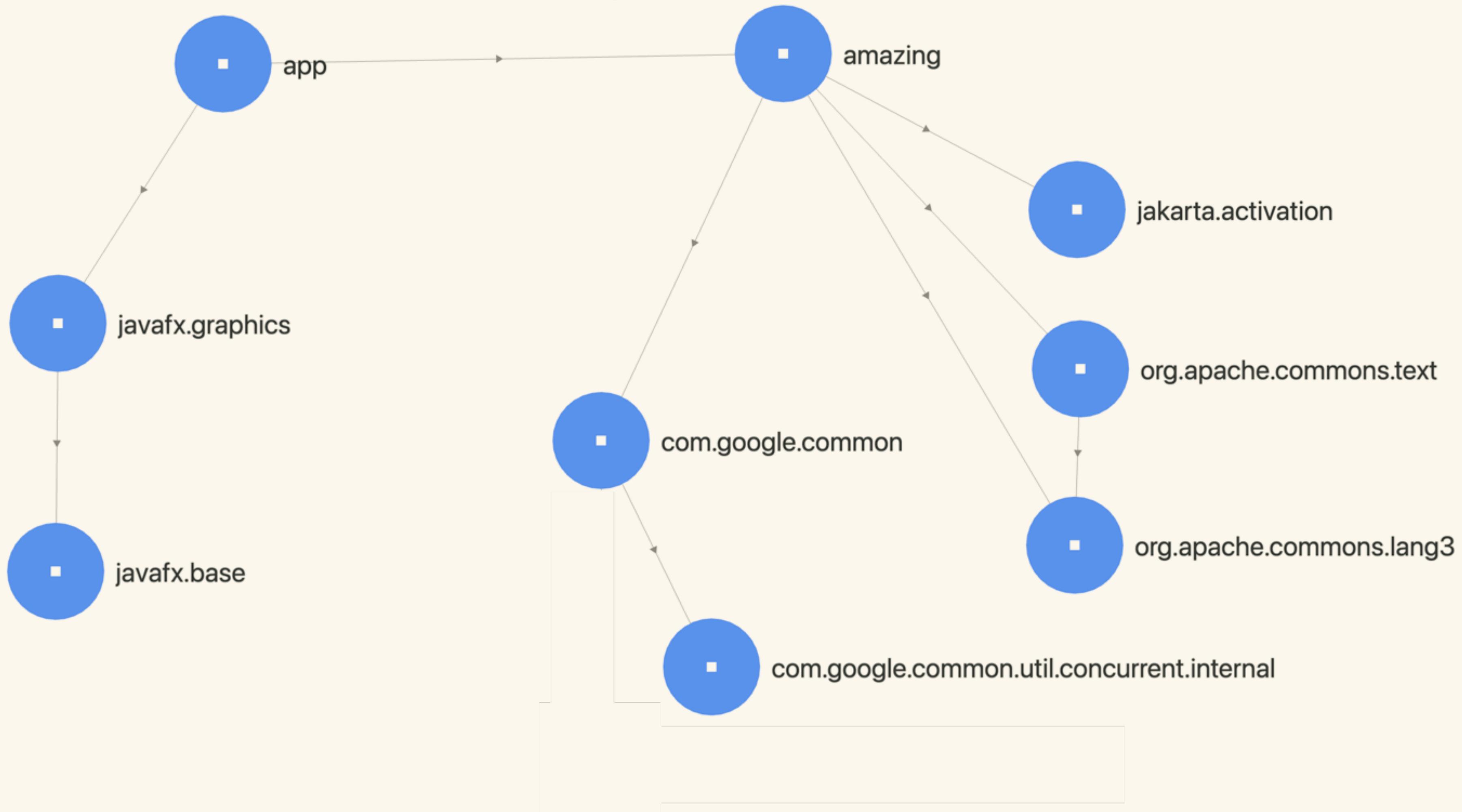
JVM Runtime
Module System

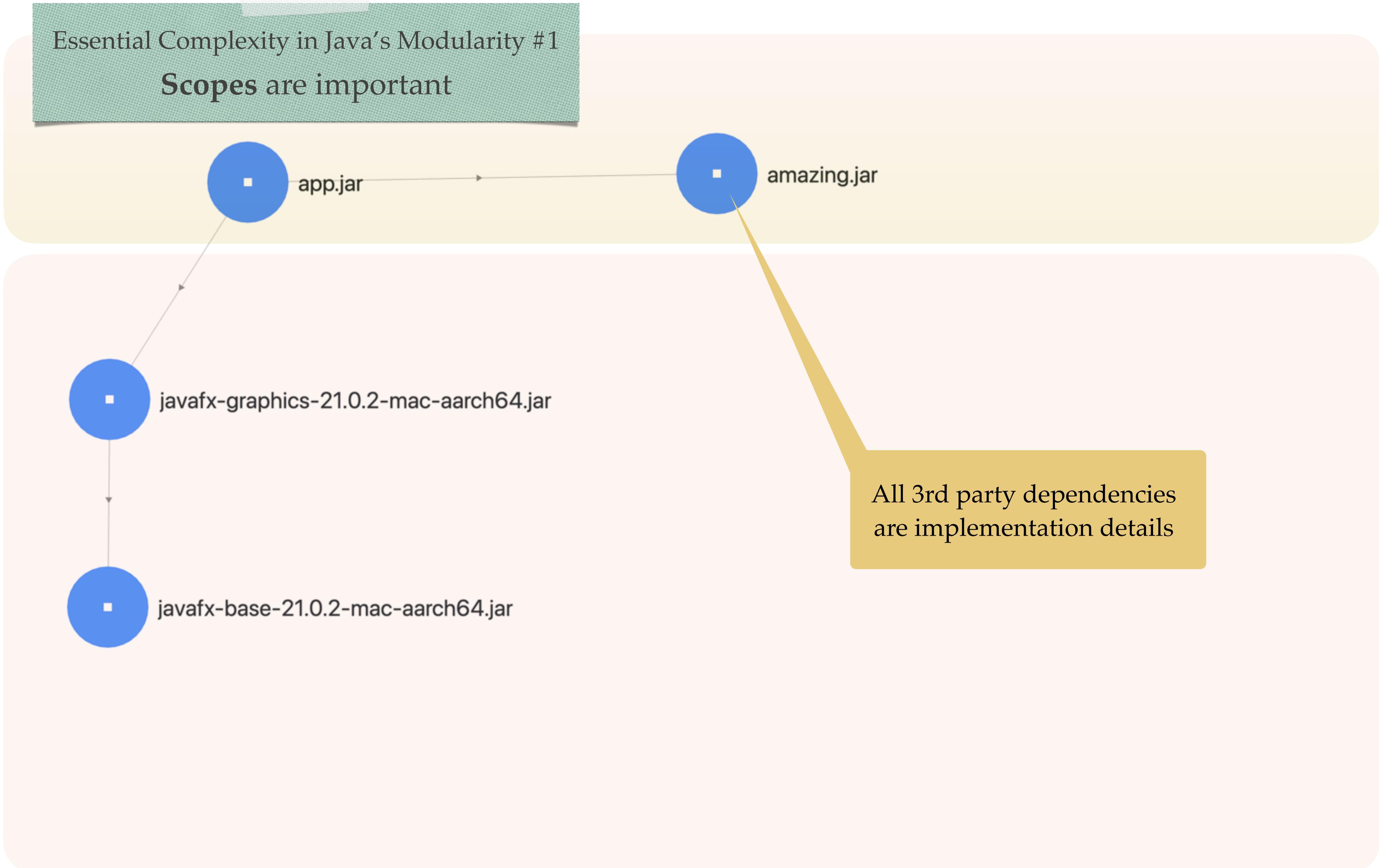
JVM Compile Time
Module app

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

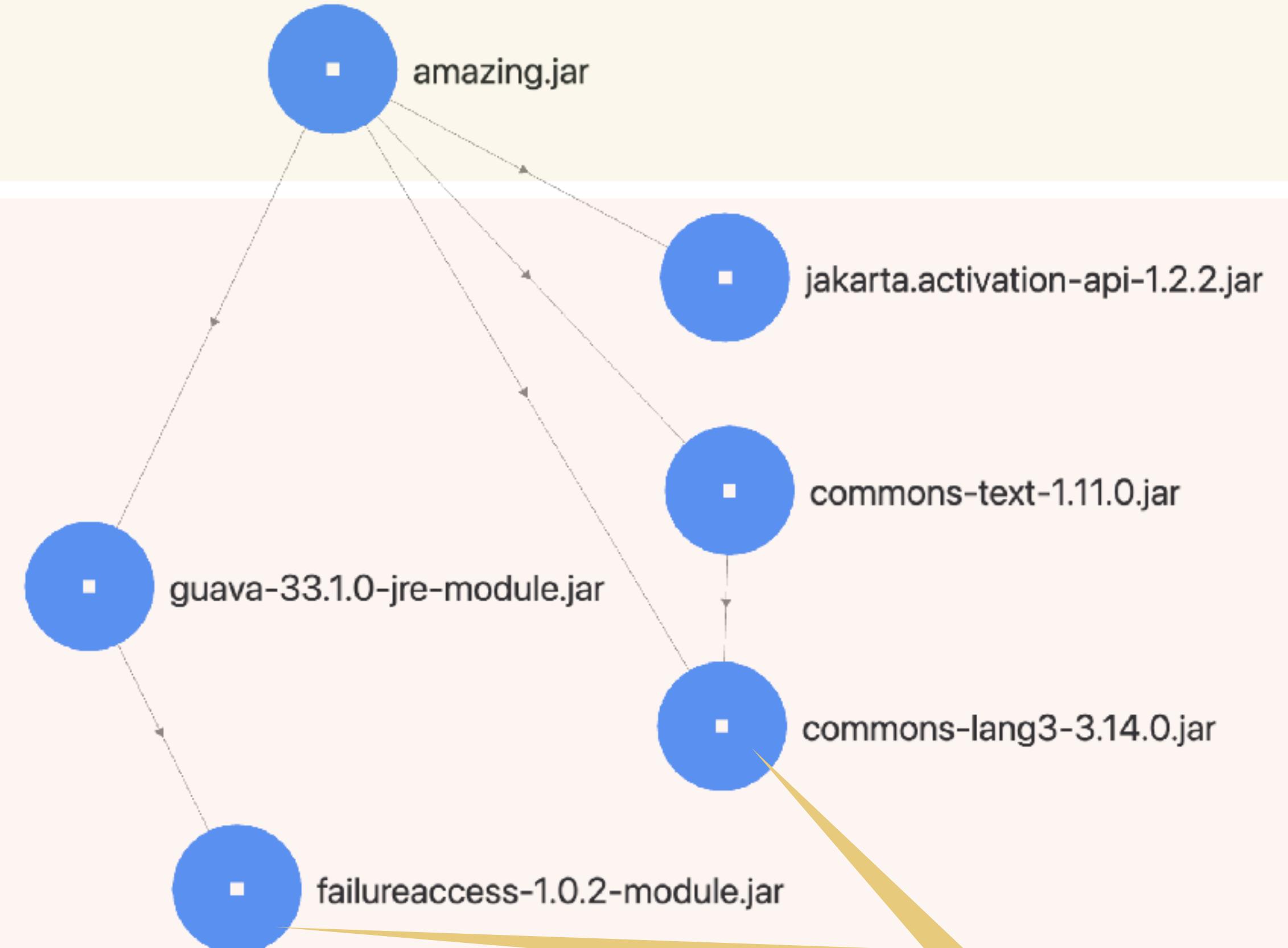
JVM Runtime
Module System

JVM Compile Time
Module app

JVM Compile Time
Module amazing

Essential Complexity in Java's Modularity #1

Scopes are important



Should these be
visible?

Software
Architecture

Jar Files
(uncleaned)

Jar Files

JVM Runtime
Classic

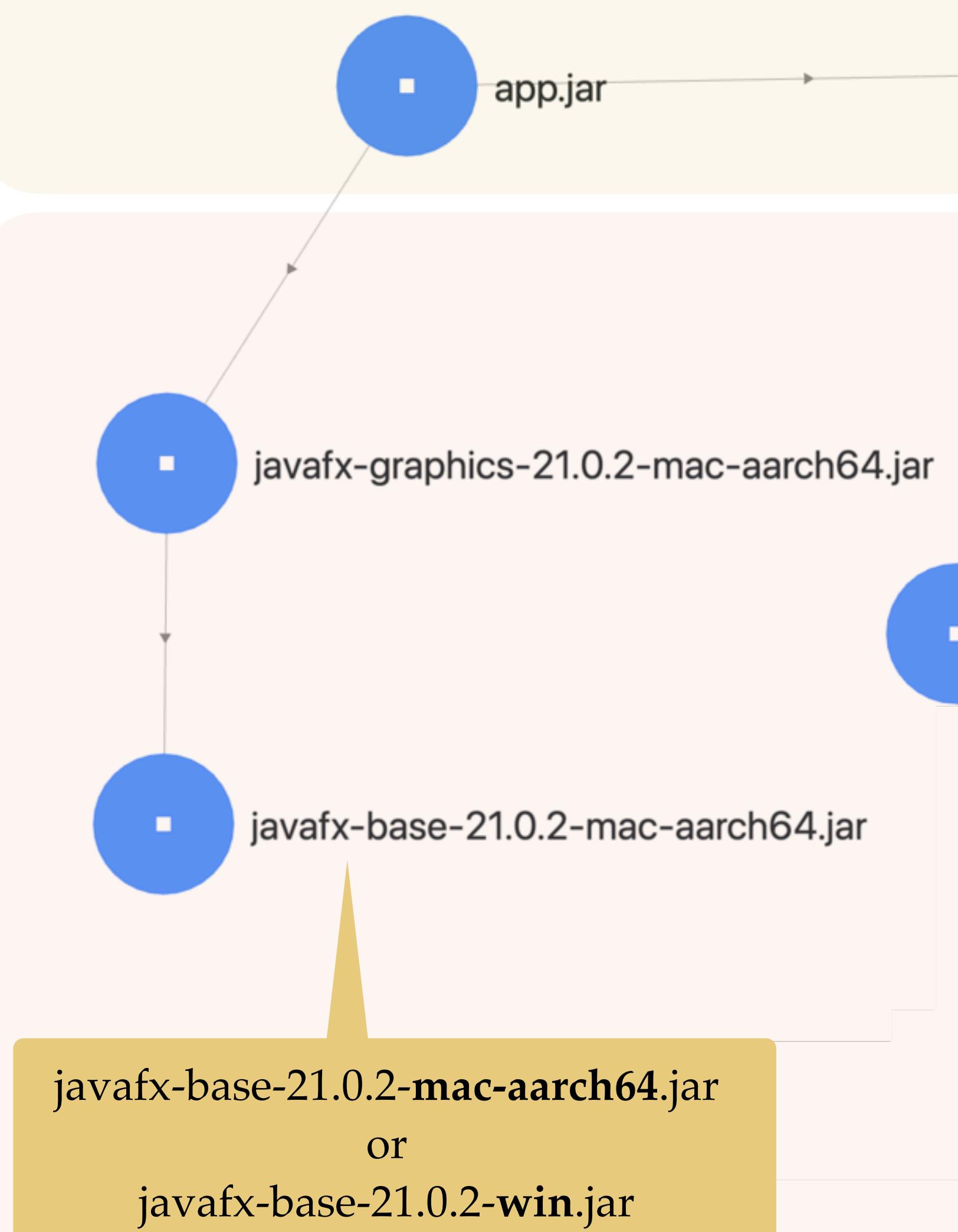
JVM Runtime
Module System

JVM Compile Time
Module app

JVM Compile Time
Module amazing

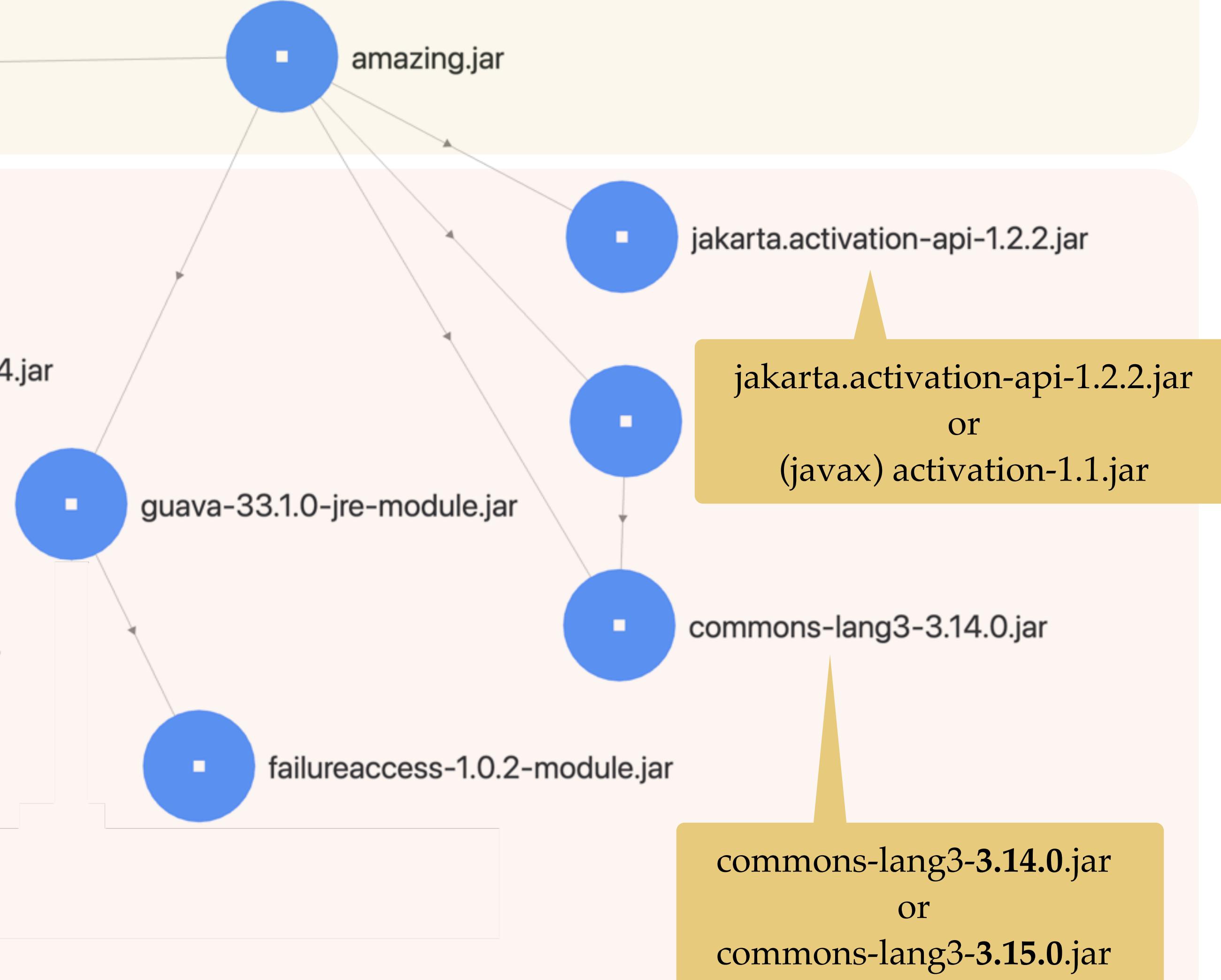
Essential Complexity in Java's Modularity #1

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

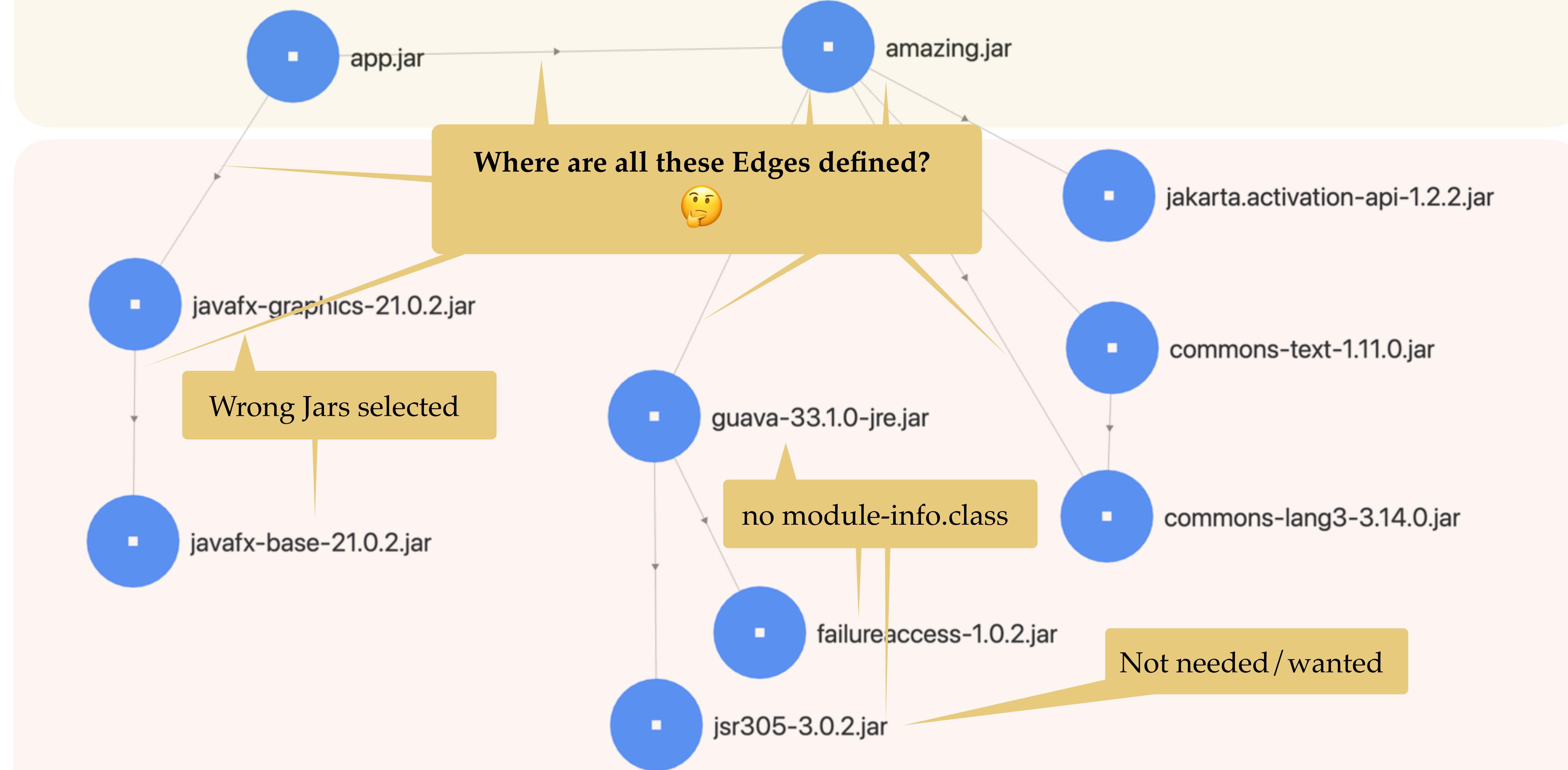
JVM Compile Time
Module amazing

Essential Complexity in Java's Modularity #1

Scopes are important

Essential Complexity in Java's Modularity #2

Multiple Versions/Variants of 3rd party Jars



Software
Architecture

Jar Files
(uncleaned)

Jar Files

JVM Runtime
Classic

JVM Runtime
Module System

JVM Compile Time
Module app

JVM Compile Time
Module amazing

Accidental Complexity #1

Scopes
not cared about (enough)

Essential
Dependencies

Maven Repository: 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)

Maven Repository: 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 #1 in String Utilities
Used By	3,599 artifacts

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?

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

~300 Lines total
~20 Lines Module Definition

```
29
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>
46
```

```
29
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
44
```

~300 Lines total
~100 Line Module Definition

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

Accidental Complexity #1
Scopes
not cared about (enough)

Accidental Complexity #2
Module definition mixed
with other build concerns

■ jsr305-3.0.2.jar

Software
Architecture

Jar Files
(uncleaned)

Jar Files

JVM Runtime
Classic

JVM Runtime
Module System

JVM Compile Time
Module app

JVM Compile Time
Module amazing

Accidental Complexity #1
Scopes
not cared about (enough)

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 #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

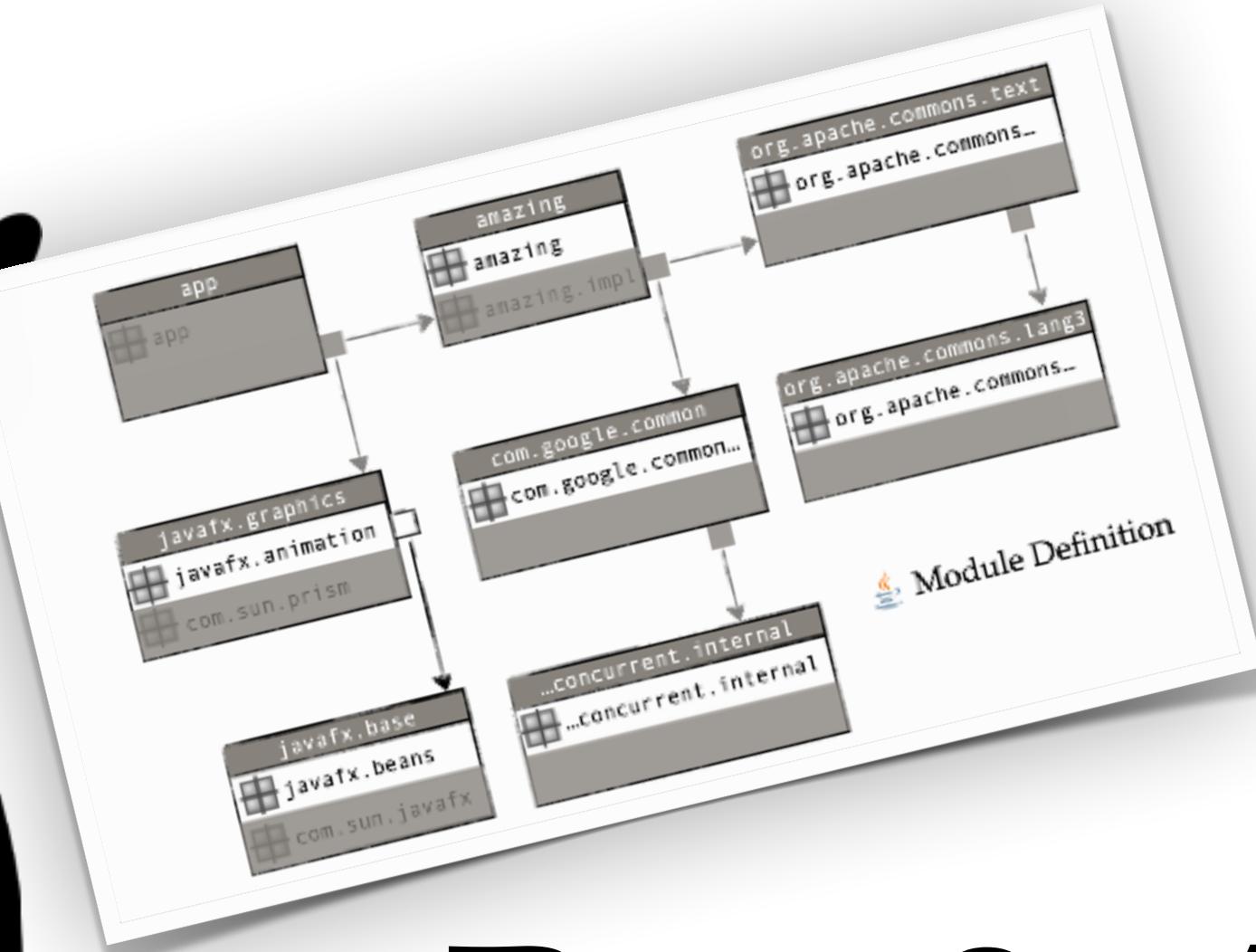
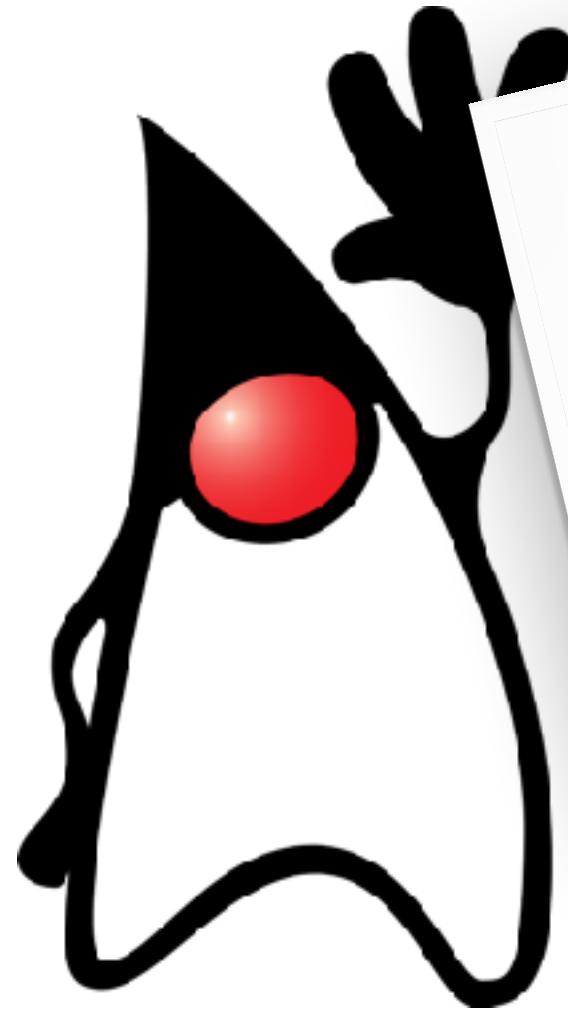
Empty Jar to get Maven to do
a OS-based selection
-> such hacks shouldn't be needed
-> does not work for Gradle

Accidental Complexity #1
Scopes
not cared about (enough)

Accidental Complexity #2
Module definition mixed
with other build concerns

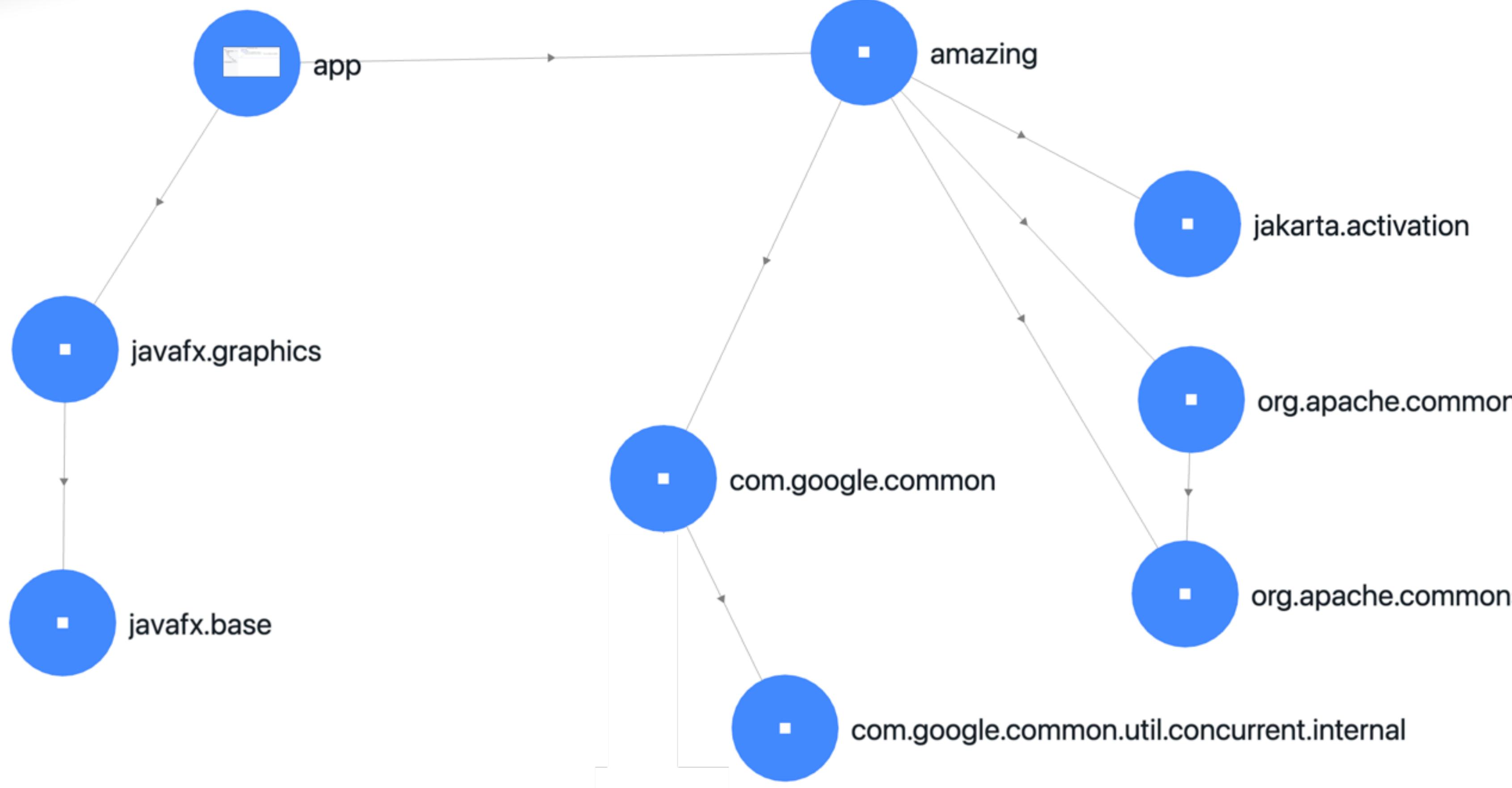
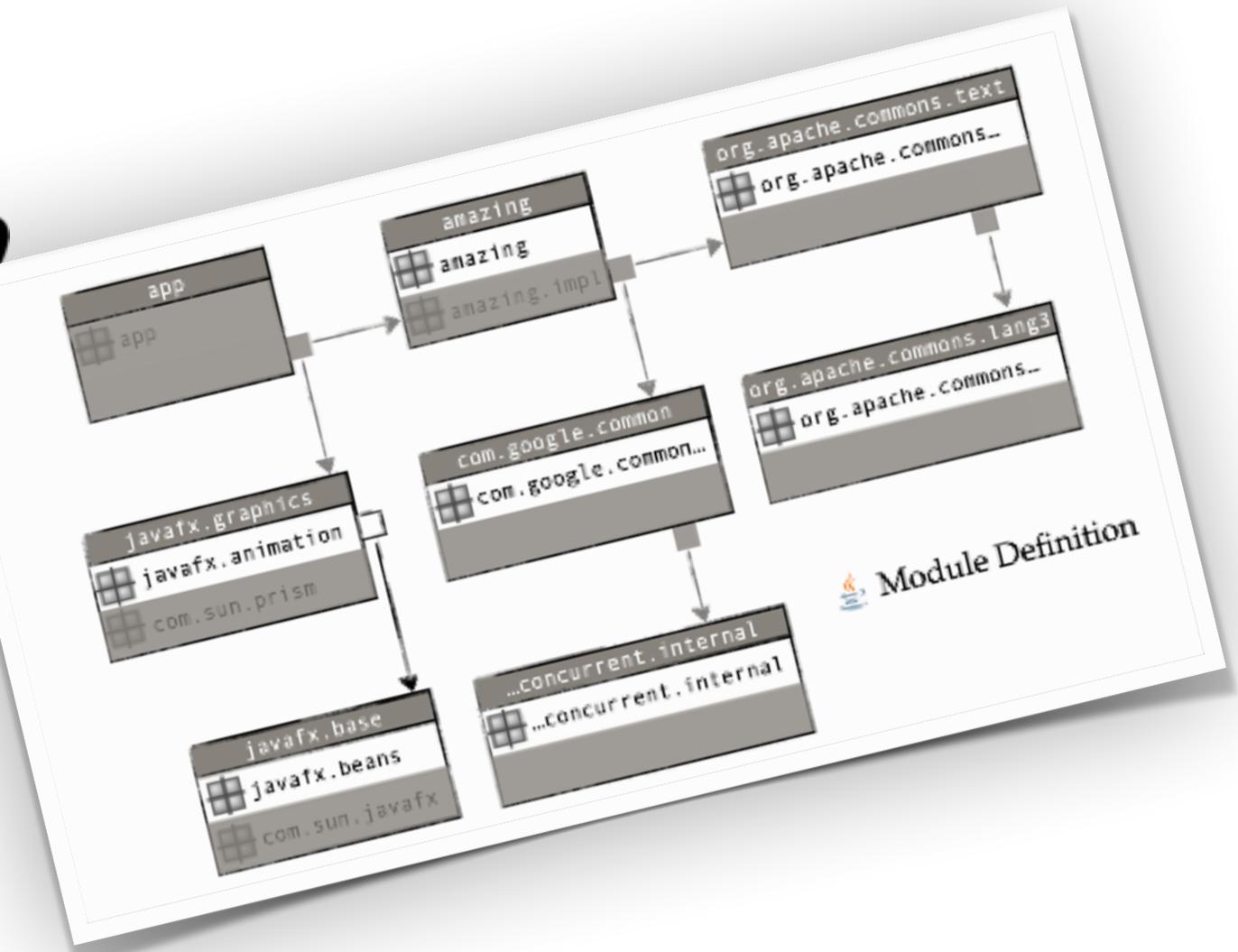
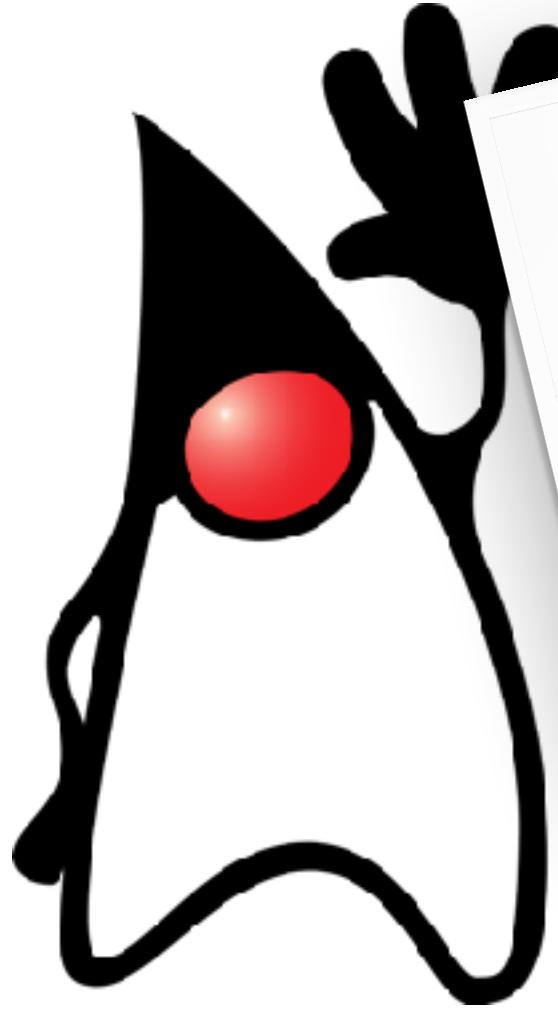
Accidental Complexity #3
Multiple Version/Variants
not cared about (enough)

Accidental Complexity #4
Metadata of 3rd party modules
is incomplete or wrong



Part 2: A New Approach

Using Java for
Module Definition



java-module-system – App.java

Project ▾

java-module-system ~/projects/gran...

- 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());
    }
}
```

off

java-module-system – App.java

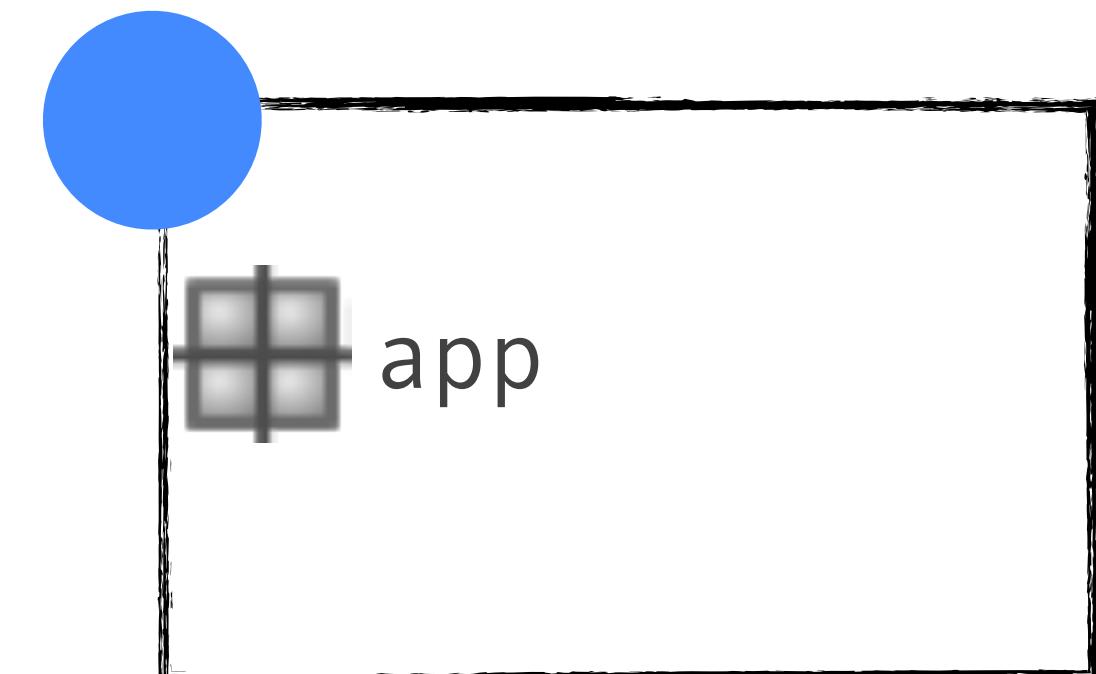
Project ▾

java-module-system ~/projects/gran...

- 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());
    }
}
```



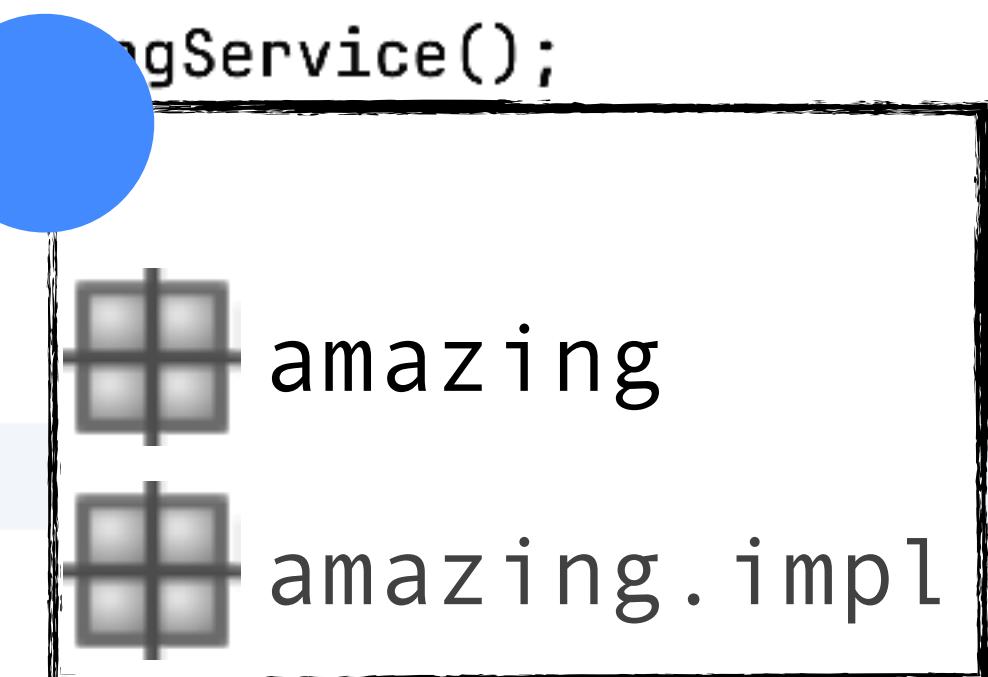
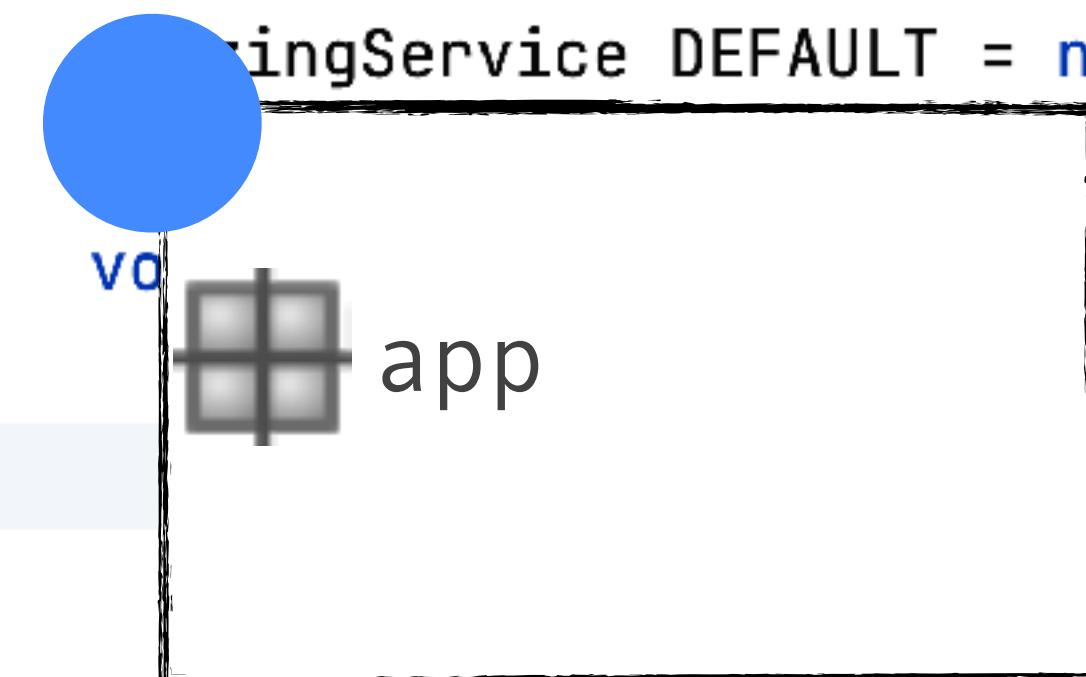
java-module-system – DefaultAmazingService.java

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

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

```
package amazing;  
  
import amazing.impl.DefaultAmazingService;  
  
public interface AmazingService {  
    AmazingService DEFAULT = new DefaultAmazingService();  
    void app();  
}
```



java-module-system – App.java

Project ▾

java-module-system ~/projects/gradle/howto/java-mo

modules

amazing

src

main

java

amazing

impl

DefaultAmazingService.java

AmazingService.java

app

src

main

java

app

App.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();
    }
}
```

The diagram illustrates the module dependency relationship between the 'app' module and the 'amazing' module. On the left, a box labeled 'app' contains a blue circle at the top and a square icon representing a module at the bottom. An arrow points from this box to a second box on the right, which is also labeled 'app' at the top and contains a blue circle at the top. This second box is divided into two sections: 'amazing' at the top and 'amazing.impl' at the bottom, each with its own square icon.

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

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

Project

java-module-system ~/projects/gradle/howto/java-mo

modules

amazing

src

main

java

amazing

impl

DefaultAmazingService.java

AmazingService.java

module-info.java

app

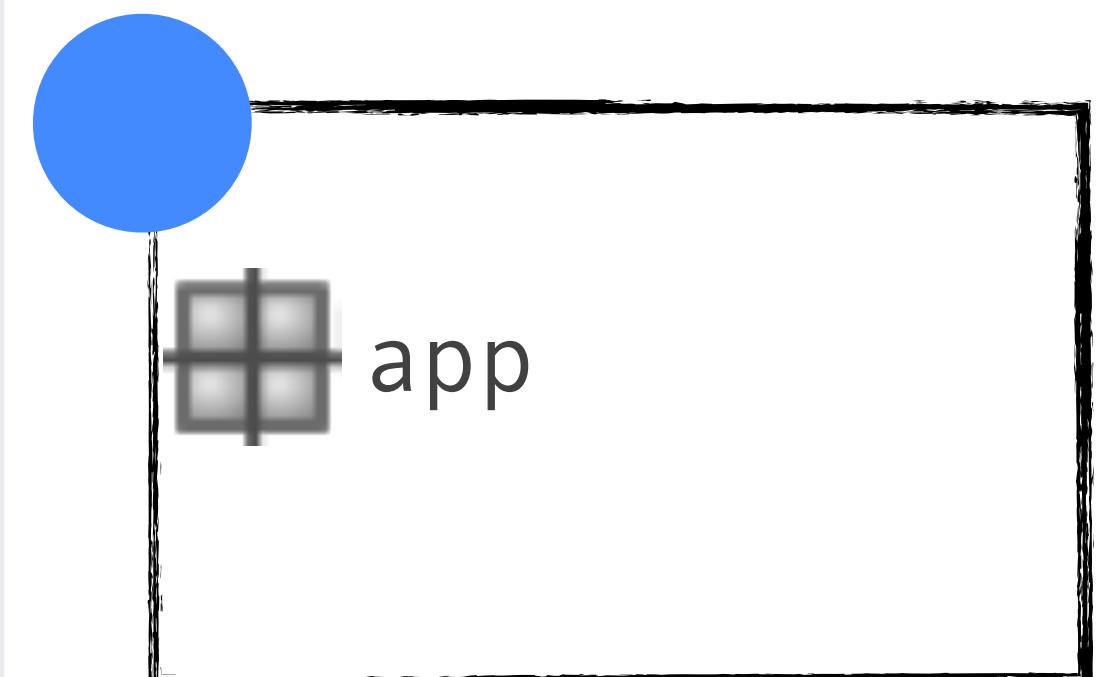
src

main

java

app

App.java



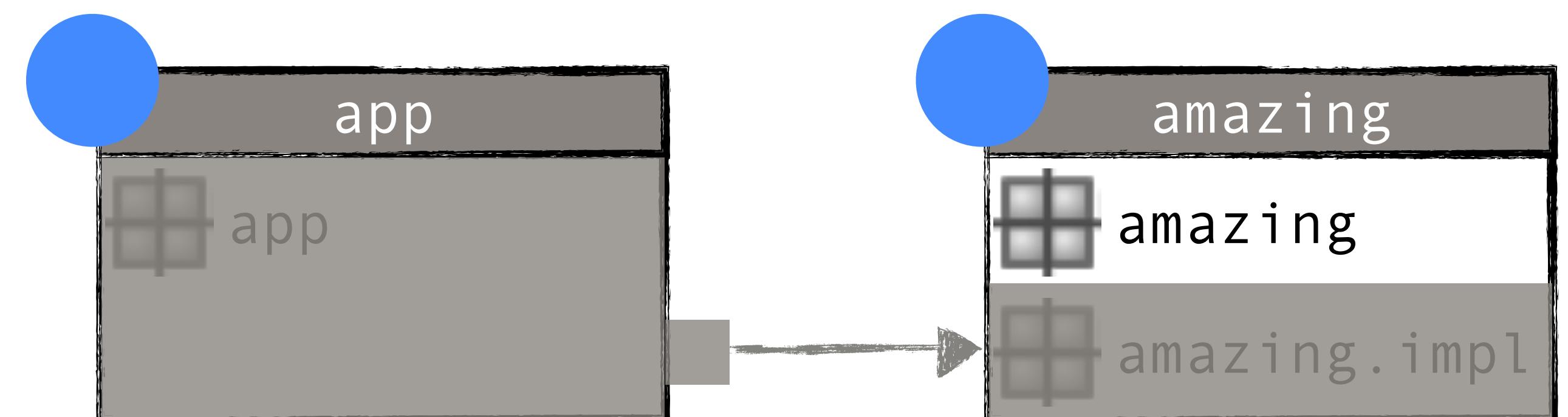
java-module-system – module-info.java (amazing)

Project

```
java-module-system ~/projects/gradle/howto/java-mo  
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-mo

modules

amazing

src

main

java

amazing

impl

DefaultAmazingService.java

AmazingService.java

module-info.java

app

src

main

java

app

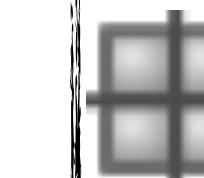
App.java

module-info.java

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

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

org.apache.commons.text



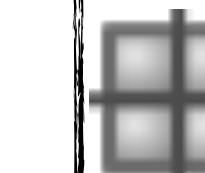
org.apache.common...

app



app

amazing



amazing



amazing.impl

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

Project

java-module-system ~/projects/gradle/howto/java-mo

modules

amazing

src

main

java

amazing

impl

DefaultAmazingService

AmazingService.java

module-info.java

app

src

main

java

app

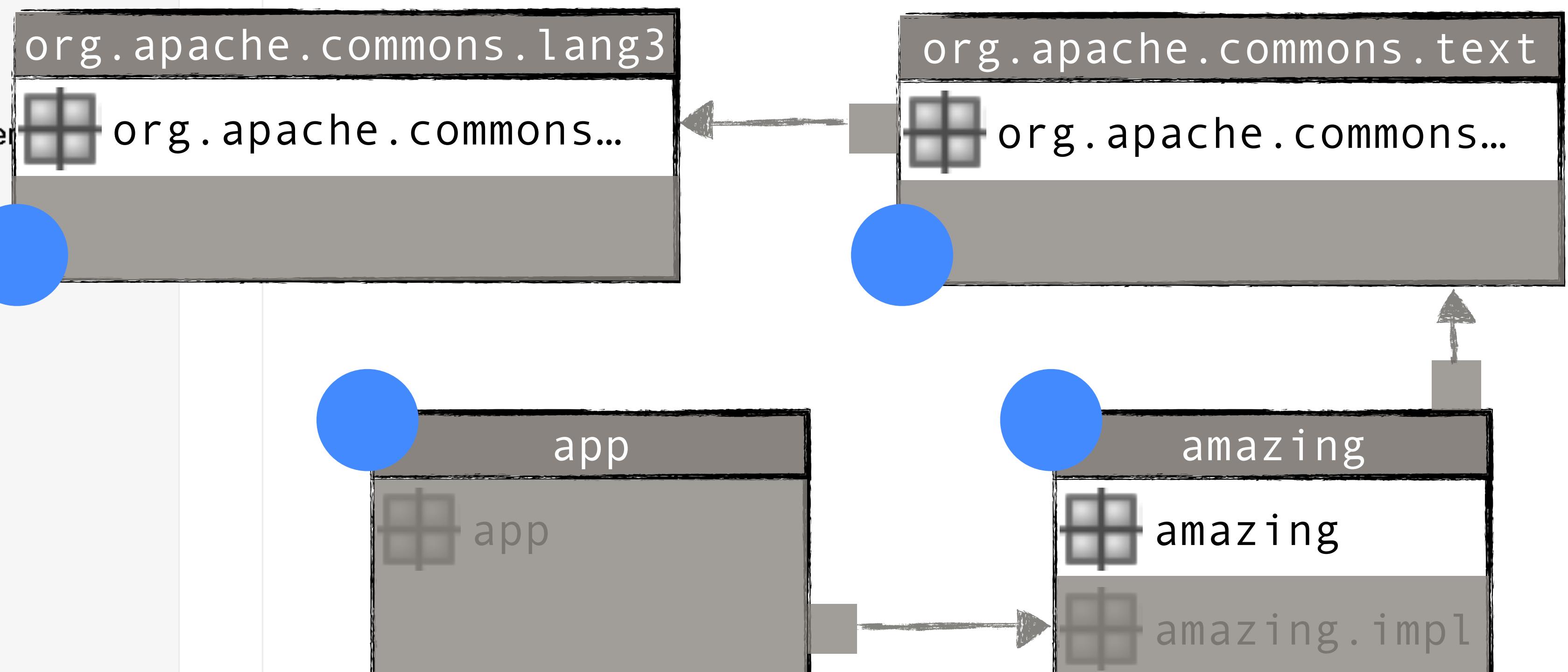
App.java

module-info.java

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

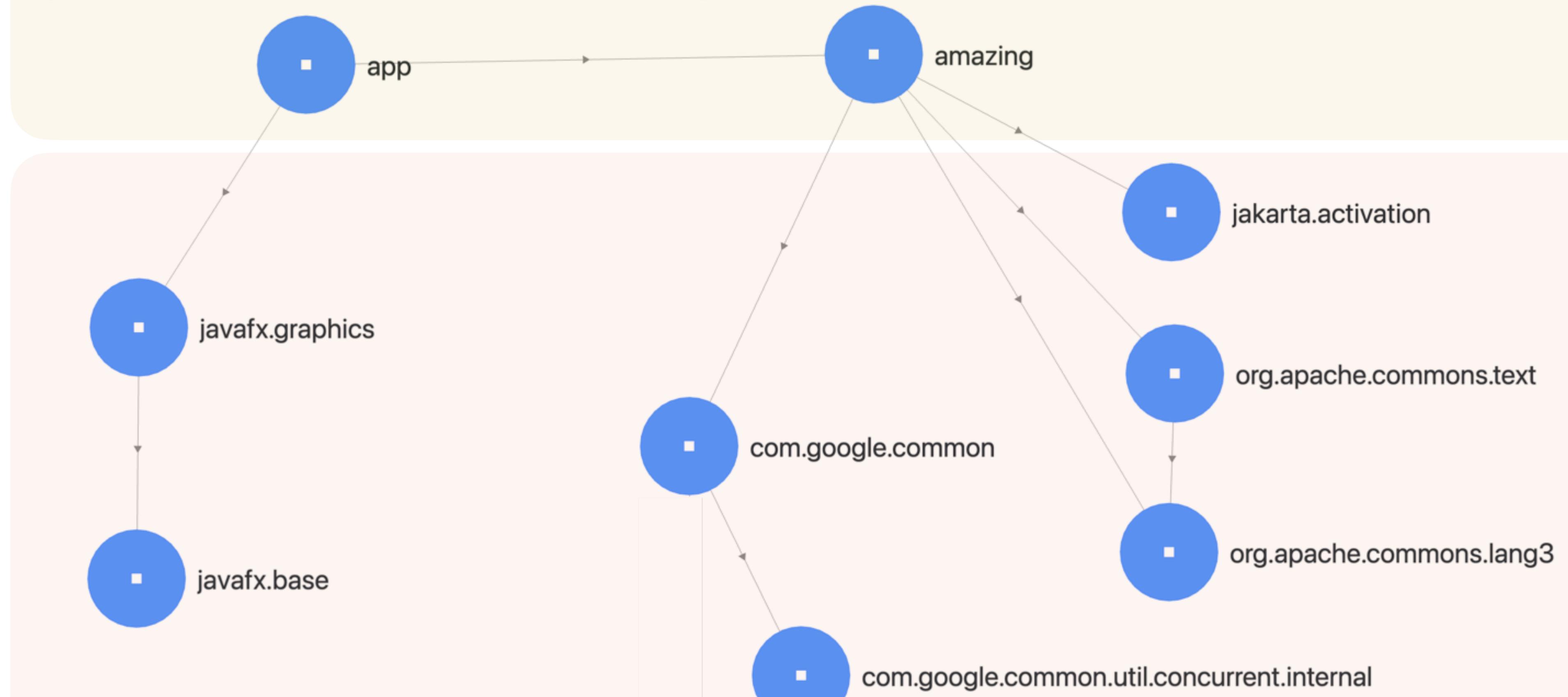
Scopes

- requires
- requires transitive
- requires static



Essential Complexity in Java's Modularity #1

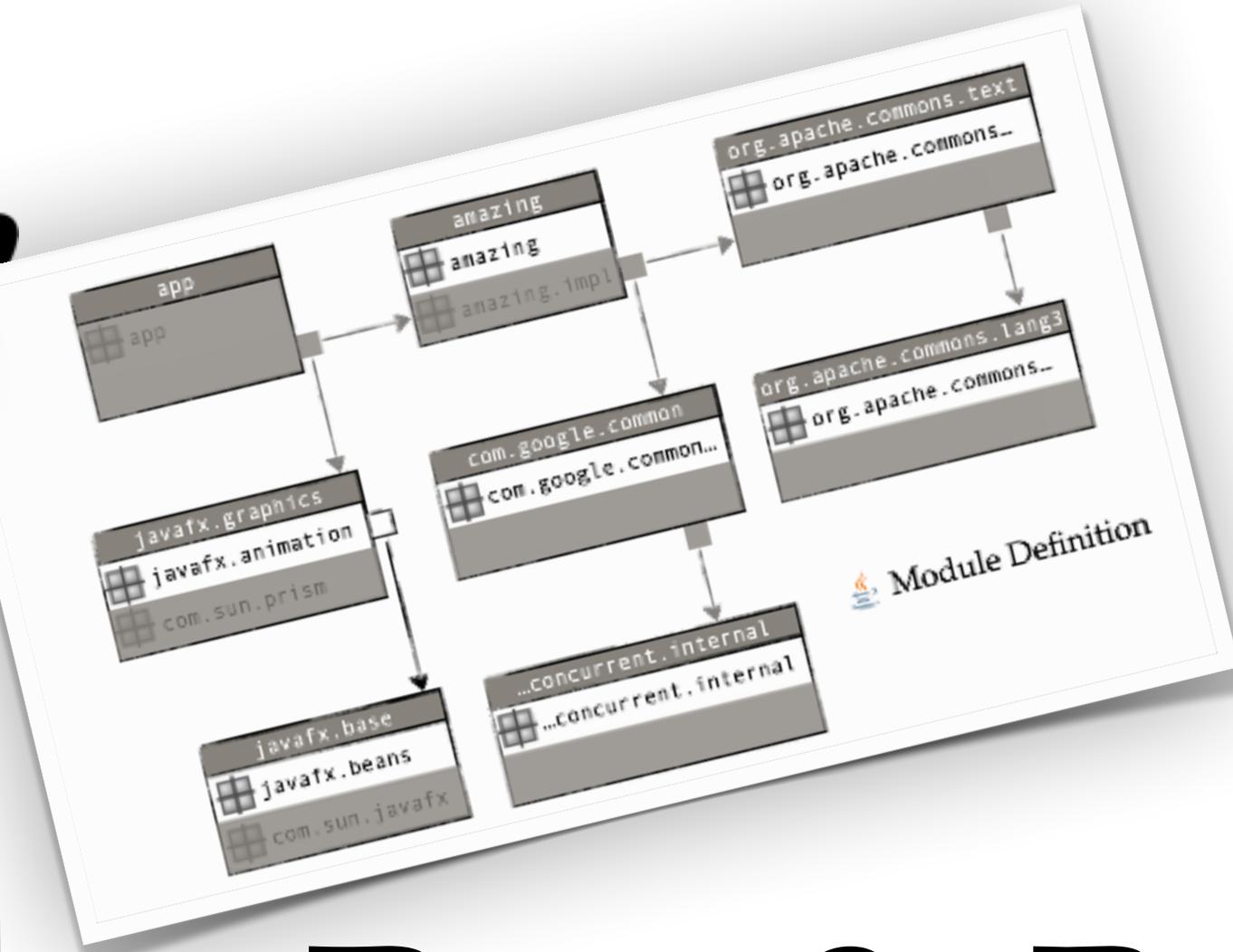
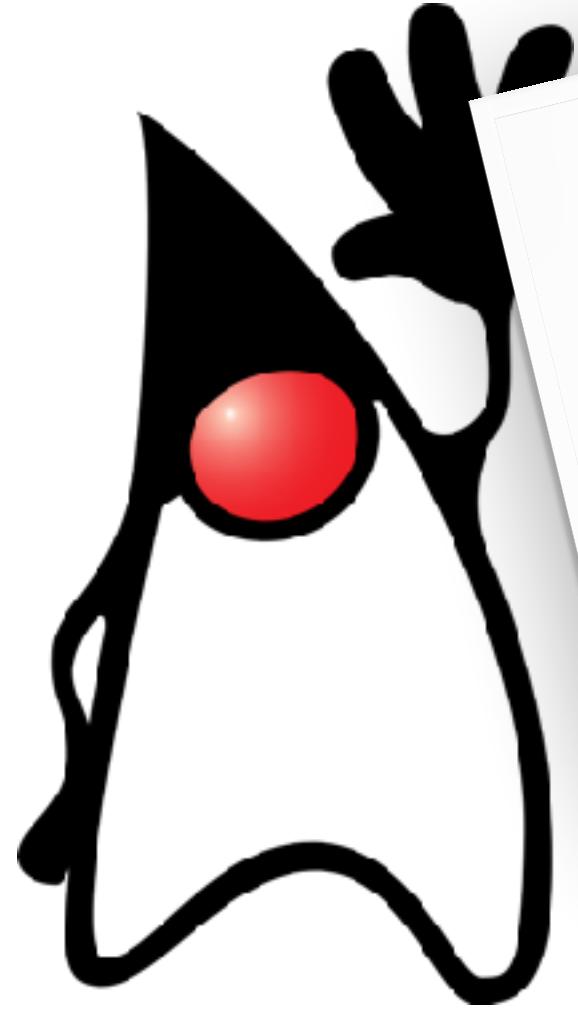
Scopes are important



module-info.java

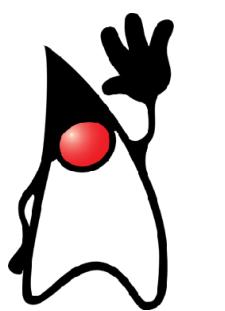
- strict standard Syntax
- javac and java enforce it

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



Part 3: Best of Both Worlds

Using Gradle not (!) for



Module Definition

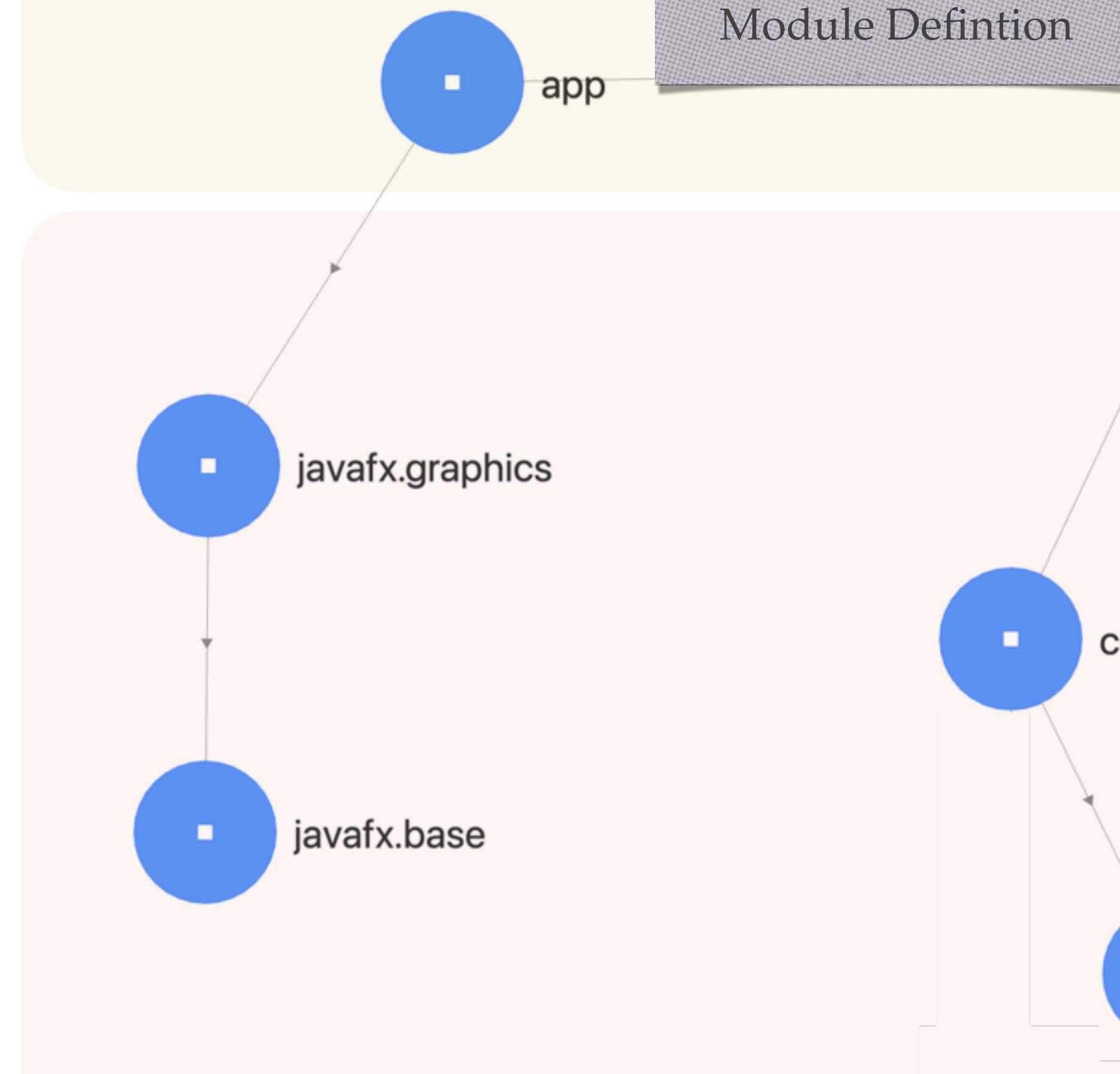
but for



Module Discovery

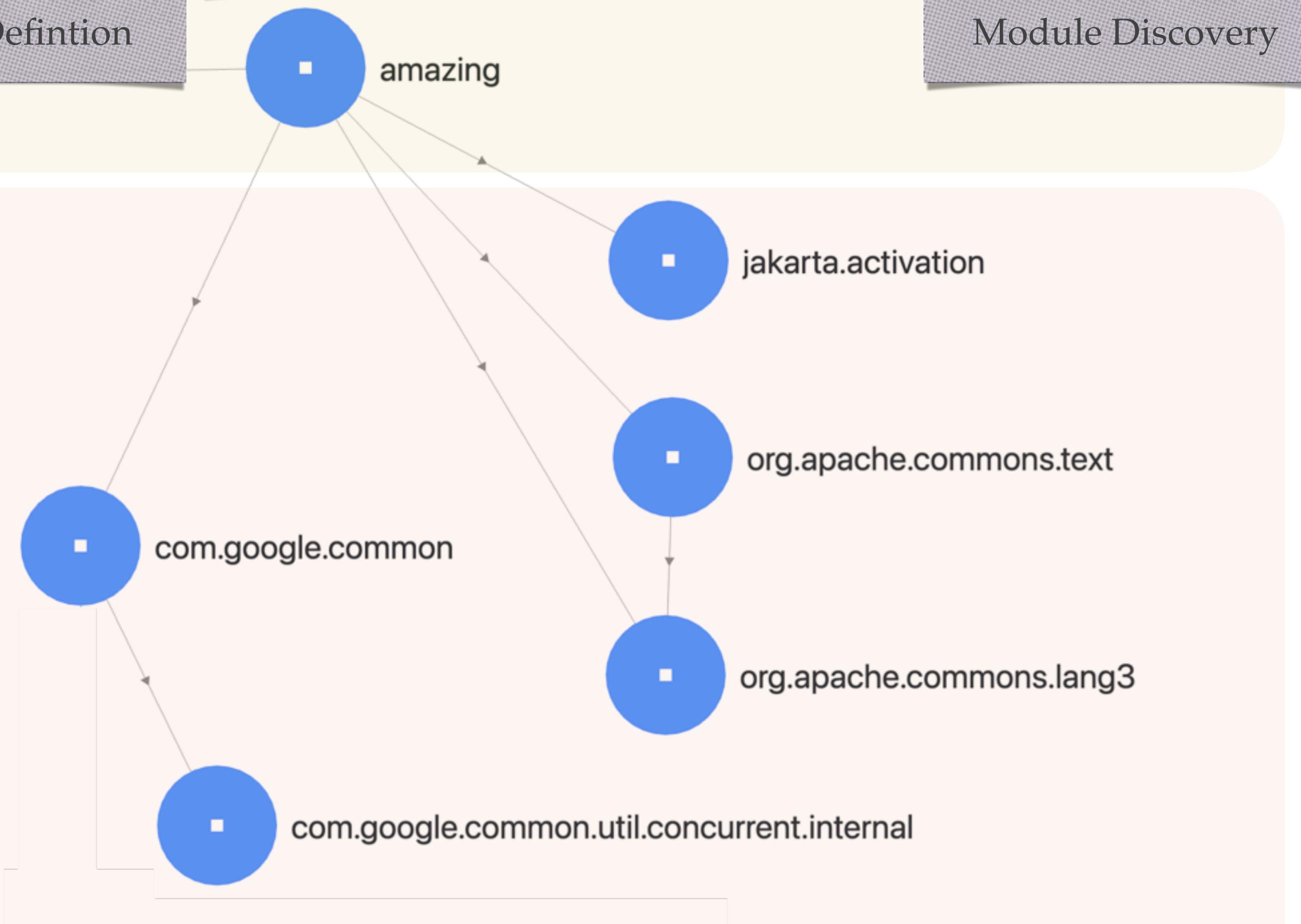
Essential Complexity in Java's Modularity #1

Scopes are important



Essential Complexity in Java's Modularity #2

Multiple Versions/Variants of 3rd party Jars



module-info.java

- strict standard Syntax
- javac and 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-module-system

- 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)

```
plugins {
    `kotlin-dsl`
}

repositories {
    gradlePluginPortal()
}

dependencies {
    implementation("com.autonomousapps:dependency-analysis-gradle-plugin:1.32.0")
    implementation("com.gradle:velocity-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

java-module-system ~/projects/gradle
gradle
plugins
src
build.gradle.kts
versions [java-module-system]
build.gradle.kts
modules
settings.gradle.kts



Project Structure

java-module-system – build.gradle.kts (:versions)

```
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")  
}
```

Project ▾

java-module-system ~/projects/gradle
 gradle
 plugins
 src
 build.gradle.kts
 versions [java-module-system]
 build.gradle.kts
 modules
 settings.gradle.kts



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

java-module-system – compile-and-test.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("java")  
    id("org.gradlex.java-module-testing")  
}  
  
java {  
    toolchain.languageVersion = JavaLanguageVersion.of(21)  
}  
  
tasks.withType<JavaCompile>().configureEach {  
    options.encoding = "UTF-8"  
}  
  
tasks.withType<Test>().configureEach {  
    maxParallelForks = 4  
    maxHeapSize = "1g"  
}
```



Other Build Concerns

java-module-system – targets.gradle.kts [plugins.main]

Project

java-module-system ~/projects/gradle/howto/java-mod
 └── gradle
 └── plugins
 └── main
 └── kotlin
 └── 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-14"))
```



OS / Arch Specifics

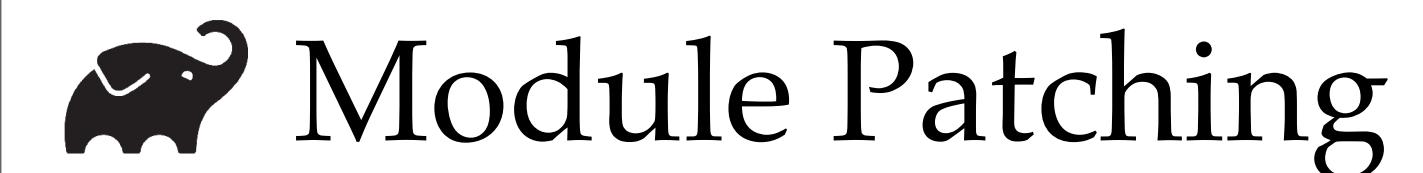
java-module-system – metadata-patch.gradle.kts [plugins.main]

Project

java-module-system ~/projects/gradle/howto/
gradle
 plugins
 src
 main
 kotlin
 build-performance.settings.gradle
 compile-and-test.gradle.kts
 dependency-analysis.gradle.kts
 java-module.gradle.kts
 metadata-patch.gradle.kts
 module-locations.settings.gradle
 targets.gradle.kts
 build.gradle.kts
 versions [java-module-system.versions]
 build.gradle.kts
modules
settings.gradle.kts

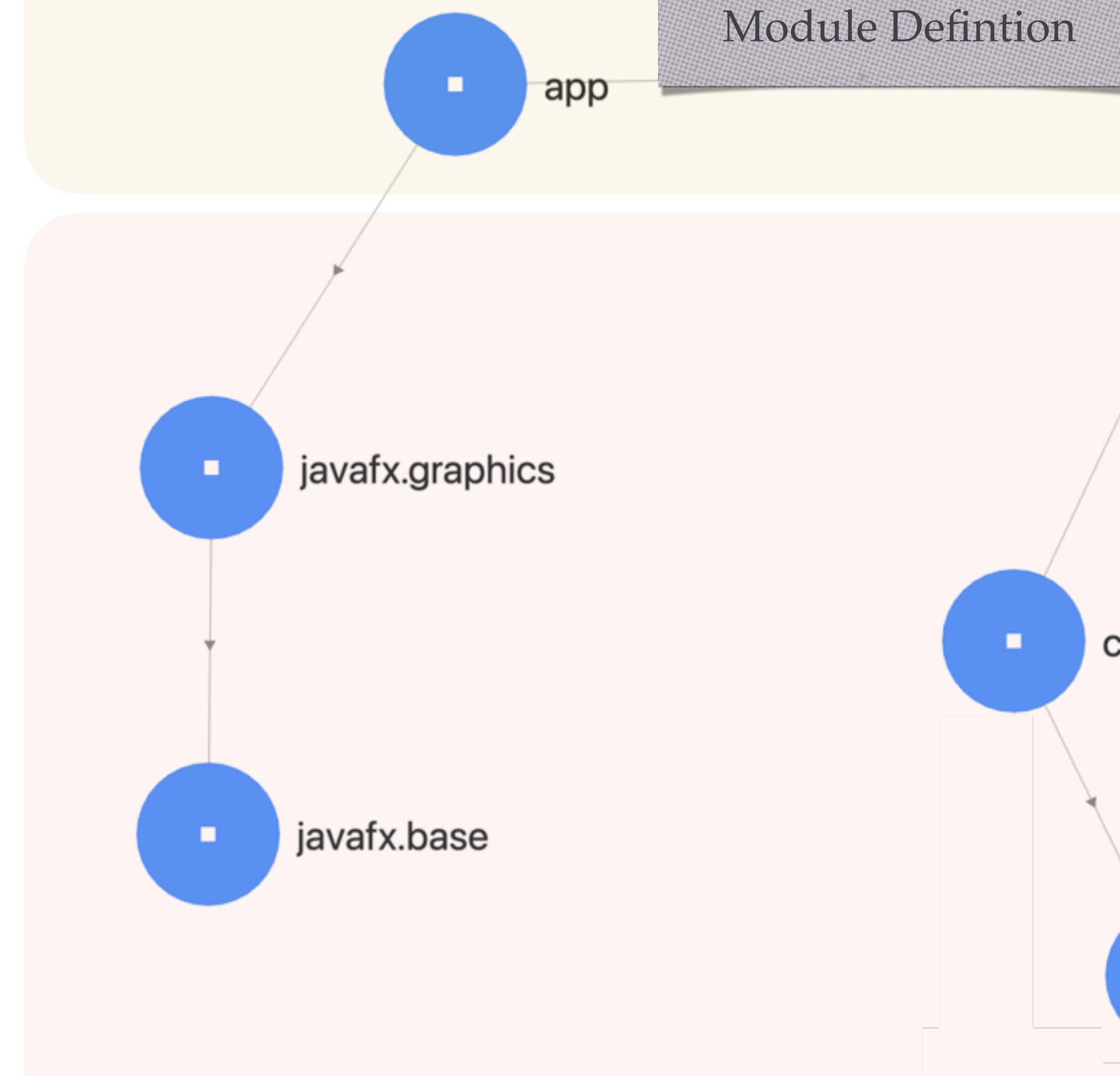
```
jvmDependencyConflicts { // Patch Maven/Gradle metadata (*.pom / *.module) ✓
    patch {
        module("com.google.guava:guava") {
            removeDependency("com.google.code.findbugs:jsr305")
            removeDependency("org.checkerframework:checker-qual")
            removeDependency("com.google.errorprone:error_prone_annotations")
        }
        listOf("base", "graphics", "controls").forEach { jfxModule ->
            module("org.openjfx:javafx-$jfxModule") {
                addTargetPlatformVariant("linux", OperatingSystemFamily.LINUX)
                addTargetPlatformVariant("mac", OperatingSystemFamily.MACOS)
                addTargetPlatformVariant("mac-aarch64", OperatingSystemFamily.MACOS_ARM)
                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:failureaccess", "com.google.errorprone")
}
```



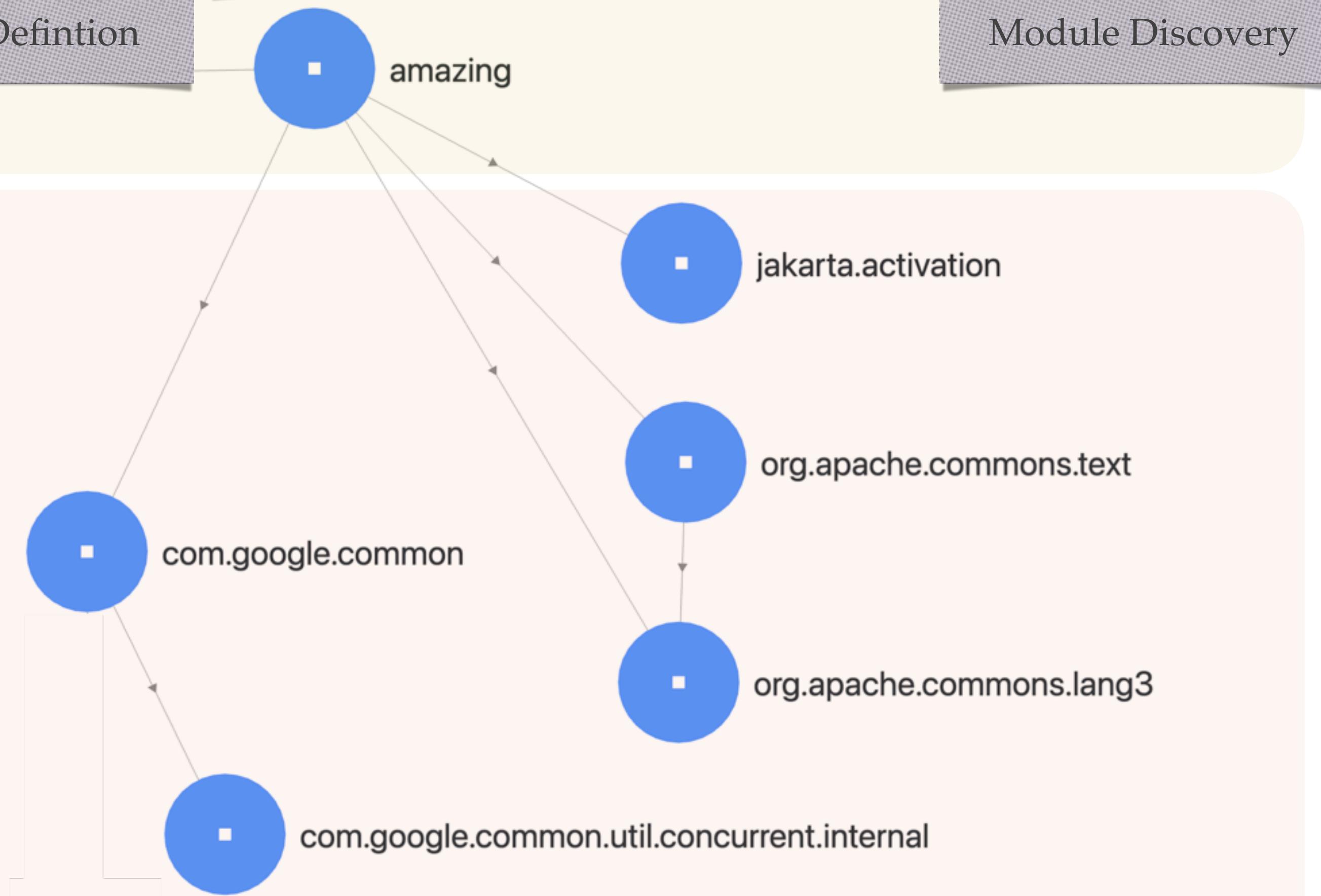
Essential Complexity in Java's Modularity #1

Scopes are important



Essential Complexity in Java's Modularity #2

Multiple Versions/Variants of 3rd party Jars

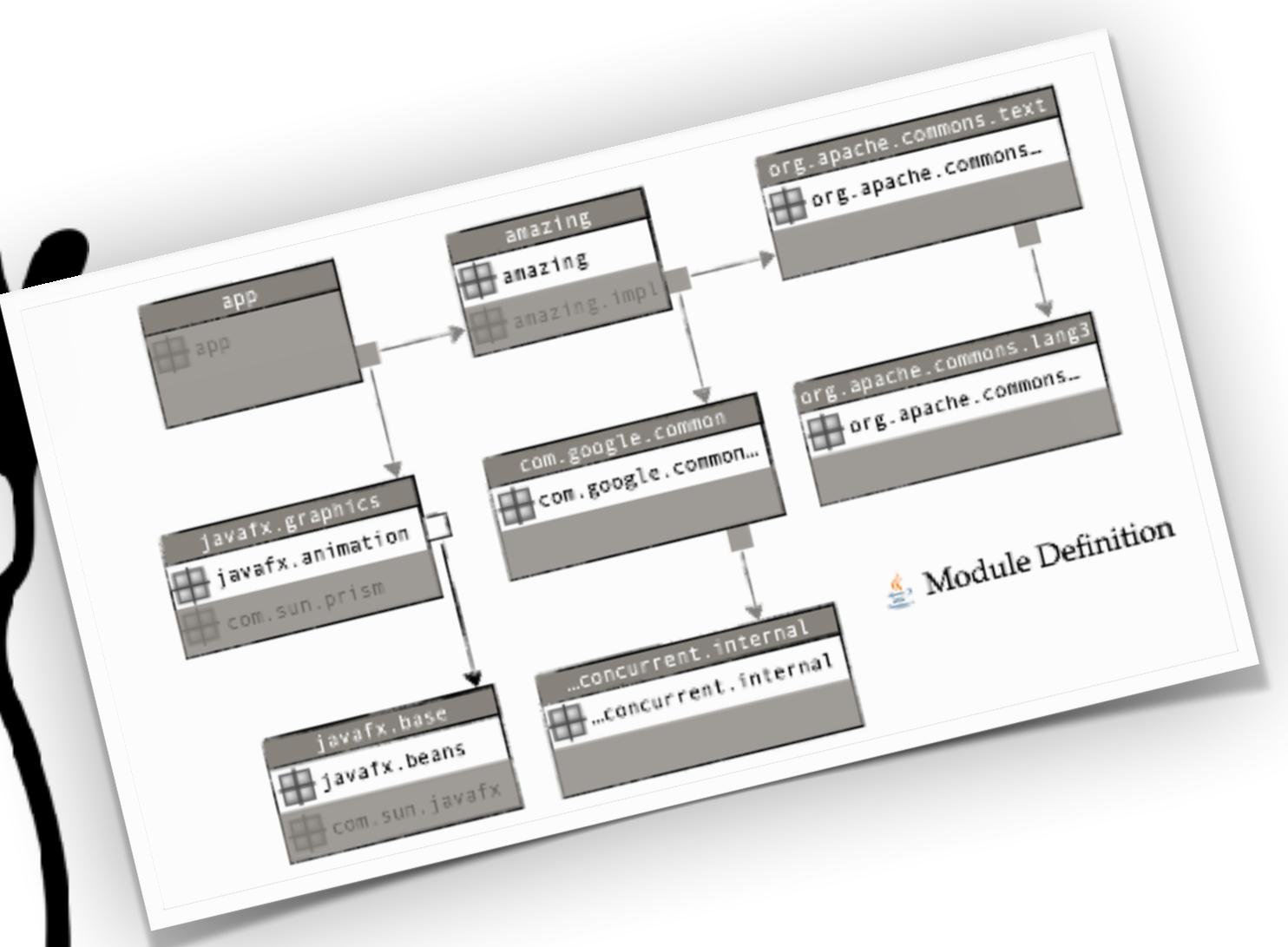
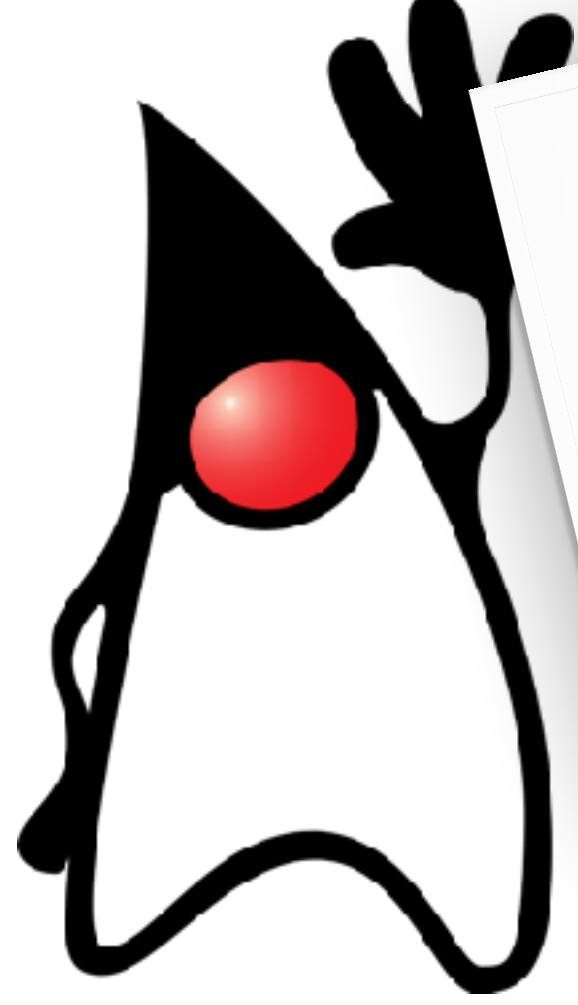


module-info.java
- strict standard Syntax
- javac and java enforce it

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

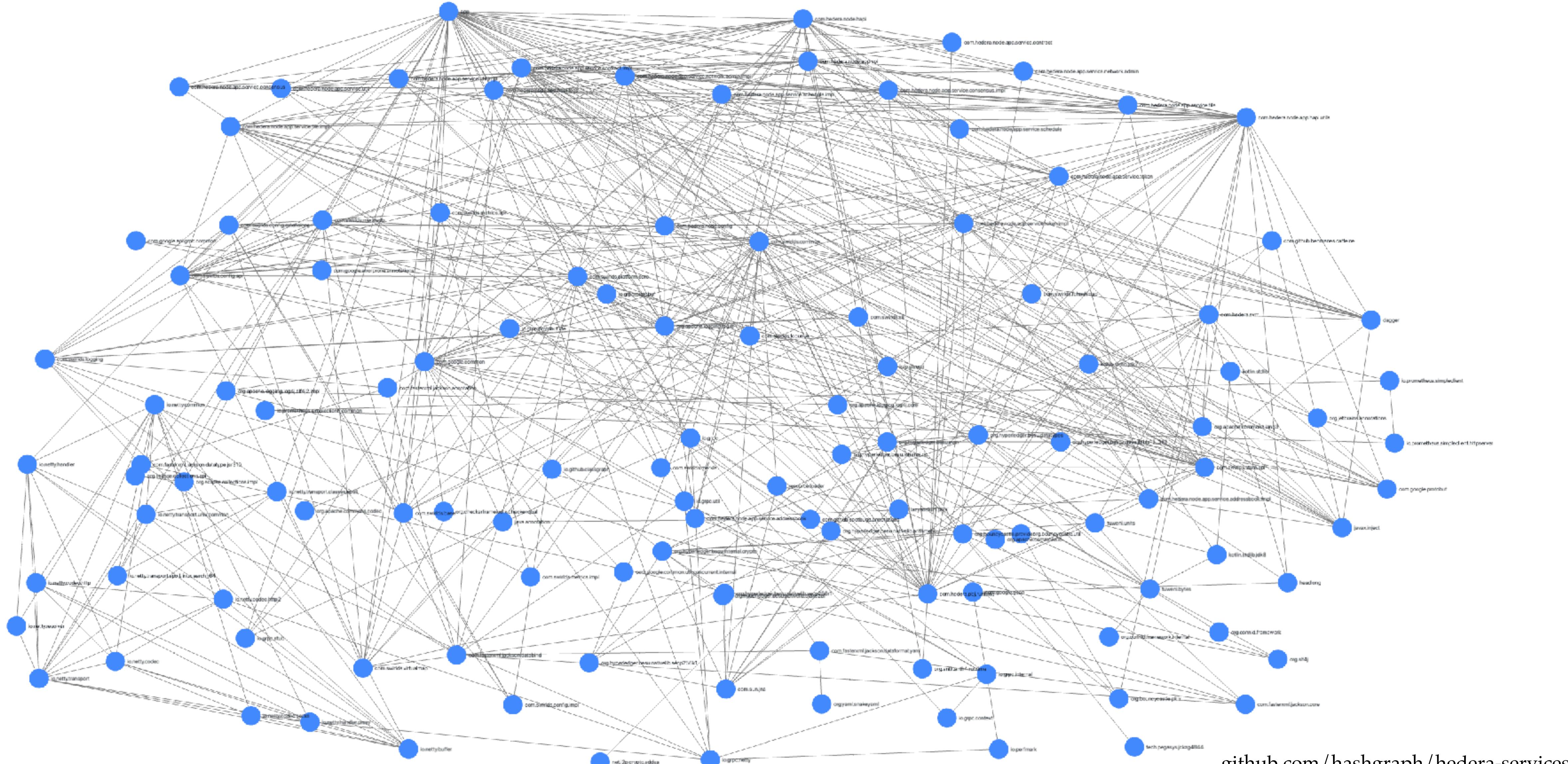
Use Gradle's **variant-aware**
conflict detection and
resolution

Central patch file(s) to
enrich / correct metadata



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

The screenshot shows a YouTube channel page for 'onepiece.Software by Jendrik Johannes'. The channel has 1340 subscribers. The main navigation bar includes 'ÜBERSICHT', 'VIDEOS', 'PLAYLISTS', 'COMMUNITY', 'KANÄLE', and 'KANALINFO'. Below the navigation, a section titled 'Java Modularity' shows four video thumbnails:

- 'Understanding Gradle #26 Java Modularity (Part 1)' - 18:15
- 'Understanding Gradle #27 Java Modularity Multiple Compile Classpaths' - 18:20
- 'Understanding Gradle #28 (Part 3) Java Modularity Clean Compile Classpaths with the Dependency Analysis Plugin' - 13:11
- 'Understanding Gradle #29 (Part 4) Java Modularity Detect and Resolve Collisions in Classpath' - 13:11

Below the thumbnails, there are two rows of video cards:

Video Title	Length	Views	Last Update
Understanding Gradle #26 – The Classpath	18:15	559	vor 2 Monaten
Untertitel			
Understanding Gradle #27 – Multiple Compile Classpaths	18:20	onepiece.Software by Jendrik Jo... 280 Aufrufe • vor 1 Monat	
Untertitel			
Understanding Gradle #28 – Clean Compile Classpaths...	13:11	onepiece.Software by Jendrik Jo... 297 Aufrufe • vor 2 Wochen	
Untertitel			
Understanding Gradle #29 (Part 4) Java Modularity Detect and Resolve Collisions in Classpath	13:11	onepiece.Software by Jendrik Jo... 95 Aufrufe • vor 1 Tag	
Untertitel			

youtube.com/@jjohannes

The screenshot shows the GitHub organization 'GradleX Project'. It has 35 repositories, 52 stars, and 1340 subscribers. The organization's name is 'Gradle How-To GitHub repositories'. It lists several repositories:

- gradle-plugins-howto How to write Gradle plugins - answers to questions and alternative implementation solutions
- gradle-project-setup-howto How to structure a growing Gradle project with smart dependency management?
- idiomatic-gradle How to idiomatically structure a large build
- gradle-demos A collection of samples demonstrating how to do different things in Gradles

It also lists 'Gradle Plugins maintained by me in the GradleX project':

- build-parameters Compile-safe access to parameters supplied to a Cradle build
- java-ecosystem-capabilities Adds Capabilities to well-known Components hosted on Maven Central
- extra-java-module-info Add Java Module information to legacy Java libraries
- java-module-dependencies Makes Gradle respect the dependencies defined in 'module-info.java' files
- java-module-testing Test Java Modules (whitebox and blackbox) without the hassle

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