Modularizing Projects

with the

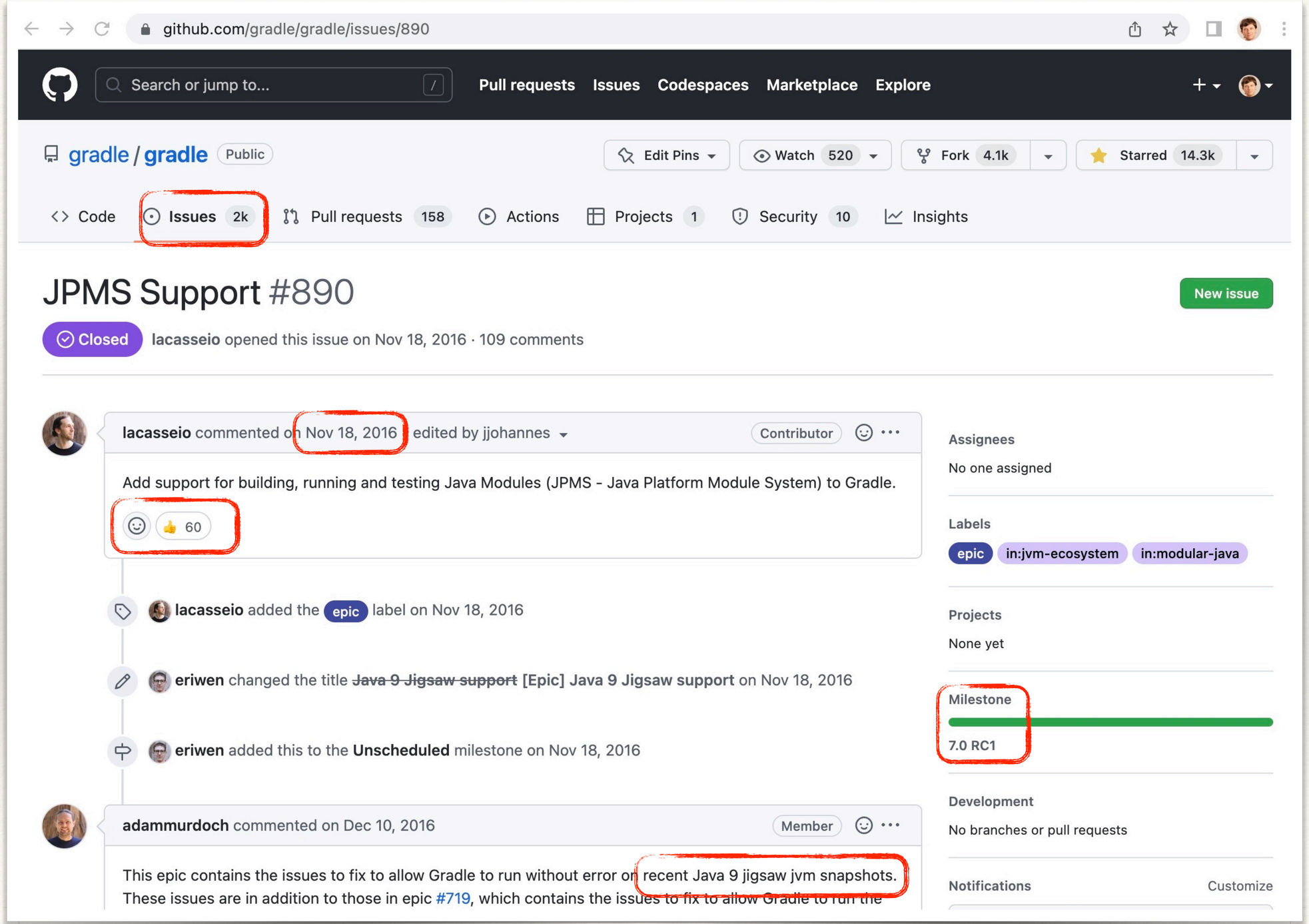
Java Module System

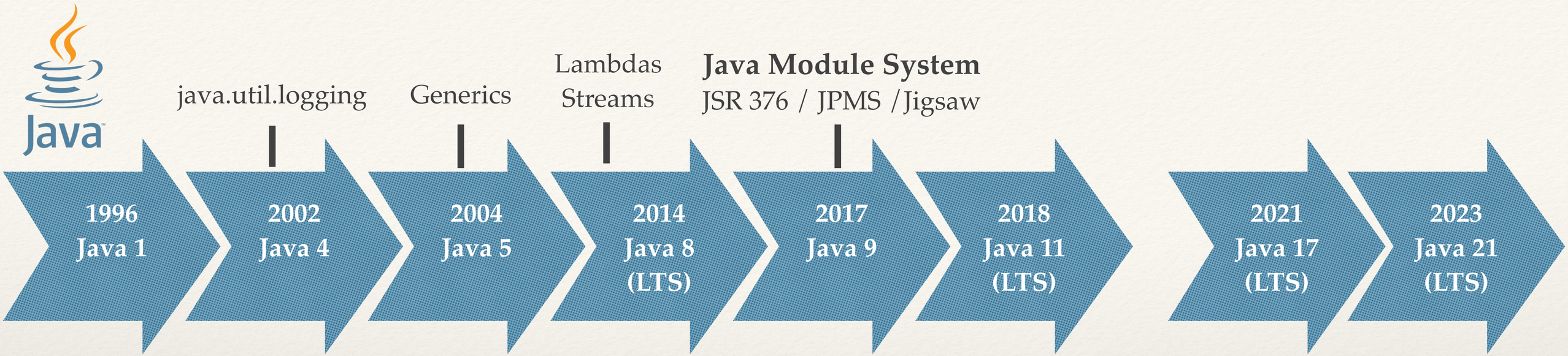
and

Gradle

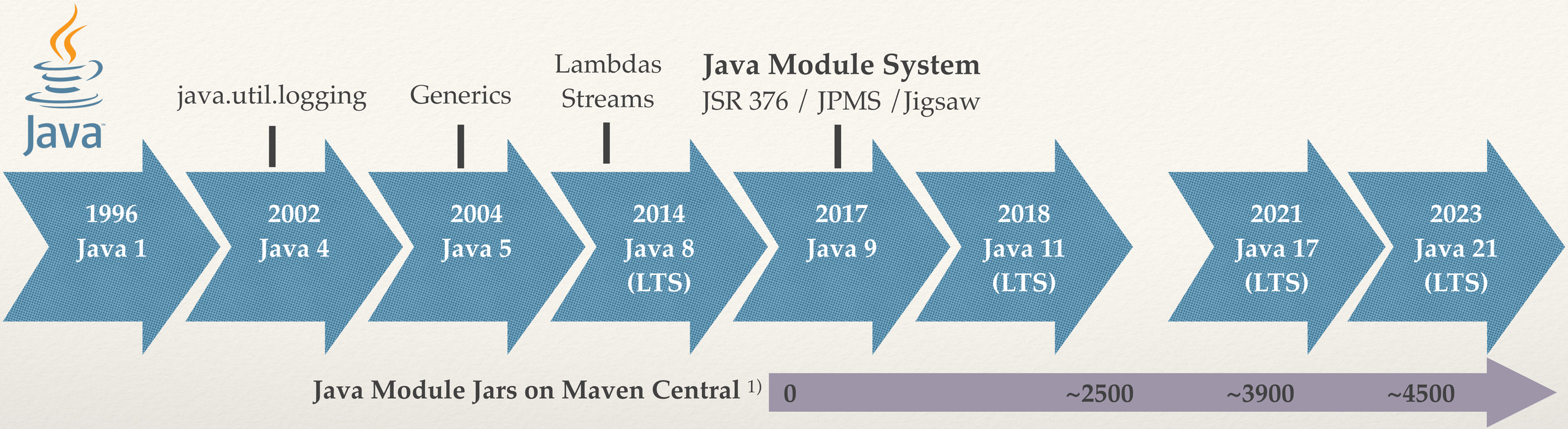
Jendrik Johannes

2022


github.com/gradle/gradle/issues/890
Search or jump to...
Pull requests
Issues
Codespaces
Marketplace
Explore
gradle / gradle
Public
Edit Pins
Watch 520
Fork 4.1k
Starred 14.3k
Code
Issues 2k
Pull requests 158
Actions
Projects 1
Security 10
Insights
JPMS Support #890
New issue
Closed
lacasseio opened this issue on Nov 18, 2016 · 109 comments
lacasseio commented on Nov 18, 2016 edited by jjohannes
Contributor
Add support for building, running and testing Java Modules (JPMS - Java Platform Module System) to Gradle.
60
lacasseio added the epic label on Nov 18, 2016
eriwen changed the title Java 9 Jigsaw support [Epic] Java 9 Jigsaw support on Nov 18, 2016
eriwen added this to the Unscheduled milestone on Nov 18, 2016
adammurdoch commented on Dec 10, 2016
Member
This epic contains the issues to fix to allow Gradle to run without error on recent Java 9 jigsaw jvm snapshots.
These issues are in addition to those in epic #719, which contains the issues to fix to allow Gradle to run the
Assignees
No one assigned
Labels
epic
in:jvm-ecosystem
in:modular-java
Projects
None yet
Milestone
7.0 RC1
Development
No branches or pull requests
Notifications
Customize



1) github.com/sormuras/modules
2) github.com/gradle/gradle/issues/890
3) gradlex.org
4) github.com/jjohannes



1) github.com/sormuras/modules
2) github.com/gradle/gradle/issues/890
3) gradlex.org
4) github.com/jjohannes



java.util.logging

Generics

Lambdas
Streams

Java Module System
JSR 376 / JPMS / Jigsaw

1996
Java 1

2002
Java 4

2004
Java 5

2014
Java 8
(LTS)

2017
Java 9

2018
Java 11
(LTS)

2021
Java 17
(LTS)

2023
Java 21
(LTS)

Java Module Jars on Maven Central [1]

0 ~2500 ~3900 ~4500



2012
Gradle 1.0

2017
Gradle 3.4

2020
Gradle 6.4

2021
Gradle 7.0

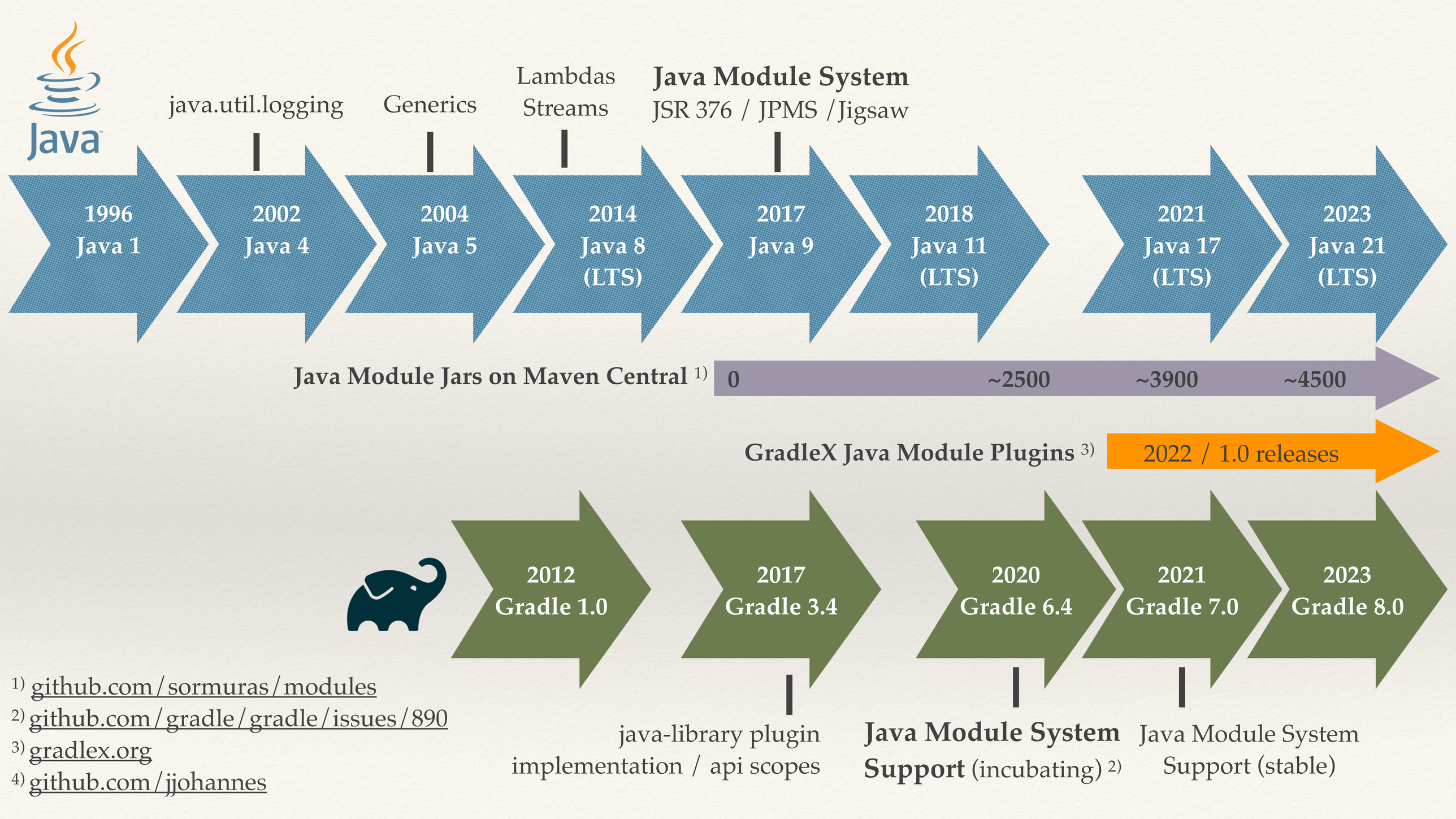
2023
Gradle 8.0

java-library plugin
implementation / api scopes

Java Module System Support (incubating) [2]

Java Module System Support (stable)

1) github.com/sormuras/modules
2) github.com/gradle/gradle/issues/890
3) gradlex.org
4) github.com/jjohannes



1) github.com/sormuras/modules
2) github.com/gradle/gradle/issues/890
3) gradlex.org
4) github.com/jjohannes



java.util.logging

Generics

Lambdas
Streams

Java Module System
JSR 376 / JPMS / Jigsaw

1996
Java 1

2002
Java 4

2004
Java 5

2014
Java 8
(LTS)

2017
Java 9

2018
Java 11
(LTS)

2021
Java 17
(LTS)

2023
Java 21
(LTS)

Java Module Jars on Maven Central [1)]

0 ~2500 ~3900 ~4500

GradleX Java Module Plugins [3)]

2022 / 1.0 releases



4)



2012
Gradle 1.0

2017
Gradle 3.4

2020
Gradle 6.4

2021
Gradle 7.0

2023
Gradle 8.0

java-library plugin
implementation / api scopes

Java Module System Support (incubating) [2)]

Java Module System Support (stable)

1) github.com/sormuras/modules
2) github.com/gradle/gradle/issues/890
3) gradlex.org
4) github.com/jjohannes


Java




```java
package my.product.app;

import static my.product.utilities.StringUtils.join;
import static my.product.utilities.StringUtils.split;

public class App {
    public static void main(String[] args) {
        Message message = new Message();
        String result = splitAndJoin(message.getMessage());
        System.out.println(result);
    }

    private static String splitAndJoin(String s) {
        return join(split(s));
    }
}
```




```java
package my.product.app;

import static my.product.utilities.StringUtils.join;
import static my.product.utilities.StringUtils.split;

public class App {
    public static void main(String[] args) {
        Message message = new Message();
        String result = splitAndJoin(message.getMessage());
        System.out.println(result);
    }

    private static String splitAndJoin(String s) {
        return join(split(s));
    }
}
```




```java
package my.product.app;

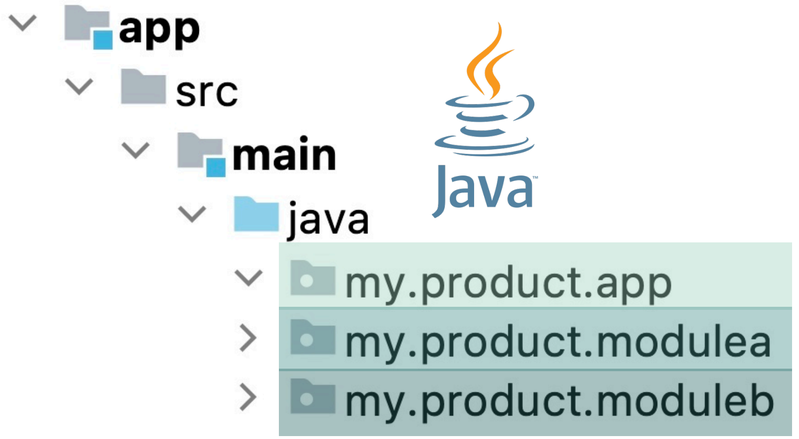
import static my.product.utilities.StringUtils.join;
import static my.product.utilities.StringUtils.split;

public class App {
    public static void main(String[] args) {
        Message message = new Message();
        String result = splitAndJoin(message.getMessage());
        System.out.println(result);
    }

    private static String splitAndJoin(String s) {
        return join(split(s));
    }
}
```

```java
package my.product.app;

class Message {
    public String getMessage() {
        return "Hello      World!";
    }
}
```




```java
package my.product.app;

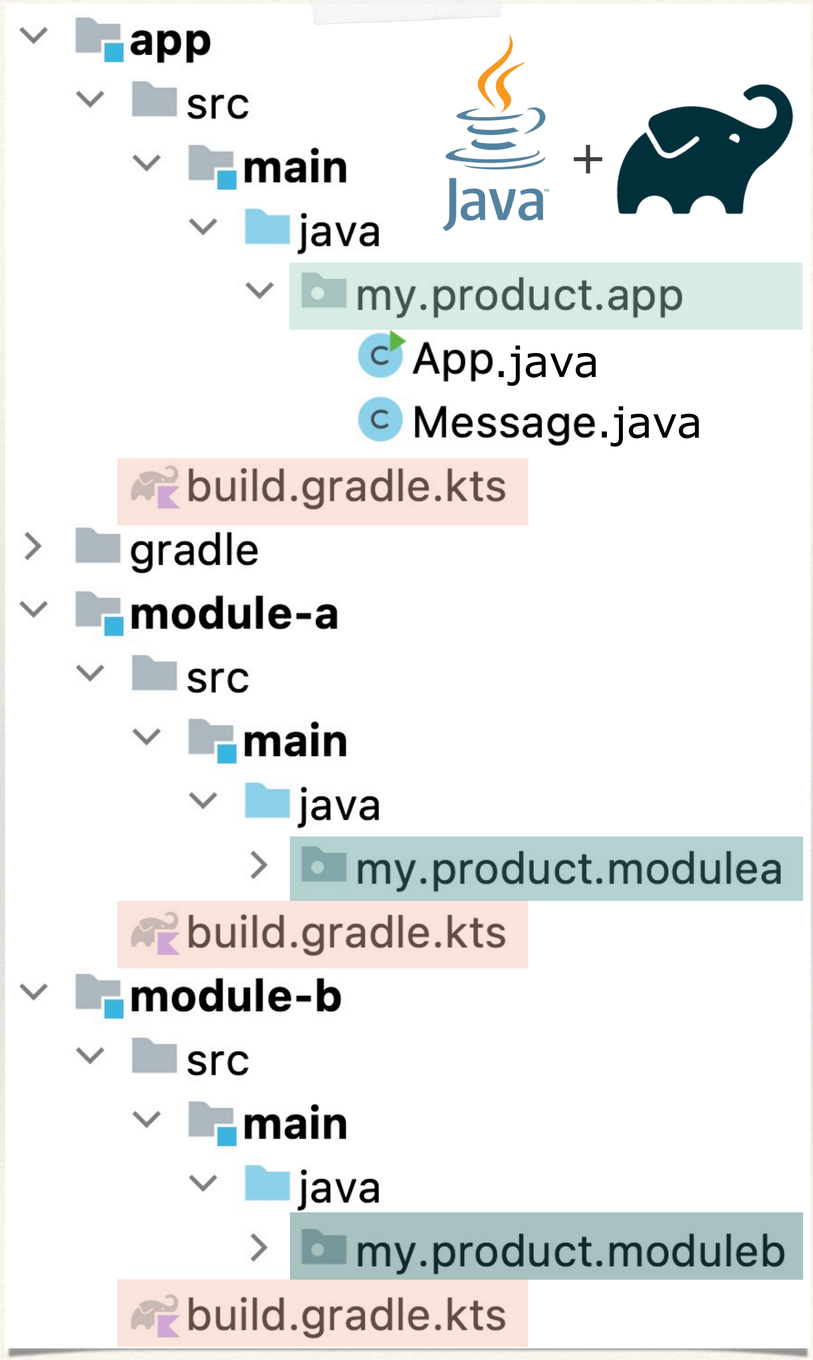
import static my.product.utilities.StringUtils.join;
import static my.product.utilities.StringUtils.split;

public class App {
    public static void main(String[] args) {
        Message message = new Message();
        String result = splitAndJoin(message.getMessage());
        System.out.println(result);
    }

    private static String splitAndJoin(String s) {
        return join(split(s));
    }
}
```

```java
package my.product.app;

class Message {
    public String getMessage() {
        return "Hello      World!";
    }
}
```




```java
package my.product.app;

import static my.product.utilities.StringUtils.join;
import static my.product.utilities.StringUtils.split;

public class App {
    public static void main(String[] args) {
        Message message = new Message();
        String result = splitAndJoin(message.getMessage());
        System.out.println(result);
    }

    private static String splitAndJoin(String s) {
        return join(split(s));
    }
}
```

```java
package my.product.app;

class Message {
    public String getMessage() {
        return "Hello      World!";
    }
}
```

Modularizing Java Projects

*(1) Modularization with the **Classpath** + Gradle*

*(2) Modularization with the **Module Path***

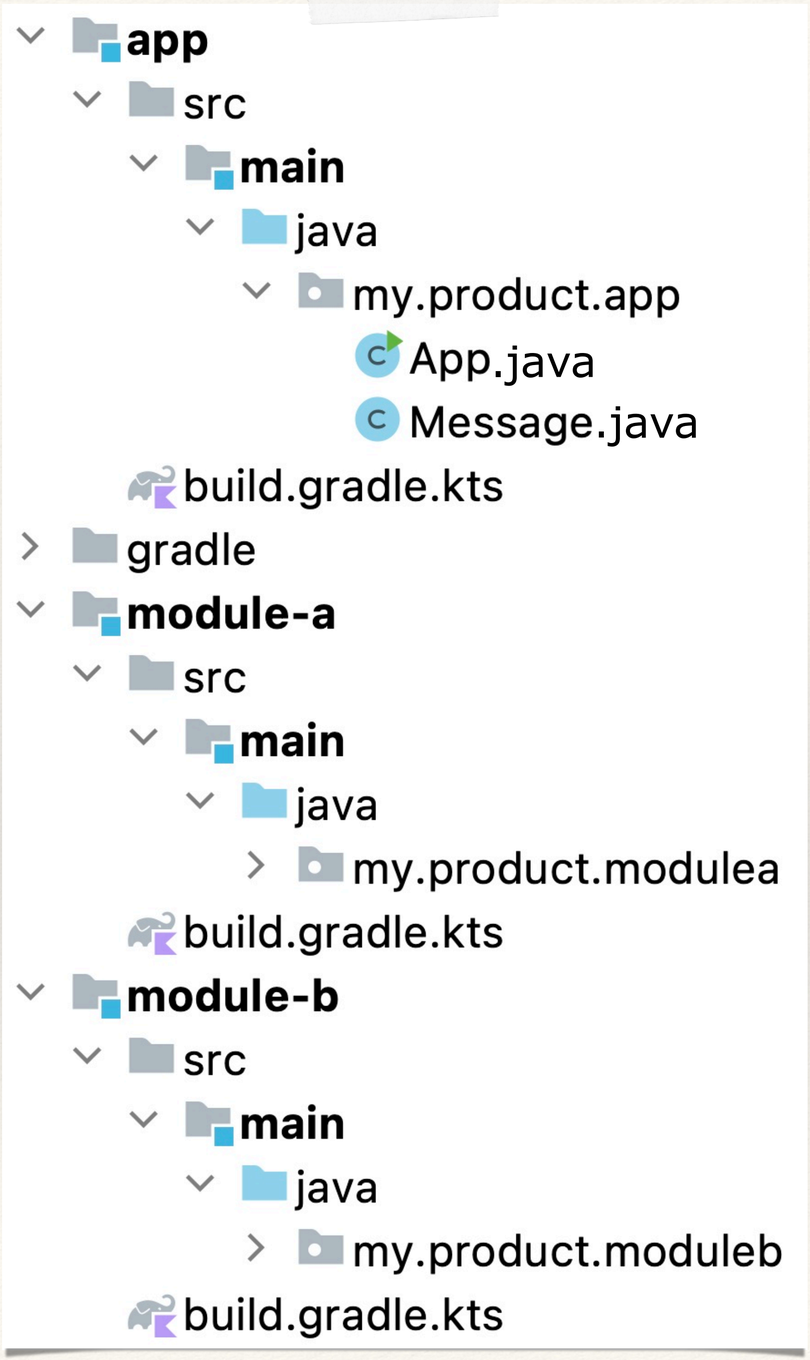
*(3) Modularization with the **Module Path** + Gradle + X*

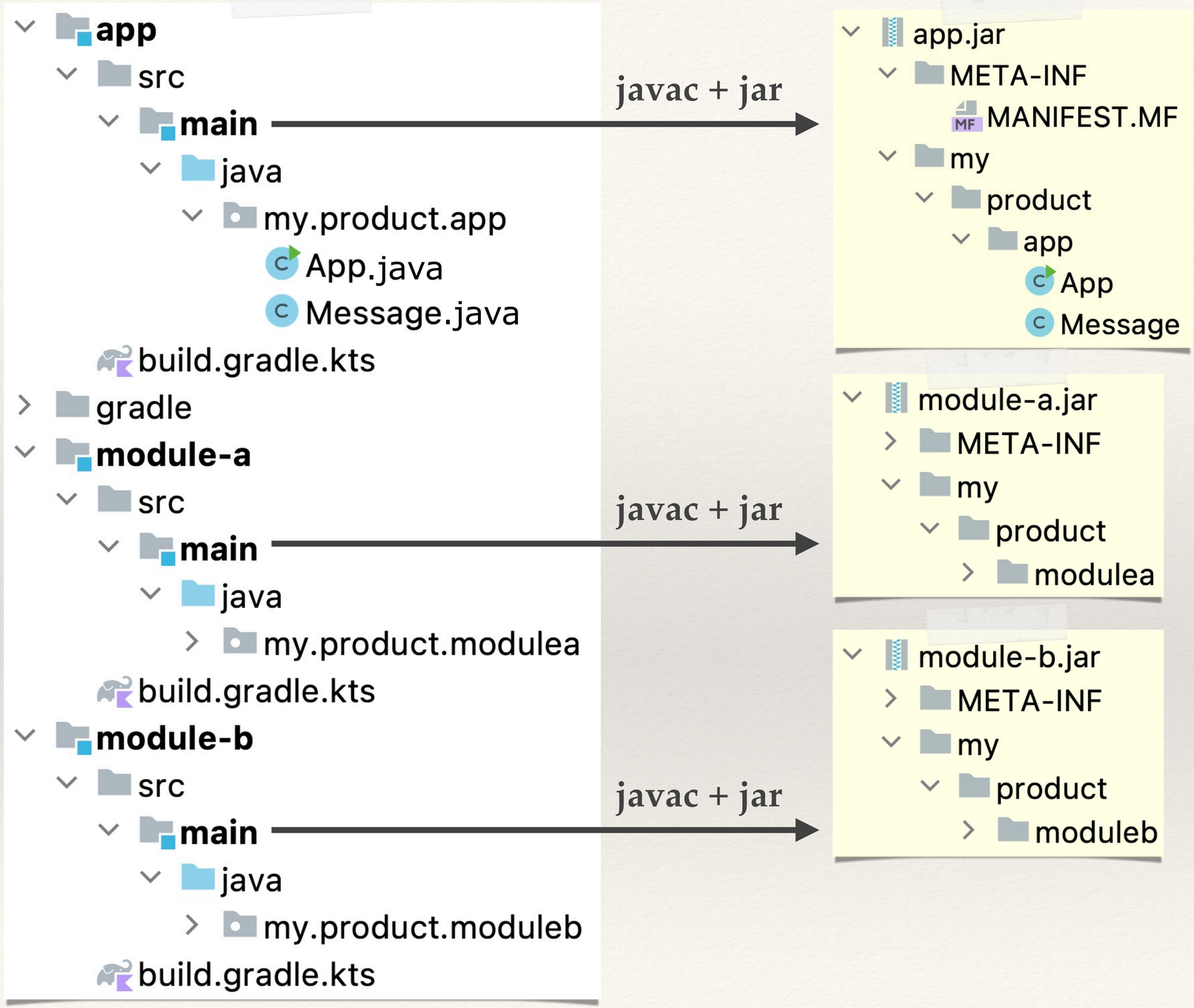
Modularizing Java Projects

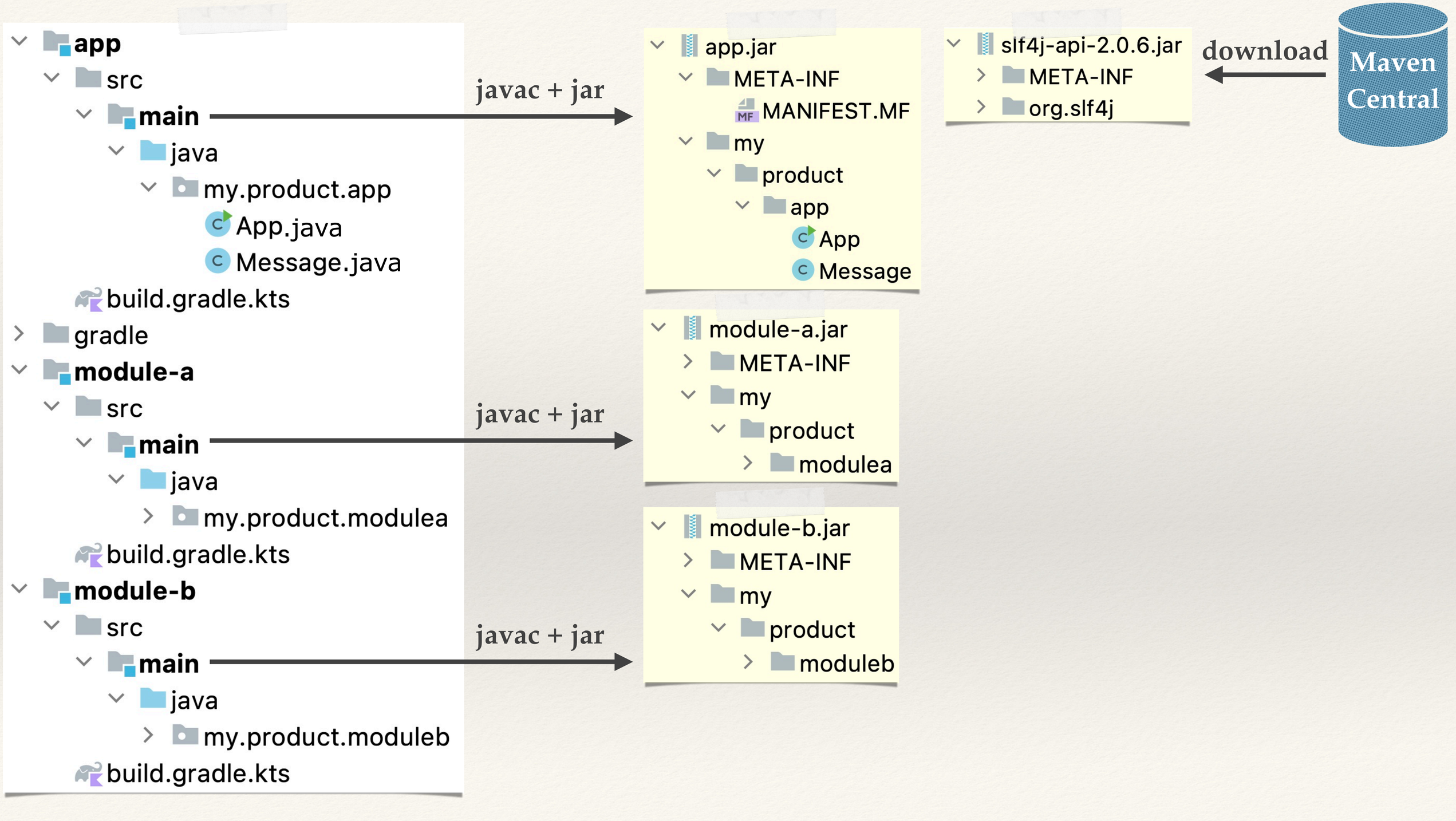
*(1) Modularization with the **Classpath** + Gradle*

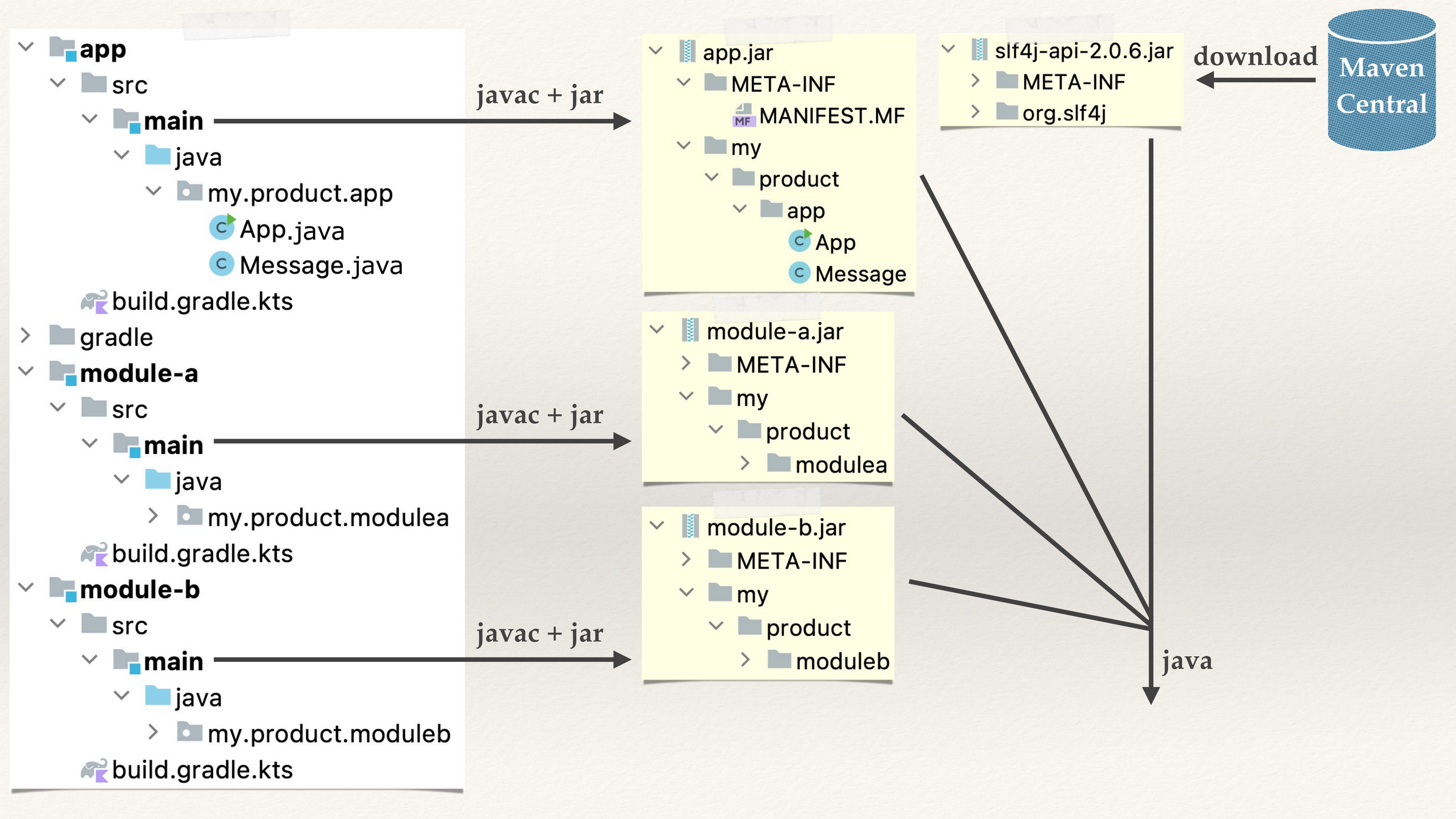
*(2) Modularization with the **Module Path***

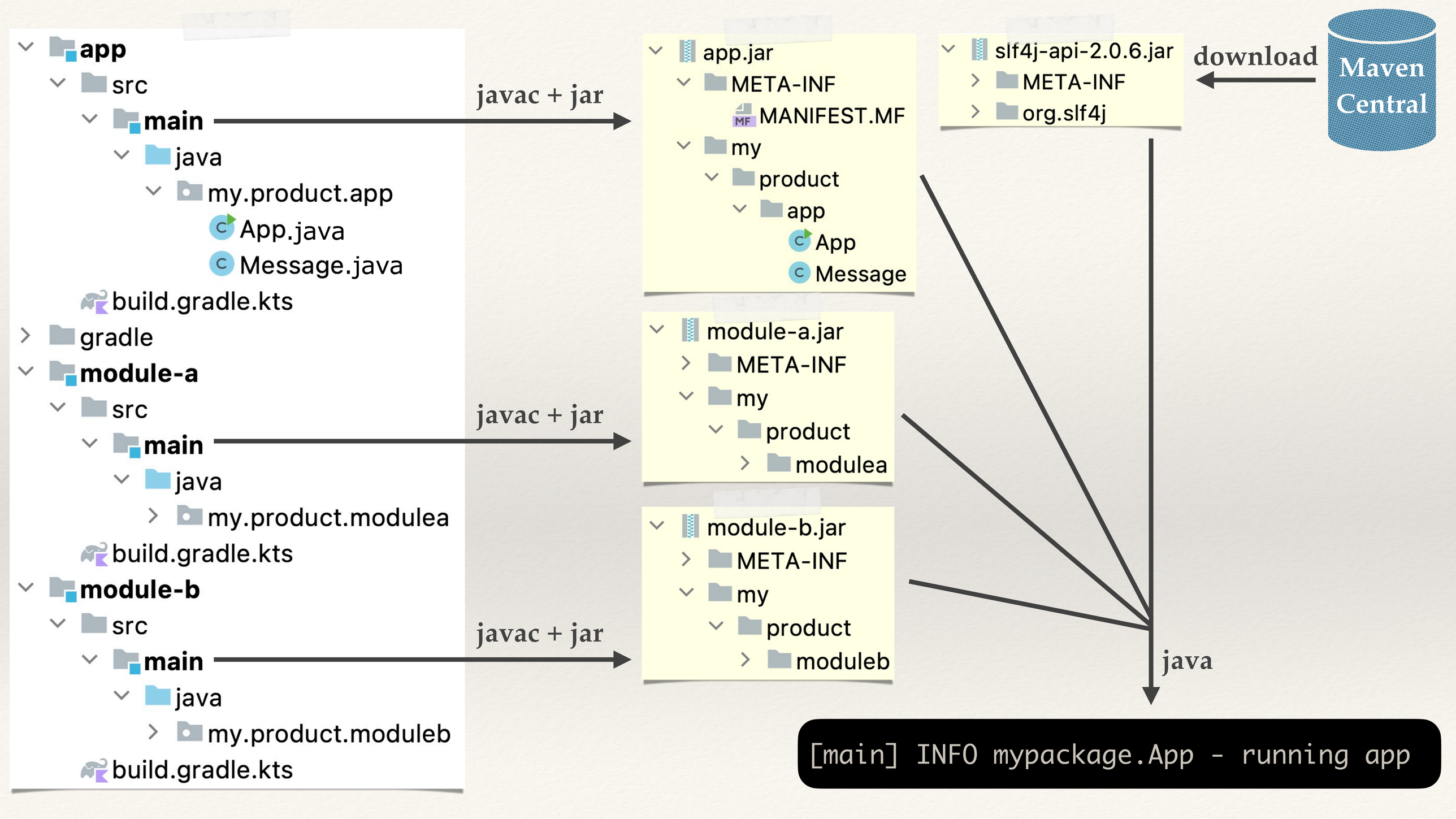
*(3) Modularization with the **Module Path** + Gradle + X*

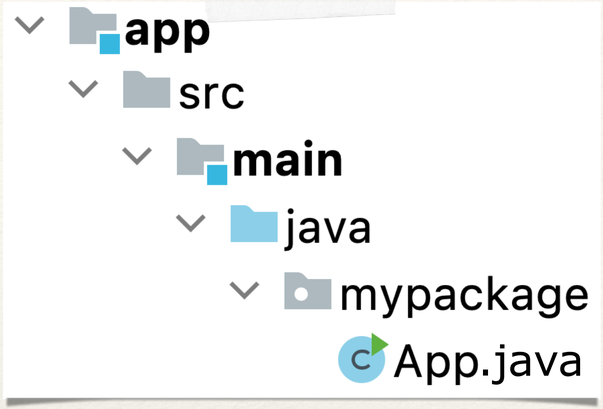

app
src
main
java
my.product.app
App.java
Message.java
build.gradle.kts
gradle
module-a
src
main
java
my.product.modulea
build.gradle.kts
module-b
src
main
java
my.product.moduleb
build.gradle.kts

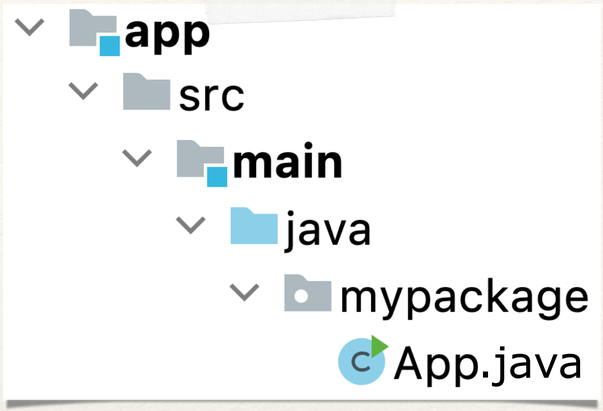

app
src
main
java
my.product.app
App.java
Message.java
build.gradle.kts
gradle
module-a
src
main
java
my.product.modulea
build.gradle.kts
module-b
src
main
java
my.product.moduleb
build.gradle.kts
javac + jar
javac + jar
javac + jar
app.jar
META-INF
MANIFEST.MF
my
product
app
App
Message
module-a.jar
META-INF
my
product
modulea
module-b.jar
META-INF
my
product
moduleb


app
src
main
java
my.product.app
App.java
Message.java
build.gradle.kts
gradle
module-a
src
main
java
my.product.modulea
build.gradle.kts
module-b
src
main
java
my.product.moduleb
build.gradle.kts
javac + jar
javac + jar
javac + jar
app.jar
META-INF
MANIFEST.MF
my
product
app
App
Message
module-a.jar
META-INF
my
product
modulea
module-b.jar
META-INF
my
product
moduleb
slf4j-api-2.0.6.jar
META-INF
org.slf4j
download
Maven Central


app
src
main
java
my.product.app
App.java
Message.java
build.gradle.kts
gradle
module-a
src
main
java
my.product.modulea
build.gradle.kts
module-b
src
main
java
my.product.moduleb
build.gradle.kts
javac + jar
javac + jar
javac + jar
app.jar
META-INF
MANIFEST.MF
my
product
app
App
Message
module-a.jar
META-INF
my
product
modulea
module-b.jar
META-INF
my
product
moduleb
slf4j-api-2.0.6.jar
META-INF
org.slf4j
download
Maven Central
java


app
src
main
java
my.product.app
App.java
Message.java
build.gradle.kts
gradle
module-a
src
main
java
my.product.modulea
build.gradle.kts
module-b
src
main
java
my.product.moduleb
build.gradle.kts
javac + jar
javac + jar
javac + jar
app.jar
META-INF
MANIFEST.MF
my
product
app
App
Message
module-a.jar
META-INF
my
product
modulea
module-b.jar
META-INF
my
product
moduleb
slf4j-api-2.0.6.jar
META-INF
org.slf4j
download
Maven Central
java
[main] INFO mypackage.App - running app

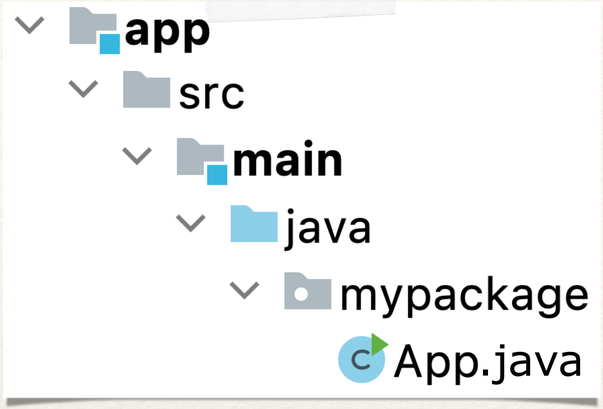

app
src
main
java
mypackage
App.java




```java
package mypackage;



public class App {
    public static void main(String[] args) {
        System.out.println("running app");
    }
}
```

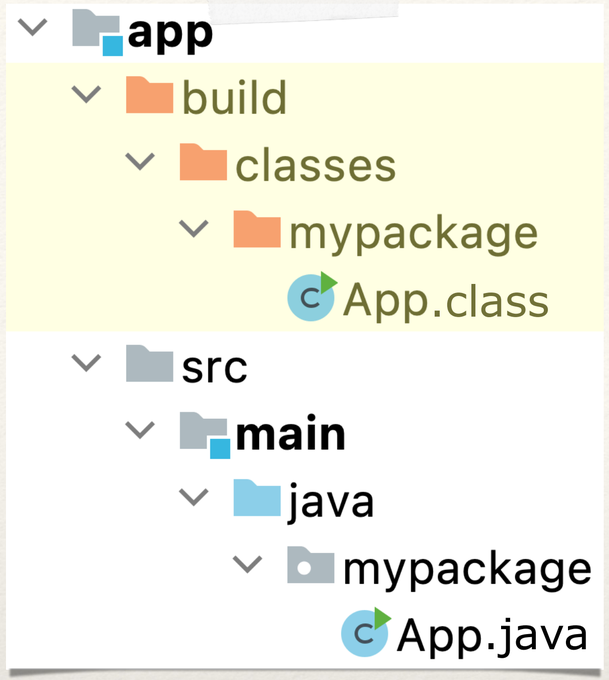



```java
package mypackage;



public class App {
    public static void main(String[] args) {
        System.out.println("running app");
    }
}
```

```
$ javac -d ../../../build/classes mypackage/App.java
```

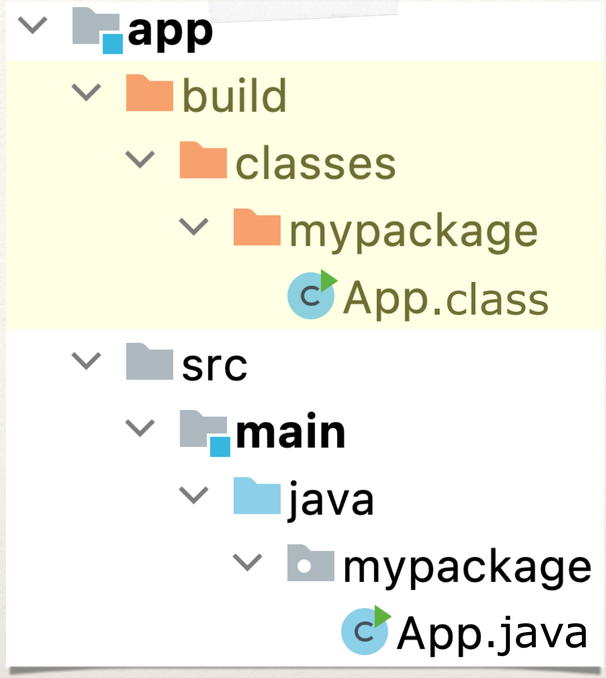



```java
package mypackage;



public class App {
    public static void main(String[] args) {
        System.out.println("running app");
    }
}
```

```
$ javac -d ../../../build/classes mypackage/App.java
```

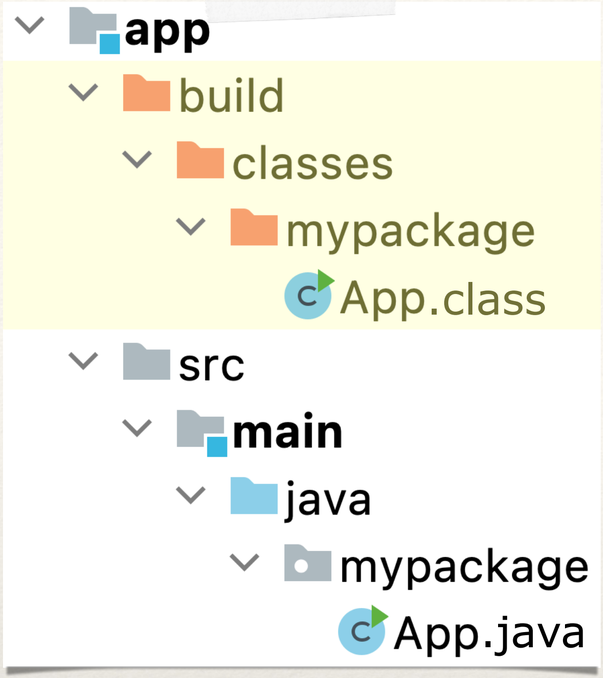



```java
package mypackage;



public class App {
    public static void main(String[] args) {
        System.out.println("running app");
    }
}
```

```
$ javac -d ../../../build/classes mypackage/App.java
$ java mypackage.App
```

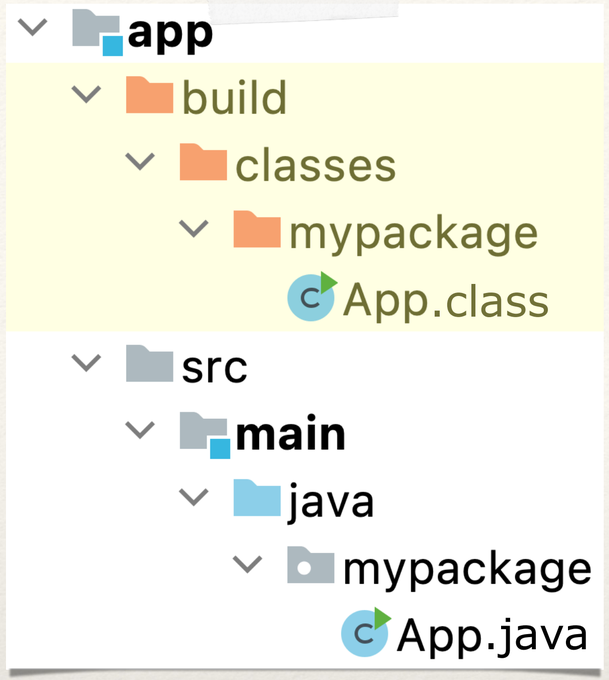



```java
package mypackage;



public class App {
    public static void main(String[] args) {
        System.out.println("running app");
    }
}
```

```
$ javac -d ../../../build/classes mypackage/App.java
$ java mypackage.App
Error: Could not find or load main class mypackage.App
Caused by: java.lang.ClassNotFoundException: mypackage.App
```

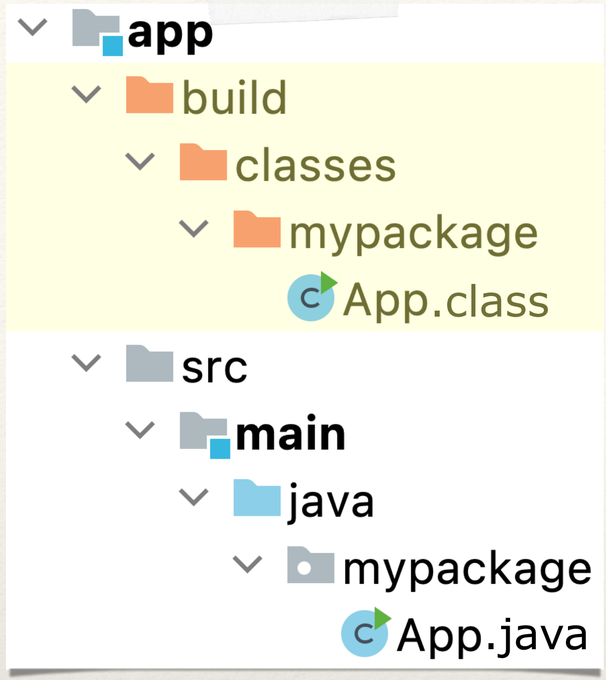



```java
package mypackage;



public class App {
    public static void main(String[] args) {
        System.out.println("running app");
    }
}
```

```
$ javac -d ../../../build/classes mypackage/App.java
$ java mypackage.App
Error: Could not find or load main class mypackage.App
Caused by: java.lang.ClassNotFoundException: mypackage.App
$ java -classpath ../../../build/classes mypackage.App
```

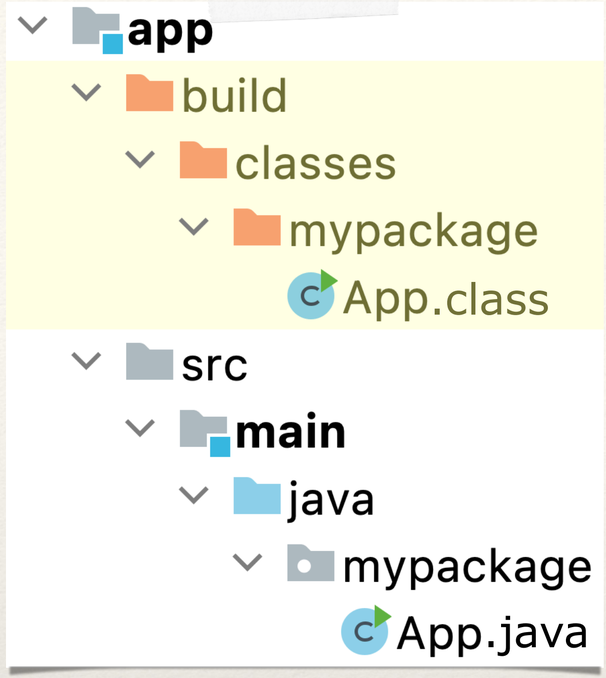



```java
package mypackage;



public class App {
    public static void main(String[] args) {
        System.out.println("running app");
    }
}
```

```
$ javac -d ../../../build/classes mypackage/App.java
$ java mypackage.App
Error: Could not find or load main class mypackage.App
Caused by: java.lang.ClassNotFoundException: mypackage.App
$ java -classpath ../../../build/classes mypackage.App
running app
```

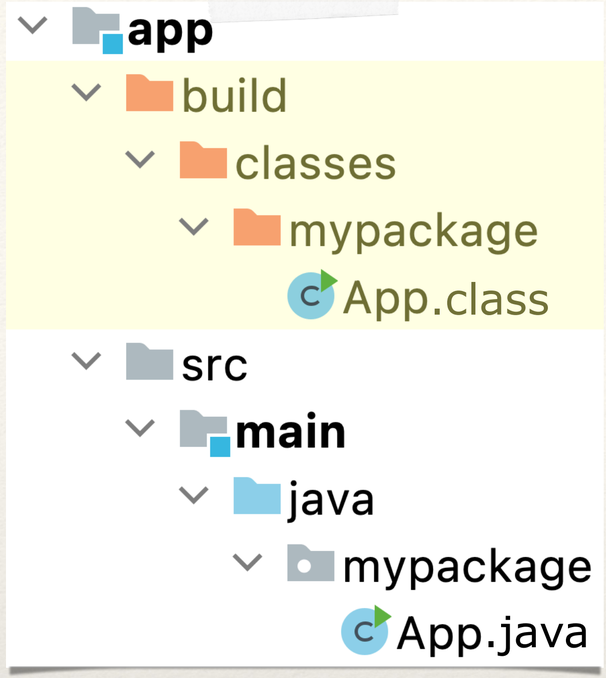



```java
package mypackage;



public class App {
    public static void main(String[] args) {
        System.out.println("running app");
    }
}
```



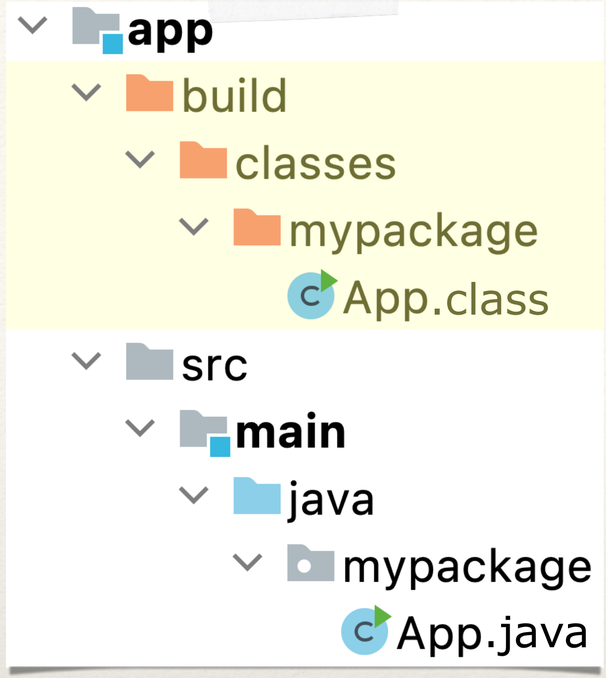



```java
package mypackage;

import org.slf4j.LoggerFactory;

public class App {
    public static void main(String[] args) {
        LoggerFactory.getLogger(App.class).info("running app");
    }
}
```



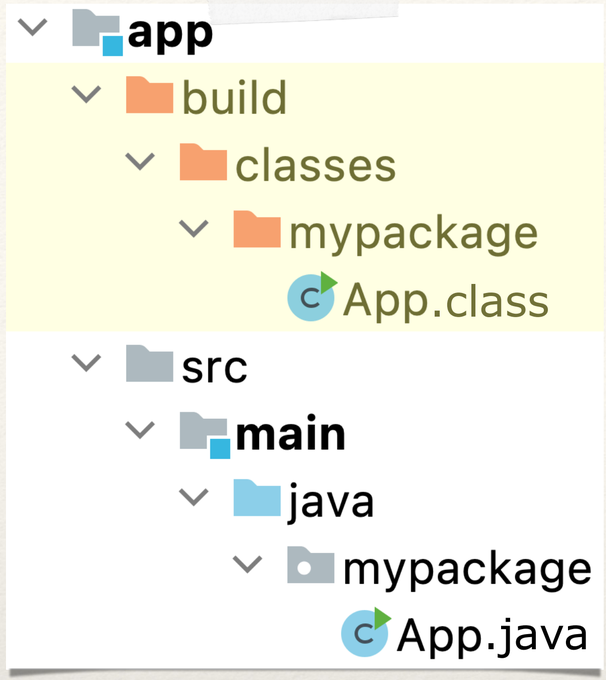



```java
package mypackage;

import org.slf4j.LoggerFactory;

public class App {
    public static void main(String[] args) {
        LoggerFactory.getLogger(App.class).info("running app");
    }
}
```

```
$ javac -d ../../../build/classes mypackage/App.java
```

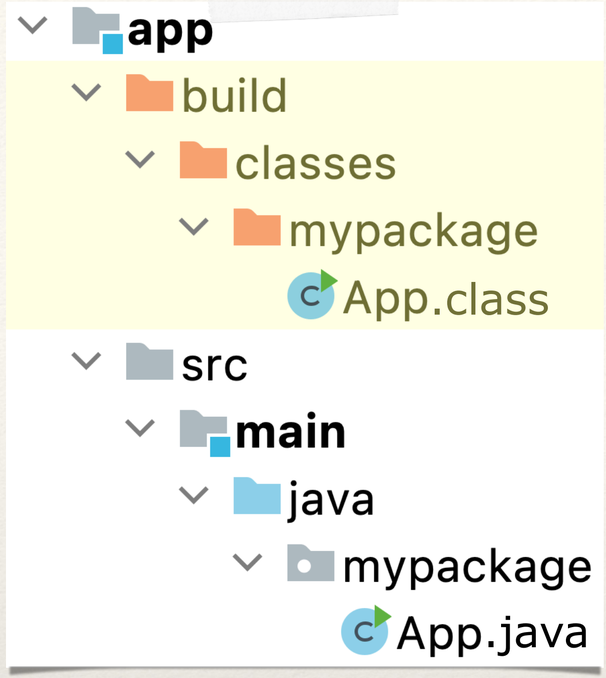



```java
package mypackage;

import org.slf4j.LoggerFactory;

public class App {
    public static void main(String[] args) {
        LoggerFactory.getLogger(App.class).info("running app");
    }
}
```

```
$ javac -d ../../../build/classes mypackage/App.java
mypackage/App.java:3: error: package org.slf4j does not exist
import org.slf4j.LoggerFactory;
                ^
mypackage/App.java:7: error: cannot find symbol
        LoggerFactory.getLogger(App.class).info("running app");
        ^
  symbol:   variable LoggerFactory
  location: class App
2 errors
```

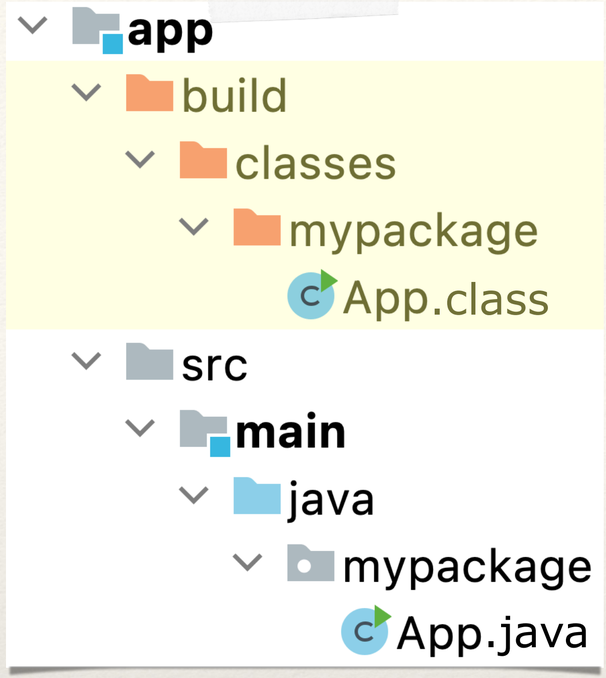



```java
package mypackage;

import org.slf4j.LoggerFactory;

public class App {
    public static void main(String[] args) {
        LoggerFactory.getLogger(App.class).info("running app");
    }
}
```



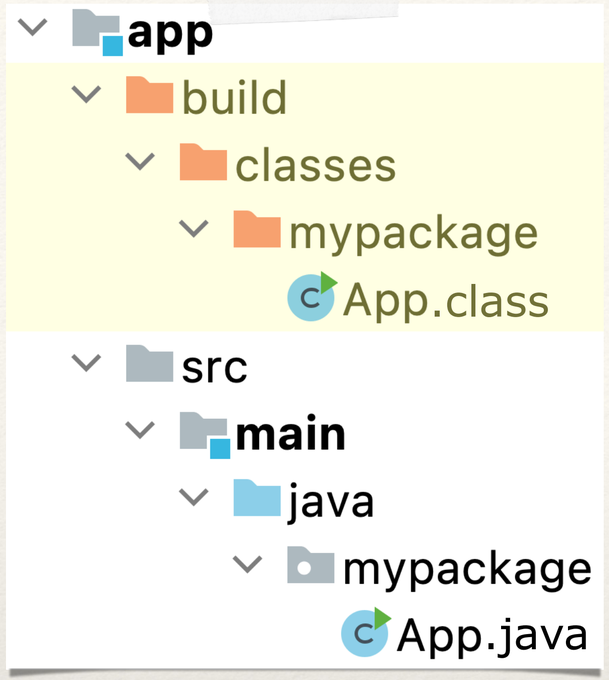



```java
package mypackage;

import org.slf4j.LoggerFactory;

public class App {
    public static void main(String[] args) {
        LoggerFactory.getLogger(App.class).info("running app");
    }
}
```

```
$ javac -d ../../../build/classes -classpath ../../../libs/slf4j-api.jar mypackage/App.java
```

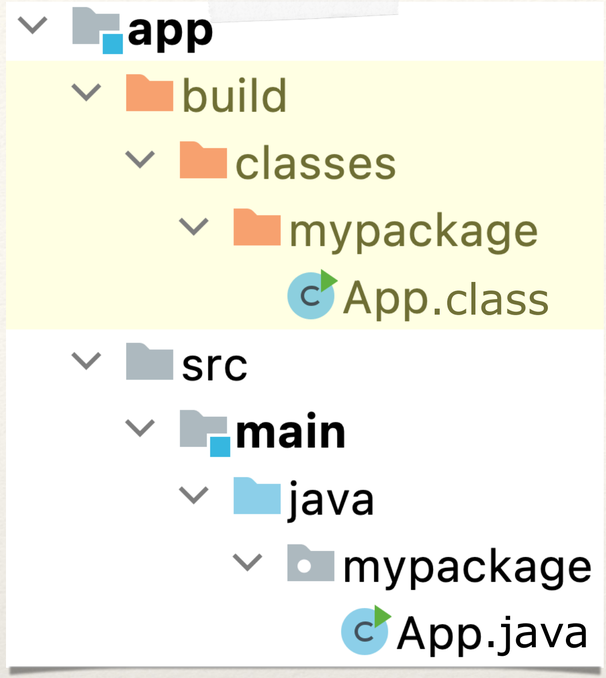



```java
package mypackage;

import org.slf4j.LoggerFactory;

public class App {
    public static void main(String[] args) {
        LoggerFactory.getLogger(App.class).info("running app");
    }
}
```

```
$ javac -d ../../../build/classes -classpath ../../../libs/slf4j-api.jar mypackage/App.java
$ java -classpath ../../../build/classes:../../../libs/slf4j-api.jar mypackage.App
```

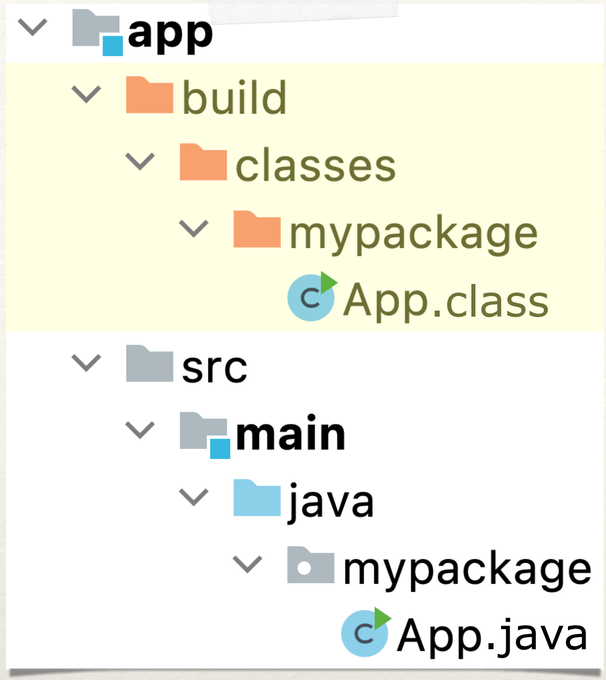



```java
package mypackage;

import org.slf4j.LoggerFactory;

public class App {
    public static void main(String[] args) {
        LoggerFactory.getLogger(App.class).info("running app");
    }
}
```

```
$ javac -d ../../../build/classes -classpath ../../../libs/slf4j-api.jar mypackage/App.java
$ java -classpath ../../../build/classes:../../../libs/slf4j-api.jar mypackage.App
SLF4J: No SLF4J providers were found.
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See https://www.slf4j.org/codes.html#noProviders for further details.
```

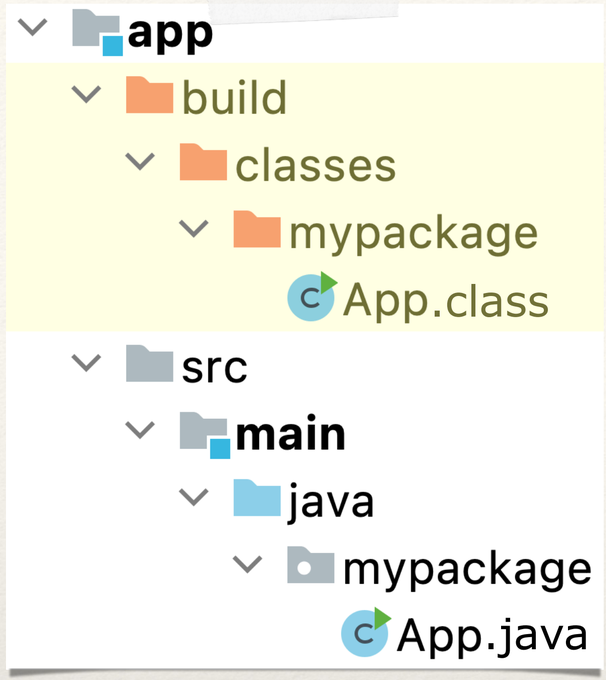



```java
package mypackage;

import org.slf4j.LoggerFactory;

public class App {
    public static void main(String[] args) {
        LoggerFactory.getLogger(App.class).info("running app");
    }
}
```

```
$ javac -d ../../../build/classes -classpath ../../../libs/slf4j-api.jar mypackage/App.java
$ java -classpath ../../../build/classes:../../../libs/slf4j-api.jar mypackage.App
SLF4J: No SLF4J providers were found.
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See https://www.slf4j.org/codes.html#noProviders for further details.
$ java -classpath ../../../build/classes:../../../libs/slf4j-api.jar:
  ../../../libs/slf4j-simple.jar mypackage.App
```

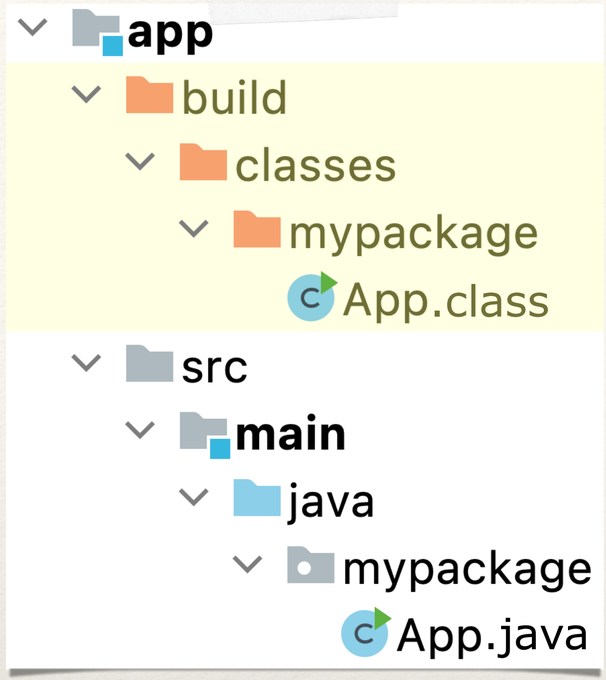



```java
package mypackage;

import org.slf4j.LoggerFactory;

public class App {
    public static void main(String[] args) {
        LoggerFactory.getLogger(App.class).info("running app");
    }
}
```

```
$ javac -d ../../../build/classes -classpath ../../../libs/slf4j-api.jar mypackage/App.java
$ java -classpath ../../../build/classes:../../../libs/slf4j-api.jar mypackage.App
SLF4J: No SLF4J providers were found.
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See https://www.slf4j.org/codes.html#noProviders for further details.
$ java -classpath ../../../build/classes:../../../libs/slf4j-api.jar:
  ../../../libs/slf4j-simple.jar mypackage.App
[main] INFO mypackage.App - running app
```

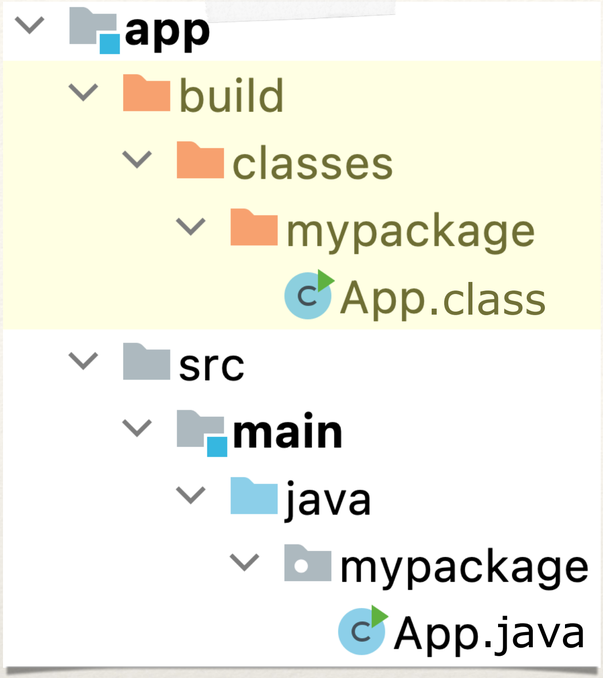



```java
package mypackage;

import org.slf4j.LoggerFactory;

public class App {
    public static void main(String[] args) {
        LoggerFactory.getLogger(App.class).info("running app");
    }
}
```

```
$ javac -d ../../../build/classes -classpath ../../../libs/slf4j-api.jar mypackage/App.java
$ java -classpath ../../../build/classes:../../../libs/slf4j-api.jar mypackage.App
SLF4J: No SLF4J providers were found.
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See https://www.slf4j.org/codes.html#noProviders for further details.
$ java -classpath ../../../build/classes:../../../libs/slf4j-api.jar:
  ../../../libs/                    mypackage.App
```

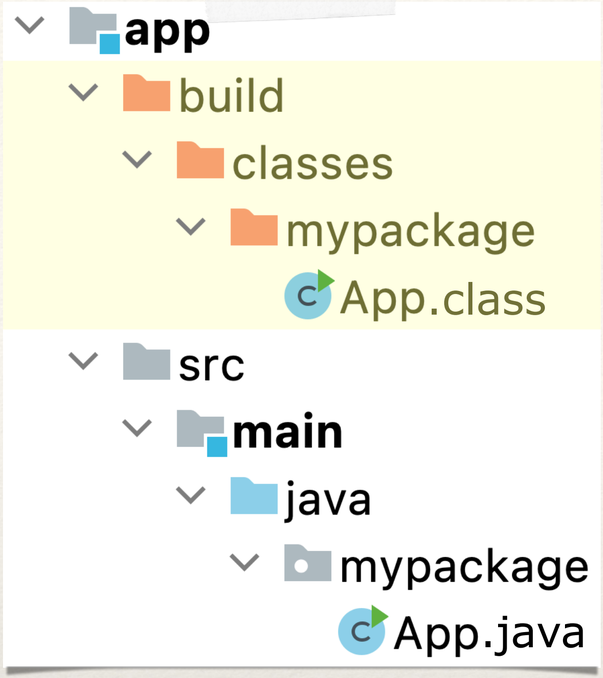



```java
package mypackage;

import org.slf4j.LoggerFactory;

public class App {
    public static void main(String[] args) {
        LoggerFactory.getLogger(App.class).info("running app");
    }
}
```

```
$ javac -d ../../../build/classes -classpath ../../../libs/slf4j-api.jar mypackage/App.java
$ java -classpath ../../../build/classes:../../../libs/slf4j-api.jar mypackage.App
SLF4J: No SLF4J providers were found.
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See https://www.slf4j.org/codes.html#noProviders for further details.
$ java -classpath ../../../build/classes:../../../libs/slf4j-api.jar:
  ../../../libs/slf4j-jdk14.jar  mypackage.App
```

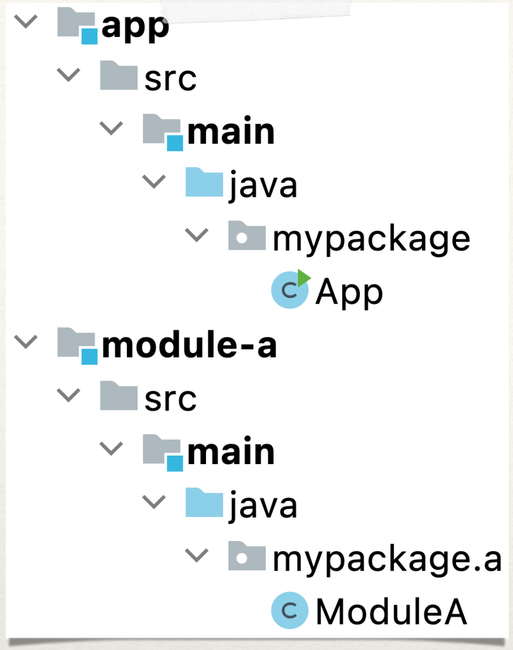



```java
package mypackage;

import org.slf4j.LoggerFactory;

public class App {
    public static void main(String[] args) {
        LoggerFactory.getLogger(App.class).info("running app");
    }
}
```

```
$ javac -d ../../../build/classes -classpath ../../../libs/slf4j-api.jar mypackage/App.java
$ java -classpath ../../../build/classes:../../../libs/slf4j-api.jar mypackage.App
SLF4J: No SLF4J providers were found.
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See https://www.slf4j.org/codes.html#noProviders for further details.
$ java -classpath ../../../build/classes:../../../libs/slf4j-api.jar:
  ../../../libs/slf4j-jdk14.jar  mypackage.App
Feb 02, 2023 7:07:07 PM mypackage.App main
INFO: running app
```

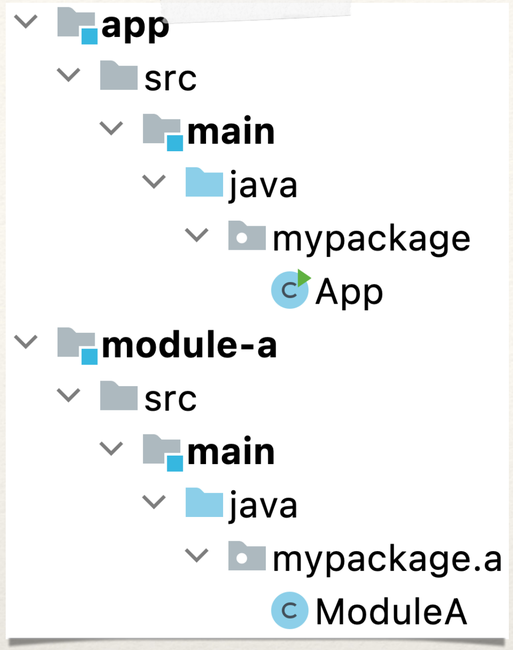



```java
package mypackage;

import org.slf4j.LoggerFactory;


public class App {
    public static void main(String[] args) {
        LoggerFactory.getLogger(App.class).info("running app");

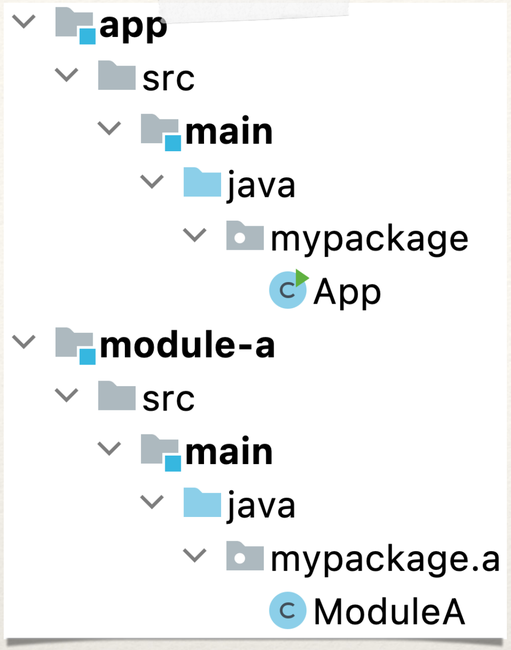
    }
}
```




```java
package mypackage;

import org.slf4j.LoggerFactory;
import mypackage.a.ModuleA;

public class App {
    public static void main(String[] args) {
        LoggerFactory.getLogger(App.class).info("running app");
        new ModuleA().doWork();
    }
}
```

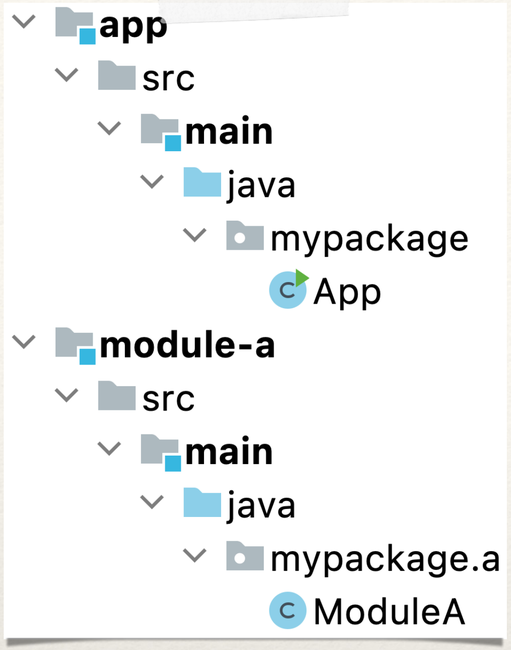



```java
package mypackage;

import org.slf4j.LoggerFactory;
import mypackage.a.ModuleA;

public class App {
    public static void main(String[] args) {
        LoggerFactory.getLogger(App.class).info("running app");
        new ModuleA().doWork();
    }
}
```

```
$ cd ../../../../module-a/src/main/java
```



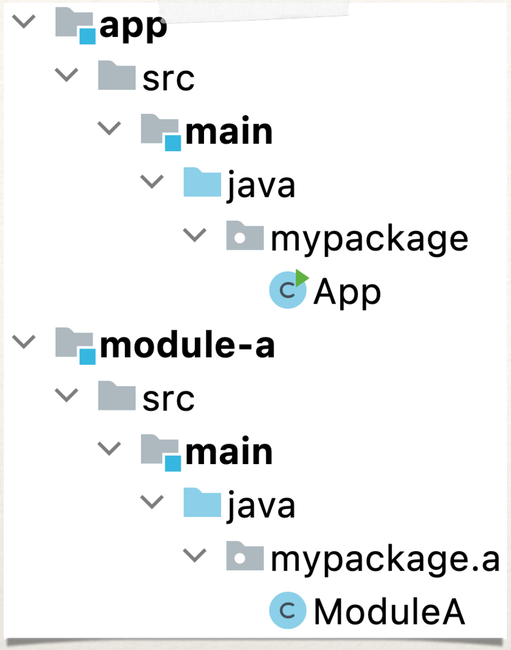

```java
package mypackage;

import org.slf4j.LoggerFactory;
import mypackage.a.ModuleA;

public class App {
    public static void main(String[] args) {
        LoggerFactory.getLogger(App.class).info("running app");
        new ModuleA().doWork();
    }
}
```

```
$ cd ../../../../module-a/src/main/java

$ javac -d ../../../build/classes -classpath ../../../libs/error_prone_annotations.jar
  mypackage/a/ModuleA.java
```

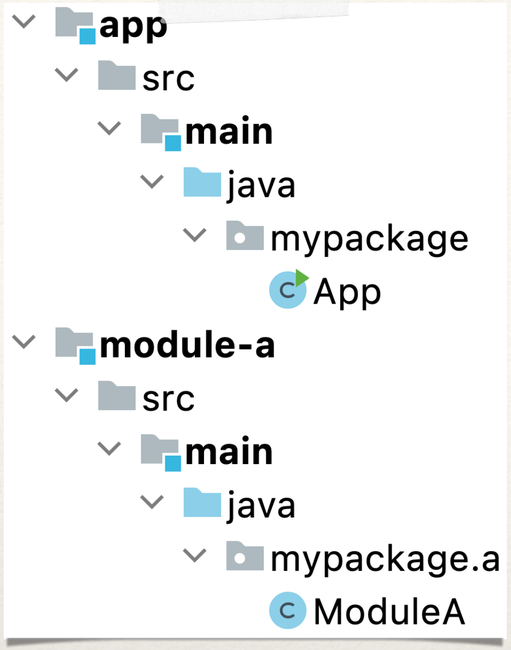



```java
package mypackage;

import org.slf4j.LoggerFactory;
import mypackage.a.ModuleA;

public class App {
    public static void main(String[] args) {
        LoggerFactory.getLogger(App.class).info("running app");
        new ModuleA().doWork();
    }
}
```

```
$ cd ../../../../module-a/src/main/java
$ javac -d ../../../build/classes -classpath ../../../libs/error_prone_annotations.jar
  mypackage/a/ModuleA.java
$ cd ../../../../app/src/main/java
```

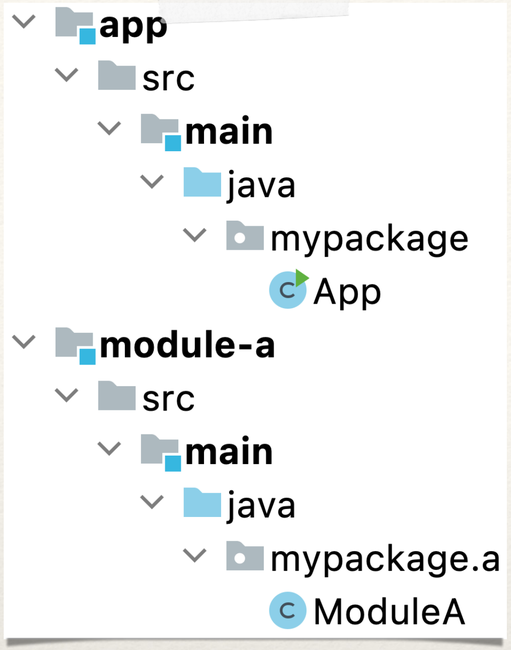



```java
package mypackage;

import org.slf4j.LoggerFactory;
import mypackage.a.ModuleA;

public class App {
    public static void main(String[] args) {
        LoggerFactory.getLogger(App.class).info("running app");
        new ModuleA().doWork();
    }
}
```

```
$ cd ../../../../module-a/src/main/java
$ javac -d ../../../build/classes -classpath ../../../libs/error_prone_annotations.jar
  mypackage/a/ModuleA.java
$ cd ../../../../app/src/main/java
$ javac -d ../../../build/classes -classpath ../../../../module-a/build/classes:
  ../../../libs/slf4j-api.jar mypackage/App.java
```

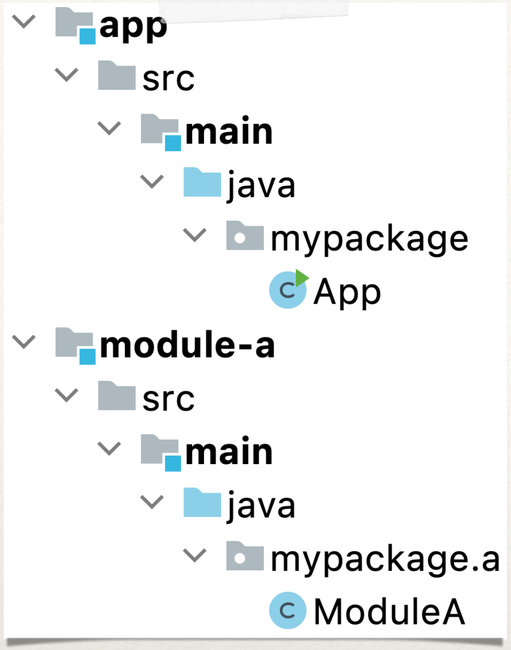



```java
package mypackage;

import org.slf4j.LoggerFactory;
import mypackage.a.ModuleA;

public class App {
    public static void main(String[] args) {
        LoggerFactory.getLogger(App.class).info("running app");
        new ModuleA().doWork();
    }
}
```

```
$ cd ../../../../module-a/src/main/java
$ javac -d ../../../build/classes -classpath ../../../libs/error_prone_annotations.jar
  mypackage/a/ModuleA.java
$ cd ../../../../app/src/main/java
$ javac -d ../../../build/classes -classpath ../../../../module-a/build/classes:
  ../../../libs/slf4j-api.jar mypackage/App.java
$ java -classpath ../../../build/classes:../../../../module-a/build/classes:
  ../../../libs/slf4j-api.jar:../../../libs/slf4j-simple.jar mypackage.App
```




```java
package mypackage;

import org.slf4j.LoggerFactory;
import mypackage.a.ModuleA;

public class App {
    public static void main(String[] args) {
        LoggerFactory.getLogger(App.class).info("running app");
        new ModuleA().doWork();
    }
}
```

```
$ cd ../../../../module-a/src/main/java
$ javac -d ../../../build/classes -classpath ../../../libs/error_prone_annotations.jar
  mypackage/a/ModuleA.java
$ cd ../../../../app/src/main/java
$ javac -d ../../../build/classes -classpath ../../../../module-a/build/classes:
  ../../../libs/slf4j-api.jar mypackage/App.java
$ java -classpath ../../../build/classes:../../../../module-a/build/classes:
  ../../../libs/slf4j-api.jar:../../../libs/slf4j-simple.jar mypackage.App
[main] INFO mypackage.App - running app
running module-a
```

Modularization by multiple Classpaths

- **Compile Classpath** of **module-a**
- **Compile Classpath** of **app**
- **Runtime Classpath** of complete application

```
$ cd ../../../../module-a/src/main/java
$ javac -d ../../../build/classes -classpath ../../../libs/error_prone_annotations.jar
  mypackage/a/ModuleA.java
$ cd ../../../../app/src/main/java
$ javac -d ../../../build/classes -classpath ../../../../module-a/build/classes:
  ../../../libs/slf4j-api.jar mypackage/App.java
$ java -classpath ../../../build/classes:../../../../module-a/build/classes:
  ../../../libs/slf4j-api.jar:../../../libs/slf4j-simple.jar mypackage.App
[main] INFO mypackage.App - running app
running module-a
```

Modularization by multiple Classpaths

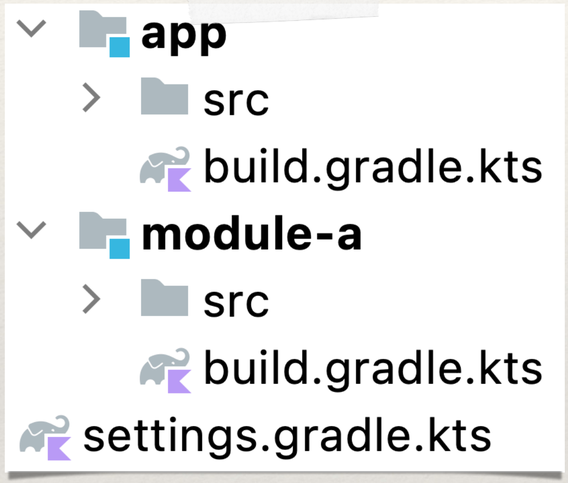
- **Compile Classpath** of **module-a**
- **Compile Classpath** of **app**
- **Runtime Classpath** of complete application

- `slf4j-api.jar module-a/build/classes`
  Implementation details of **app** (required at runtime!)
- `slf4j-simple.jar app/build/classes`
  Runtime-only modules for the application
- `error_prone_annotations.jar`
  Compile time only (for tool) in **module-a**

```
$ cd ../../../../module-a/src/main/java
$ javac -d ../../../build/classes -classpath ../../../libs/error_prone_annotations.jar
  mypackage/a/ModuleA.java
$ cd ../../../../app/src/main/java
$ javac -d ../../../build/classes -classpath ../../../../module-a/build/classes:
  ../../../libs/slf4j-api.jar mypackage/App.java
$ java -classpath ../../../build/classes:../../../../module-a/build/classes:
  ../../../libs/slf4j-api.jar:../../../libs/slf4j-simple.jar mypackage.App
[main] INFO mypackage.App - running app
running module-a
```

Modularization by multiple Classpaths

- **Compile Classpath** of **module-a**
- **Compile Classpath** of **app**
- **Runtime Classpath** of complete application

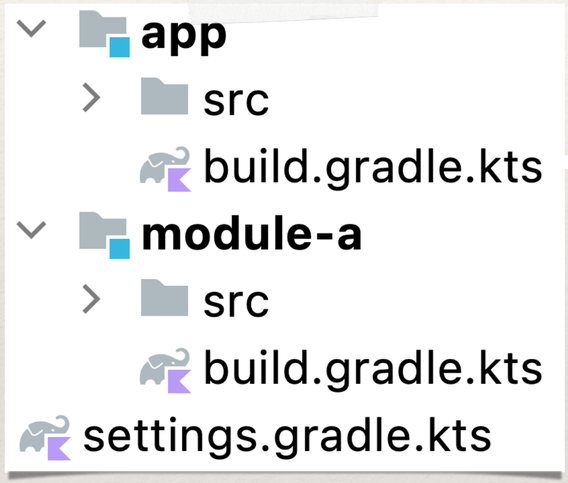


- `slf4j-api.jar module-a/build/classes`
  Implementation details of **app** (required at runtime!)
- `slf4j-simple.jar app/build/classes`
  Runtime-only modules for the application
- `error_prone_annotations.jar`
  Compile time only (for tool) in **module-a**

Modularization by multiple Classpaths

- **Compile Classpath** of **module-a**
- **Compile Classpath** of **app**
- **Runtime Classpath** of complete application

- `slf4j-api.jar module-a/build/classes`
  Implementation details of **app** (required at runtime!)
- `slf4j-simple.jar app/build/classes`
  Runtime-only modules for the application
- `error_prone_annotations.jar`
  Compile time only (for tool) in **module-a**



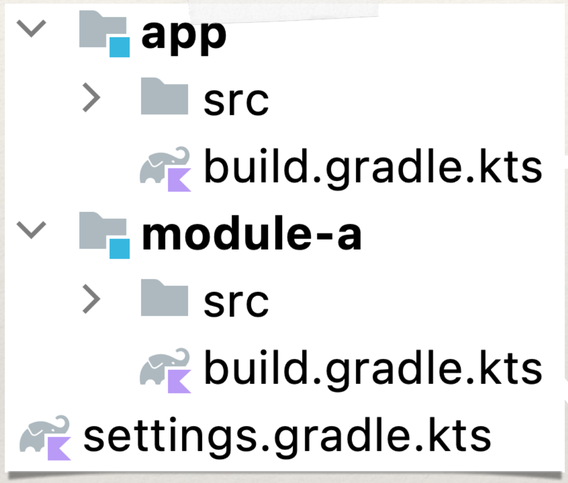

```
plugins {
  id("application") // java-application
}

dependencies {
  implementation(project(":module-a"))
  implementation("org.slf4j:slf4j-api:2.0.6")
  runtimeOnly("org.slf4j:slf4j-simple:2.0.6")
}
```

Modularization by multiple Classpaths

- **Compile Classpath** of **module-a**
- **Compile Classpath** of **app**
- **Runtime Classpath** of complete application

- `slf4j-api.jar module-a/build/classes`
  Implementation details of **app** (required at runtime!)
- `slf4j-simple.jar app/build/classes`
  Runtime-only modules for the application
- `error_prone_annotations.jar`
  Compile time only (for tool) in **module-a**




```
plugins {
  id("application") // java-application
}

dependencies {
  implementation(project(":module-a"))
  implementation("org.slf4j:slf4j-api:2.0.6")
  runtimeOnly("org.slf4j:slf4j-simple:2.0.6")
}
```

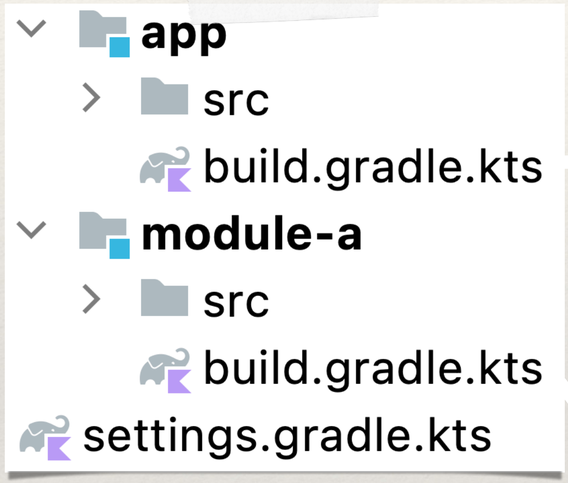
```
plugins {
  id("java-library")
}

dependencies {
  compileOnly("com.google.errorprone:error_prone_annotations:2.16")
}
```

Modularization by multiple Classpaths

- **Compile Classpath** of **module-a**
- **Compile Classpath** of **app**
- **Runtime Classpath** of complete application

- `slf4j-api.jar module-a/build/classes`
  Implementation details of **app** (required at runtime!)
- `slf4j-simple.jar app/build/classes`
  Runtime-only modules for the application
- `error_prone_annotations.jar`
  Compile time only (for tool) in **module-a**




```
plugins {
  id("application")
}

dependencies {
  implementation(pro
  implementation("or
  runtimeOnly("org.s
}
```

```
plugins {
  id("java-library")
}

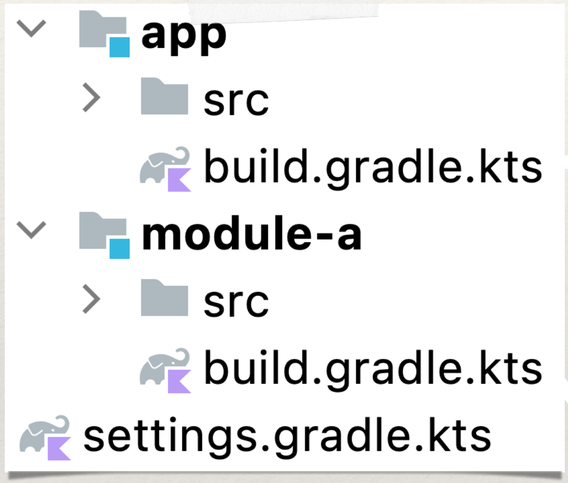
dependencies {
  compileOnly("com.g
}
```



Modularization by multiple Classpaths

- **Compile Classpath** of **module-a**
- **Compile Classpath** of **app**
- **Runtime Classpath** of complete application

- `slf4j-api.jar module-a/build/classes`
  Implementation details of **app** (required at runtime!)
- `slf4j-simple.jar app/build/classes`
  Runtime-only modules for the application
- `error_prone_annotations.jar`
  Compile time only (for tool) in **module-a**




```
plugins {
  id("application")
}

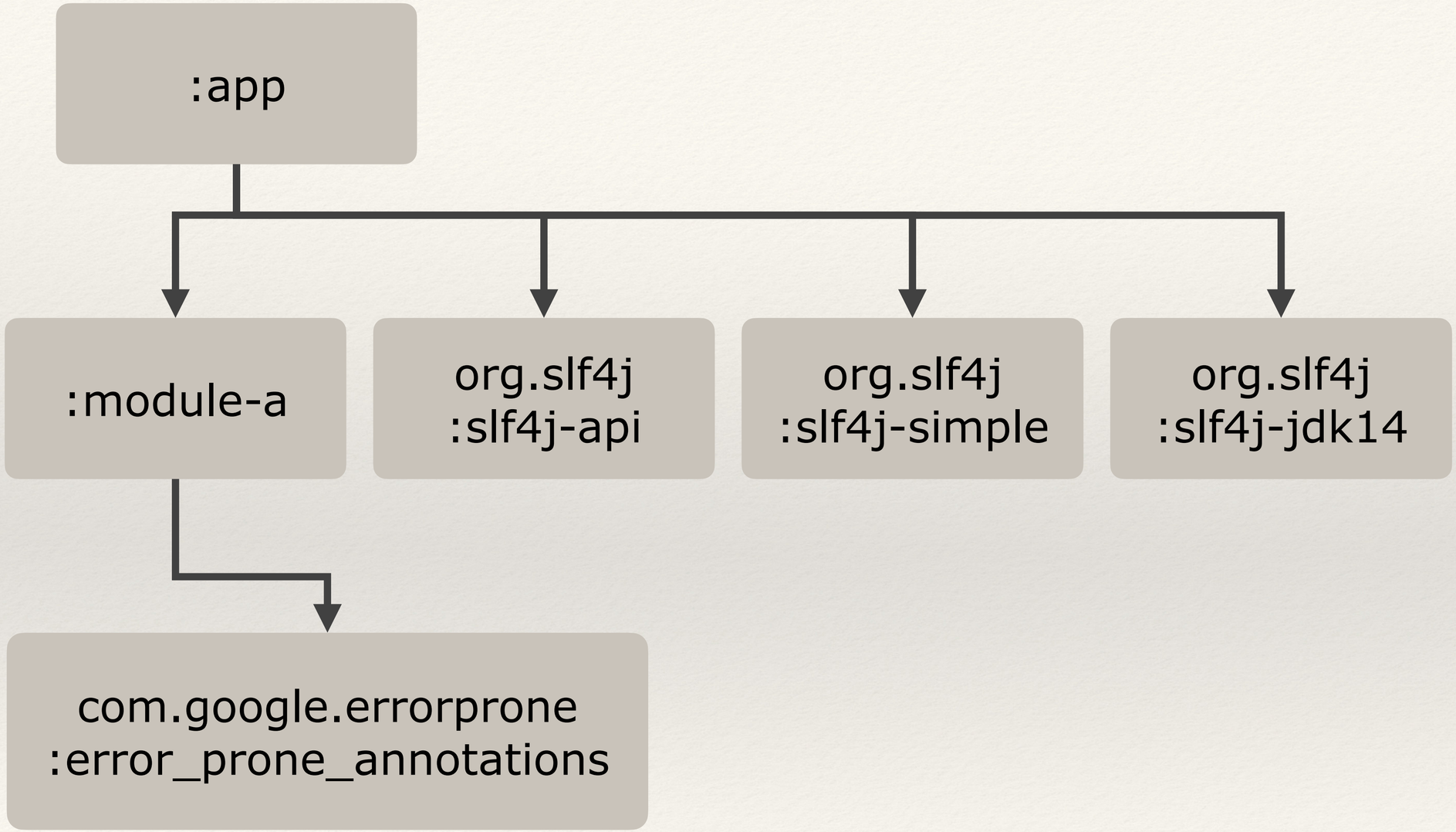
dependencies {
  implementation(pro
  implementation("or
  runtimeOnly("org.s
}
```

```
plugins {
  id("java-library")
}

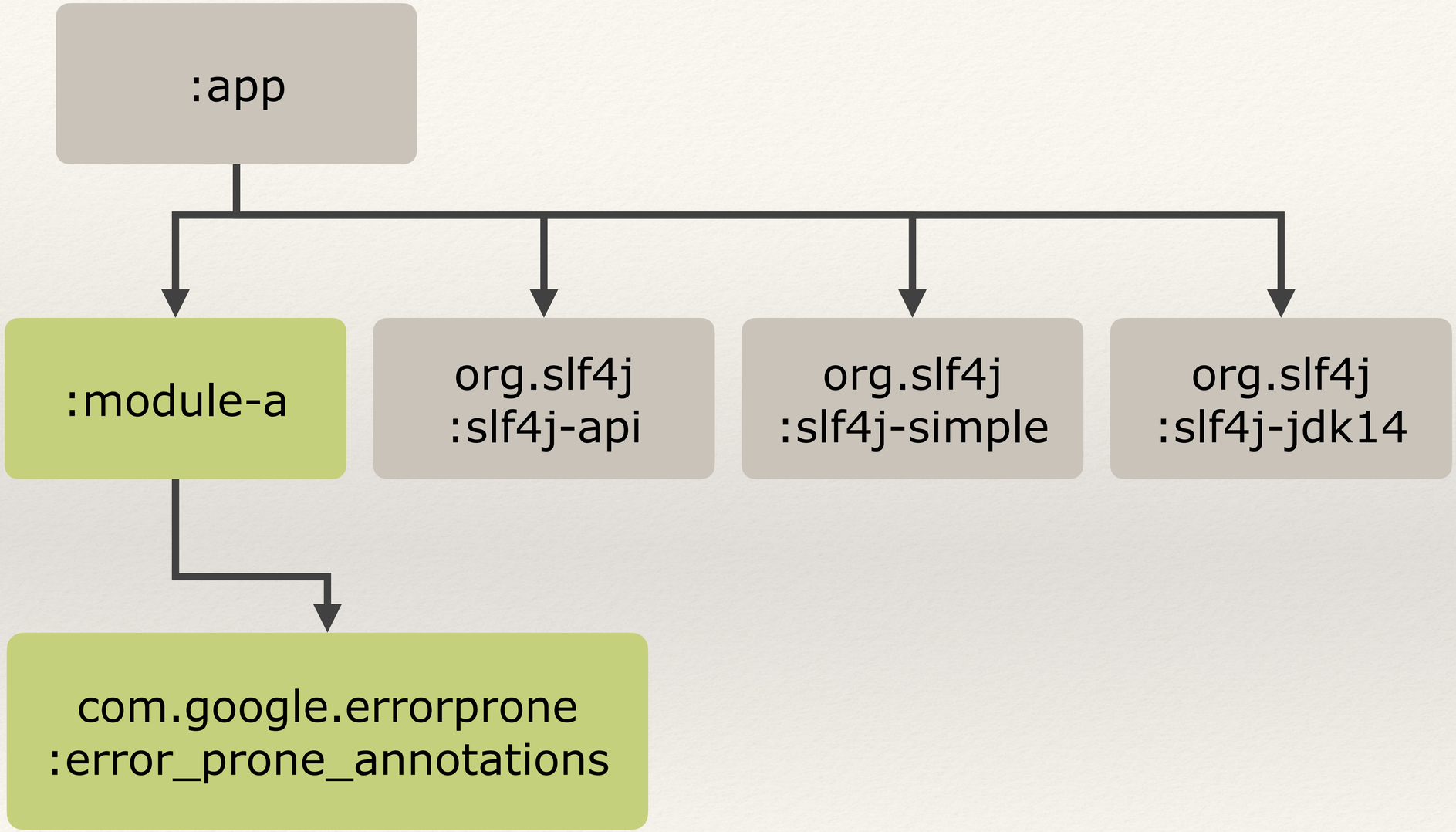
dependencies {
  compileOnly("com.g
}
```

```
$ gradle :app:run
> Task :module-a:compileJava UP-TO-DATE
> Task :app:compileJava
> Task :app:processResources NO-SOURCE
> Task :app:classes UP-TO-DATE
> Task :module-a:processResources NO-SOURCE
> Task :module-a:classes UP-TO-DATE
> Task :module-a:jar

> Task :app:run
[main] INFO mypackage.App - running app
running module-a
```

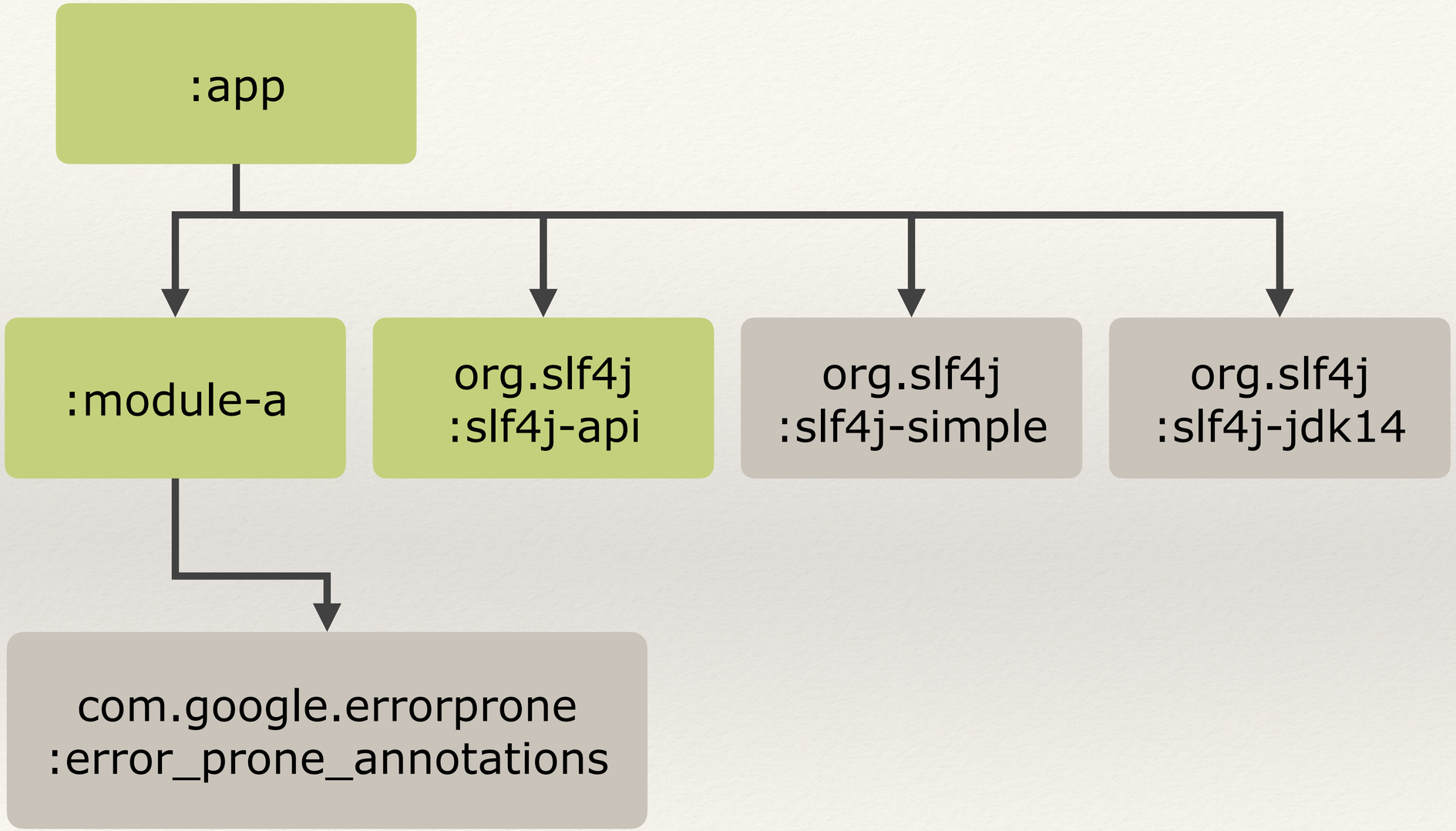

:app
:module-a
org.slf4j
:slf4j-api
org.slf4j
:slf4j-simple
org.slf4j
:slf4j-jdk14
com.google.errorprone
:error_prone_annotations

```
javac -classpath error_prone_annotations.jar
```

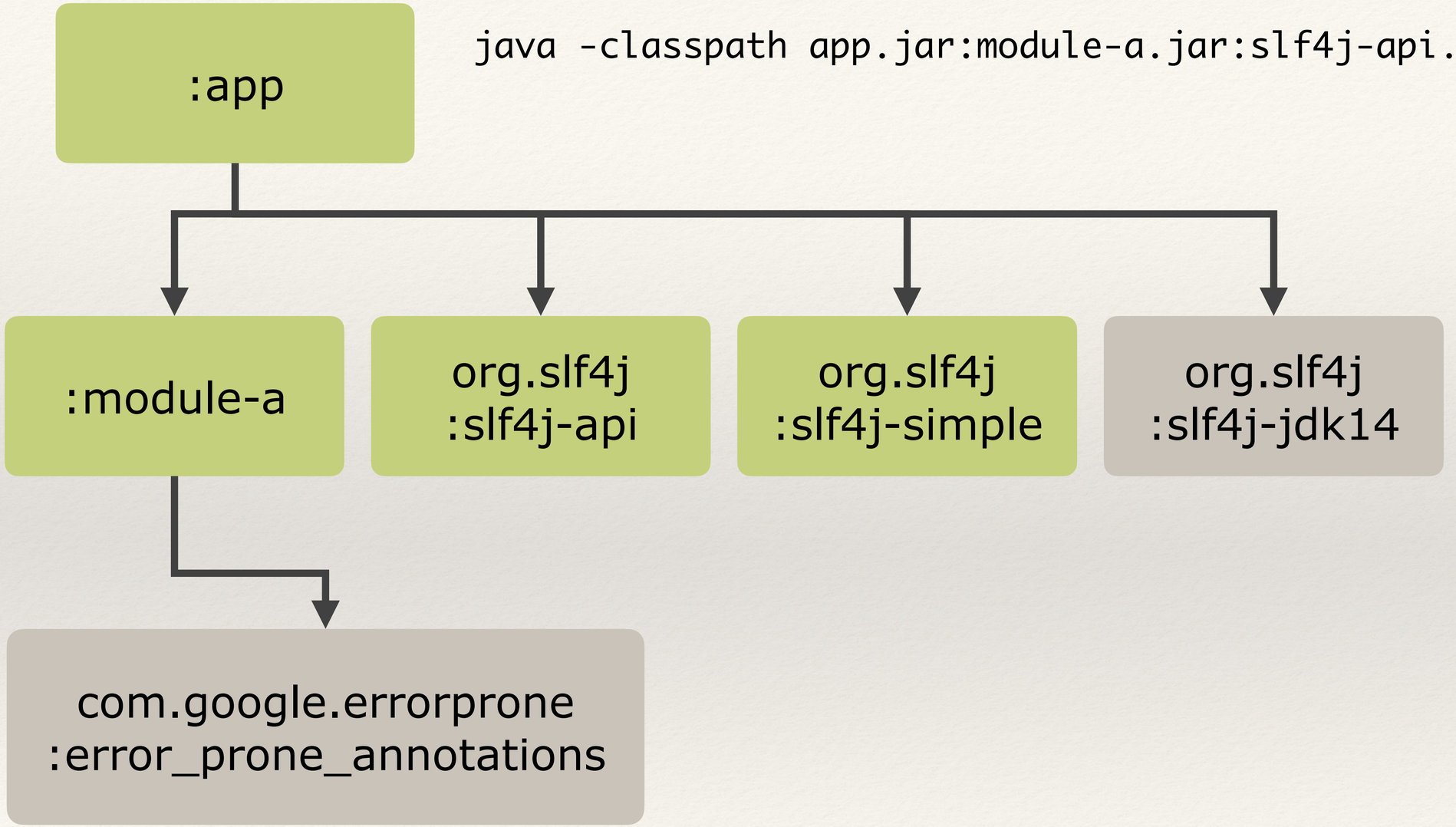


```
javac -classpath error_prone_annotations.jar
```

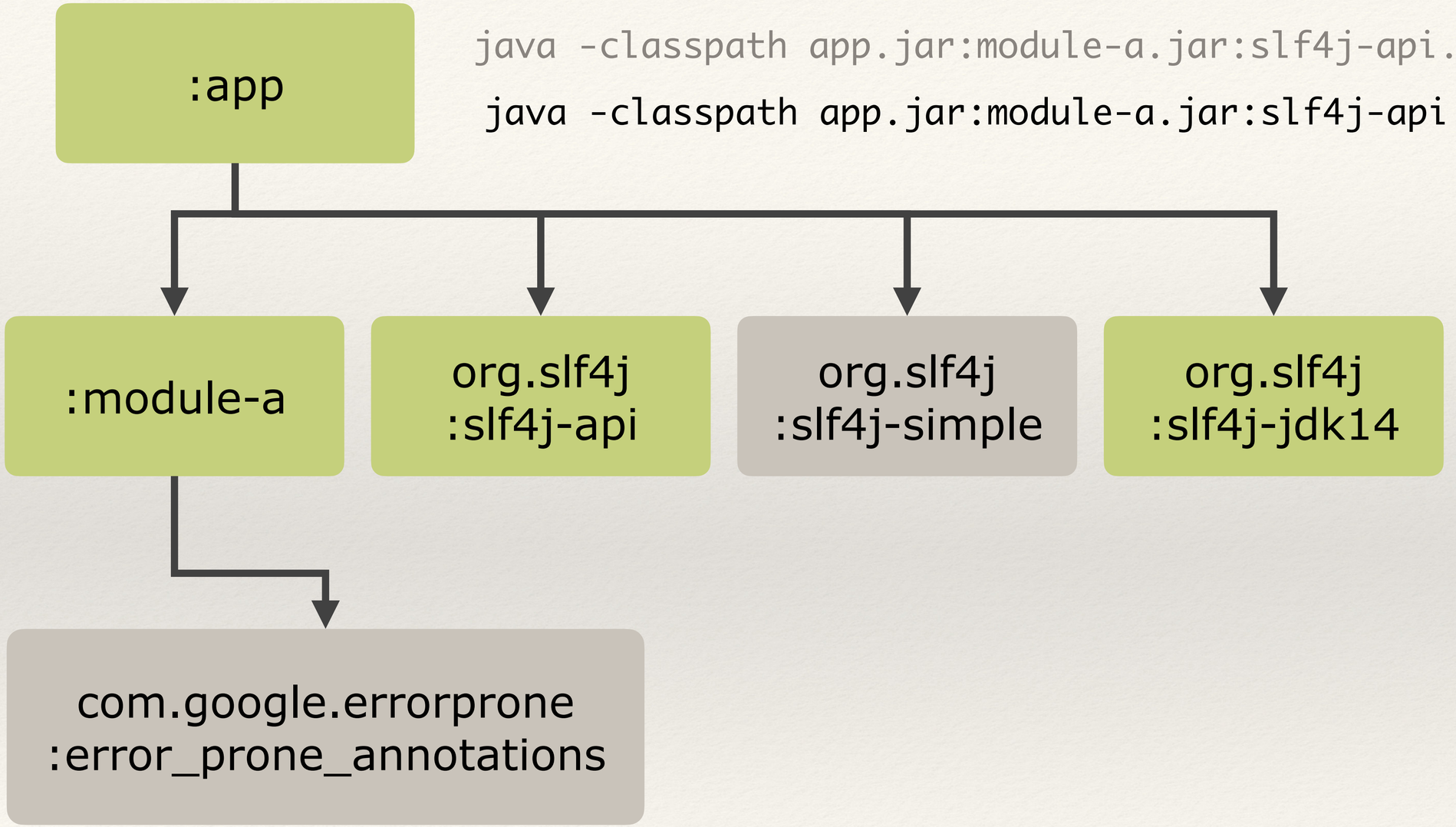
```
javac -classpath module-a.jar:slf4j-api.jar
```



```
javac -classpath error_prone_annotations.jar
javac -classpath module-a.jar:slf4j-api.jar
java -classpath app.jar:module-a.jar:slf4j-api.jar:slf4j-simple.jar
```



```
javac -classpath error_prone_annotations.jar
javac -classpath module-a.jar:slf4j-api.jar
java -classpath app.jar:module-a.jar:slf4j-api.jar:slf4j-simple.jar
java -classpath app.jar:module-a.jar:slf4j-api.jar:slf4j-jdk14.jar
```



Modularizing Java Projects

*(1) Modularization with the **Classpath** + Gradle*

*(2) Modularization with the **Module Path***

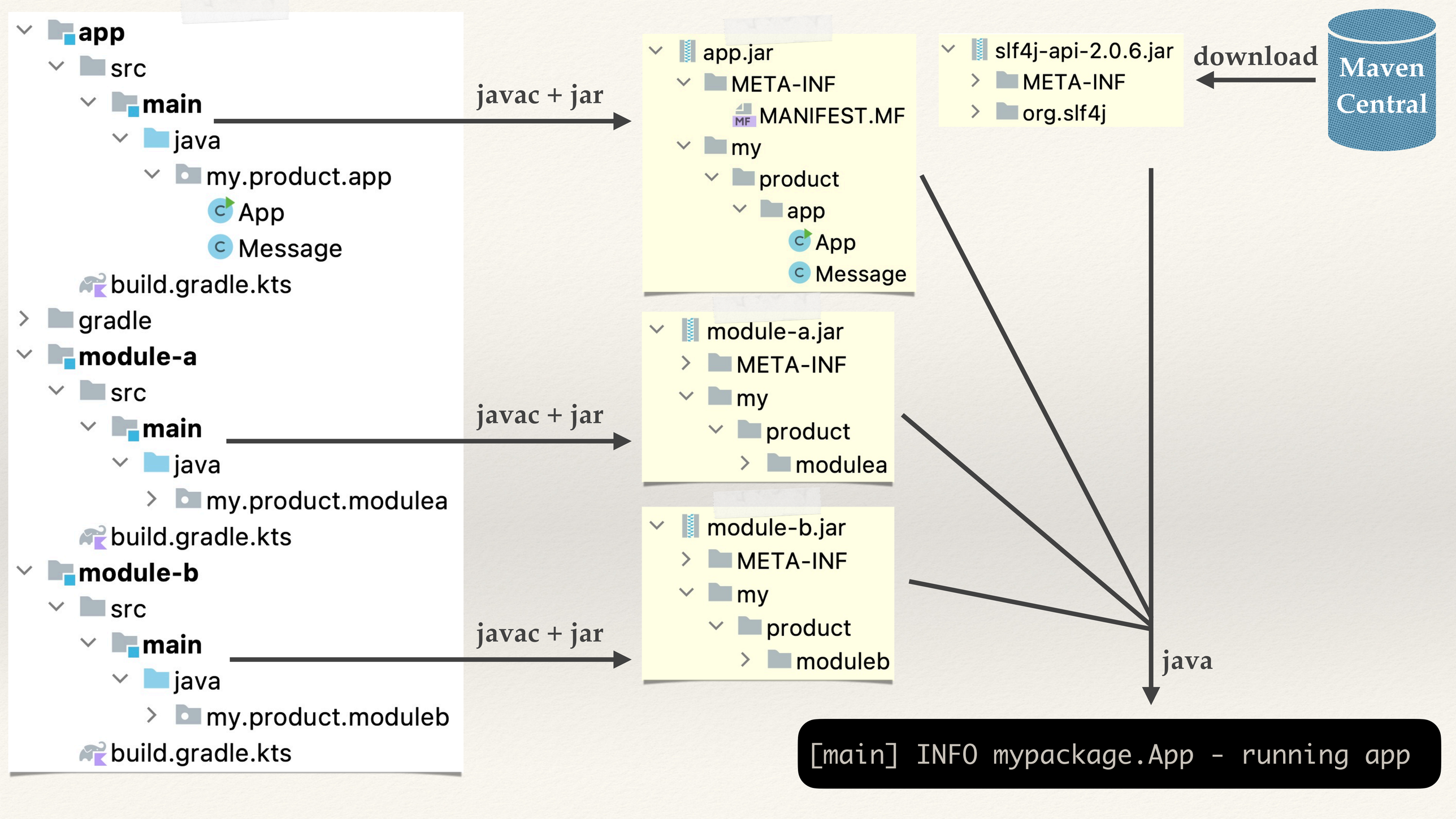
*(3) Modularization with the **Module Path** + Gradle + X*

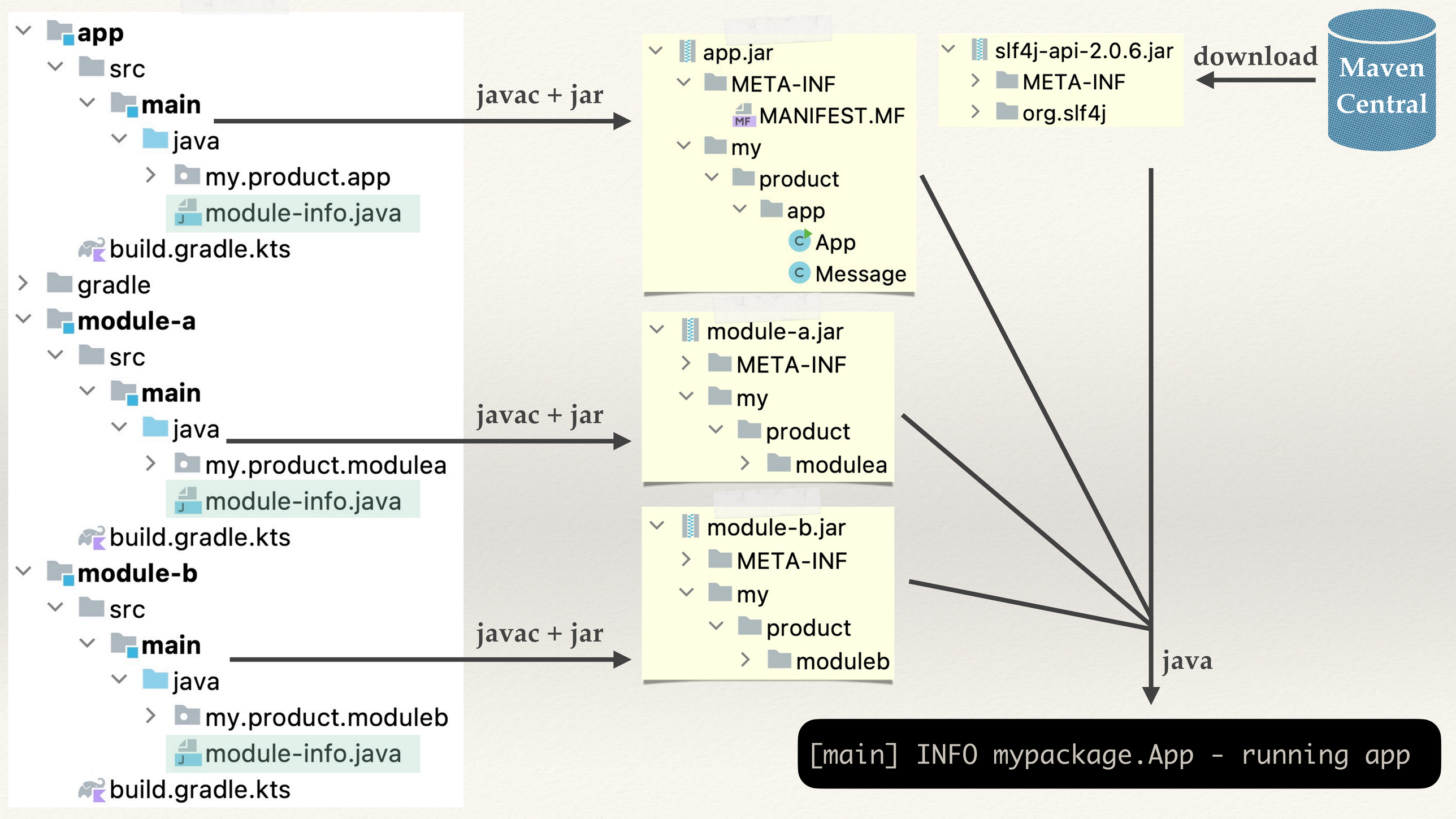
Modularizing Java Projects

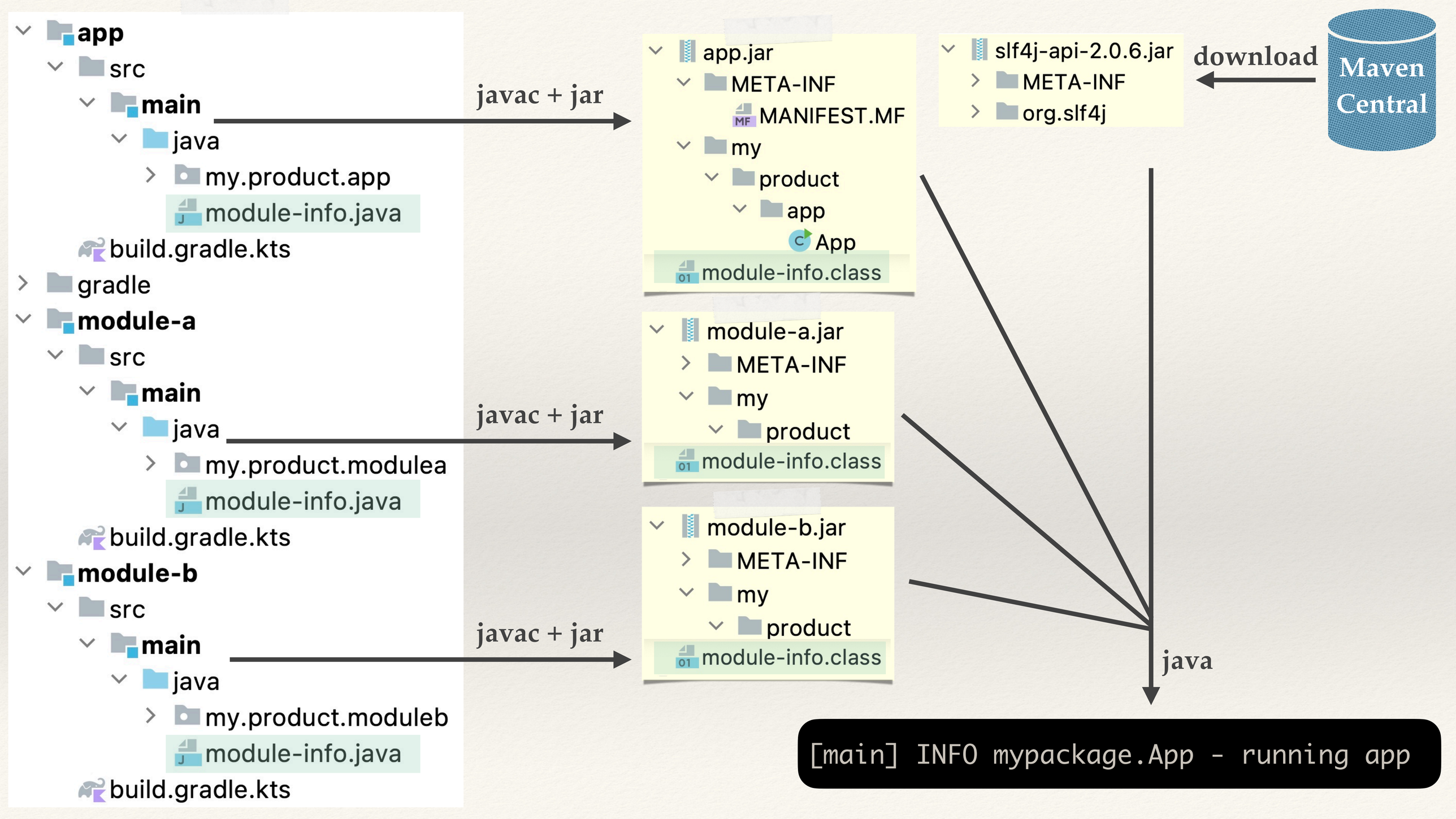
*(1) Modularization with the **Classpath** + Gradle*

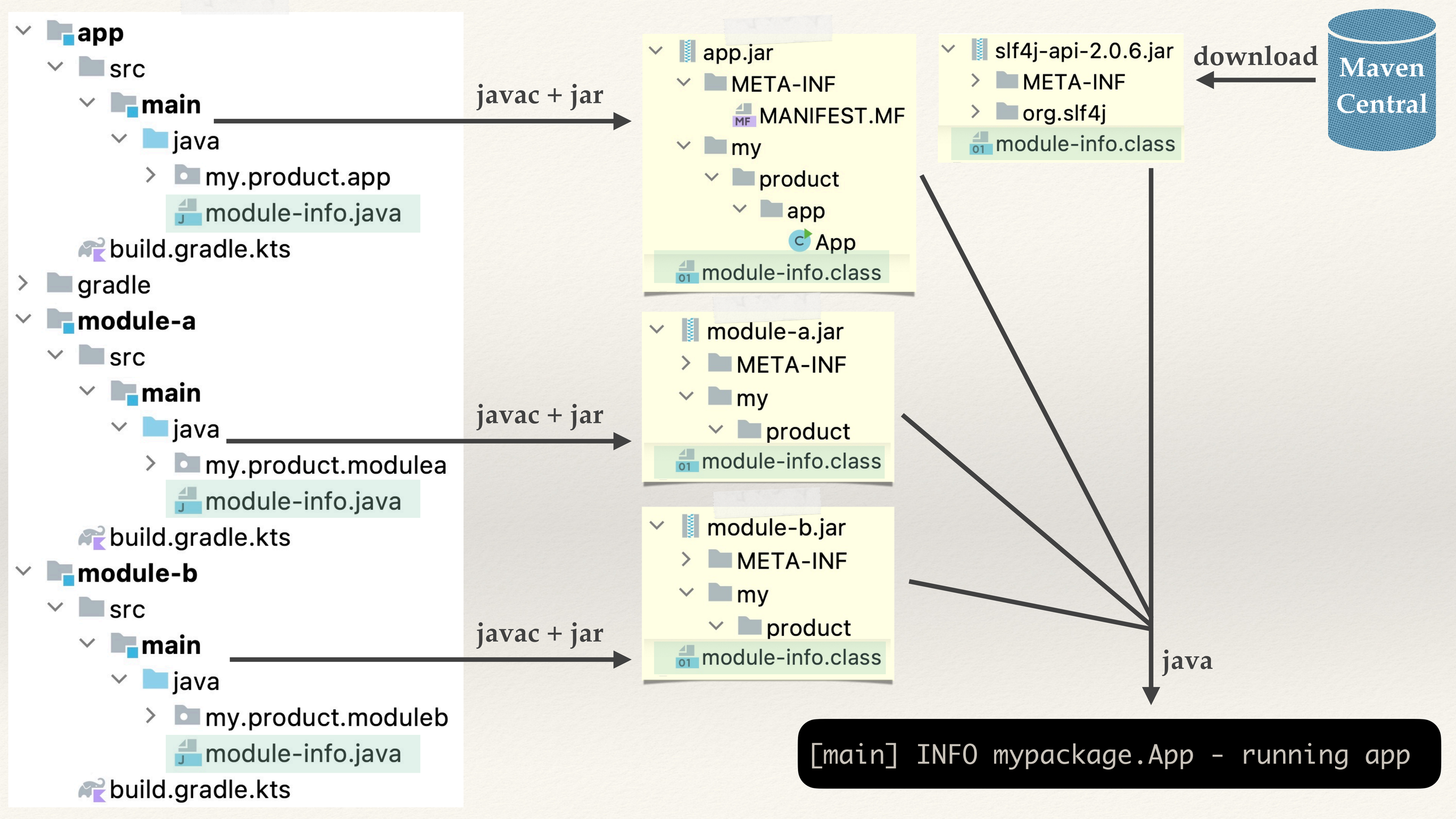
*(2) Modularization with the **Module Path***

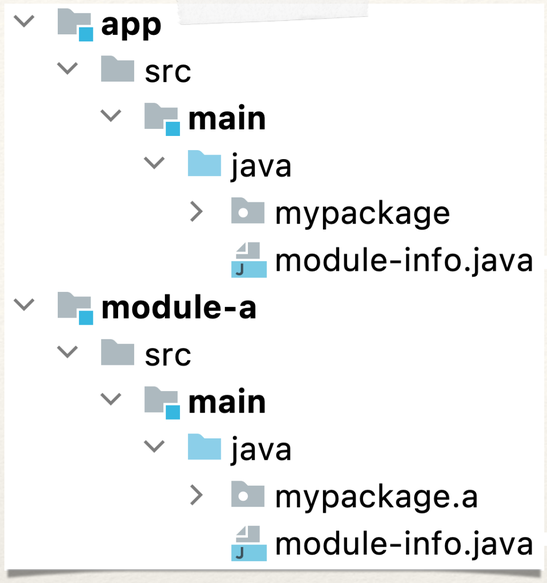
*(3) Modularization with the **Module Path** + Gradle + X*


app
src
main
java
my.product.app
App
Message
build.gradle.kts
gradle
module-a
src
main
java
my.product.modulea
build.gradle.kts
module-b
src
main
java
my.product.moduleb
build.gradle.kts
javac + jar
javac + jar
javac + jar
app.jar
META-INF
MANIFEST.MF
my
product
app
App
Message
module-a.jar
META-INF
my
product
modulea
module-b.jar
META-INF
my
product
moduleb
slf4j-api-2.0.6.jar
META-INF
org.slf4j
download
Maven Central
java
[main] INFO mypackage.App - running app


app
src
main
java
my.product.app
module-info.java
build.gradle.kts
gradle
module-a
src
main
java
my.product.modulea
module-info.java
build.gradle.kts
module-b
src
main
java
my.product.moduleb
module-info.java
build.gradle.kts
javac + jar
javac + jar
javac + jar
app.jar
META-INF
MANIFEST.MF
my
product
app
App
Message
module-a.jar
META-INF
my
product
modulea
module-b.jar
META-INF
my
product
moduleb
slf4j-api-2.0.6.jar
META-INF
org.slf4j
download
Maven Central
java
[main] INFO mypackage.App - running app


app
src
main
java
my.product.app
module-info.java
build.gradle.kts
gradle
module-a
src
main
java
my.product.modulea
module-info.java
build.gradle.kts
module-b
src
main
java
my.product.moduleb
module-info.java
build.gradle.kts
javac + jar
javac + jar
javac + jar
app.jar
META-INF
MANIFEST.MF
my
product
app
App
module-info.class
module-a.jar
META-INF
my
product
module-info.class
module-b.jar
META-INF
my
product
module-info.class
slf4j-api-2.0.6.jar
META-INF
org.slf4j
download
Maven Central
java
[main] INFO mypackage.App - running app


app
src
main
java
my.product.app
module-info.java
build.gradle.kts
gradle
module-a
src
main
java
my.product.modulea
module-info.java
build.gradle.kts
module-b
src
main
java
my.product.moduleb
module-info.java
build.gradle.kts
javac + jar
javac + jar
javac + jar
app.jar
META-INF
MANIFEST.MF
my
product
app
App
module-info.class
module-a.jar
META-INF
my
product
module-info.class
module-b.jar
META-INF
my
product
module-info.class
slf4j-api-2.0.6.jar
META-INF
org.slf4j
module-info.class
download
Maven Central
java
[main] INFO mypackage.App - running app

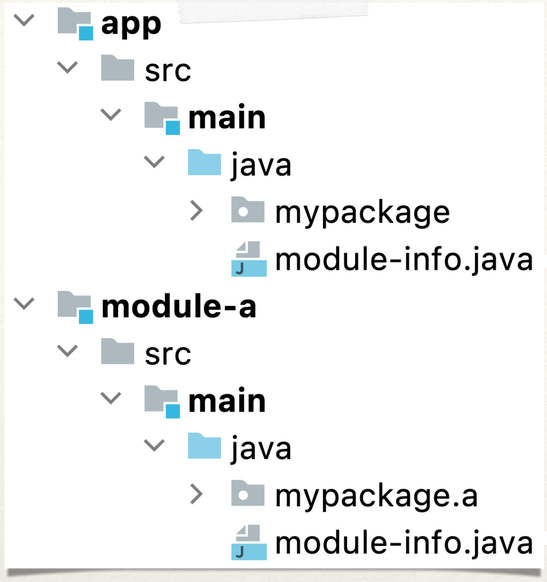



```
module my.app {


}
```

```
module my.module.a {

}
```



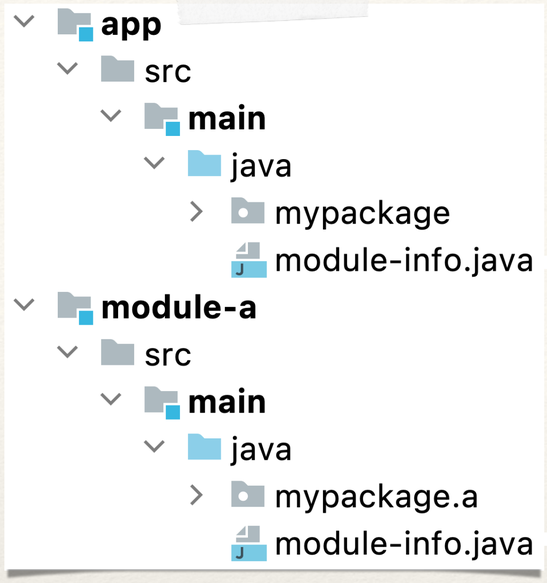

```
module my.app {


}
```

```
module my.module.a {

}
```

```
$ javac -d ../../../build/classes -classpath ../../../../module-a/build/classes:
  ../../../libs/slf4j-api.jar mypackage/App.java
```



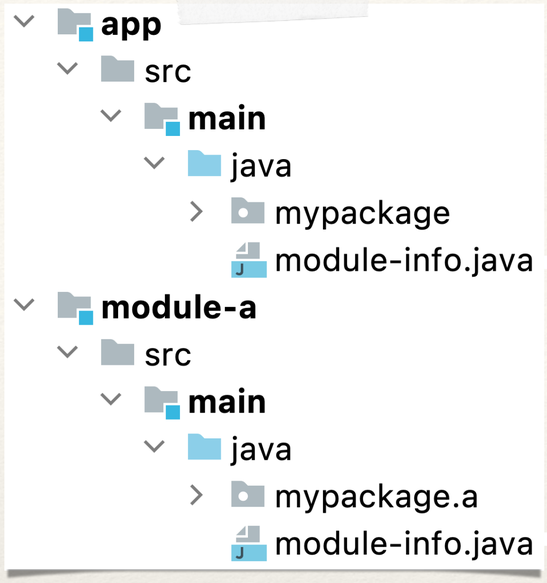

```
module my.app {


}
```

```
module my.module.a {

}
```

```
$ javac -d ../../../build/classes -classpath ../../../../module-a/build/classes:
  ../../../libs/slf4j-api.jar mypackage/App.java
$ javac -d ../../../build/classes --module-path ../../../../module-a/build/classes:
  ../../../libs/slf4j-api.jar mypackage/App.java module-info.java
```




```
module my.app {

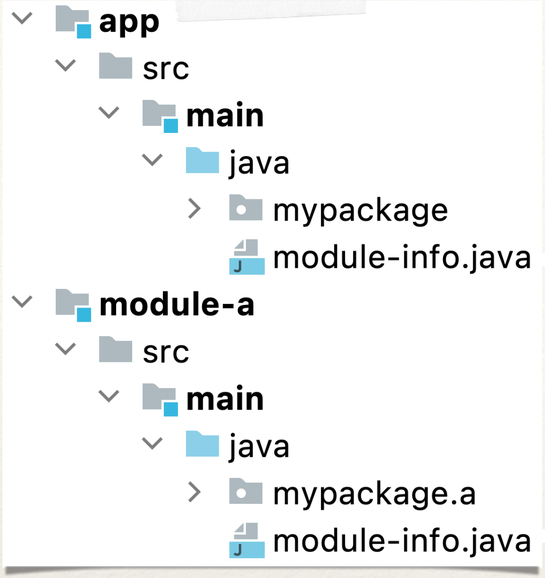

}
```

```
module my.module.a {

}
```

```
$ javac -d ../../../build/classes -classpath ../../../../module-a/build/classes:
  ../../../libs/slf4j-api.jar mypackage/App.java
$ javac -d ../../../build/classes --module-path ../../../../module-a/build/classes:
  ../../../libs/slf4j-api.jar mypackage/App.java module-info.java
mypackage/App.java:3: error: package org.slf4j is not visible
import org.slf4j.LoggerFactory;
          ^
 (package org.slf4j is declared in module org.slf4j, but module my.app does not read it)

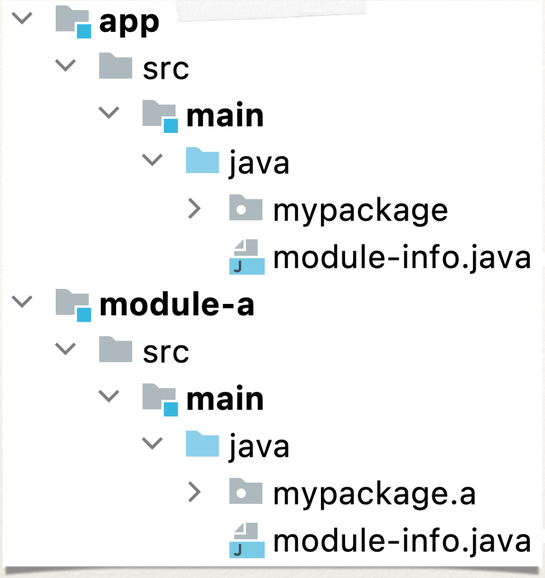
import mypackage.a.ModuleA;
                  ^
 (package mypackage.a is declared in module my.module.a, but module my.app does not read it)
```




```
module my.app {
    requires my.module.a;
    requires org.slf4j;
}
```

```
module my.module.a {

}
```

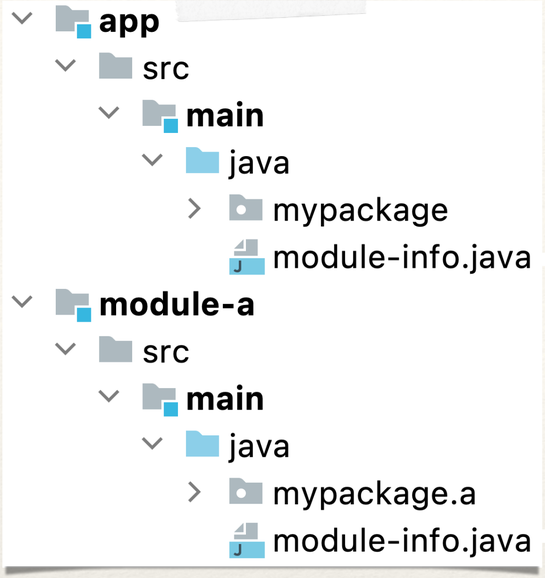



```
module my.app {
    requires my.module.a;
    requires org.slf4j;
}
```

```
module my.module.a {

}
```

```
$ javac -d ../../../build/classes --module-path ../../../../module-a/build/classes:
  ../../../libs/slf4j-api.jar mypackage/App.java module-info.java
```

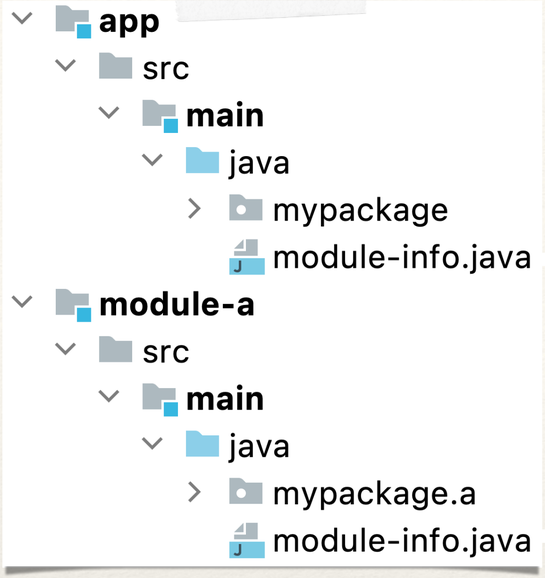



```
module my.app {
    requires my.module.a;
    requires org.slf4j;
}
```

```
module my.module.a {

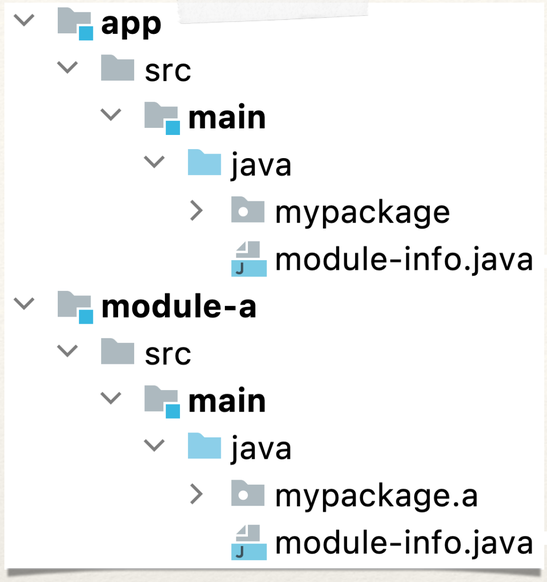
}
```

```
$ javac -d ../../../build/classes --module-path ../../../../module-a/build/classes:
  ../../../libs/slf4j-api.jar mypackage/App.java module-info.java
mypackage/App.java:4: error: package mypackage.a is not visible
import mypackage.a.ModuleA;
                  ^
  (package mypackage.a is declared in module my.module.a, which does not export it)
```




```
module my.app {
    requires my.module.a;
    requires org.slf4j;
}
```

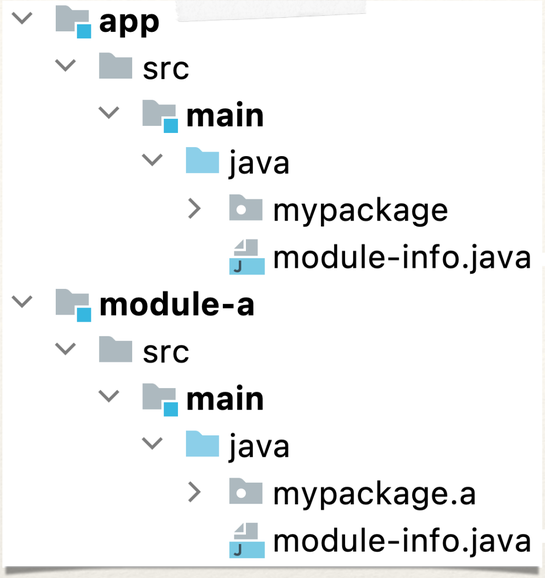
```
module my.module.a {
    exports mypackage.a;
}
```




```
module my.app {
    requires my.module.a;
    requires org.slf4j;
}
```

```
module my.module.a {
    exports mypackage.a;
}
```

```
$ javac -d ../../../build/classes --module-path ../../../../module-a/build/classes:
  ../../../libs/slf4j-api.jar mypackage/App.java module-info.java
```

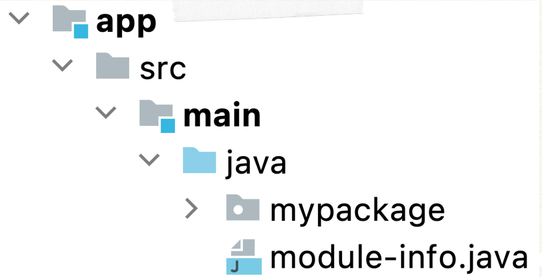



```
module my.app {
    requires my.module.a;
    requires org.slf4j;
}
```

```
module my.module.a {
    exports mypackage.a;
}
```

```
$ javac -d ../../../build/classes --module-path ../../../../module-a/build/classes:
  ../../../libs/slf4j-api.jar mypackage/App.java module-info.java

$ java -classpath ../../../build/classes:../../../../module-a/build/classes:
  ../../../libs/slf4j-api.jar:../../../libs/slf4j-simple.jar mypackage.App
```



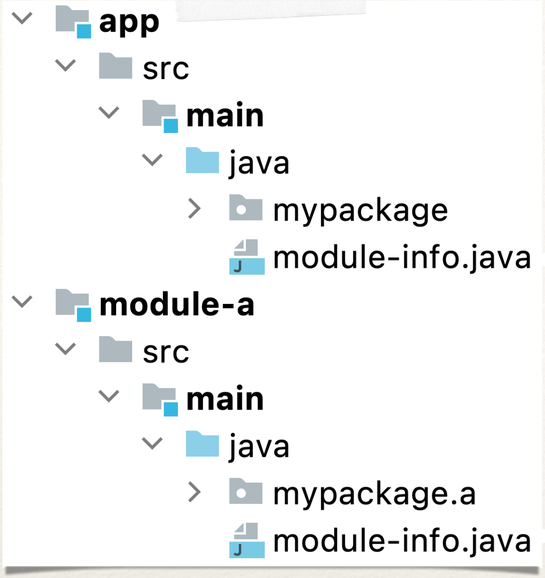

```
module my.app {
    requires my.module.a;
    requires org.slf4j;
}
```

```
module my.module.a {
    exports mypackage.a;
}
```

```
$ javac -d ../../../build/classes --module-path ../../../../module-a/build/classes:
  ../../../libs/slf4j-api.jar mypackage/App.java module-info.java

$ java -classpath ../../../build/classes:../../../../module-a/build/classes:
  ../../../libs/slf4j-api.jar:../../../libs/slf4j-simple.jar mypackage.App

[main] INFO mypackage.App - running app
running module-a
```




```
module my.app {
    requires my.module.a;
    requires org.slf4j;
}
```

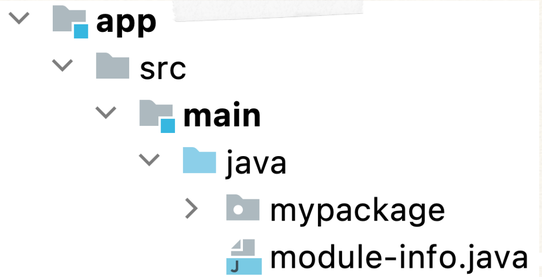
```
module my.module.a {
    exports mypackage.a;
}
```

```
$ javac -d ../../../build/classes --module-path ../../../../module-a/build/classes:
  ../../../libs/slf4j-api.jar mypackage/App.java module-info.java

$ java -classpath ../../../build/classes:../../../../module-a/build/classes:
  ../../../libs/slf4j-api.jar:../../../libs/slf4j-simple.jar mypackage.App

[main] INFO mypackage.App - running app
running module-a

$ java --module-path ../../../build/classes:../../../../module-a/build/classes:
  ../../../libs/slf4j-api.jar:../../../libs/slf4j-simple.jar --module my.app/mypackage.App
```




```
module my.app {
    requires my.module.a;
    requires org.slf4j;
}
```

```
module my.module.a {
    exports mypackage.a;
}
```

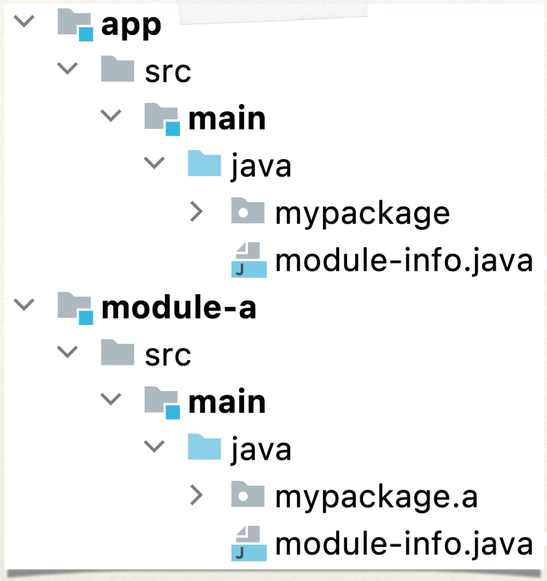
```
$ javac -d ../../../build/classes --module-path ../../../../module-a/build/classes:
  ../../../libs/slf4j-api.jar mypackage/App.java module-info.java

$ java -classpath ../../../build/classes:../../../../module-a/build/classes:
  ../../../libs/slf4j-api.jar:../../../libs/slf4j-simple.jar mypackage.App

[main] INFO mypackage.App - running app
running module-a

 $ java --module-path ../../../build/classes:../../../../module-a/build/classes:
   ../../../libs/slf4j-api.jar:../../../libs/slf4j-simple.jar --module my.app/mypackage.App

[main] INFO mypackage.App - running app
running module-a
```




```
module my.app {
    requires my.module.a;
    requires org.slf4j;
}
```

```
module my.module.a {
    exports mypackage.a;
}
```

```
LoggerFactory.getLogger(App.class).info(
  "running app in module" +
  App.class.getModule().getName()
);
```

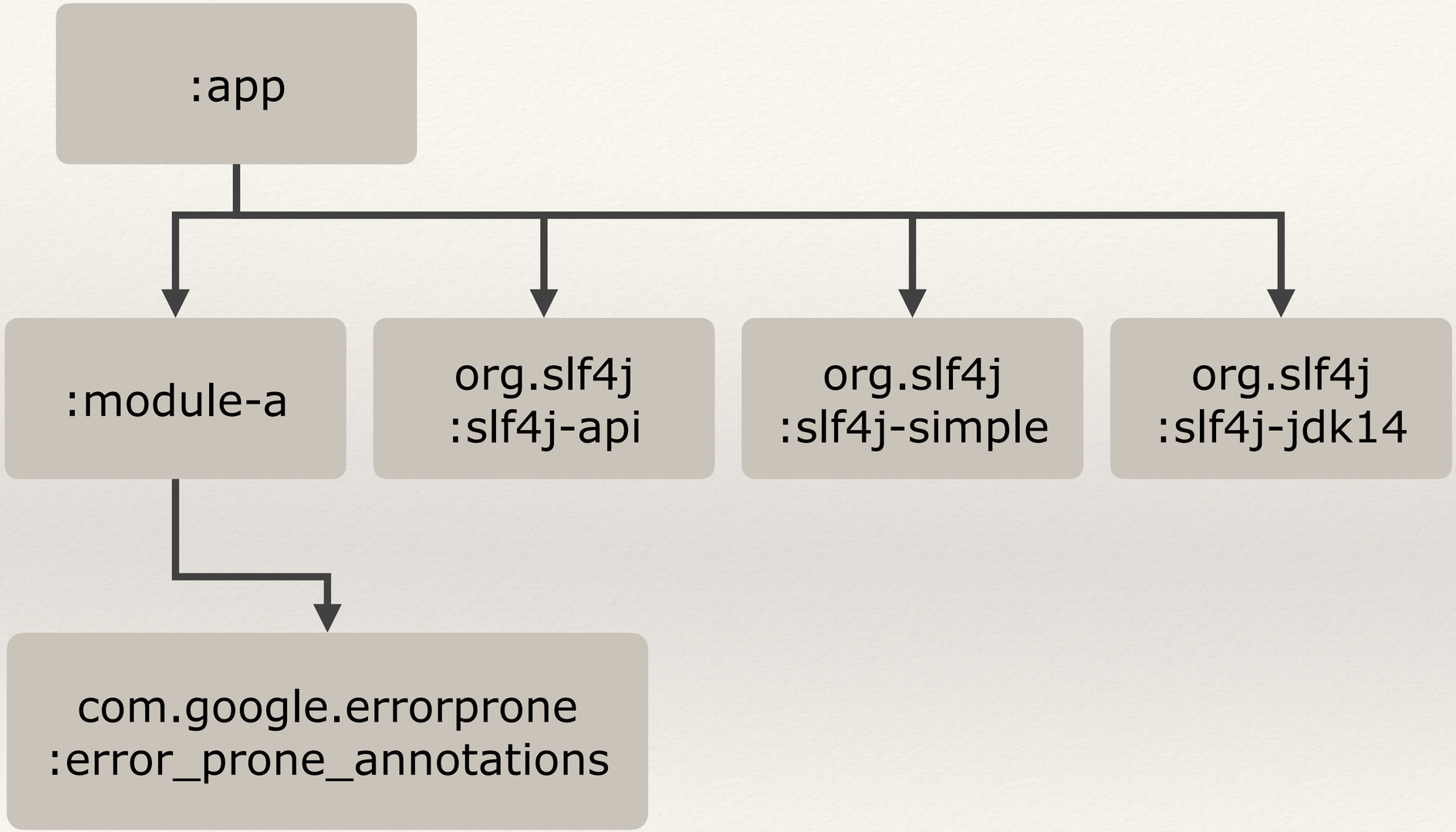
```
$ javac -d ../../../build/classes --module-path ../../../../module-a/build/classes:
  ../../../libs/slf4j-api.jar mypackage/App.java module-info.java

$ java -classpath ../../../build/classes:../../../../module-a/build/classes:
  ../../../libs/slf4j-api.jar:../../../libs/slf4j-simple.jar mypackage.App

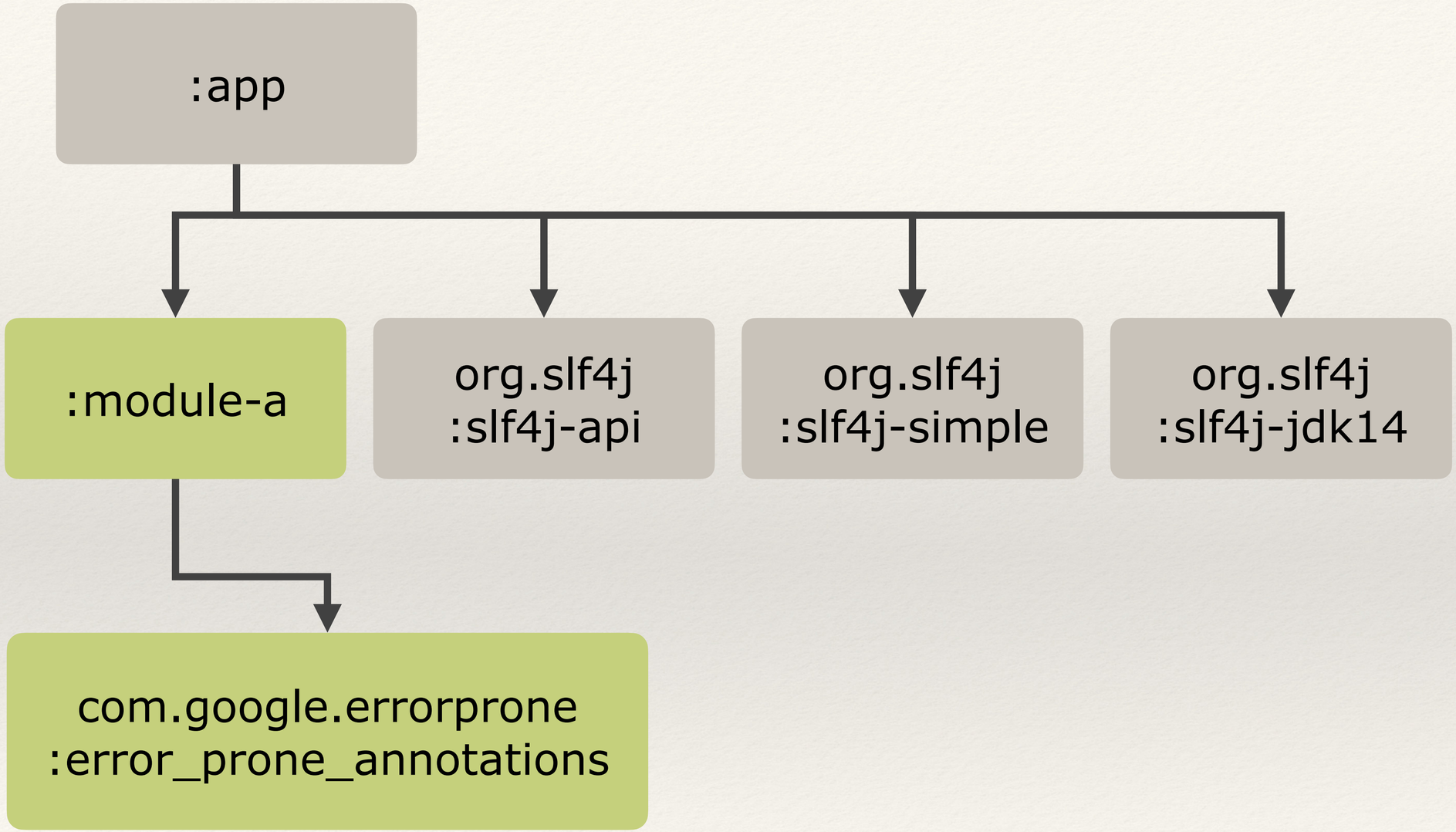
[main] INFO mypackage.App - running app in module null
running module-a

 $ java --module-path ../../../build/classes:../../../../module-a/build/classes:
   ../../../libs/slf4j-api.jar:../../../libs/slf4j-simple.jar --module my.app/mypackage.App

[main] INFO mypackage.App - running app in module my.app
running module-a
```

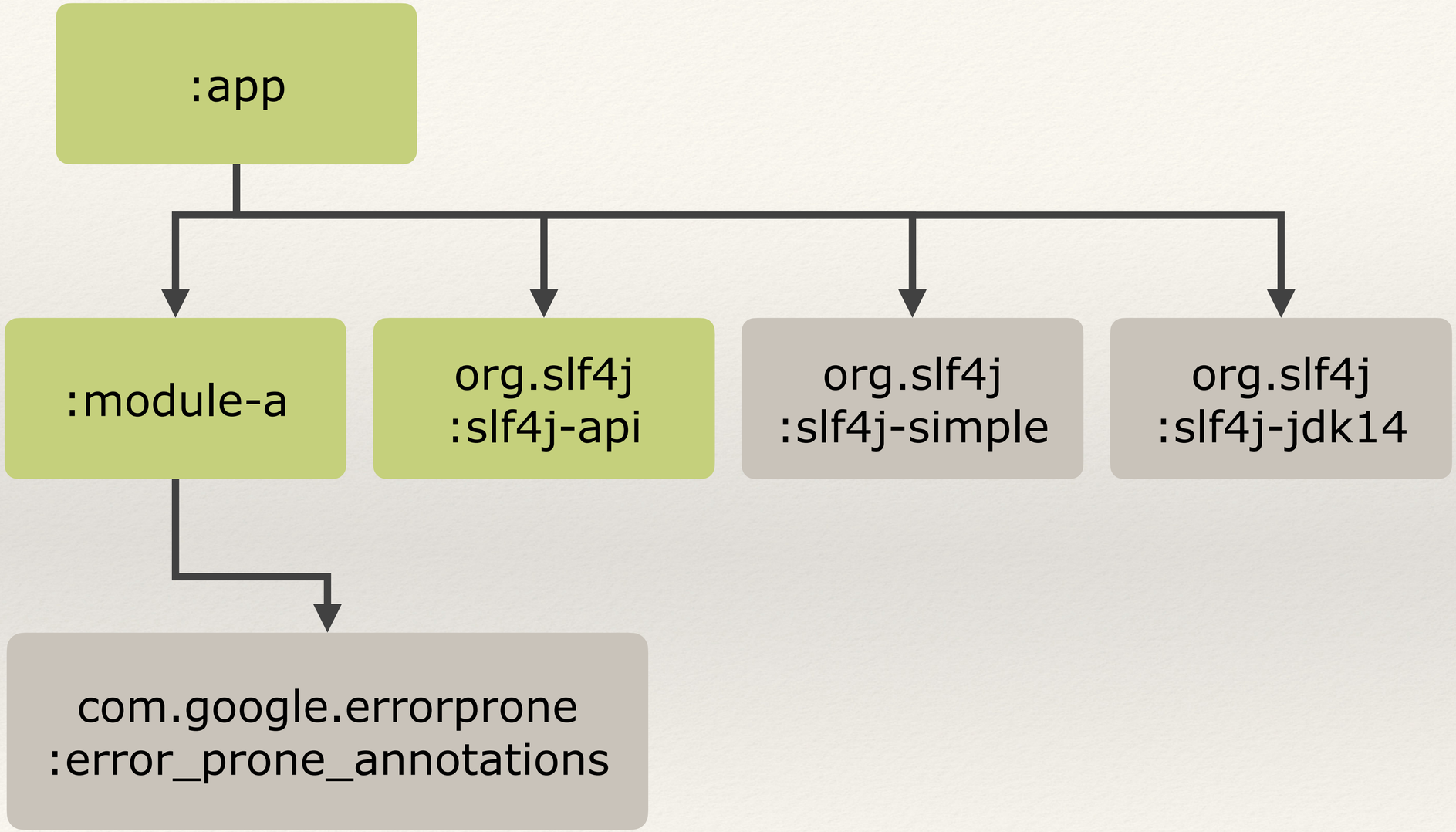

:app
:module-a
org.slf4j
:slf4j-api
org.slf4j
:slf4j-simple
org.slf4j
:slf4j-jdk14
com.google.errorprone
:error_prone_annotations

```
javac --module-path slf4j-api.jar:slf4j-jdk14.jar:error_prone_annotations.jar
```

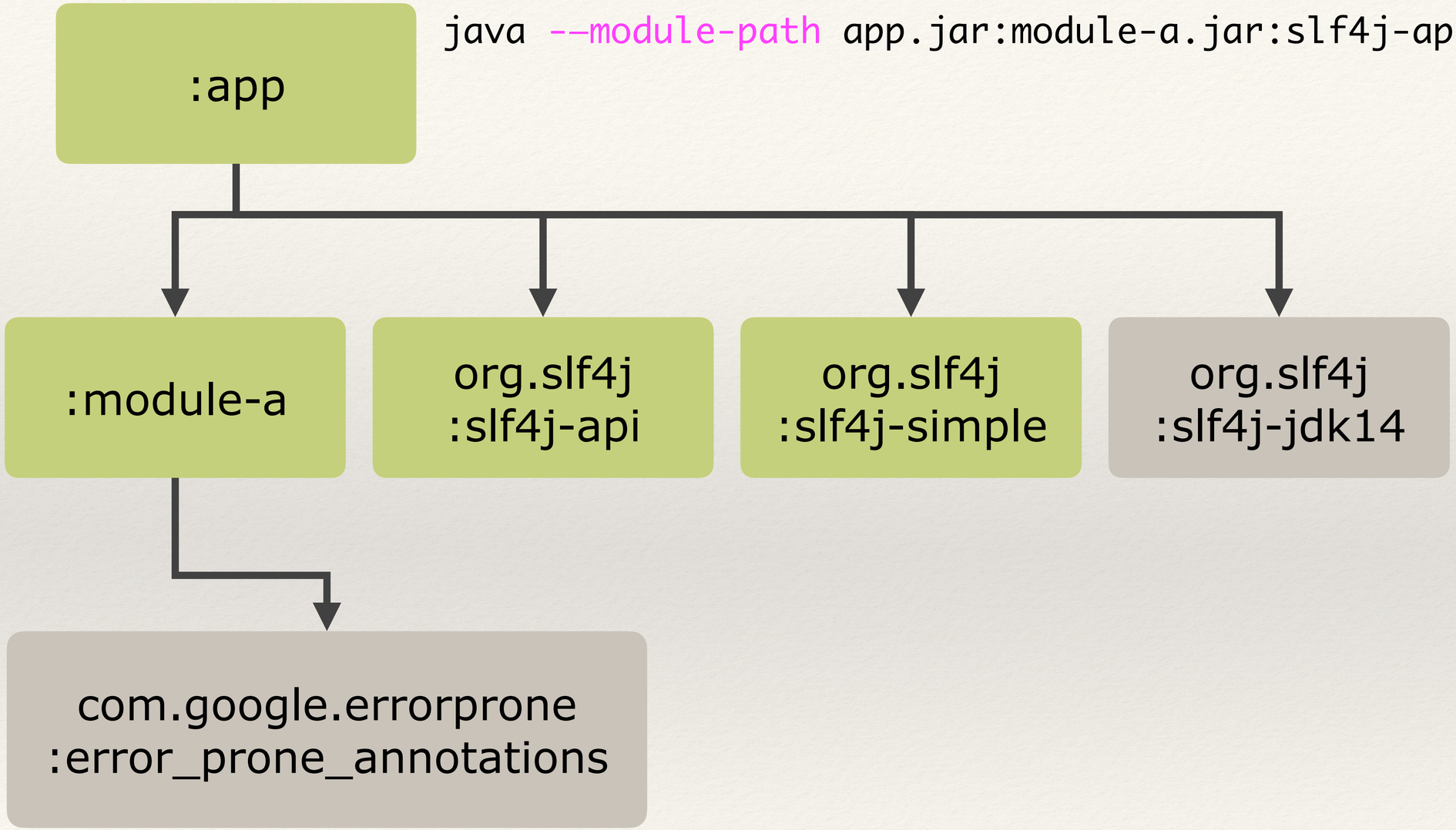


```
javac --module-path slf4j-api.jar:slf4j-jdk14.jar:error_prone_annotations.jar
```

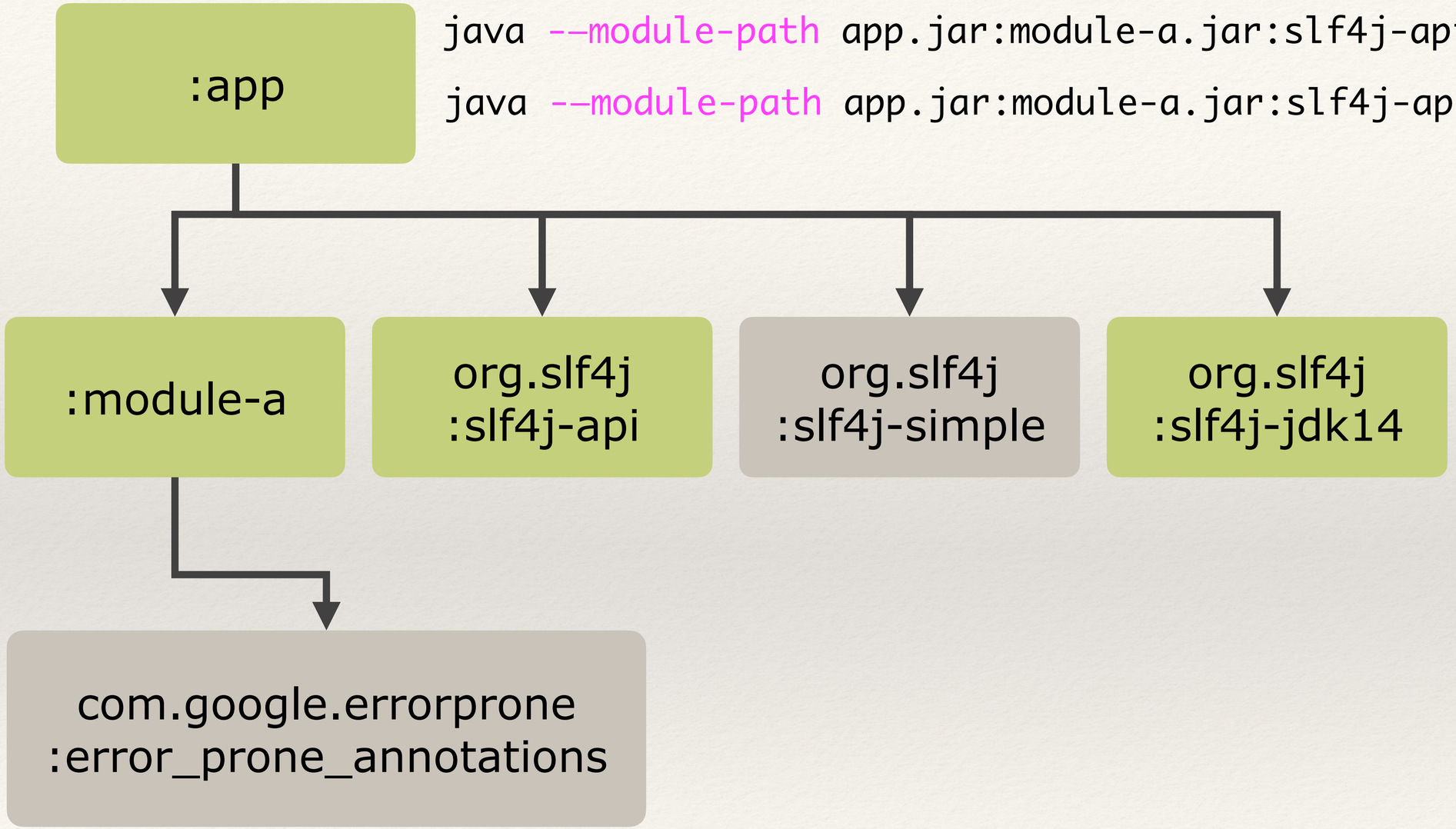
```
javac --module-path module-a.jar:slf4j-api.jar:slf4j-jdk14.jar:error_prone_annotations.jar
```

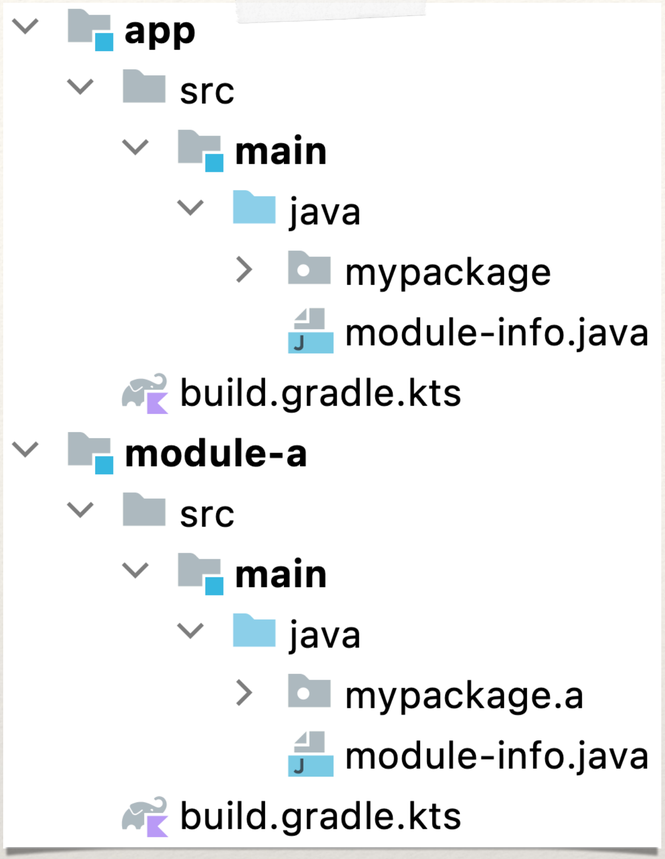


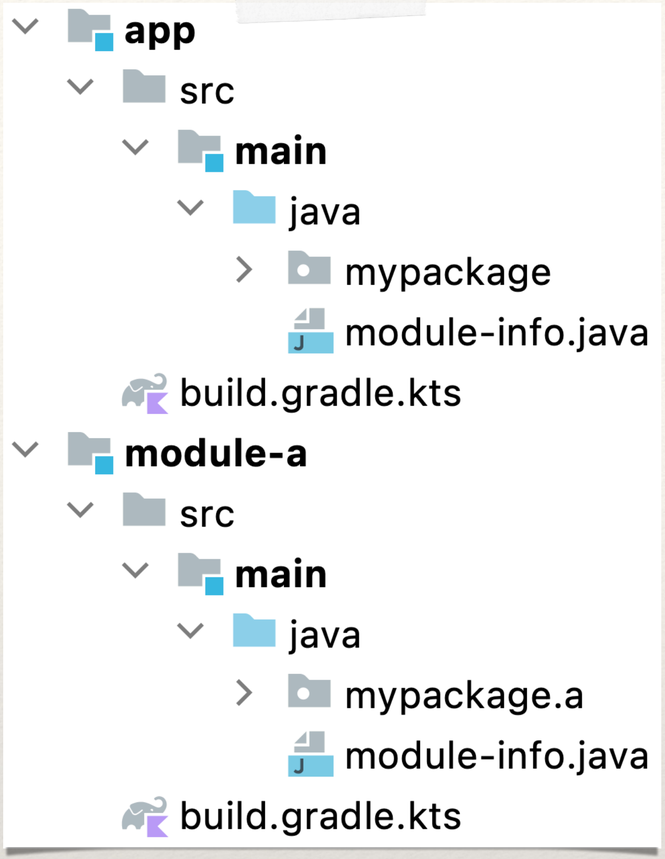
```
javac --module-path slf4j-api.jar:slf4j-jdk14.jar:error_prone_annotations.jar
javac --module-path module-a.jar:slf4j-api.jar:slf4j-jdk14.jar:error_prone_annotations.jar
java --module-path app.jar:module-a.jar:slf4j-api.jar:slf4j-simple.jar
```



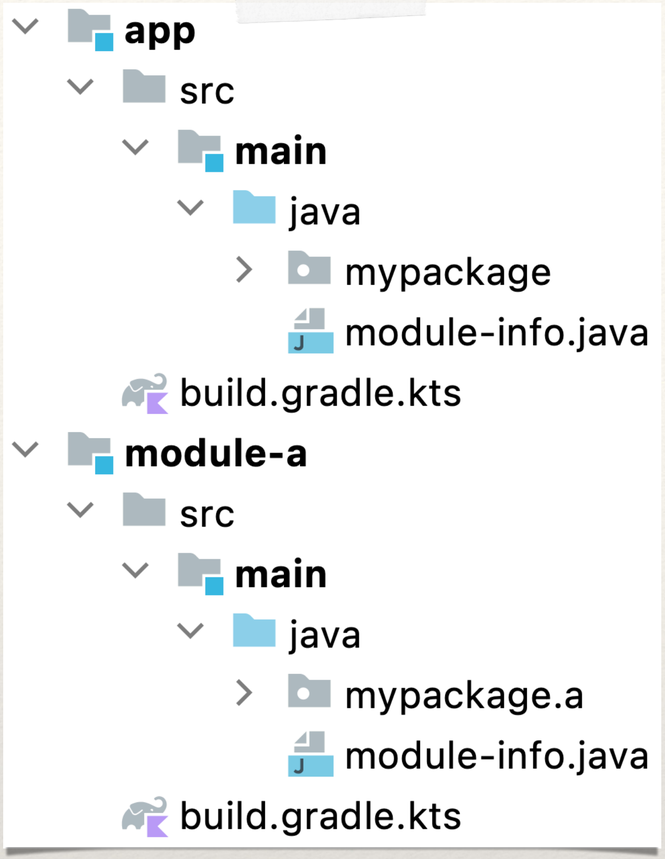
```
javac --module-path slf4j-api.jar:slf4j-jdk14.jar:error_prone_annotations.jar
javac --module-path module-a.jar:slf4j-api.jar:slf4j-jdk14.jar:error_prone_annotations.jar
java --module-path app.jar:module-a.jar:slf4j-api.jar:slf4j-simple.jar
java --module-path app.jar:module-a.jar:slf4j-api.jar:slf4j-jdk14.jar
```




app
src
main
java
mypackage
module-info.java
build.gradle.kts
module-a
src
main
java
mypackage.a
module-info.java
build.gradle.kts



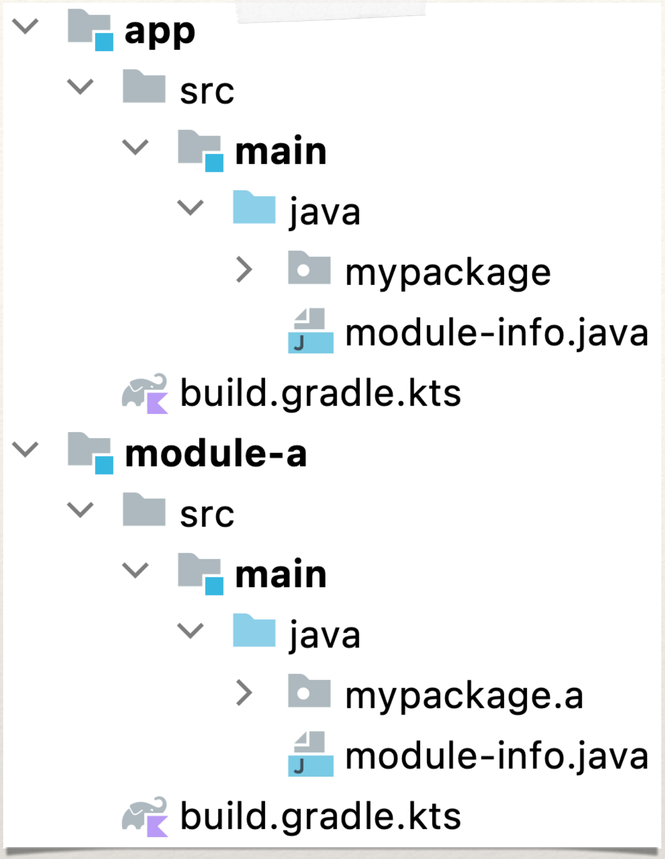

```
module my.app {
    requires my.module.a;
    requires org.slf4j;
}
```




```
module my.app {
    requires my.module.a;
    requires org.slf4j;
}
```

```
plugins {
  id("application") // java-application
}

dependencies {
  implementation(project(":module-a"))
  implementation("org.slf4j:slf4j-api:2.0.6")
  runtimeOnly("org.slf4j:slf4j-simple:2.0.6")
}
```

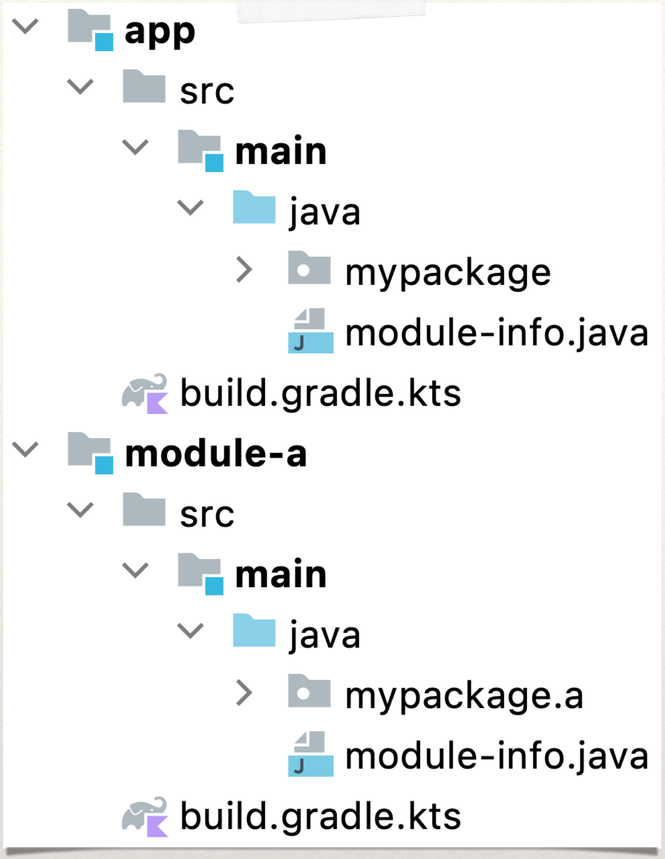



```
module my.app {
    requires my.module.a;
    requires org.slf4j;
}
```

```
plugins {
  id("application") // java-application
}

dependencies {
  implementation(project(":module-a"))
  implementation("org.slf4j:slf4j-api:2.0.6")
  runtimeOnly("org.slf4j:slf4j-simple:2.0.6")
}
```

```
module my.module.a {
  requires static com.google.errorprone.annotations;
  exports mypackage.a;
}
```




```
module my.app {
    requires my.module.a;
    requires org.slf4j;
}
```

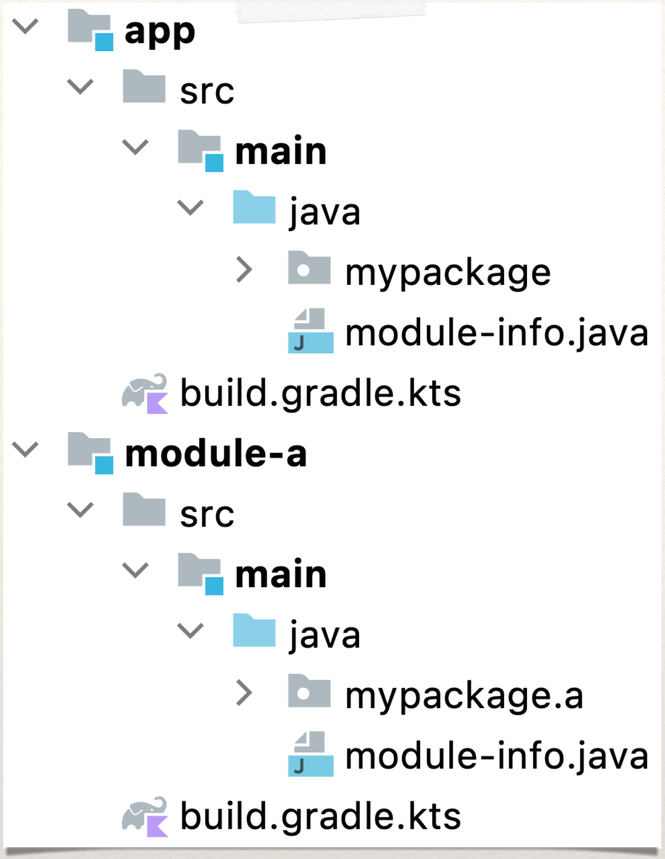
```
plugins {
  id("application") // java-application
}

dependencies {
  implementation(project(":module-a"))
  implementation("org.slf4j:slf4j-api:2.0.6")
  runtimeOnly("org.slf4j:slf4j-simple:2.0.6")
}
```

```
module my.module.a {
  requires static com.google.errorprone.annotations;
  exports mypackage.a;
}
```

```
plugins {
  id("java-library")
}

dependencies {
  compileOnly("com.google.errorprone:error_prone_annotations:2.16")
}
```



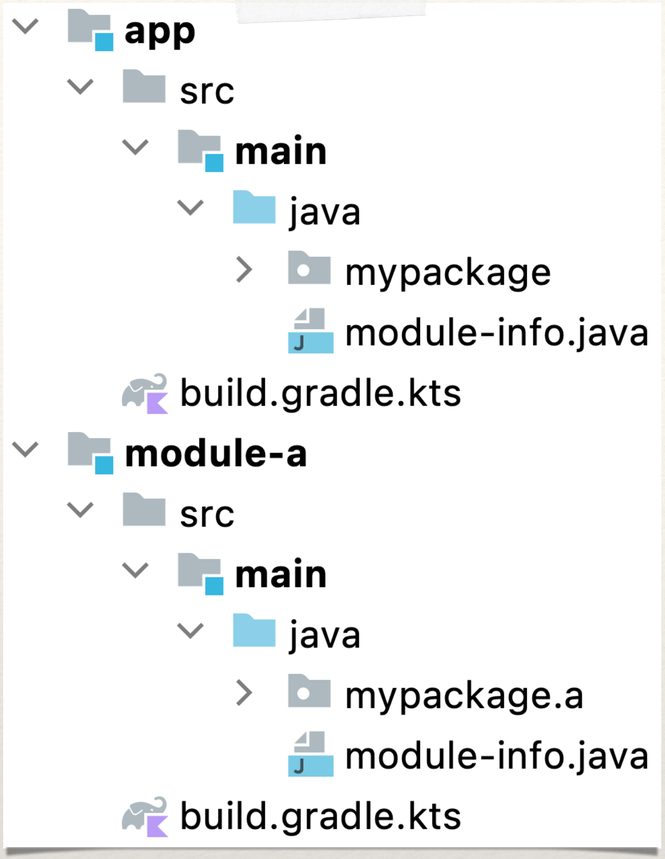

```
module my.app {
    requires my.module.a;
    requires org.slf4j;
}
```

```
p
}
de
}
```

```
mo
}
```

```
p
}

dependencies {
  compileOnly("com.google.errorprone:error_prone_annotations:2.16")
}
```




```
module my.app {
    requires my.module.a;
    requires org.slf4j;
}
```

```
$ gradle :app:run
> Task :module-a:compileJava UP-TO-DATE
> Task :app:compileJava
> Task :app:processResources NO-SOURCE
> Task :app:classes UP-TO-DATE
> Task :module-a:processResources NO-SOURCE
> Task :module-a:classes UP-TO-DATE
> Task :module-a:jar

> Task :app:run
[main] INFO mypackage.App - running app in module my.app
running module-a
```

```
}

dependencies {
  compileOnly("com.google.errorprone:error_prone_annotations:2.16")
}
```

Modularizing Java Projects

*(1) Modularization with the **Classpath** + Gradle*

*(2) Modularization with the **Module Path***

*(3) Modularization with the **Module Path** + Gradle + X*

Modularizing Java Projects

(1) Modularization with the ***Classpath*** *+ Gradle*

(2) Modularization with the ***Module Path***

(3) Modularization with the ***Module Path*** *+ Gradle + X*

	Classpath w/ Gradle	*Module Path (plain Java)*
discover modules in repos	✅	❌
module version management	✅	❌
conflict detection	✅	✅
(automatic) conflict resolution	✅	❌
integrate non-module Jars	✅	⚠️
scopes at compile time	✅	✅
private packages	❌	✅
testing of private units	✅	⚠️
JDK tools (e.g. jlink)	❌	✅
visibility at runtime	❌	✅

	Classpath w/ Gradle	*Module Path (plain Java)*
discover modules in repos	✅	❌
module version management	✅	❌
conflict detection	✅	✅
(automatic) conflict resolution	✅	❌
integrate non-module Jars	✅	⚠️
scopes at compile time	✅	✅
private packages	❌	✅
testing of private units	✅	⚠️
JDK tools (e.g. jlink)	❌	✅
visibility at runtime	❌	✅

	Module Path w/ plain Gradle
discover modules in repos	✅
module version management	✅
conflict detection	✅ ✅
(automatic) conflict resolution	✅
integrate non-module Jars	⚠️
scopes at compile time	✅ ⚠️ (redundancy)
private packages	⭐ ✅ ⭐
testing of private units	⚠️
JDK tools (e.g. jlink)	✅
visibility at runtime	✅

	Module Path w/ plain Gradle
discover modules in repos	✅
module version management	✅
conflict detection	✅✅
(automatic) conflict resolution	✅
integrate non-module Jars	⚠️ - org.gradlex.extra-java-module-info
scopes at compile time	⭐✅⭐ - org.gradlex.java-module-dependencies
private packages	⭐✅⭐
testing of private units	✅ - org.gradlex.java-module-testing
JDK tools (e.g. jlink)	✅
visibility at runtime	✅

Modularizing Java Projects

*(1) Modularization with the **Classpath** + Gradle*

*(2) Modularization with the **Module Path***

*(3) Modularization with the **Module Path** + Gradle + X*

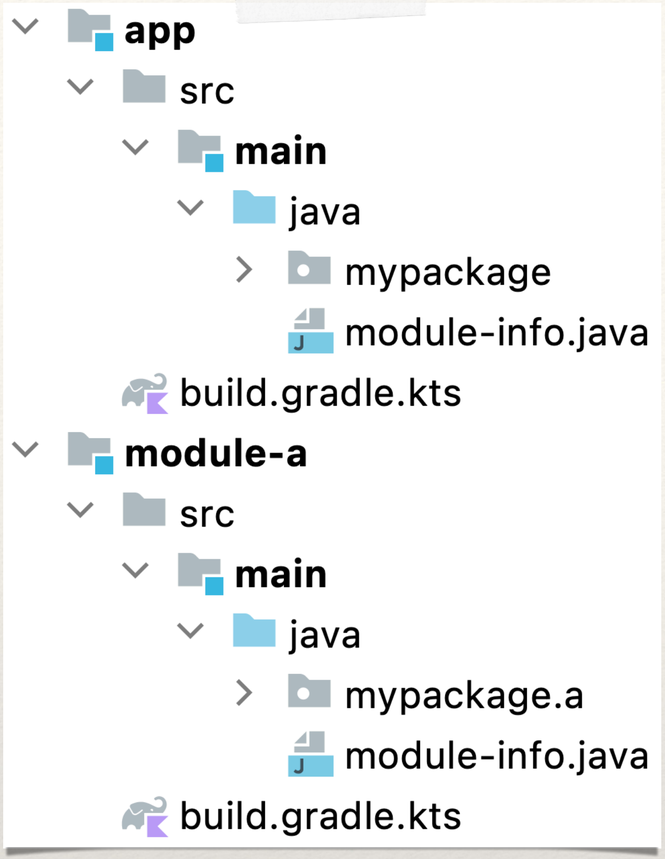
Modularizing Java Projects

*(1) Modularization with the **Classpath** + Gradle*

*(2) Modularization with the **Module Path***

*(3) Modularization with the **Module Path** + Gradle + X*

	Module Path w/ plain Gradle
discover modules in repos	✅
module version management	✅
conflict detection	✅✅
(automatic) conflict resolution	✅
integrate non-module Jars	⚠️ - org.gradlex.extra-java-module-info
scopes at compile time	⭐✅⭐ - org.gradlex.java-module-dependencies
private packages	⭐✅⭐
testing of private units	✅ - org.gradlex.java-module-testing
JDK tools (e.g. jlink)	✅
visibility at runtime	✅




```
module my.app {
  requires my.module.a;
  requires org.slf4j;
}
```

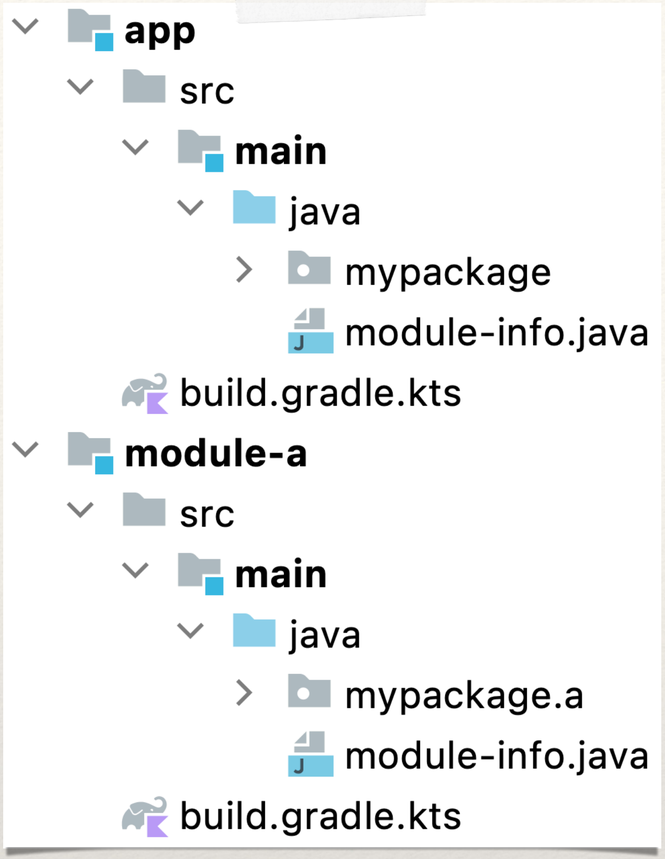
```
plugins {
  id("application")
}

dependencies {
  implementation(project(":module-a"))
  implementation("org.slf4j:slf4j-api:2.0.6")
  runtimeOnly("org.slf4j:slf4j-simple:2.0.6")
}
```

```
module my.module.a {
  requires static com.google.errorprone.annotations;
  exports mypackage.a;
}
```

```
plugins {
  id("java-library")
}

dependencies {
  compileOnly("com.google.errorprone:error_prone_annotations:2.16")
}
```




```
module my.app {
  requires my.module.a;
  requires org.slf4j;
}
```

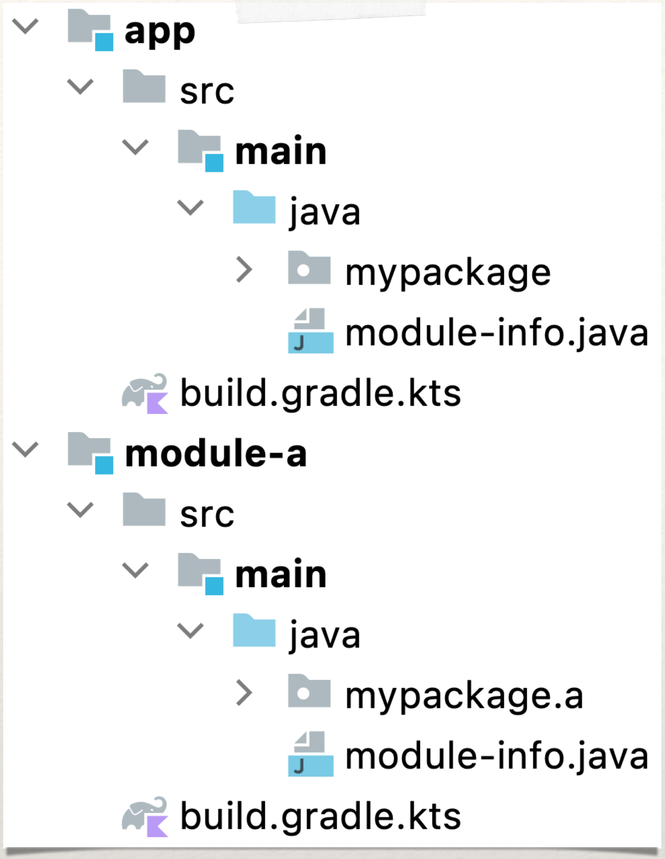
```
plugins {
  id("application")
}

dependencies {
  implementation(project(":module-a"))
  implementation("org.slf4j:slf4j-api:2.0.6")
  runtimeOnly("org.slf4j:slf4j-simple:2.0.6")
}
```

```
module my.module.a {
  requires static com.google.errorprone.annotations;
  exports mypackage.a;
}
```

```
plugins {
  id("java-library")
}

dependencies {
  compileOnly("com.google.errorprone:error_prone_annotations:2.16")
}
```




```
module my.app {
  requires my.module.a;
  requires org.slf4j;
}
```

Scope Mappings

`requires`	=>	`implementation`
`requires static`	=>	`compileOnly`
`requires transitive`	=>	`api`
`requires static transitive`	=>	`compileOnlyApi`

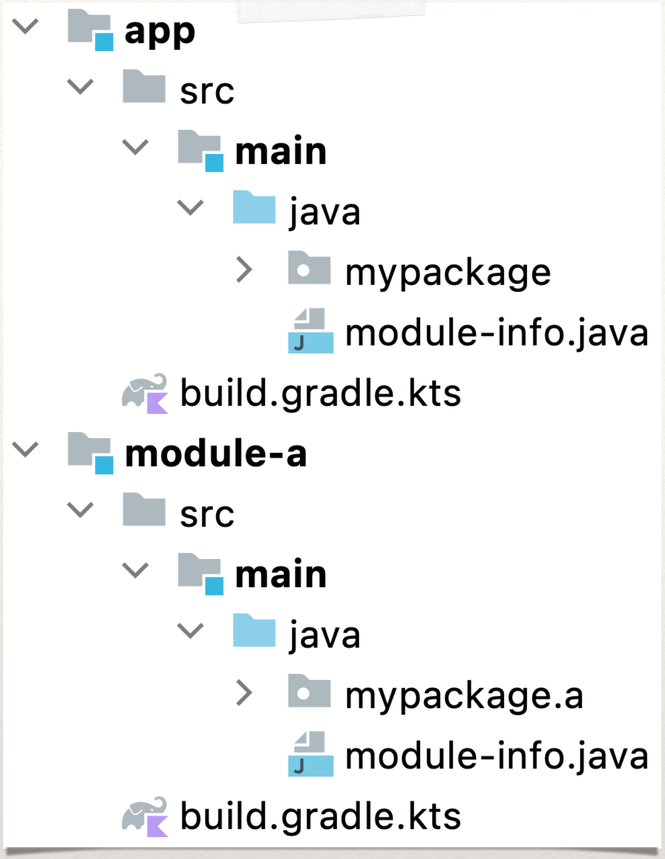
```
plugins {
  id("application")
}

dependencies {
  implementation(project(":module-a"))
  implementation("org.slf4j:slf4j-api:2.0.6")
  runtimeOnly("org.slf4j:slf4j-simple:2.0.6")
}
```

```
module my.module.a {
  requires static com.google.errorprone.annotations;
  exports mypackage.a;
}
```

```
plugins {
  id("java-library")
}

dependencies {
  compileOnly("com.google.errorprone:error_prone_annotations:2.16")
}
```




```
module my.app {
  requires my.module.a;
  requires org.slf4j;
}
```

Scope Mappings

`requires`	=>	`implementation`
`requires static`	=>	`compileOnly`
`requires transitive`	=>	`api`
`requires static transitive`	=>	`compileOnlyApi`

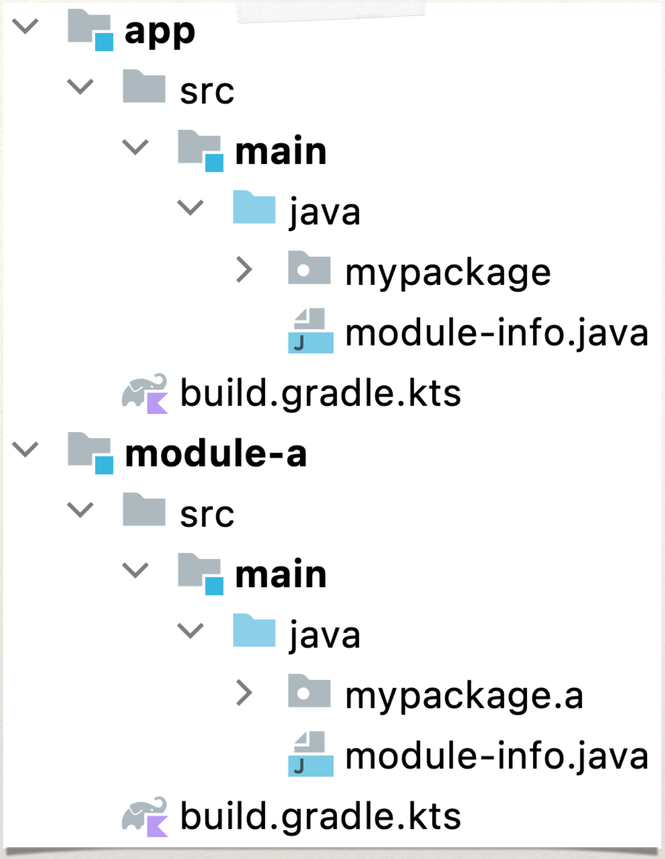
```
plugins {
  id("application")
}

dependencies {
  implementation(project(":module-a"))
  implementation("org.slf4j:slf4j-api:2.0.6")
  runtimeOnly("org.slf4j:slf4j-simple:2.0.6")
}
```

```
module my.module.a {
  requires static com.google.errorprone.annotations;
  exports mypackage.a;
}
```

```
plugins {
  id("java-library")
}

dependencies {
  compileOnly("com.google.errorprone:error_prone_annotations:2.16")
}
```




```
module my.app {
  requires my.module.a;
  requires org.slf4j;
}
```

Scope Mappings

```
requires                => implementation
requires static         => compileOnly
requires transitive     => api
requires static transitive => compileOnlyApi
```

```
plugins {
  id("application")
}

dependencies {
  implementation(project(":module-a"))
  implementation("org.slf4j:slf4j-api:2.0.6")
  runtimeOnly("org.slf4j:slf4j-simple:2.0.6")
}
```

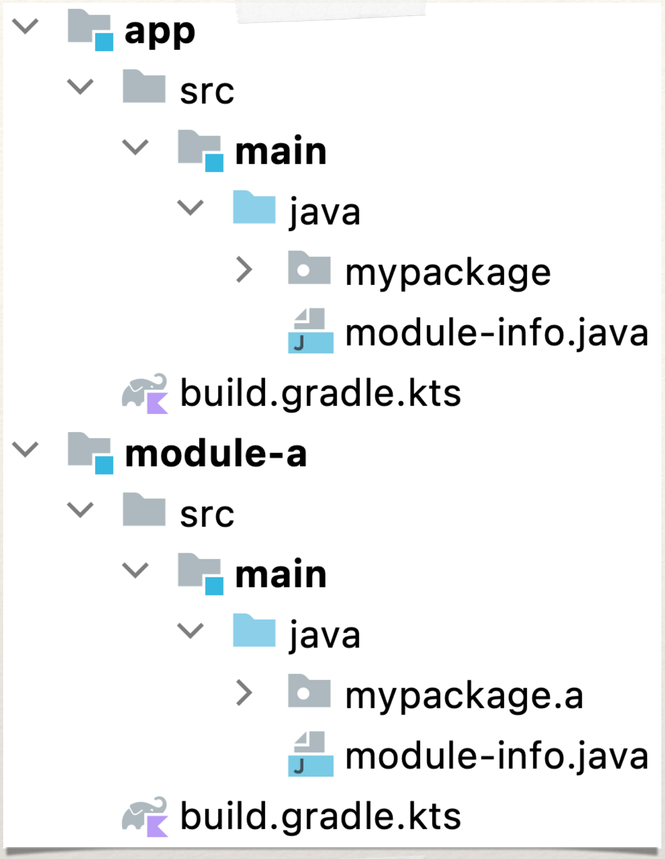
Module Name to Coordinates Mapping

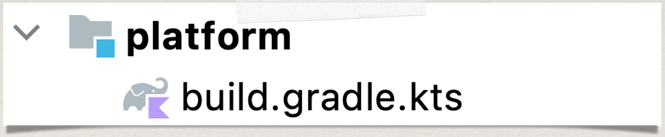
```
org.slf4j => org.slf4j:slf4j-api
com.google.errorprone.annotations
  => com.google.errorprone:error_prone_annotations
```

```
module my.module.a {
  requires static com.google.errorprone.annotations;
  exports mypackage.a;
}
```

```
plugins {
  id("java-library")
}

dependencies {
  compileOnly("com.google.errorprone:error_prone_annotations:2.16")
}
```






```
module my.app {
  requires my.module.a;
  requires org.slf4j;
}
```

Scope Mappings

```
requires                => implementation
requires static         => compileOnly
requires transitive     => api
requires static transitive => compileOnlyApi
```

```
plugins {
  id("application")
}

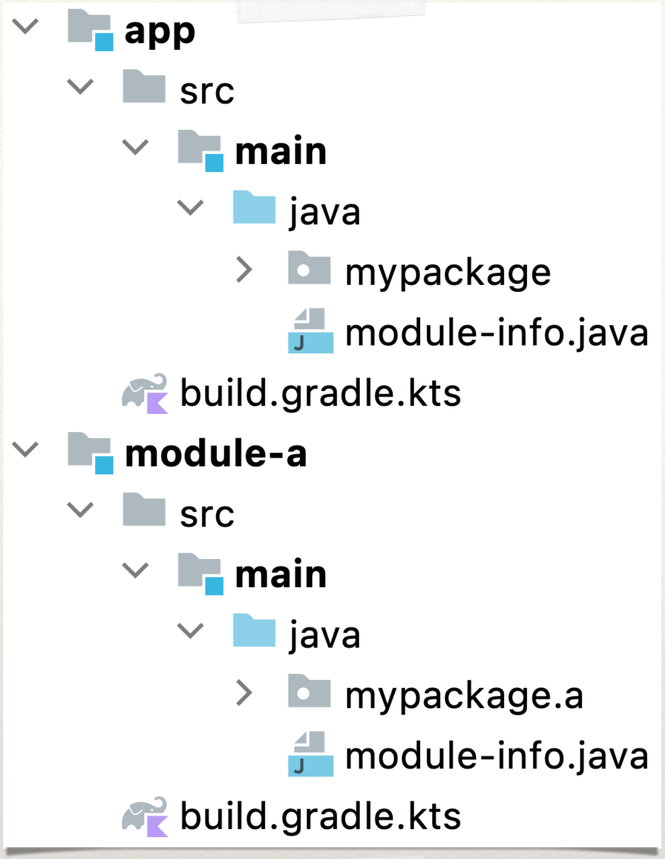
dependencies {
  implementation(project(":module-a"))
  implementation("org.slf4j:slf4j-api")
  runtimeOnly("org.slf4j:slf4j-simple")
}
```

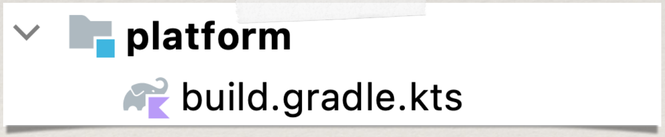
Module Name to Coordinates Mapping

```
org.slf4j => org.slf4j:slf4j-api
com.google.errorprone.annotations
  => com.google.errorprone:error_prone_annotations
```

```
plugins {
  id("java-platform")
}

dependencies.constraints {
  api("org.slf4j:slf4j-api:2.0.6")
  api("org.slf4j:slf4j-simple:2.0.6")
  api("com.google.errorprone:error_prone_annotations:2.16")
}
```






```
module my.app {
  requires my.module.a;
  requires org.slf4j;
}
```

Scope Mappings

```
requires                => implementation
requires static         => compileOnly
requires transitive     => api
requires static transitive => compileOnlyApi
```

```
plugins {
  id("application")
}

dependencies {
  implementation(project(":module-a"))
  implementation("org.slf4j:slf4j-api")
  runtimeOnly("org.slf4j:slf4j-simple")
}
```

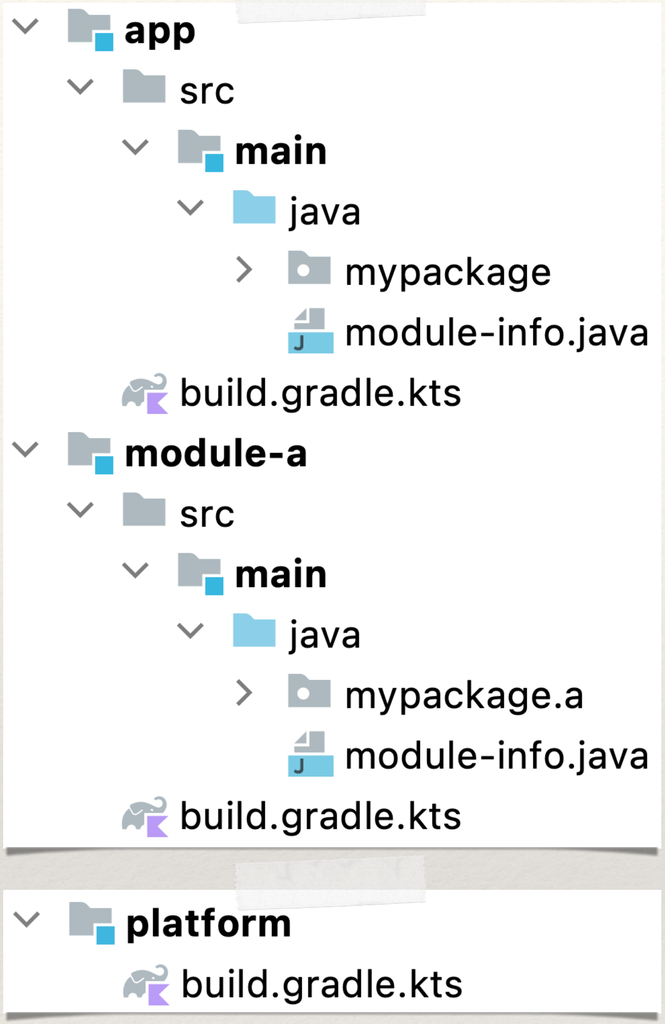
Module Name to Coordinates Mapping

```
org.slf4j => org.slf4j:slf4j-api
com.google.errorprone.annotations
  => com.google.errorprone:error_prone_annotations
```

```
module my.module.a {
  requires static com.google.errorprone.annotations;
  exports mypackage.a;
}
```

```
plugins {
  id("java-library")
}

dependencies {
  compileOnly("com.google.errorprone:error_prone_annotations")
}
```




```
module my.app {
  requires my.module.a;
  requires org.slf4j;
}
```

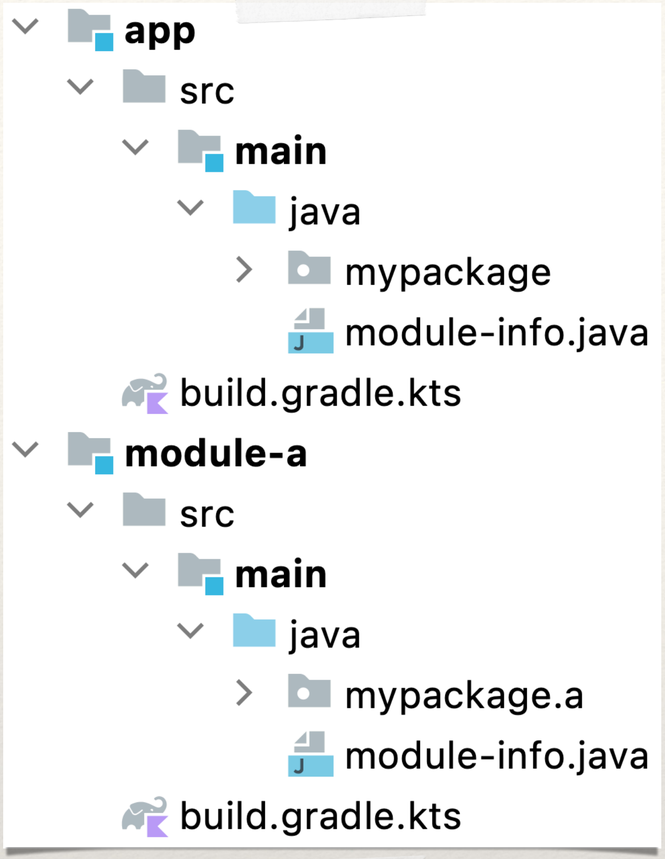
```
plugins {
  id("application")
}

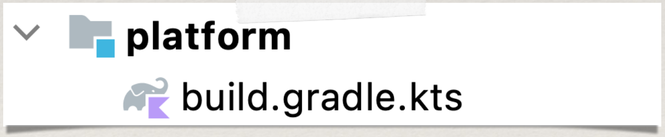
dependencies {
  implementation(project(":module-a"))
  implementation("org.slf4j:slf4j-api")
  runtimeOnly("org.slf4j:slf4j-simple")
}
```

```
module my.module.a {
  requires static com.google.errorprone.annotations;
  exports mypackage.a;
}
```

```
plugins {
  id("java-library")
}

dependencies {
  compileOnly("com.google.errorprone:error_prone_annotations")
}
```





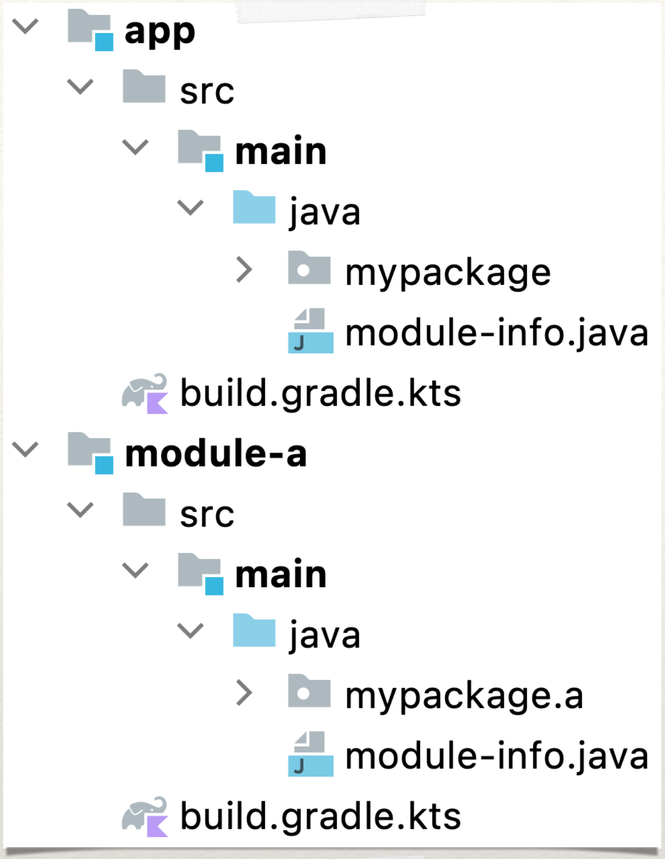

```
module my.app {
  requires my.module.a;
  requires org.slf4j;
}
```

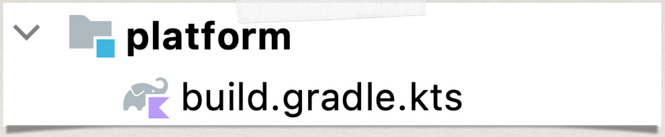
```
plugins {
  id("application")
  id("org.gradlex.java-module-dependencies")
}

dependencies {
  runtimeOnly("org.slf4j:slf4j-simple")
}
```

```
module my.module.a {
  requires static com.google.errorprone.annotations;
  exports mypackage.a;
}
```

```
plugins {
  id("java-library")
  id("org.gradlex.java-module-dependencies")
}
```






```
module my.app {
  requires my.module.a;
  requires org.slf4j;
  requires /*runtime*/ org.slf4j.simple;
}
```

```
plugins {
  id("application")
  id("org.gradlex.java-module-dependencies")
}
```

```
module my.module.a {
  requires static com.google.errorprone.annotations;
  exports mypackage.a;
}
```

```
plugins {
  id("java-library")
  id("org.gradlex.java-module-dependencies")
}
```

```
plugins {
  id("org.gradlex.java-module-dependencies")
}
```

```
com.fasterxml.jackson.annotation=com.fasterxml.jackson.core:jackson-annotations
com.fasterxml.jackson.core=com.fasterxml.jackson.core:jackson-core
com.fasterxml.jackson.databind=com.fasterxml.jackson.core:jackson-databind
com.fasterxml.jackson.dataformat.avro=com.fasterxml.jackson.dataformat:jackson-dataformat-avro
com.fasterxml.jackson.dataformat.cbor=com.fasterxml.jackson.dataformat:jackson-dataformat-cbor
com.fasterxml.jackson.dataformat.csv=com.fasterxml.jackson.dataformat:jackson-dataformat-csv
```

Built-in mappings for many Modules on Maven Central taken from: github.com/sormuras/modules
(Thanks to **Christian Stein**)

```
javaModuleDependencies {
    // Override because there are multiple options
    moduleNameToGA.put("jakarta.activation",
        "com.sun.activation:jakarta.activation")
    moduleNameToGA.put("jakarta.mail",
        "com.sun.mail:jakarta.mail")
}
```

If needed, add/override your own mappings

```
plugins {
  id("org.gradlex.java-module-dependencies")
}
```

build.gradle.kts (platform)

```
plugins {
    id("java-platform")
}

dependencies {
    api(platform("com.fasterxml.jackson:jackson-bom:2.13.2.20220328"))
    api(platform("com.google.inject:guice-bom:5.1.0"))
    api(platform("org.junit:junit-bom:5.7.2"))
}

dependencies.constraints {
    javaModuleDependencies {
        api(gav("jakarta.activation", "1.2.2"))
        api(gav("jakarta.mail", "1.6.7"))
        api(gav("java.inject", "1.0.5"))
        api(gav("java.servlet", "4.0.4")) {
            version { reject("[5.0.0,)") }
            because("Stay Tomcat 8 compatible")
        }
    }
}
```

Address modules by **Module Name** anywhere in your Gradle scripts

```
plugins {
  id("org.gradlex.java-module-dependencies")
}
```

```
$ ./gradlew :app:analyzeModulePath -q

[INFO] All Java Modules required by this project
================================================
com.fasterxml.jackson.annotation -> com.fasterxml.jackson.core:jackson-annotations (2.13.2)
com.fasterxml.jackson.core -> com.fasterxml.jackson.core:jackson-core (2.13.2)
com.fasterxml.jackson.databind -> com.fasterxml.jackson.core:jackson-databind (2.13.2)
org.apache.logging.log4j -> org.apache.logging.log4j:log4j-api (2.14.0)
org.apache.xmlbeans -> org.apache.xmlbeans:xmlbeans (5.0.1)
org.apiguardian.api -> org.apiguardian:apiguardian-api (1.1.2)
org.junit.jupiter.api -> org.junit.jupiter:junit-jupiter-api (5.8.2)
org.junit.jupiter.engine -> org.junit.jupiter:junit-jupiter-engine (5.8.2)
org.junit.platform.commons -> org.junit.platform:junit-platform-commons (1.8.2)
org.junit.platform.engine -> org.junit.platform:junit-platform-engine (1.8.2)
org.junit.platform.launcher -> org.junit.platform:junit-platform-launcher (1.8.2)
org.my.app -> project :app
org.my.lib -> project :lib
org.opentest4j -> org.opentest4j:opentest4j (1.2.0)
org.slf4j -> org.slf4j:slf4j-api (1.7.28)
org.slf4j.simple -> org.slf4j:slf4j-simple (1.7.28)

[WARN] Components that are NOT Java Modules
===========================================
commons-cli:commons-cli (1.5.0)

Notes / Options:
  - This may be ok if you use the Classpath (aka ALL-UNNAMED) in addition to the Module Path
  - Remove the dependencies or upgrade to higher versions
  - Patch legacy Jars to Modules: https://github.com/jjohannes/extra-java-module-info
```

:analyzeModulePath help task

```
plugins {
  id("org.gradlex.java-module-dependencies")
}
```

```
$ ./gradlew :app:recommendModuleVersions -q

Latest Stable Versions of Java Modules - use in your platform project's build.gradle(.kts)
================================================================================
dependencies.constraints {
    javaModuleDependencies {
        api(gav("com.fasterxml.jackson.annotation", "2.13.2"))
        api(gav("com.fasterxml.jackson.core", "2.13.2"))
        api(gav("com.fasterxml.jackson.databind", "2.13.2.2"))
        api(gav("org.apache.logging.log4j", "2.17.2"))
        api(gav("org.apache.xmlbeans", "5.0.3"))
        api(gav("org.junit.jupiter.api", "5.8.2"))
        api(gav("org.junit.jupiter.engine", "5.8.2"))
        api(gav("org.junit.platform.commons", "1.8.2"))
        api(gav("org.junit.platform.engine", "1.8.2"))
        api(gav("org.junit.platform.launcher", "1.8.2"))
        api(gav("org.opentest4j", "1.2.0"))
        api(gav("org.slf4j", "1.7.36"))
        api(gav("org.slf4j.simple", "1.7.36"))
    }
}
```

:recommendModuleVersions help task

```
plugins {
  id("org.gradlex.java-module-dependencies")
}
```

```
dependencies {
  implementation(project(":module-a"))
  implementation("org.slf4j:slf4j-api")
  runtimeOnly("org.slf4j:slf4j-simple")
}
```



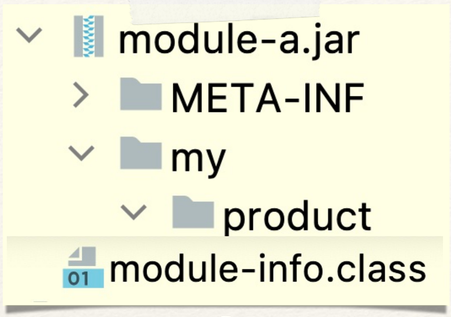
```
module my.app {
  requires org.slf4j;
  requires my.module.a;
  requires /*runtime*/ org.slf4j.simple;
}
```

: **generateAllModuleInfoFiles** for ⚠️ experimenting / getting started

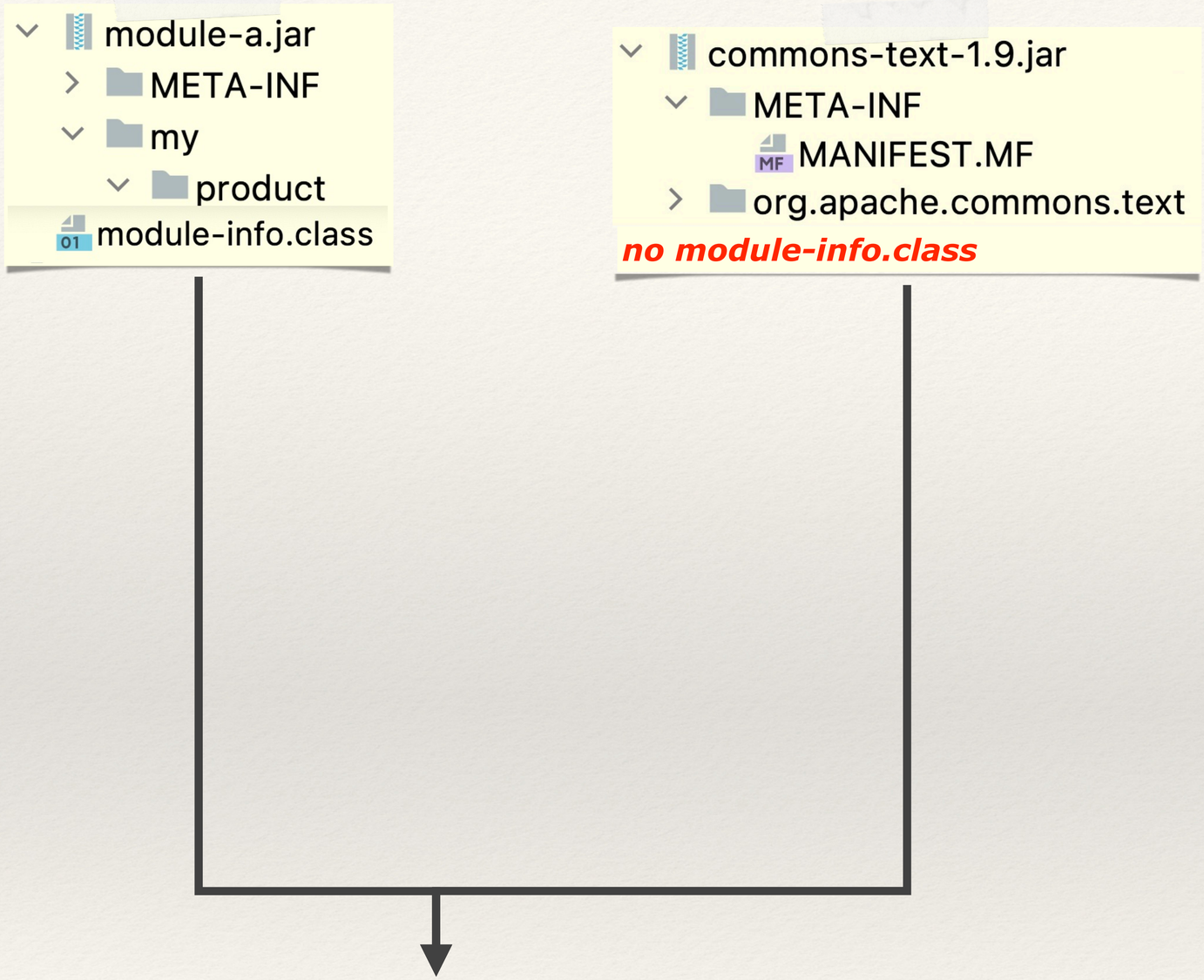
	Module Path w/ plain Gradle
discover modules in repos	✅
module version management	✅
conflict detection	✅✅
(automatic) conflict resolution	✅
integrate non-module Jars	⚠️ - org.gradlex.extra-java-module-info
scopes at compile time	⭐✅⭐ - org.gradlex.java-module-dependencies
private packages	⭐✅⭐
testing of private units	✅ - org.gradlex.java-module-testing
JDK tools (e.g. jlink)	✅
visibility at runtime	✅

	Module Path w/ plain Gradle
discover modules in repos	✅
module version management	✅
conflict detection	✅✅
(automatic) conflict resolution	✅
integrate non-module Jars	⚠️ - org.gradlex.extra-java-module-info
scopes at compile time	⭐✅⭐ - org.gradlex.java-module-dependencies
private packages	⭐✅⭐
testing of private units	✅ - org.gradlex.java-module-testing
JDK tools (e.g. jlink)	✅
visibility at runtime	✅

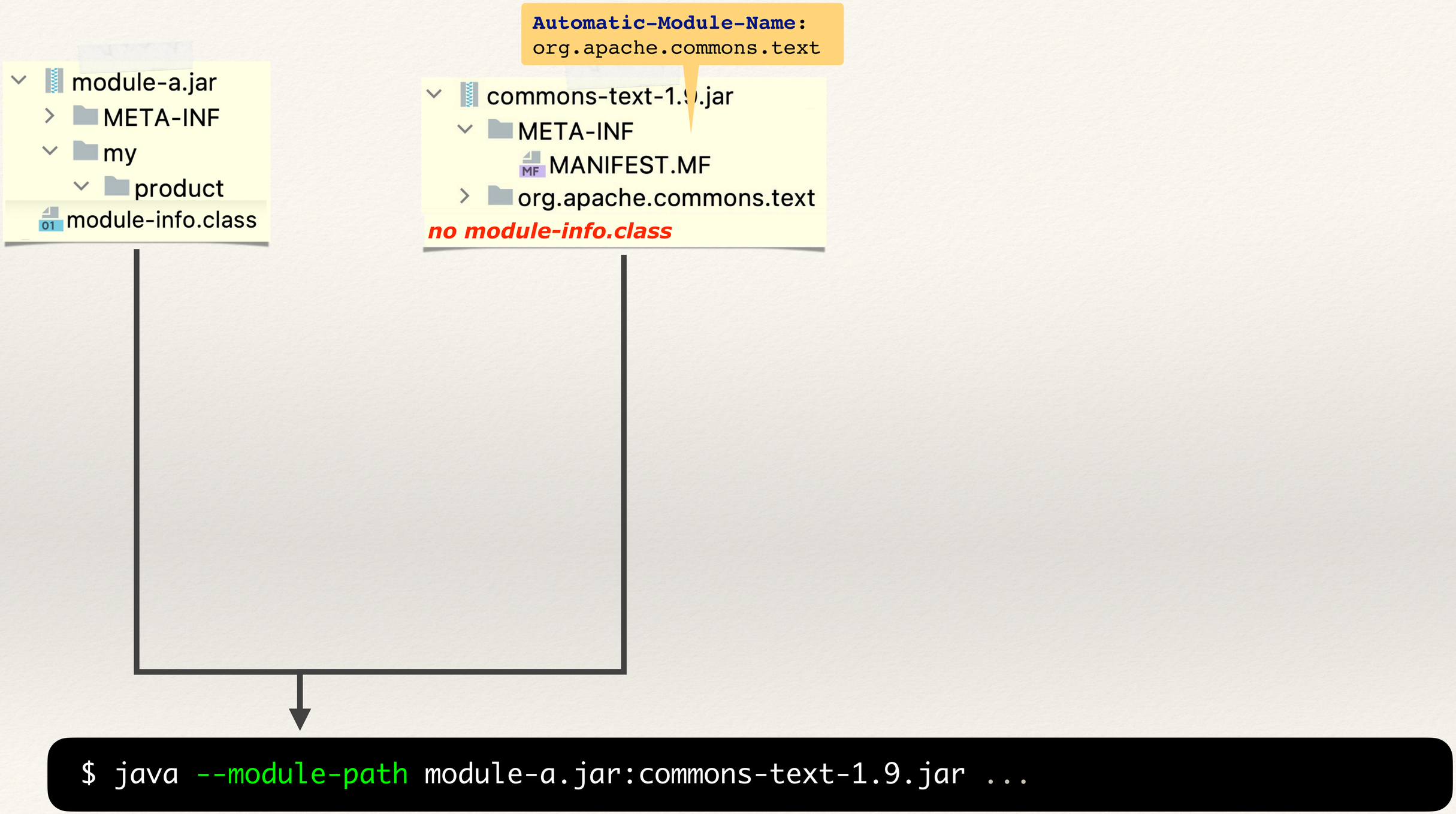
```
$ java --module-path module-a.jar:commons-text-1.9.jar ...
```

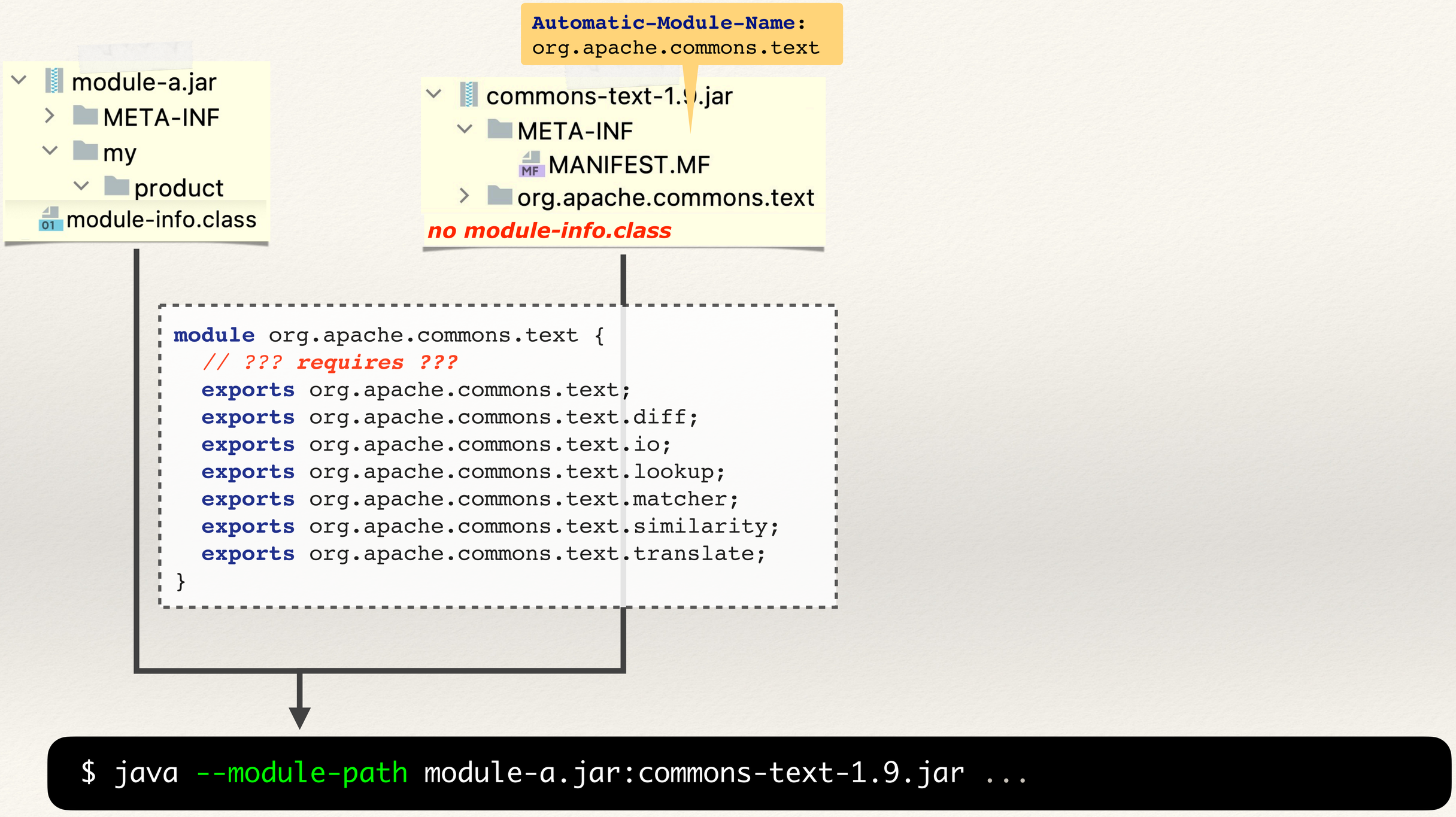


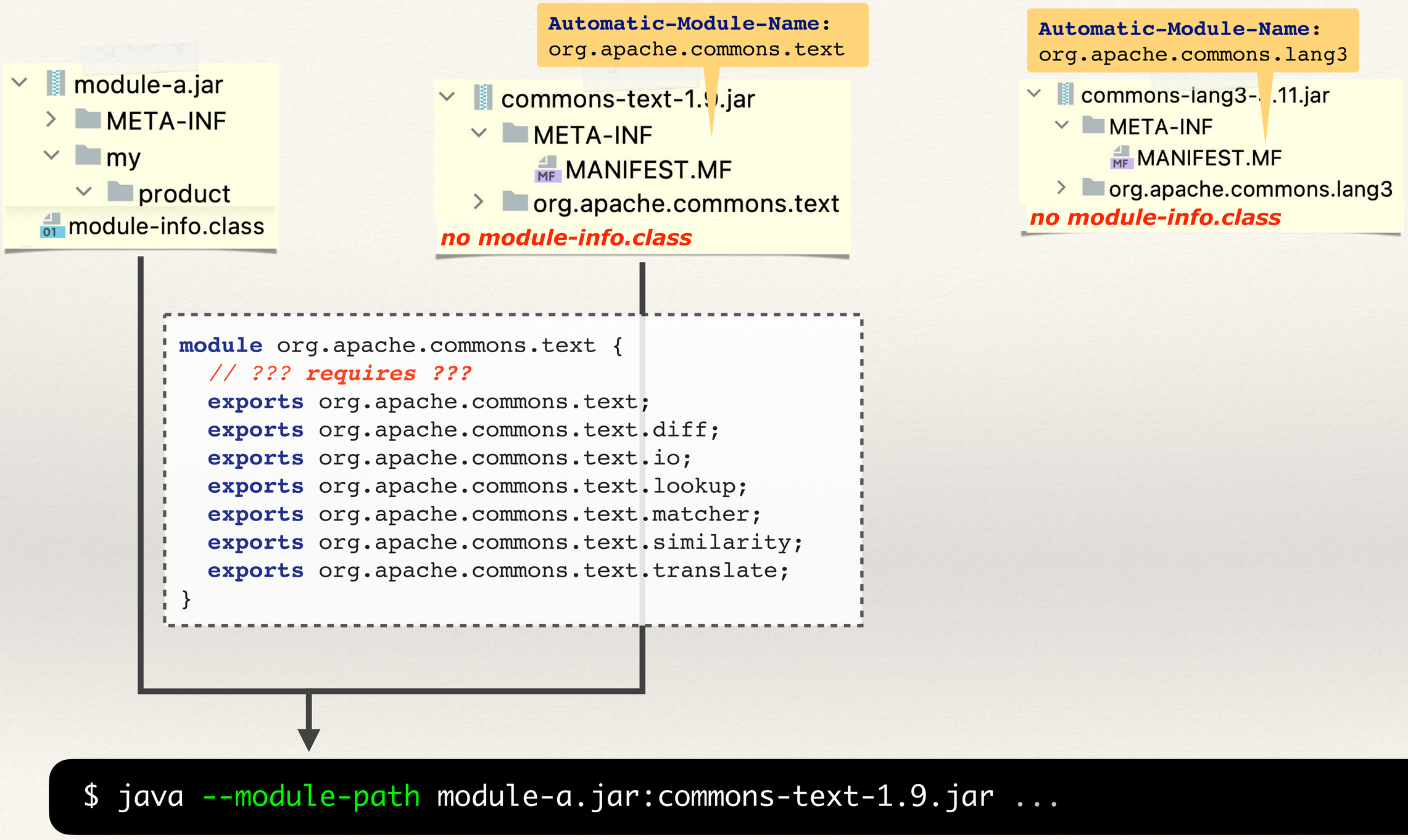

```
$ java --module-path module-a.jar:commons-text-1.9.jar ...
```

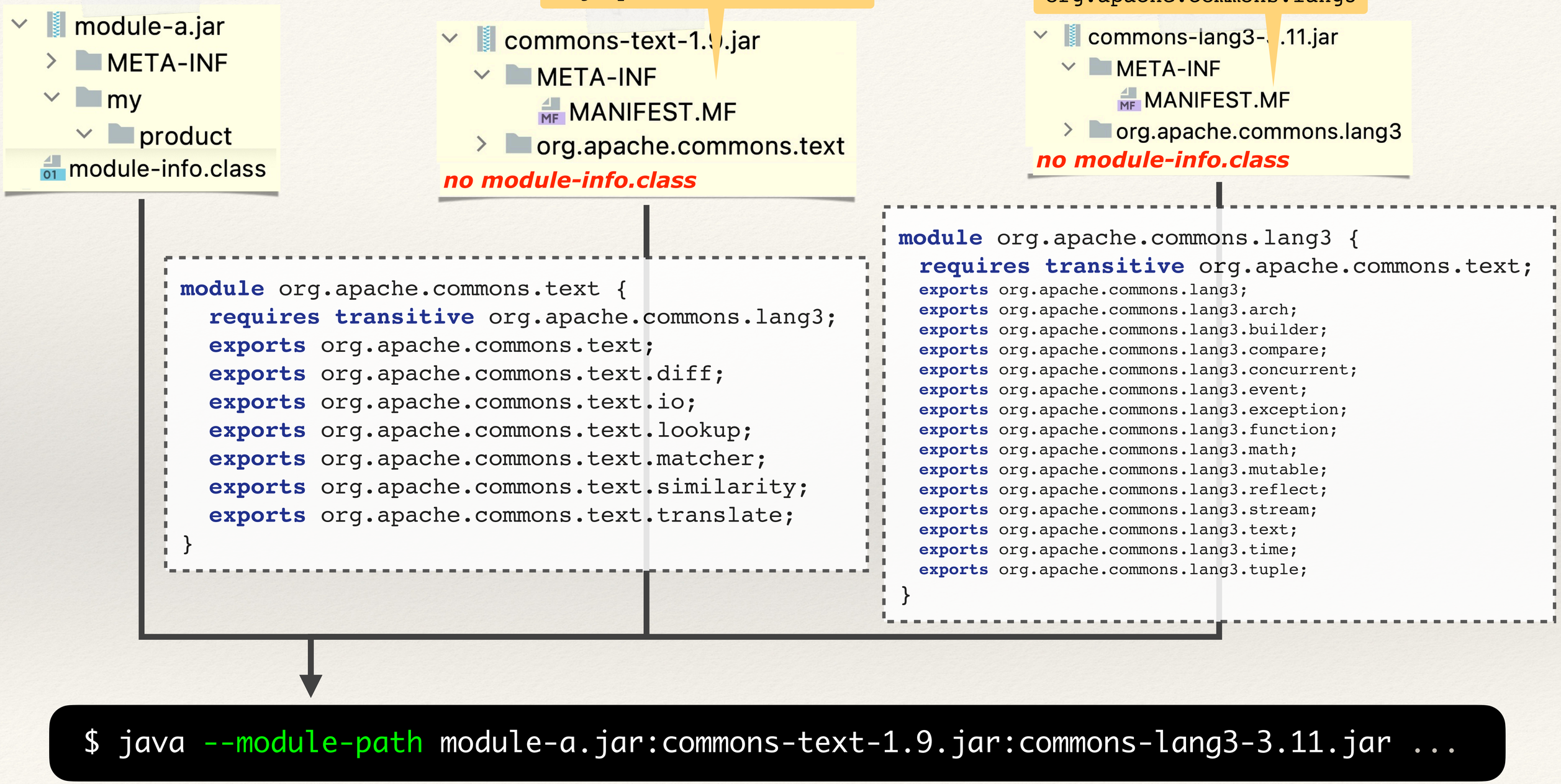


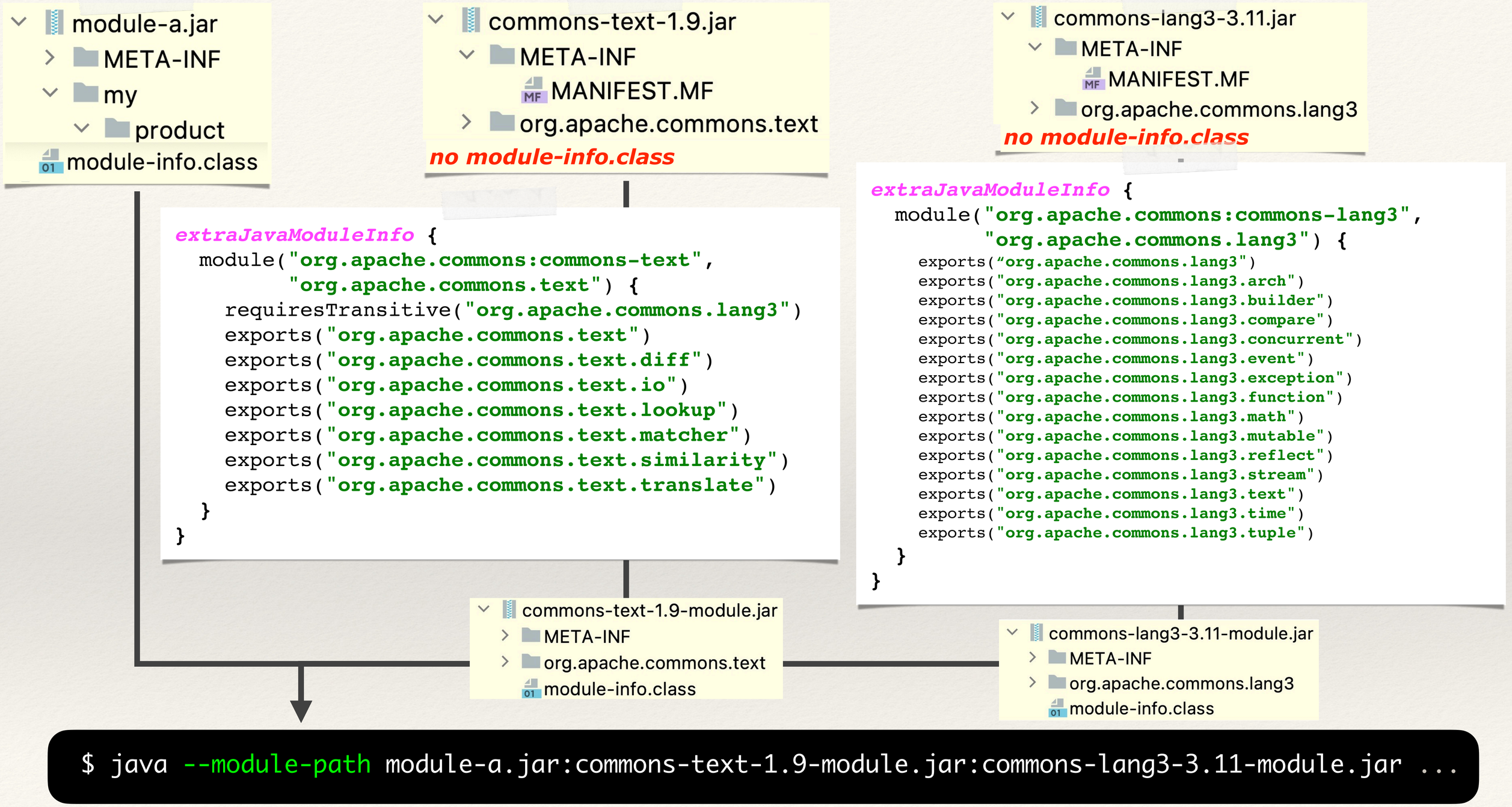

```
$ java --module-path module-a.jar:commons-text-1.9.jar ...
```

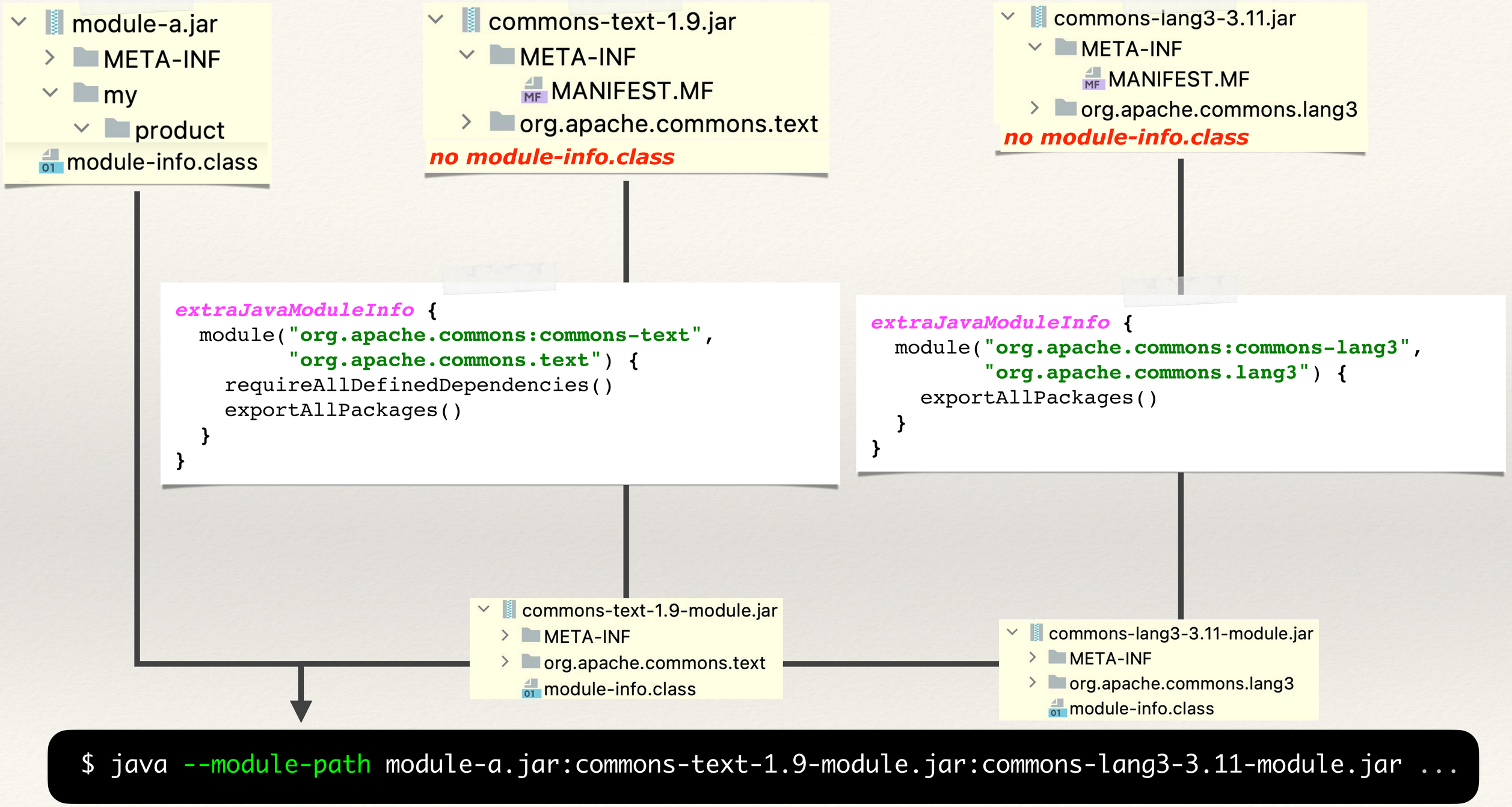

Automatic-Module-Name:
org.apache.commons.text
module-a.jar
META-INF
my
product
module-info.class
commons-text-1.9.jar
META-INF
MANIFEST.MF
org.apache.commons.text
no module-info.class
$ java --module-path module-a.jar:commons-text-1.9.jar ...


Automatic-Module-Name:
org.apache.commons.text
module-a.jar
META-INF
my
product
module-info.class
commons-text-1.9.jar
META-INF
MANIFEST.MF
org.apache.commons.text
no module-info.class
module org.apache.commons.text {
// ??? requires ???
exports org.apache.commons.text;
exports org.apache.commons.text.diff;
exports org.apache.commons.text.io;
exports org.apache.commons.text.lookup;
exports org.apache.commons.text.matcher;
exports org.apache.commons.text.similarity;
exports org.apache.commons.text.translate;
}
$ java --module-path module-a.jar:commons-text-1.9.jar ...


Automatic-Module-Name:
org.apache.commons.text
Automatic-Module-Name:
org.apache.commons.lang3
module-a.jar
META-INF
my
product
module-info.class
commons-text-1.9.jar
META-INF
MANIFEST.MF
org.apache.commons.text
no module-info.class
commons-lang3-3.11.jar
META-INF
MANIFEST.MF
org.apache.commons.lang3
no module-info.class
module org.apache.commons.text {
// ??? requires ???
exports org.apache.commons.text;
exports org.apache.commons.text.diff;
exports org.apache.commons.text.io;
exports org.apache.commons.text.lookup;
exports org.apache.commons.text.matcher;
exports org.apache.commons.text.similarity;
exports org.apache.commons.text.translate;
}
$ java --module-path module-a.jar:commons-text-1.9.jar ...

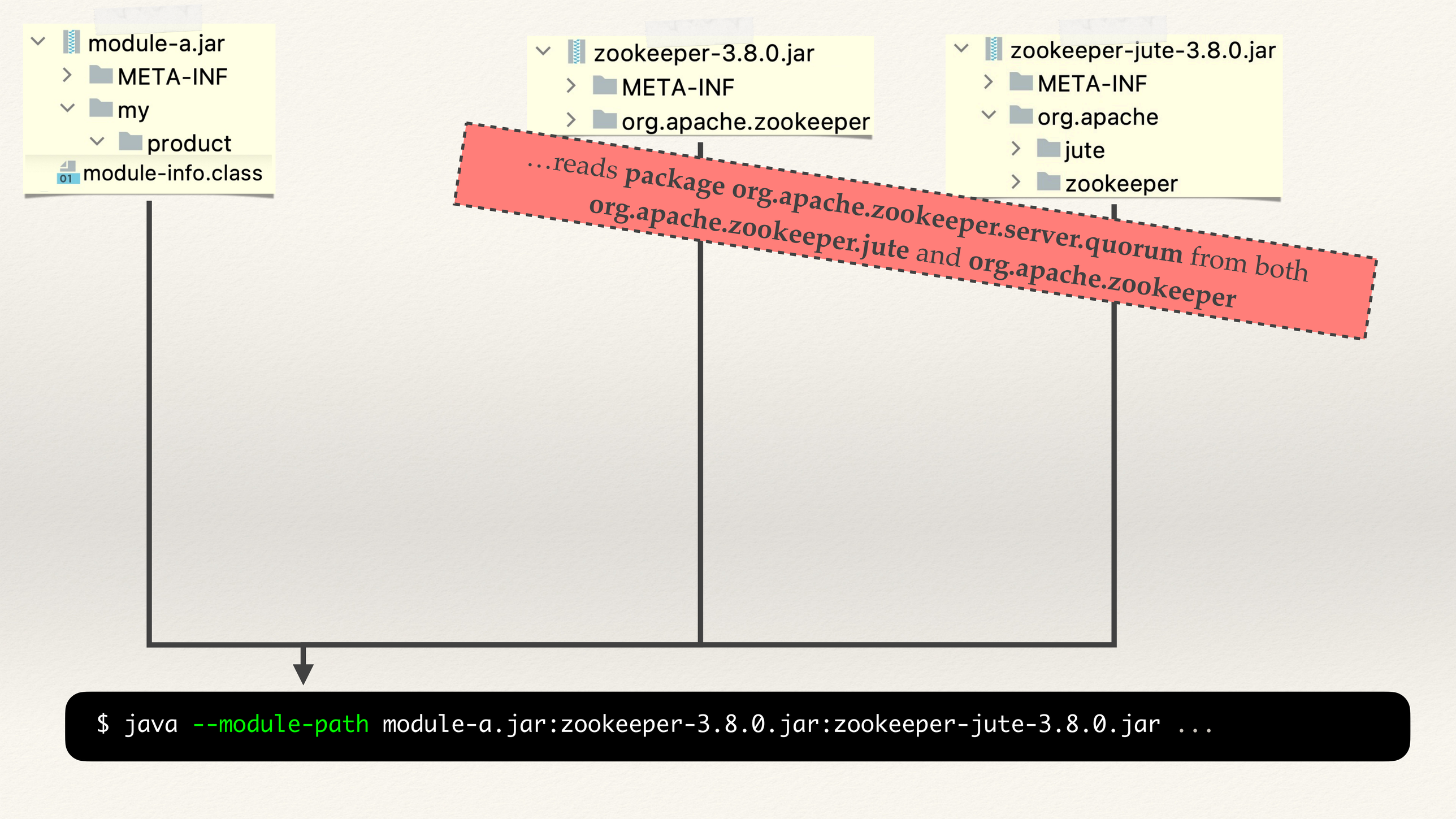

module-a.jar
META-INF
my
product
module-info.class
Automatic-Module-Name:
org.apache.commons.text
commons-text-1.9.jar
META-INF
MANIFEST.MF
org.apache.commons.text
no module-info.class
Automatic-Module-Name:
org.apache.commons.lang3
commons-lang3-3.11.jar
META-INF
MANIFEST.MF
org.apache.commons.lang3
no module-info.class
module org.apache.commons.text {
requires transitive org.apache.commons.lang3;
exports org.apache.commons.text;
exports org.apache.commons.text.diff;
exports org.apache.commons.text.io;
exports org.apache.commons.text.lookup;
exports org.apache.commons.text.matcher;
exports org.apache.commons.text.similarity;
exports org.apache.commons.text.translate;
}
module org.apache.commons.lang3 {
requires transitive org.apache.commons.text;
exports org.apache.commons.lang3;
exports org.apache.commons.lang3.arch;
exports org.apache.commons.lang3.builder;
exports org.apache.commons.lang3.compare;
exports org.apache.commons.lang3.concurrent;
exports org.apache.commons.lang3.event;
exports org.apache.commons.lang3.exception;
exports org.apache.commons.lang3.function;
exports org.apache.commons.lang3.math;
exports org.apache.commons.lang3.mutable;
exports org.apache.commons.lang3.reflect;
exports org.apache.commons.lang3.stream;
exports org.apache.commons.lang3.text;
exports org.apache.commons.lang3.time;
exports org.apache.commons.lang3.tuple;
}
$ java --module-path module-a.jar:commons-text-1.9.jar:commons-lang3-3.11.jar ...

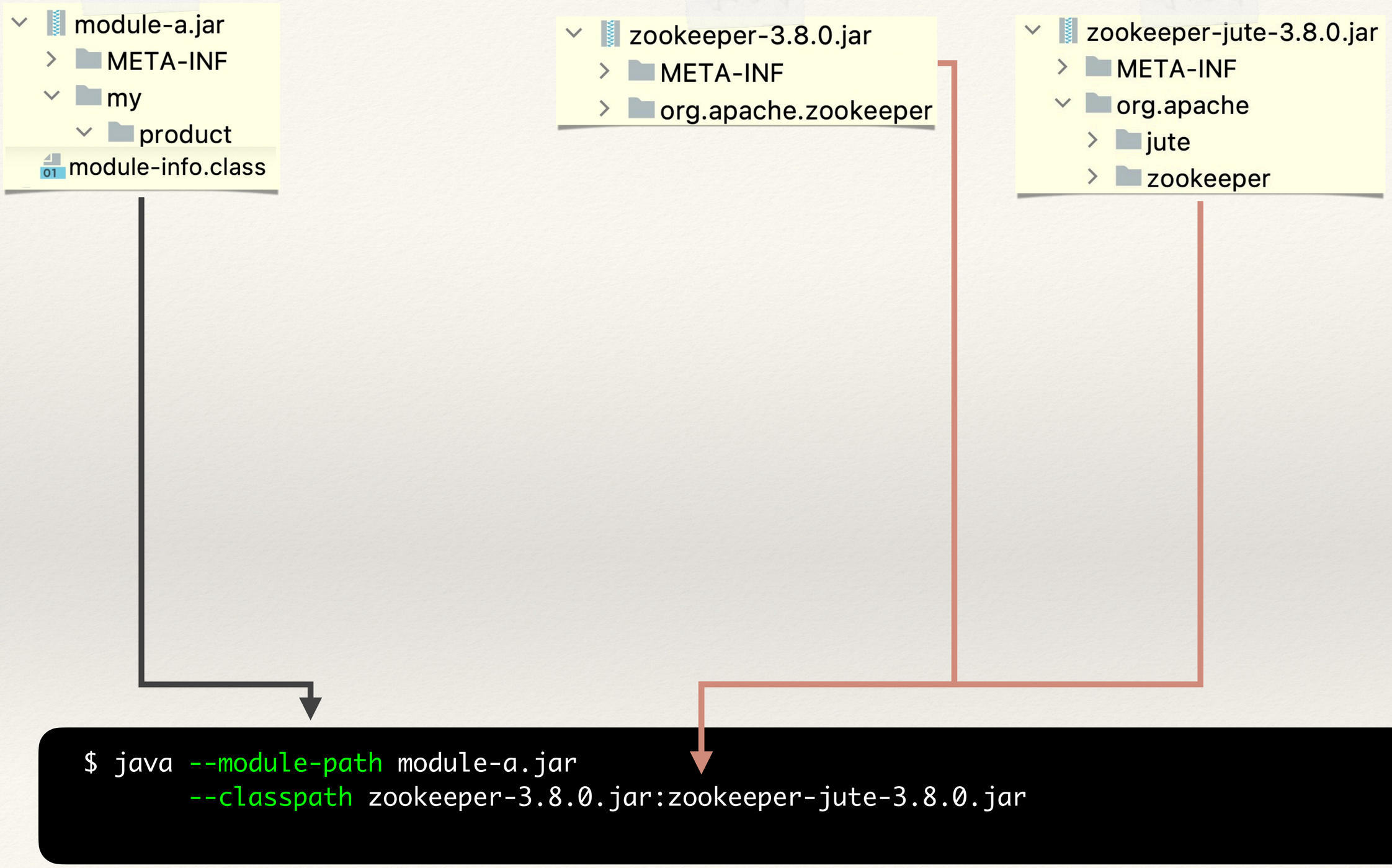

module-a.jar
META-INF
my
product
module-info.class
commons-text-1.9.jar
META-INF
MANIFEST.MF
org.apache.commons.text
no module-info.class
commons-lang3-3.11.jar
META-INF
MANIFEST.MF
org.apache.commons.lang3
no module-info.class
extraJavaModuleInfo {
module("org.apache.commons:commons-text",
"org.apache.commons.text") {
requiresTransitive("org.apache.commons.lang3")
exports("org.apache.commons.text")
exports("org.apache.commons.text.diff")
exports("org.apache.commons.text.io")
exports("org.apache.commons.text.lookup")
exports("org.apache.commons.text.matcher")
exports("org.apache.commons.text.similarity")
exports("org.apache.commons.text.translate")
}
}
extraJavaModuleInfo {
module("org.apache.commons:commons-lang3",
"org.apache.commons.lang3") {
exports("org.apache.commons.lang3")
exports("org.apache.commons.lang3.arch")
exports("org.apache.commons.lang3.builder")
exports("org.apache.commons.lang3.compare")
exports("org.apache.commons.lang3.concurrent")
exports("org.apache.commons.lang3.event")
exports("org.apache.commons.lang3.exception")
exports("org.apache.commons.lang3.function")
exports("org.apache.commons.lang3.math")
exports("org.apache.commons.lang3.mutable")
exports("org.apache.commons.lang3.reflect")
exports("org.apache.commons.lang3.stream")
exports("org.apache.commons.lang3.text")
exports("org.apache.commons.lang3.time")
exports("org.apache.commons.lang3.tuple")
}
}
commons-text-1.9-module.jar
META-INF
org.apache.commons.text
module-info.class
commons-lang3-3.11-module.jar
META-INF
org.apache.commons.lang3
module-info.class
$ java --module-path module-a.jar:commons-text-1.9-module.jar:commons-lang3-3.11-module.jar ...

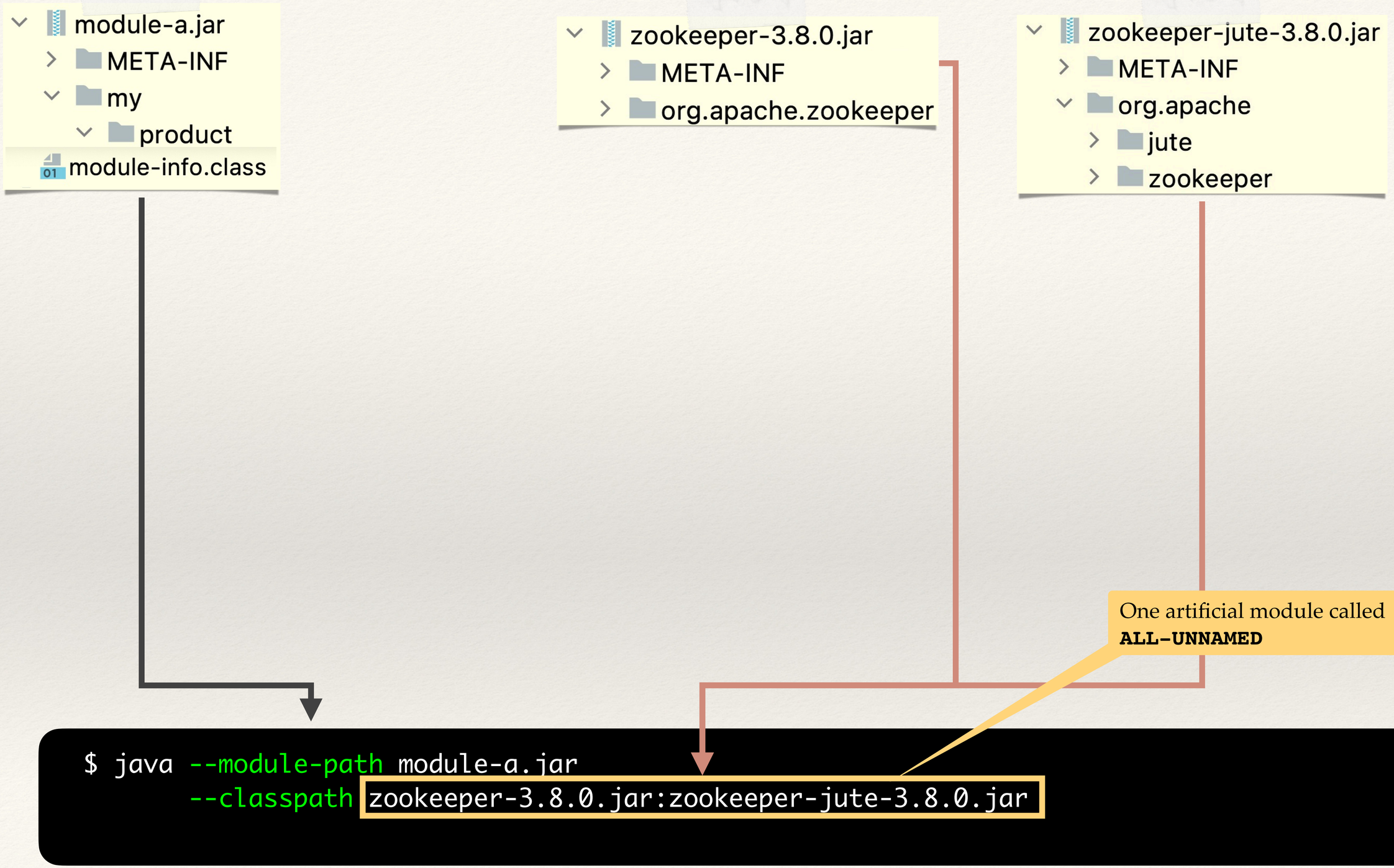

module-a.jar
META-INF
my
product
module-info.class
commons-text-1.9.jar
META-INF
MANIFEST.MF
org.apache.commons.text
no module-info.class
commons-lang3-3.11.jar
META-INF
MANIFEST.MF
org.apache.commons.lang3
no module-info.class
extraJavaModuleInfo {
module("org.apache.commons:commons-text",
"org.apache.commons.text") {
requireAllDefinedDependencies()
exportAllPackages()
}
}
extraJavaModuleInfo {
module("org.apache.commons:commons-lang3",
"org.apache.commons.lang3") {
exportAllPackages()
}
}
commons-text-1.9-module.jar
META-INF
org.apache.commons.text
module-info.class
commons-lang3-3.11-module.jar
META-INF
org.apache.commons.lang3
module-info.class
$ java --module-path module-a.jar:commons-text-1.9-module.jar:commons-lang3-3.11-module.jar ...

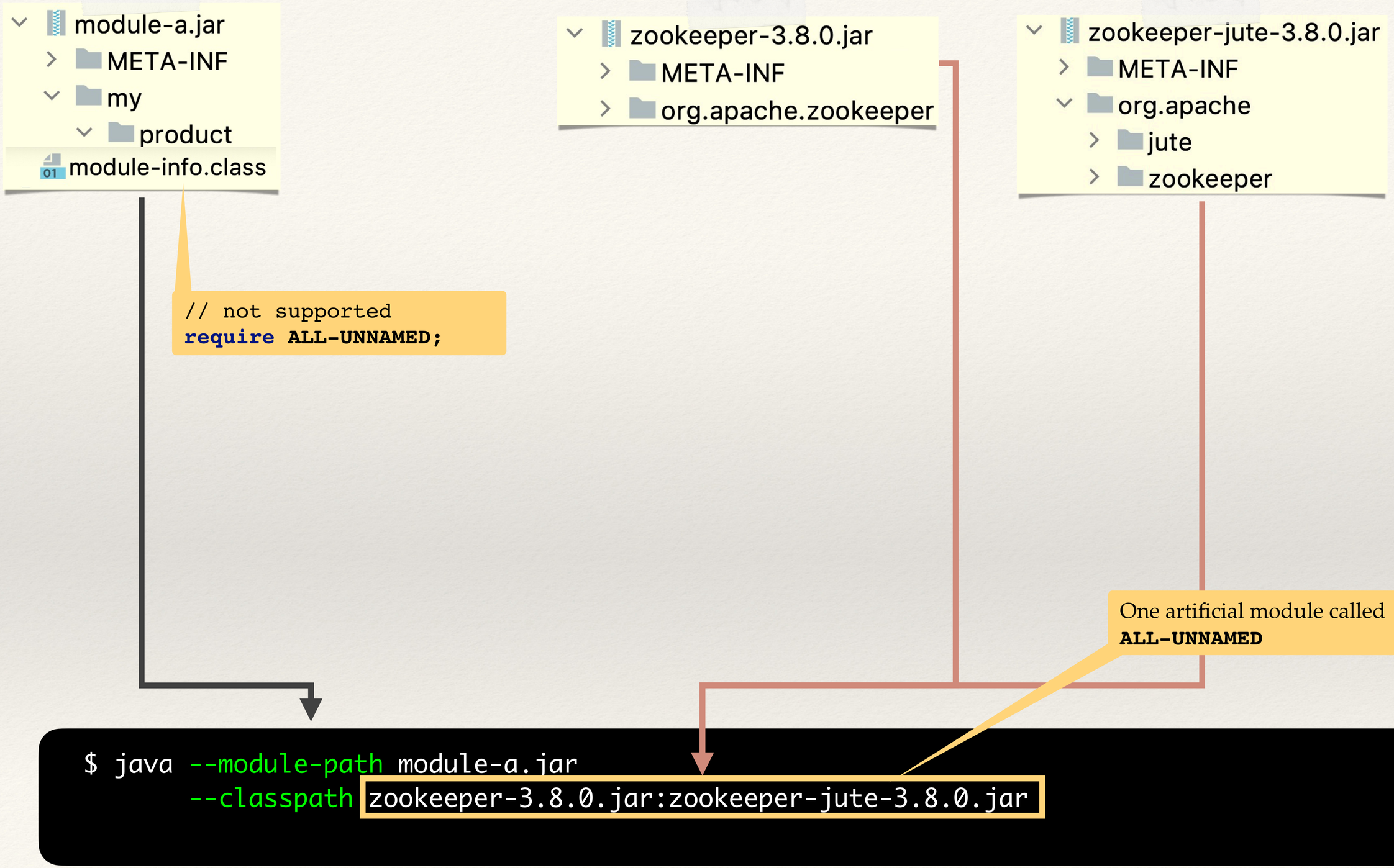


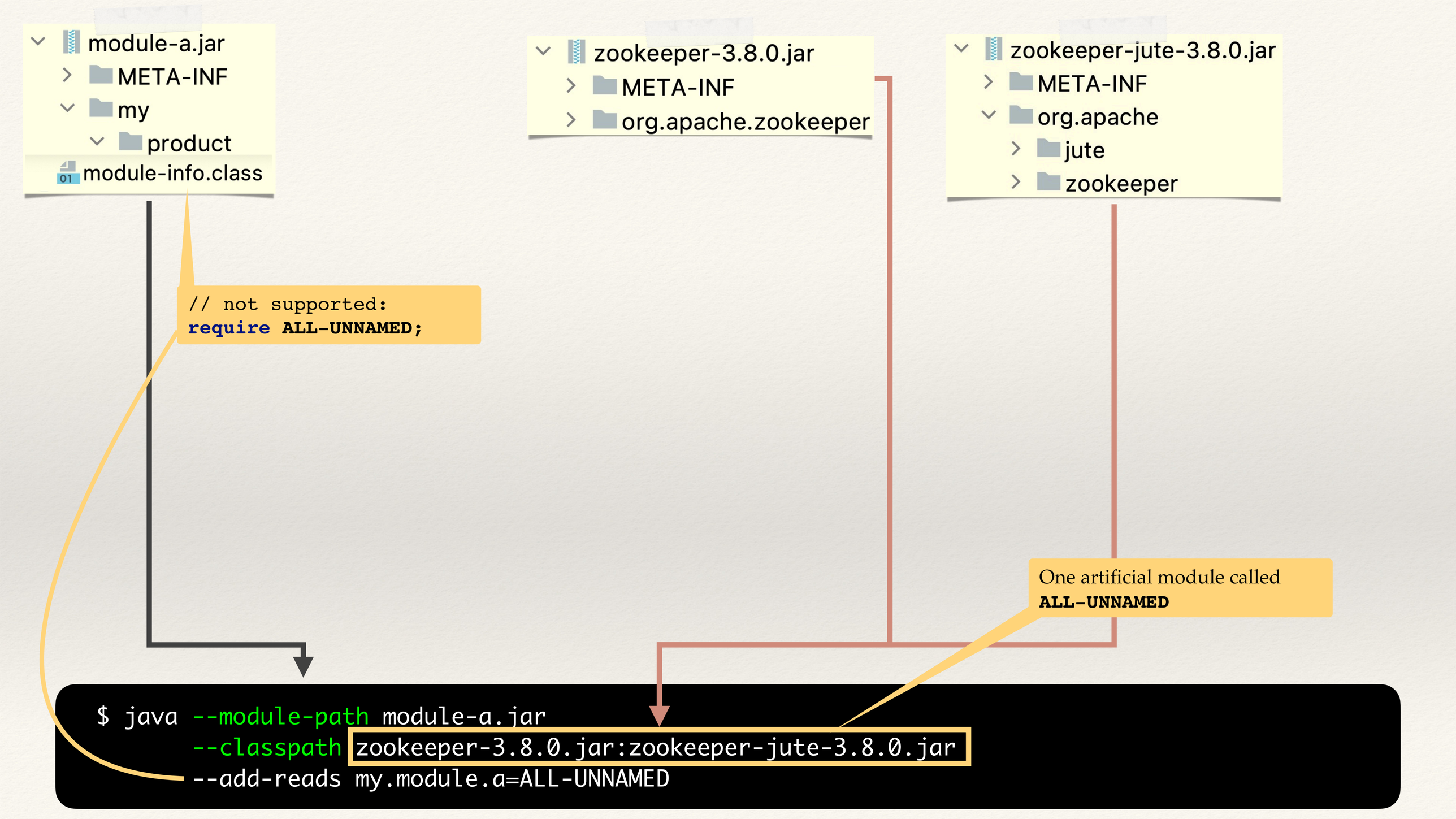

```
$ java --module-path module-a.jar:zookeeper-3.8.0.jar:zookeeper-jute-3.8.0.jar ...
```


module-a.jar
META-INF
my
product
module-info.class
zookeeper-3.8.0.jar
META-INF
org.apache.zookeeper
zookeeper-jute-3.8.0.jar
META-INF
org.apache
jute
zookeeper
…reads package org.apache.zookeeper.server.quorum from both org.apache.zookeeper.jute and org.apache.zookeeper
$ java --module-path module-a.jar:zookeeper-3.8.0.jar:zookeeper-jute-3.8.0.jar ...


module-a.jar
META-INF
my
product
module-info.class
zookeeper-3.8.0.jar
META-INF
org.apache.zookeeper
zookeeper-jute-3.8.0.jar
META-INF
org.apache
jute
zookeeper
$ java --module-path module-a.jar
--classpath zookeeper-3.8.0.jar:zookeeper-jute-3.8.0.jar

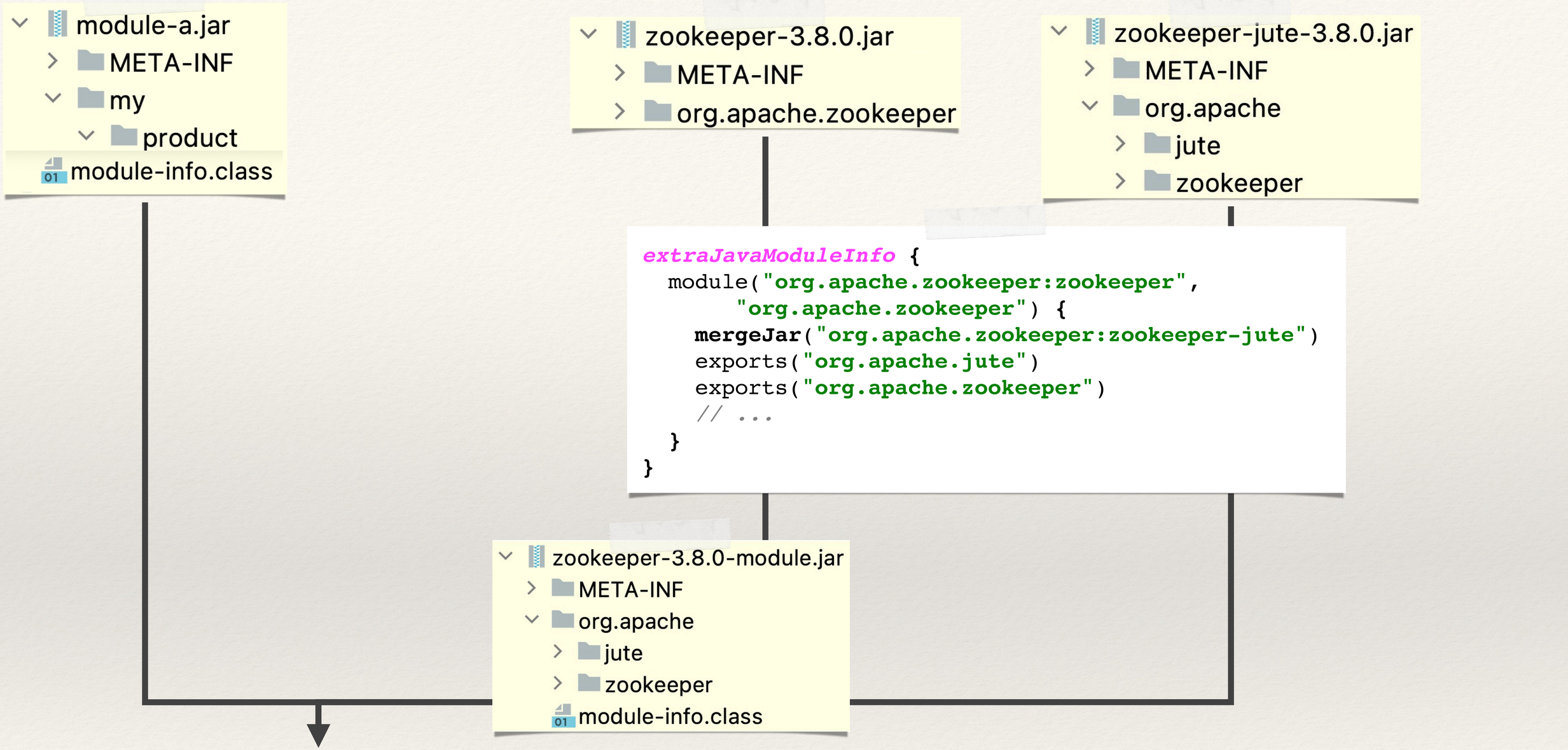

module-a.jar
META-INF
my
product
module-info.class
zookeeper-3.8.0.jar
META-INF
org.apache.zookeeper
zookeeper-jute-3.8.0.jar
META-INF
org.apache
jute
zookeeper
One artificial module called
ALL-UNNAMED
$ java --module-path module-a.jar
--classpath zookeeper-3.8.0.jar:zookeeper-jute-3.8.0.jar

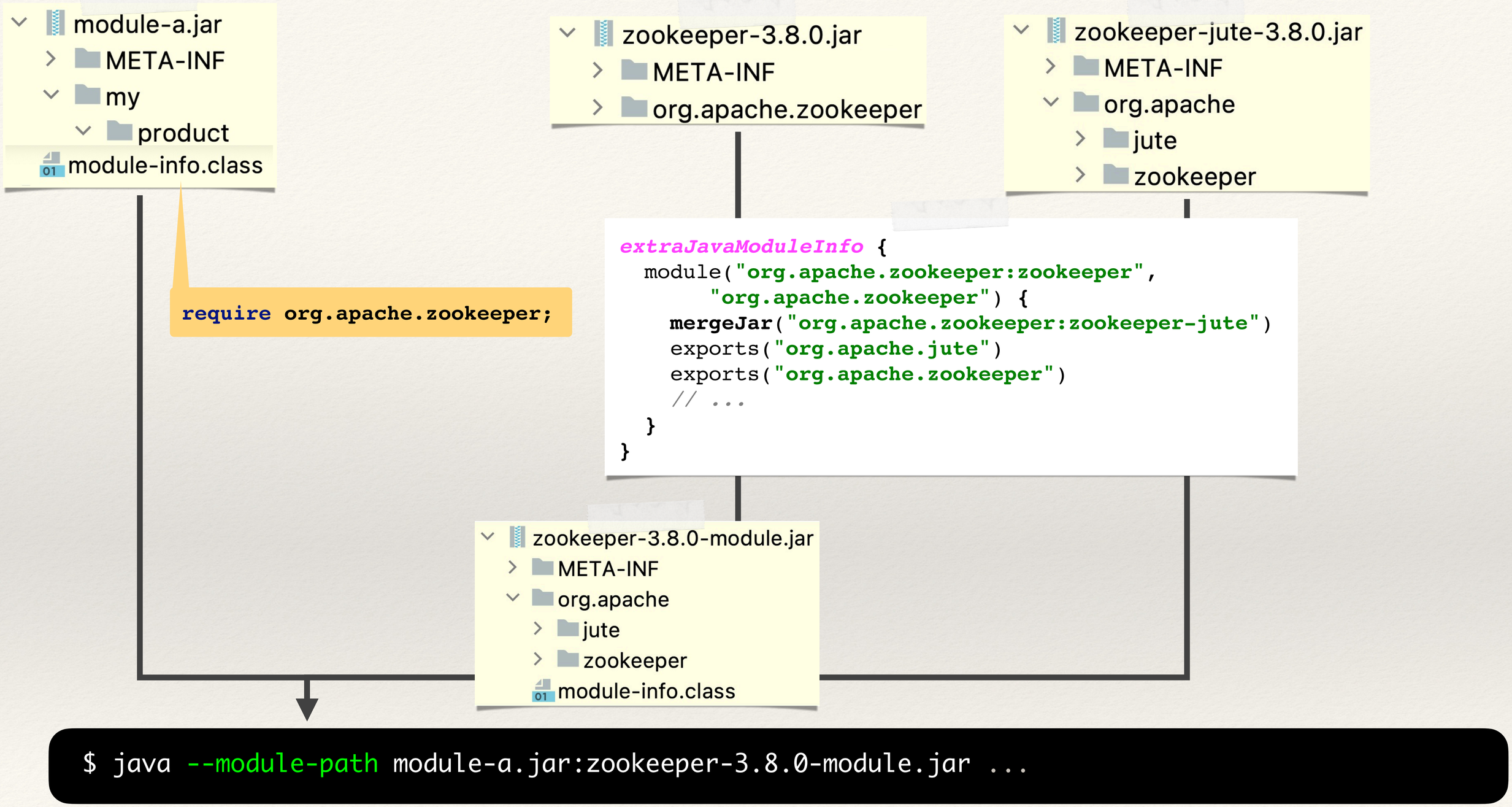

module-a.jar
META-INF
my
product
module-info.class
zookeeper-3.8.0.jar
META-INF
org.apache.zookeeper
zookeeper-jute-3.8.0.jar
META-INF
org.apache
jute
zookeeper
// not supported
require ALL-UNNAMED;
One artificial module called
ALL-UNNAMED
$ java --module-path module-a.jar
--classpath zookeeper-3.8.0.jar:zookeeper-jute-3.8.0.jar


module-a.jar
META-INF
my
product
module-info.class
zookeeper-3.8.0.jar
META-INF
org.apache.zookeeper
zookeeper-jute-3.8.0.jar
META-INF
org.apache
jute
zookeeper
// not supported:
require ALL-UNNAMED;
One artificial module called
ALL-UNNAMED
$ java --module-path module-a.jar
--classpath zookeeper-3.8.0.jar:zookeeper-jute-3.8.0.jar
--add-reads my.module.a=ALL-UNNAMED



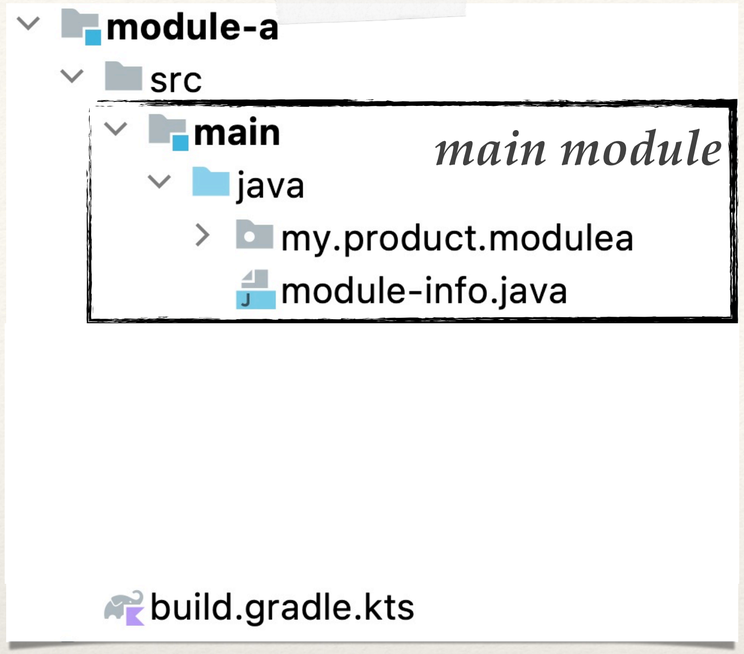

```
$ java --module-path module-a.jar:zookeeper-3.8.0-module.jar ...
```


module-a.jar
META-INF
my
product
module-info.class
zookeeper-3.8.0.jar
META-INF
org.apache.zookeeper
zookeeper-jute-3.8.0.jar
META-INF
org.apache
jute
zookeeper
extraJavaModuleInfo {
module("org.apache.zookeeper:zookeeper",
"org.apache.zookeeper") {
mergeJar("org.apache.zookeeper:zookeeper-jute")
exports("org.apache.jute")
exports("org.apache.zookeeper")
// ...
}
}
zookeeper-3.8.0-module.jar
META-INF
org.apache
jute
zookeeper
module-info.class
$ java --module-path module-a.jar:zookeeper-3.8.0-module.jar ...


module-a.jar
META-INF
my
product
module-info.class
zookeeper-3.8.0.jar
META-INF
org.apache.zookeeper
zookeeper-jute-3.8.0.jar
META-INF
org.apache
jute
zookeeper
require org.apache.zookeeper;
extraJavaModuleInfo {
module("org.apache.zookeeper:zookeeper",
"org.apache.zookeeper") {
mergeJar("org.apache.zookeeper:zookeeper-jute")
exports("org.apache.jute")
exports("org.apache.zookeeper")
// ...
}
}
zookeeper-3.8.0-module.jar
META-INF
org.apache
jute
zookeeper
module-info.class
$ java --module-path module-a.jar:zookeeper-3.8.0-module.jar ...

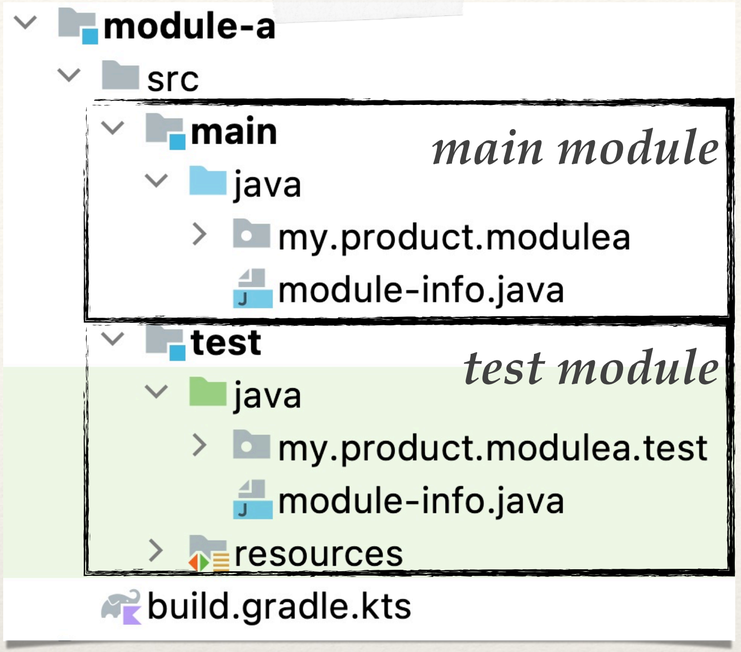
	Module Path w/ plain Gradle
discover modules in repos	✅
module version management	✅
conflict detection	✅✅
(automatic) conflict resolution	✅
integrate non-module Jars	⚠️ - org.gradlex.extra-java-module-info
scopes at compile time	⭐✅⭐ - org.gradlex.java-module-dependencies
private packages	⭐✅⭐
testing of private units	✅ - org.gradlex.java-module-testing
JDK tools (e.g. jlink)	✅
visibility at runtime	✅

	Module Path w/ plain Gradle
discover modules in repos	✅
module version management	✅
conflict detection	✅✅
(automatic) conflict resolution	✅
integrate non-module Jars	⚠️ - org.gradlex.extra-java-module-info
scopes at compile time	⭐✅⭐ - org.gradlex.java-module-dependencies
private packages	⭐✅⭐
testing of private units	✅ - org.gradlex.java-module-testing
JDK tools (e.g. jlink)	✅
visibility at runtime	✅



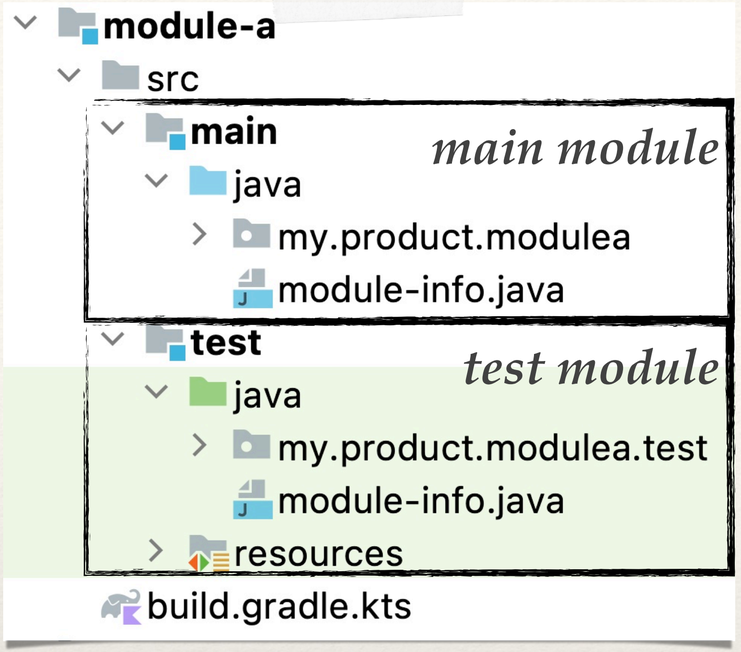
backbox / modular testing

backbox / modular testing




```
open module my.product.module.a.test {
    requires my.product.module.a;
    requires org.junit.jupiter.api;
}
```

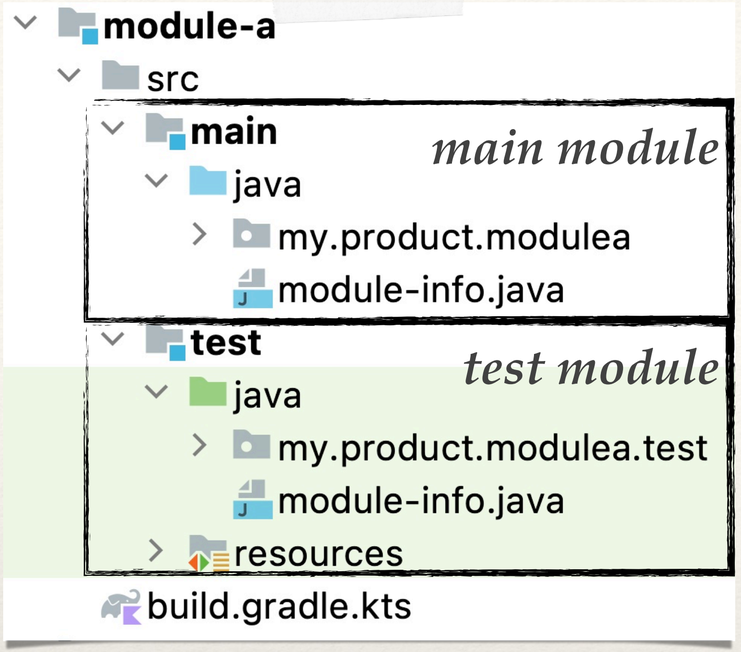
backbox / modular testing




```
open module my.product.module.a.test {
    requires my.product.module.a;
    requires org.junit.jupiter.api;
}
```

```
$ java --module-path build/libs/module-a.jar:build/classes/java/test:...
       -classpath build/resources/test
```

backbox / modular testing



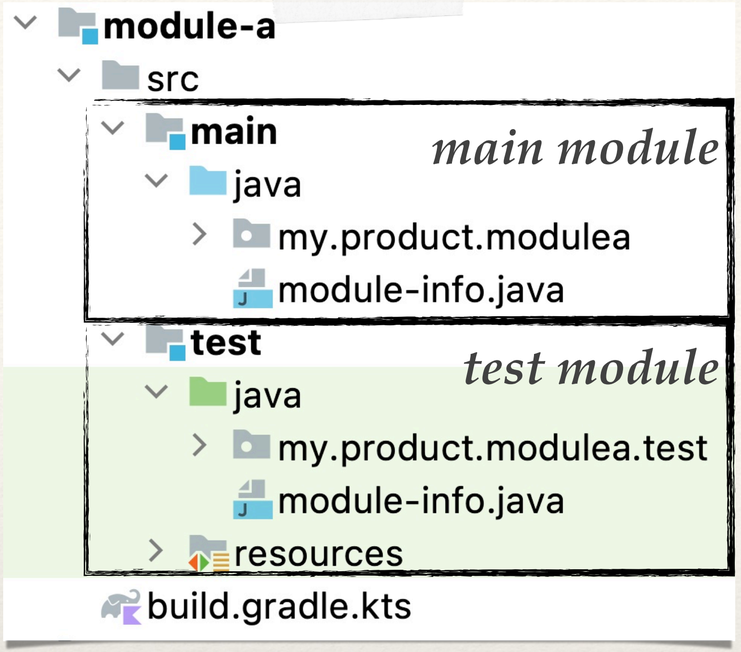

```
open module my.product.module.a.test {
    requires my.product.module.a;
    requires org.junit.jupiter.api;
}
```

```
$ java --module-path build/libs/module-a.jar:build/classes/java/test:...
       -classpath build/resources/test
```

```
plugins {
  id("org.gradlex.java-module-testing")
}

javaModuleTesting.blackbox(testing.suites["test"])
```

backbox / modular testing



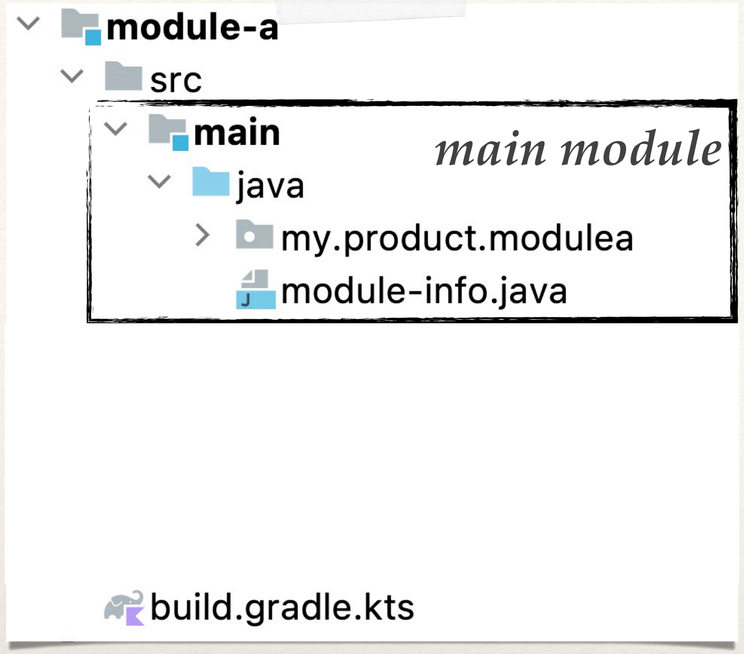

```
open module my.product.module.a.test {
    requires my.product.module.a;
    requires org.junit.jupiter.api;
}
```

```
$ java --module-path build/libs/module-a.jar:build/classes/java/test:...
       -classpath build/resources/test
```

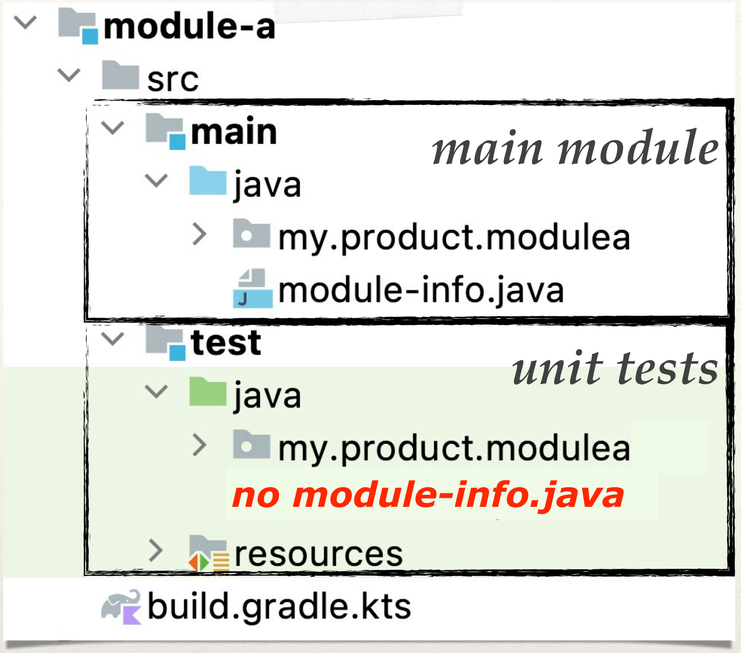
```
plugins {
  id("org.gradlex.java-module-testing")
}

javaModuleTesting.blackbox(testing.suites["test"])
```

```
$ java --module-path build/libs/module-a.jar:build/libs/build/module-a-test.jar:...
```

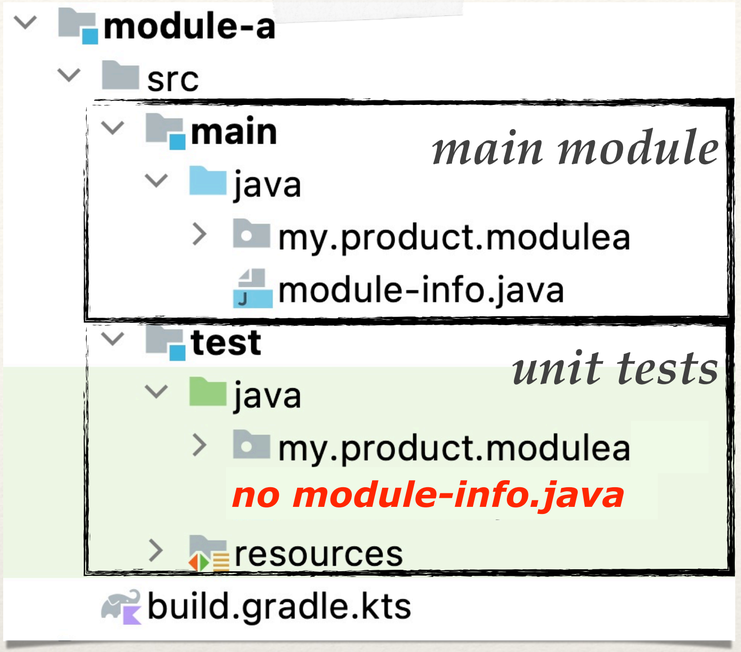


whitebox / in-module testing



whitebox / in-module testing

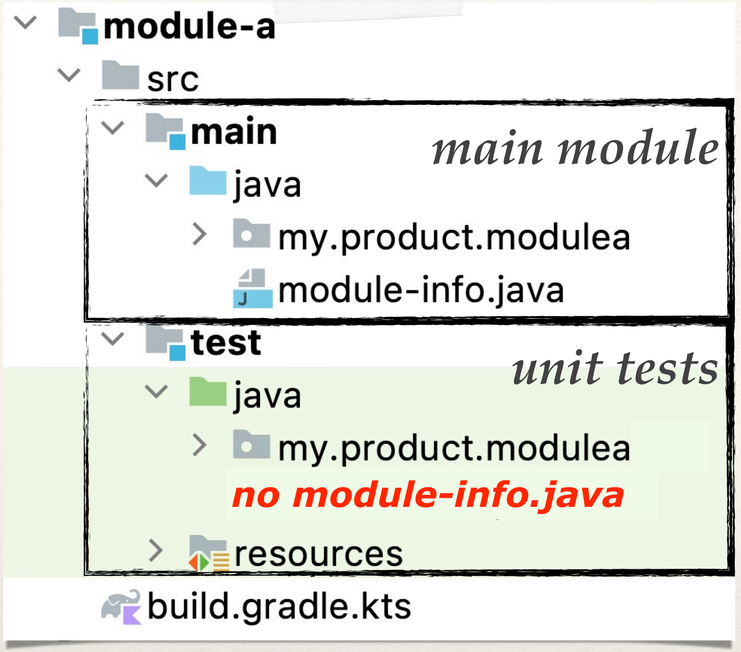
whitebox / in-module testing




```
plugins {
  id("org.gradlex.java-module-testing")
}

javaModuleTesting.whitebox(testing.suites["test"]) {
    requires.add("org.junit.jupiter.api")
    opensTo.add("org.junit.platform.commons")
}
```

whitebox / in-module testing




```
plugins {
  id("org.gradlex.java-module-testing")
}

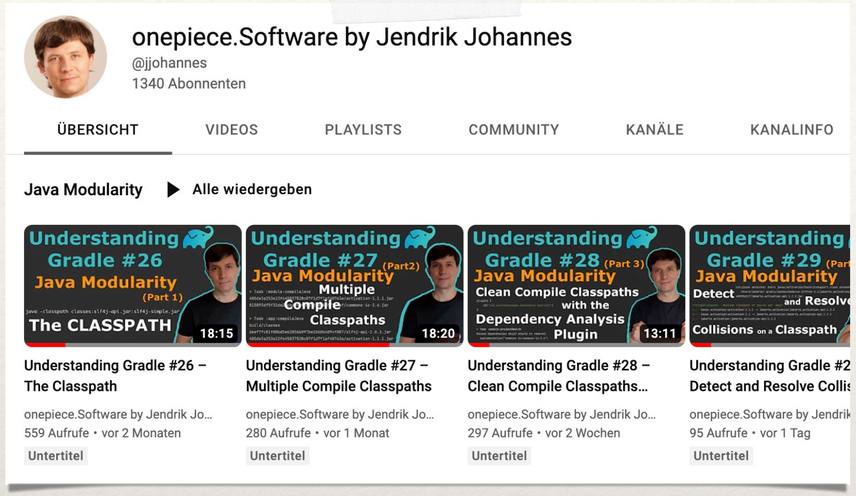
javaModuleTesting.whitebox(testing.suites["test"]) {
    requires.add("org.junit.jupiter.api")
    opensTo.add("org.junit.platform.commons")
}
```

```
$ java –module-path ...
       --add-reads my.product.module.a=org.junit.jupiter.api
       --add-opens my.product.module.a/my.product.modulea.test=org.junit.platform.commons
       --patch-module my.product.module.a=build/classes/java/test:build/resources/main:build/resources/test
```

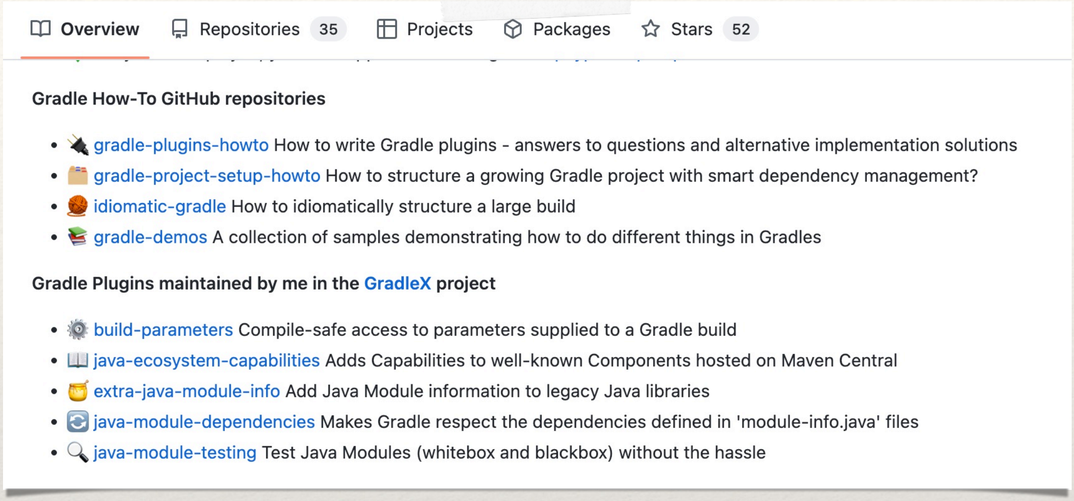
	Module Path w/ plain Gradle
discover modules in repos	✅
module version management	✅
conflict detection	✅✅
(automatic) conflict resolution	✅
integrate non-module Jars	⚠️ - org.gradlex.extra-java-module-info
scopes at compile time	⭐✅⭐ - org.gradlex.java-module-dependencies
private packages	⭐✅⭐
testing of private units	✅ - org.gradlex.java-module-testing
JDK tools (e.g. jlink)	✅
visibility at runtime	✅

	Classpath w/ Gradle	*Module Path w/ Gradle+X*
discover modules in repos	✅	✅
module version management	✅	✅
conflict detection	✅	✅✅
(automatic) conflict resolution	✅	✅
integrate non-module Jars	✅	⚠️
scopes at compile time	✅	✅⭐
private packages	❌	✅
testing of private units	✅	✅⚠️
JDK tools (e.g. jlink)	❌	✅
visibility at runtime	❌	✅

Thank You!



youtube.com/@jjohannes



github.com/jjohannes

get updates
mastodon.social/@jendrik

training and consulting
onepiece.software