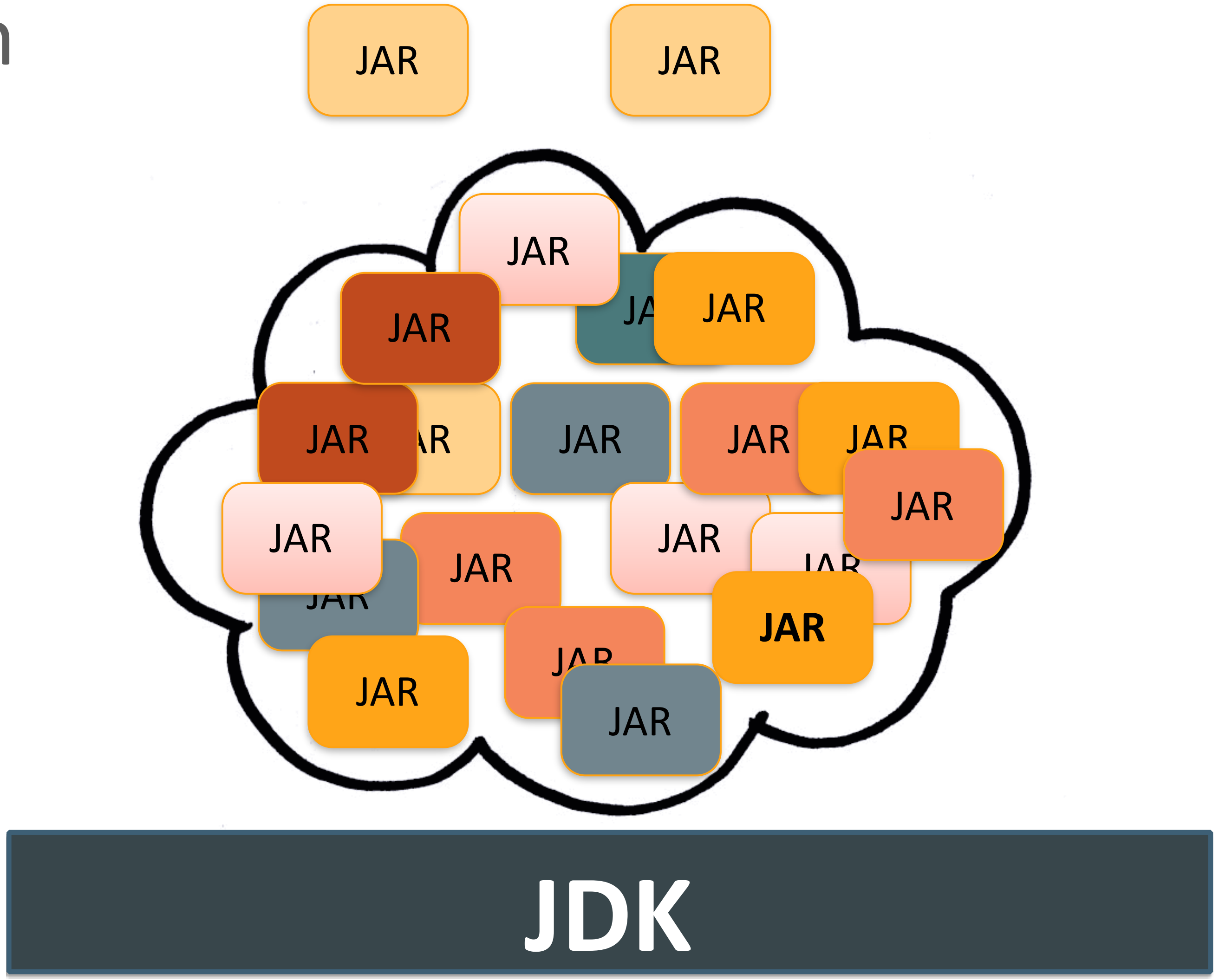
Advanced Modular Development

Alan Bateman
Alex Buckley
Java Platform Group, Oracle
September 2016

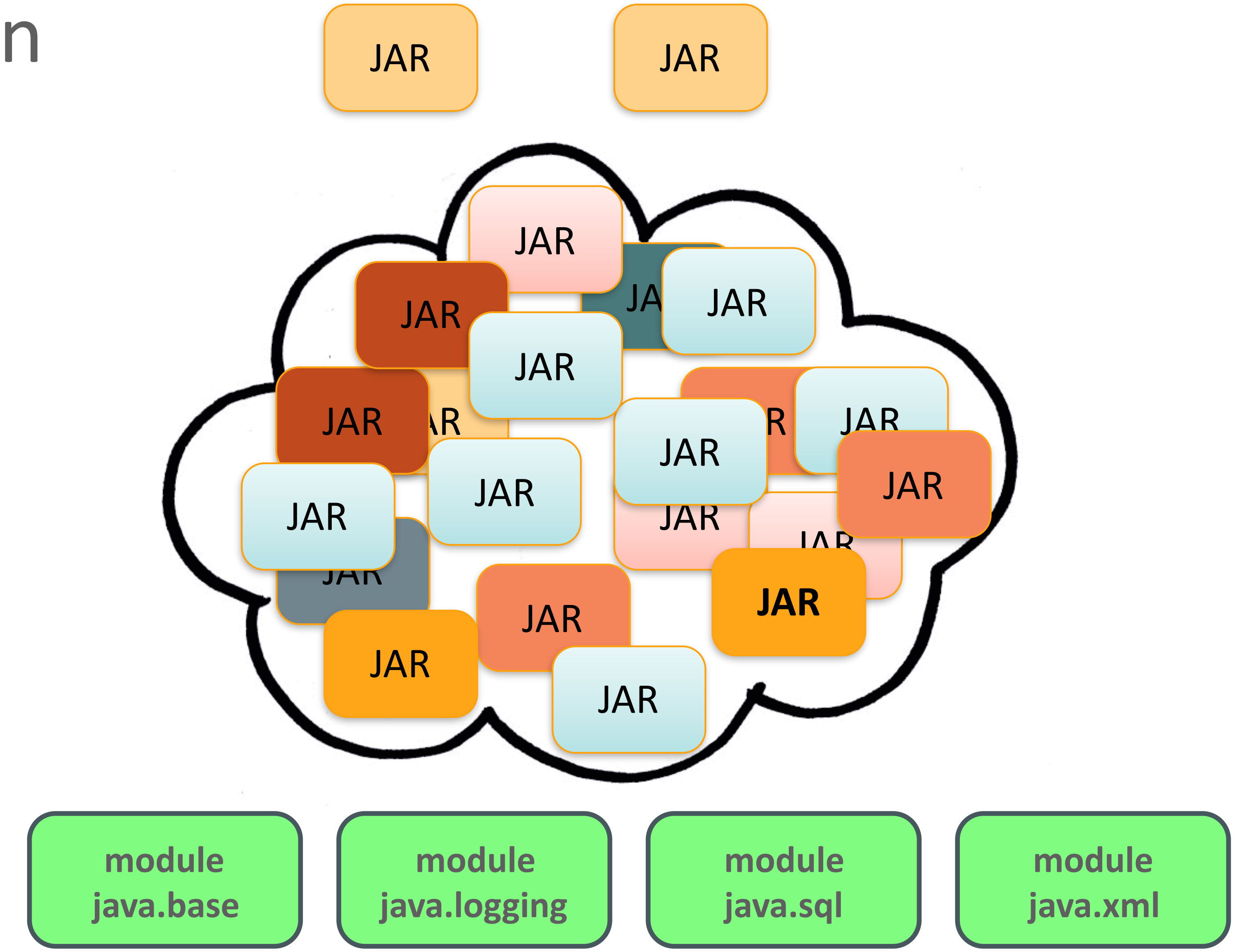
Sessions

Application Migration

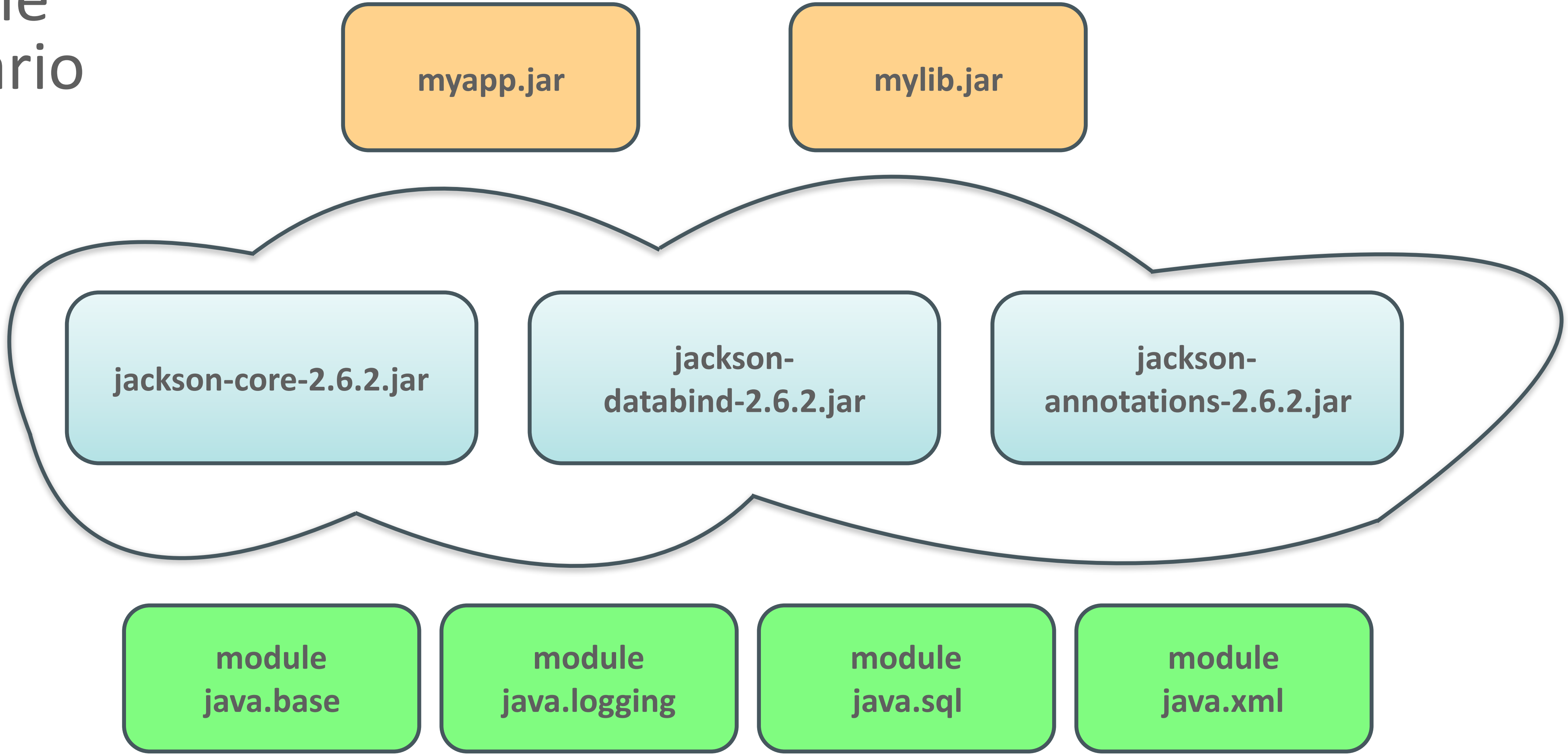
Typical application



Typical application



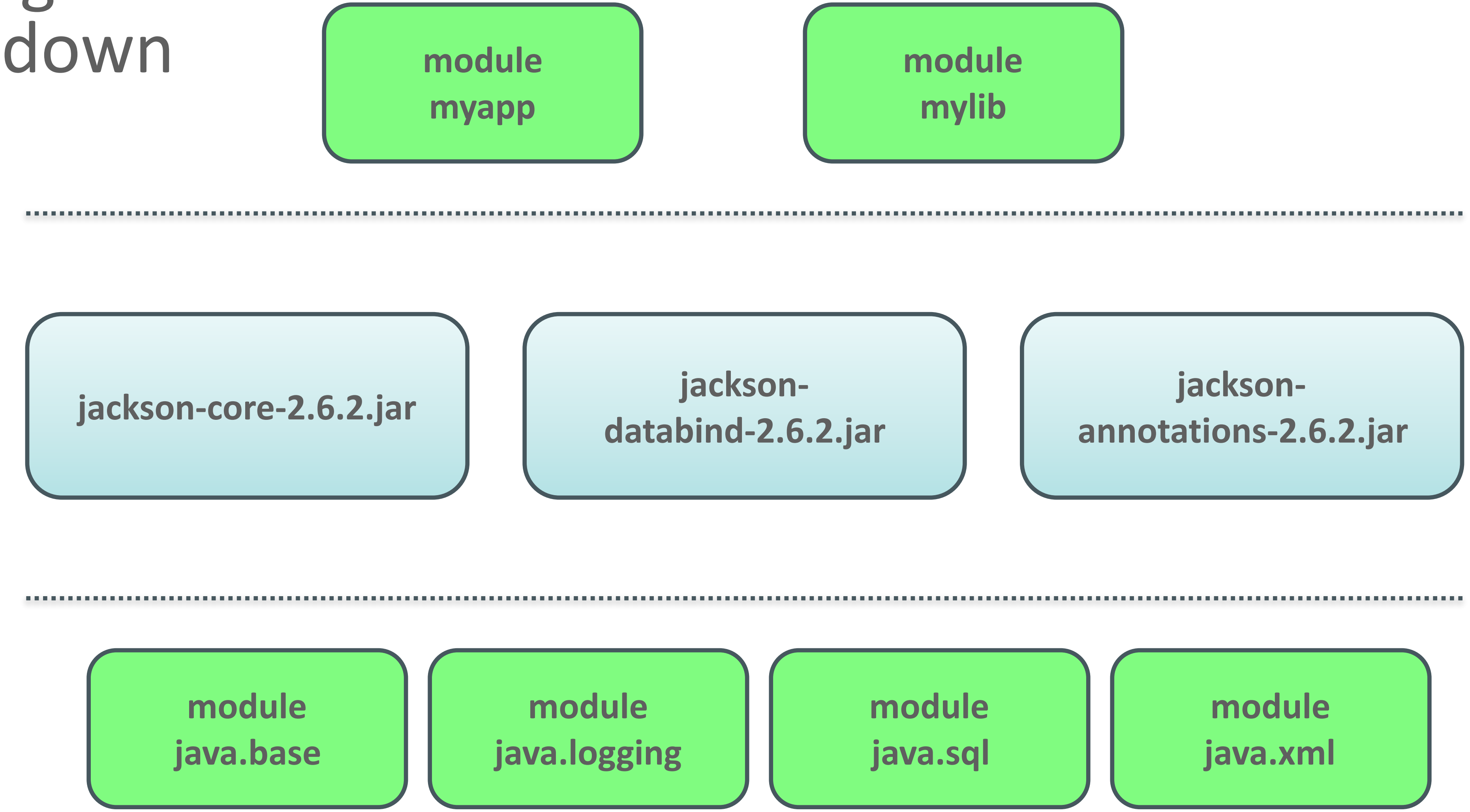
Sample Scenario



Running my application

```
$ java -cp \
     lib/myapp.jar:\
     lib/mylib.jar:\
     lib/jackson-core-2.6.2.jar:\
     lib/jackson-databind-2.6.2.jar:\
     lib/jackson-annotations-2.6.2.jar\
  myapp.Main
```

Migrating from the top down



Migrating from the top down

- module myapp requires?
- module myapp exports?

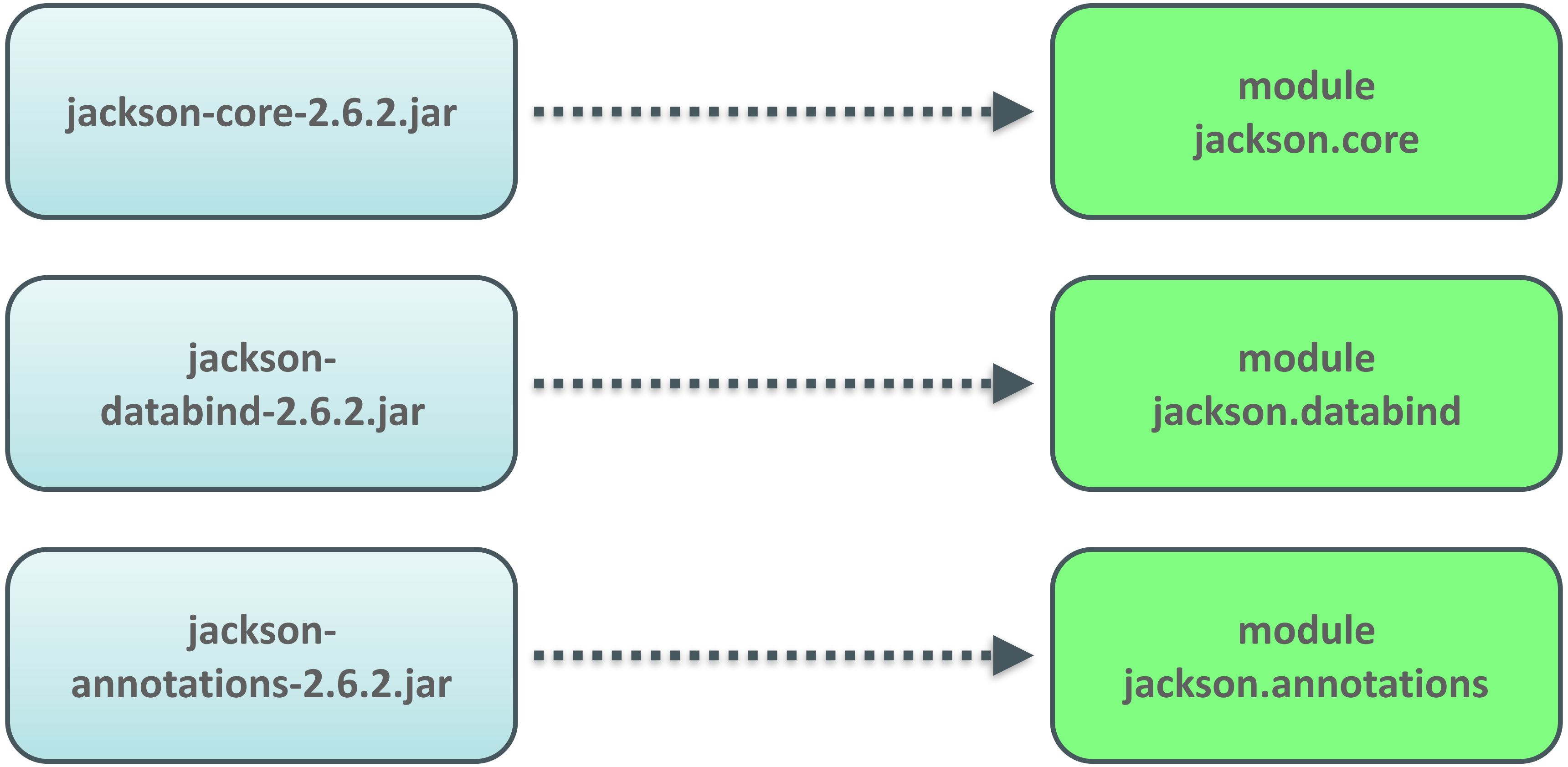
- module mylib requires?
- module mylib exports?

```
$ jdeps -s lib/myapp.jar lib/mylib.jar
myapp.jar -> lib/jackson-core-2.6.2.jar
myapp.jar -> lib/jackson-databind-2.6.2.jar
myapp.jar -> mylib.jar
myapp.jar -> java.base
myapp.jar -> java.sql
mylib.jar -> java.base
```

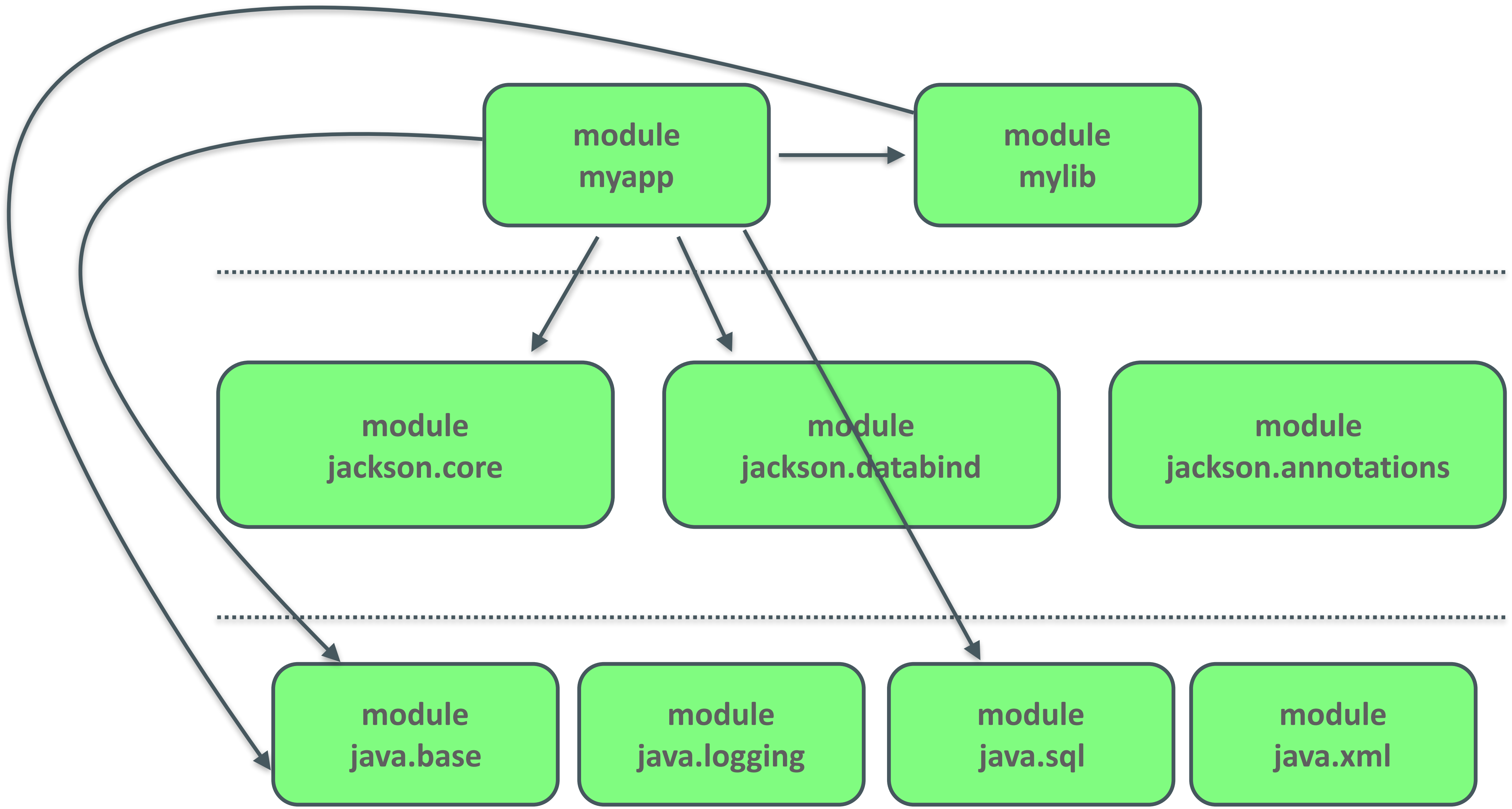
```
// src/mylib/module-info.java
module mylib {
    requires java.base;
    exports com.myapp.lib.util to myapp;
}
```

```
// src/myapp/module-info.java
module myapp {
    requires mylib;
    requires java.base;
    requires java.sql;
    ??? requires jackson.core ???
    ??? requires jackson.databind ???
}
```

If only …

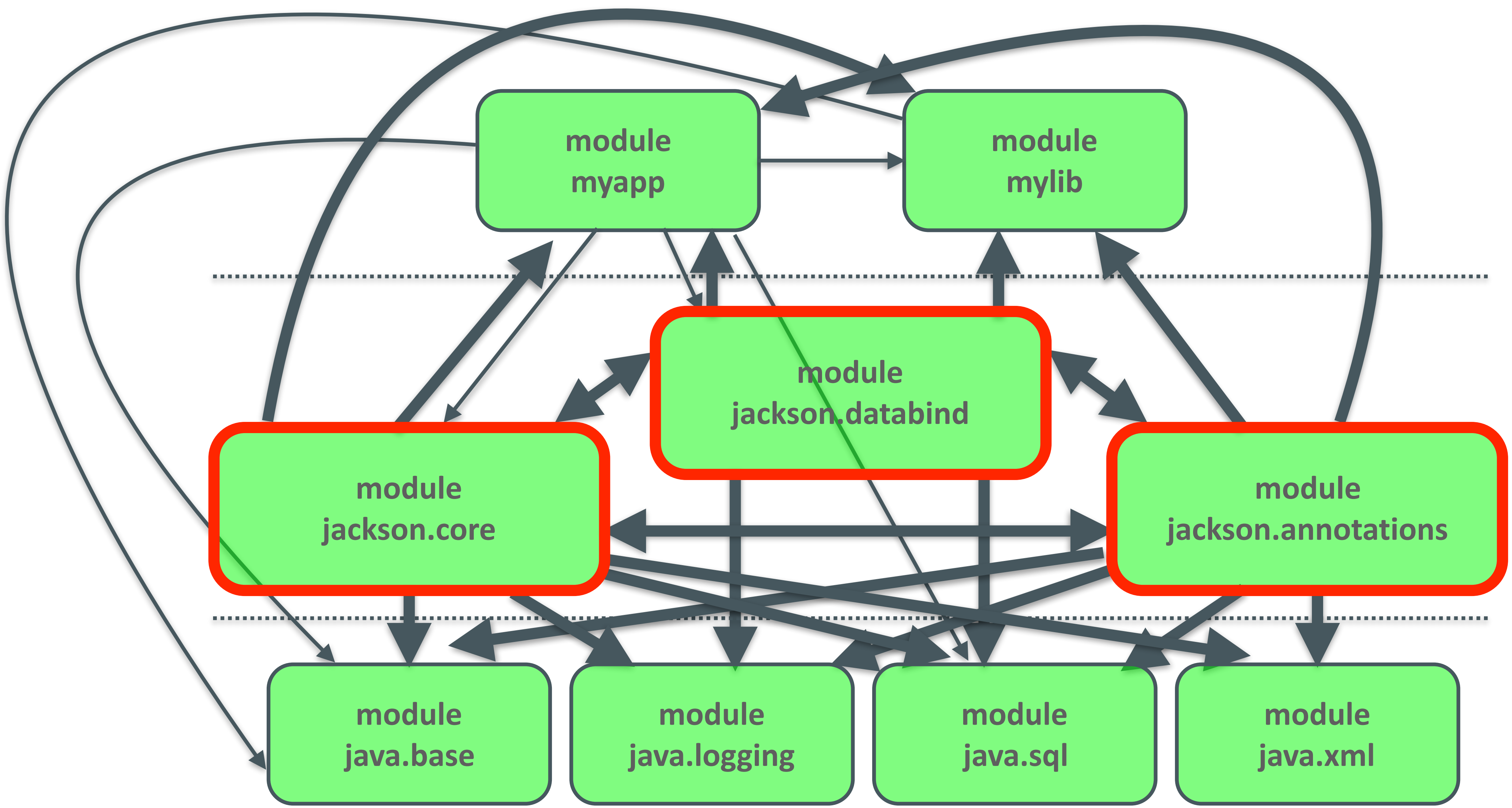

jackson-core-2.6.2.jar
module jackson.core
jackson-databind-2.6.2.jar
module jackson.databind
jackson-annotations-2.6.2.jar
module jackson.annotations

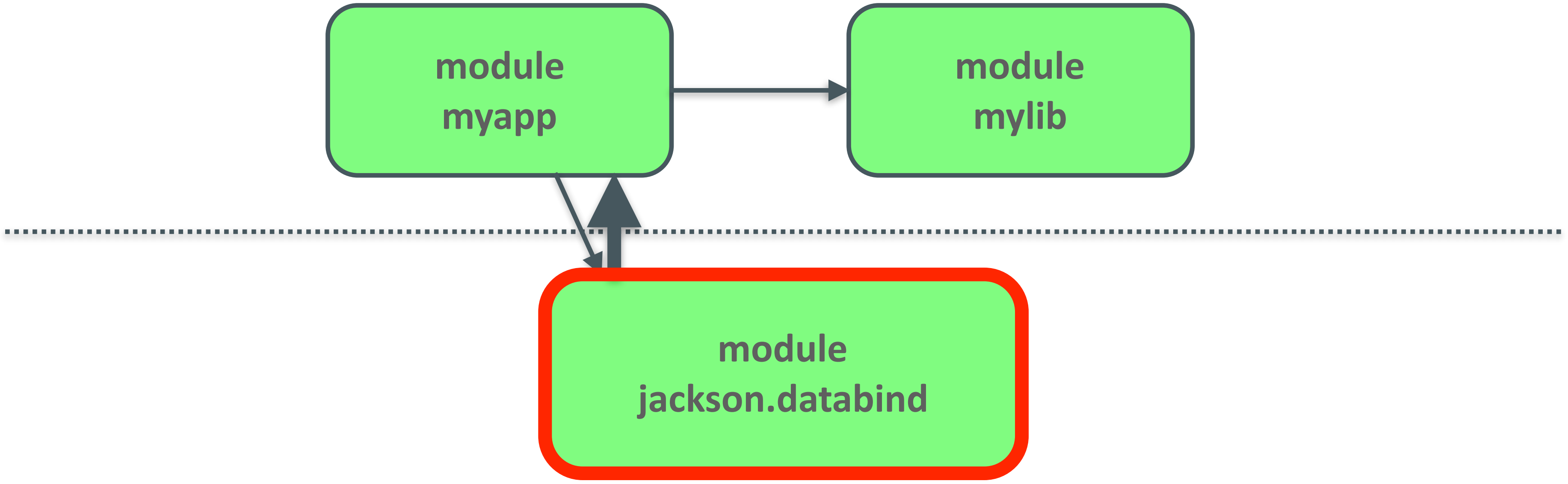
```
// src/myapp/module-info.java
module myapp {
    requires mylib;
    requires java.base;
    requires java.sql;
    requires jackson.core;
    requires jackson.databind;
}
```


module myapp
module mylib
module jackson.core
module jackson.databind
module jackson.annotations
module java.base
module java.logging
module java.sql
module java.xml

Automatic modules

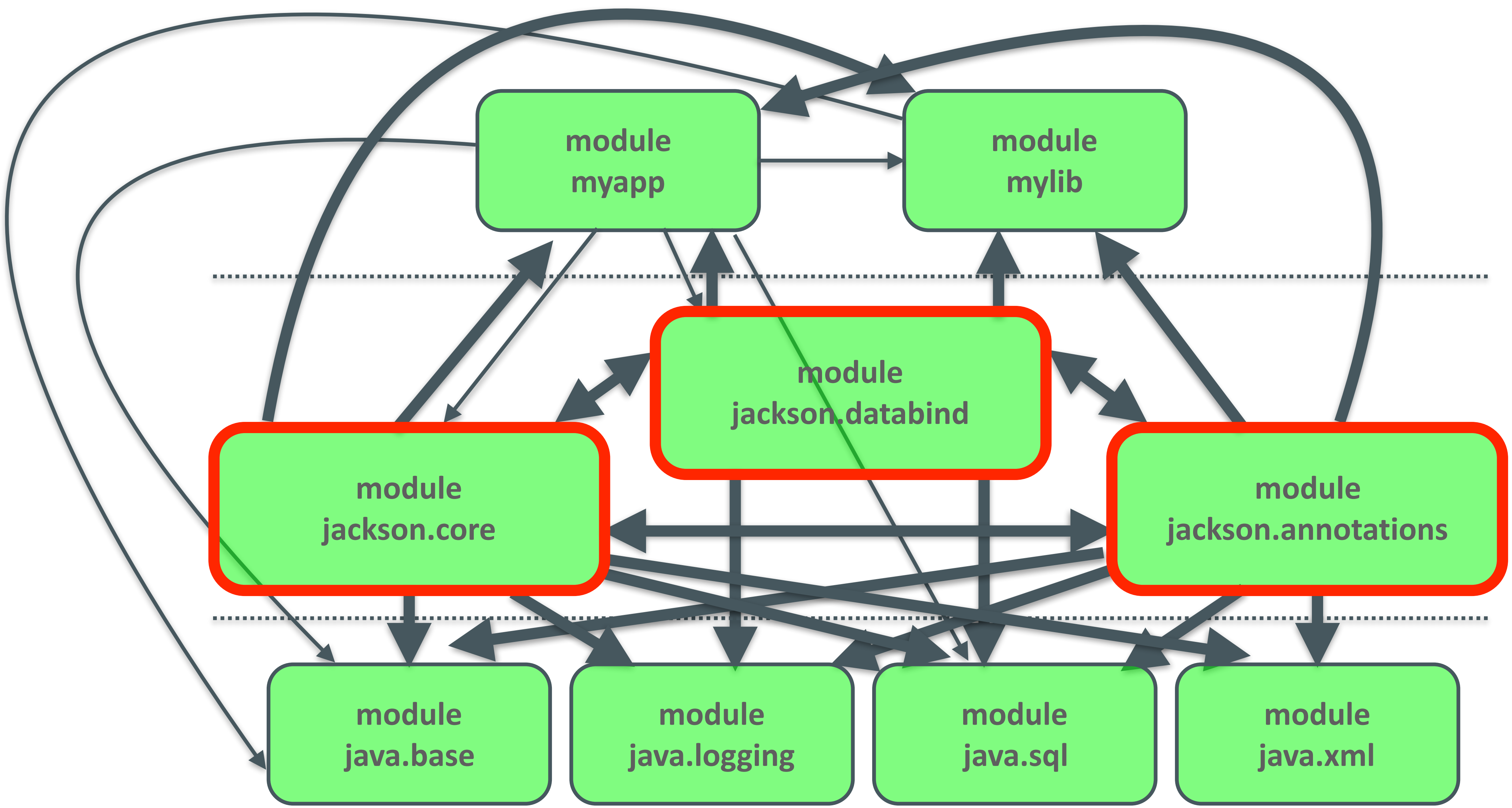
- “Real” modules
- No changes to someone else’s JAR file :-)
- Module name derived from JAR file name
- Exports all its packages
- Requires all other modules

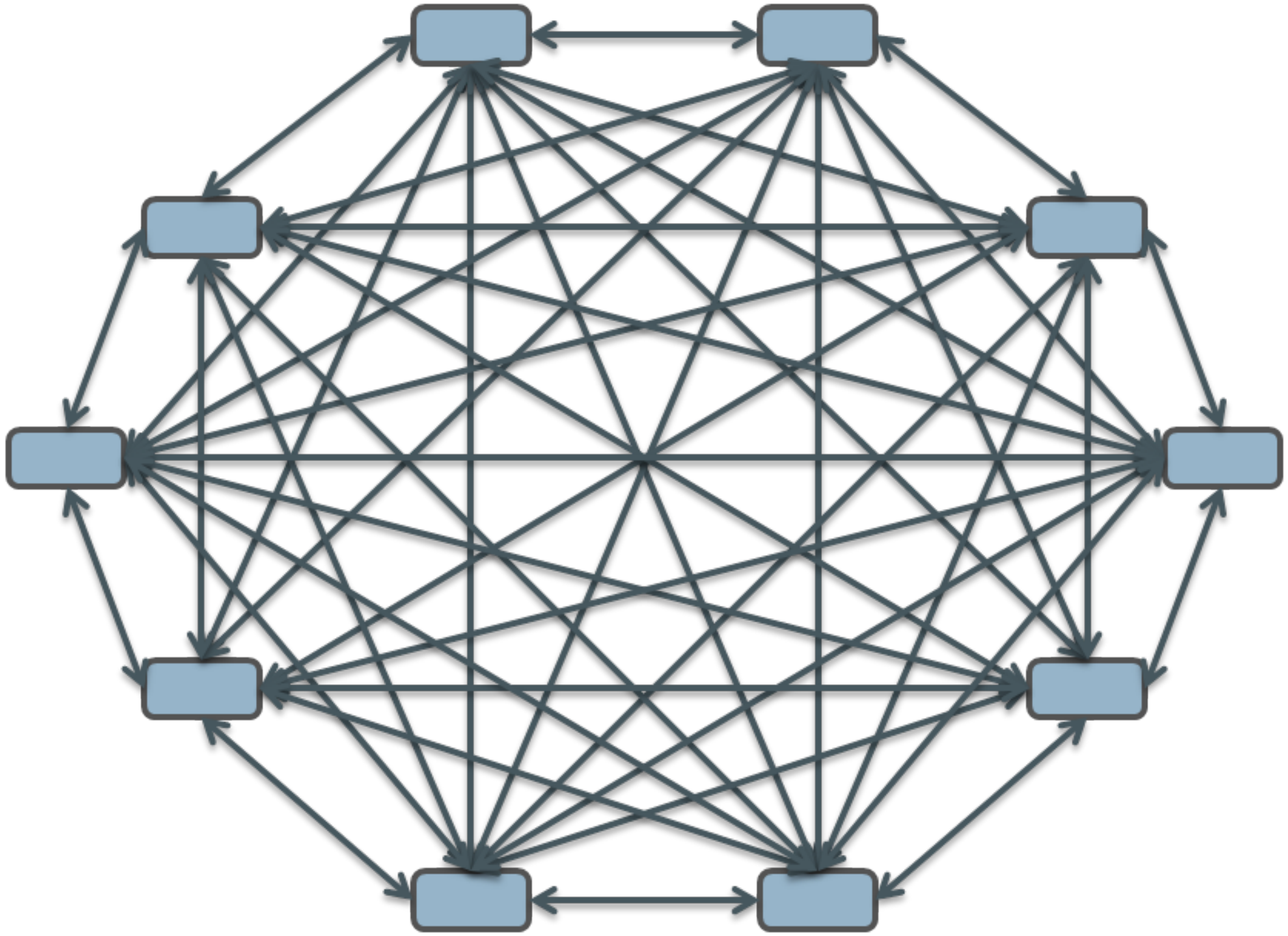

module
myapp
module
mylib
module
jackson.databind
module
jackson.core
module
jackson.annotations
module
java.base
module
java.logging
module
java.sql
module
java.xml



```
import com.fasterxml.jackson.databind.ObjectMapper;
...
ObjectMapper mapper = new ObjectMapper();
MyValue value = mapper.readValue("{\"name\":\"Bob\", \"age\":13}", MyValue.class);
```

```
// src/myapp/module-info.java
weak module myapp {
    requires mylib;
    requires java.base;
    requires java.sql;
    requires jackson.core;
    requires jackson.databind;
}
```


module
myapp
module
mylib
module
jackson.databind
module
jackson.core
module
jackson.annotations
module
java.base
module
java.logging
module
java.sql
module
java.xml



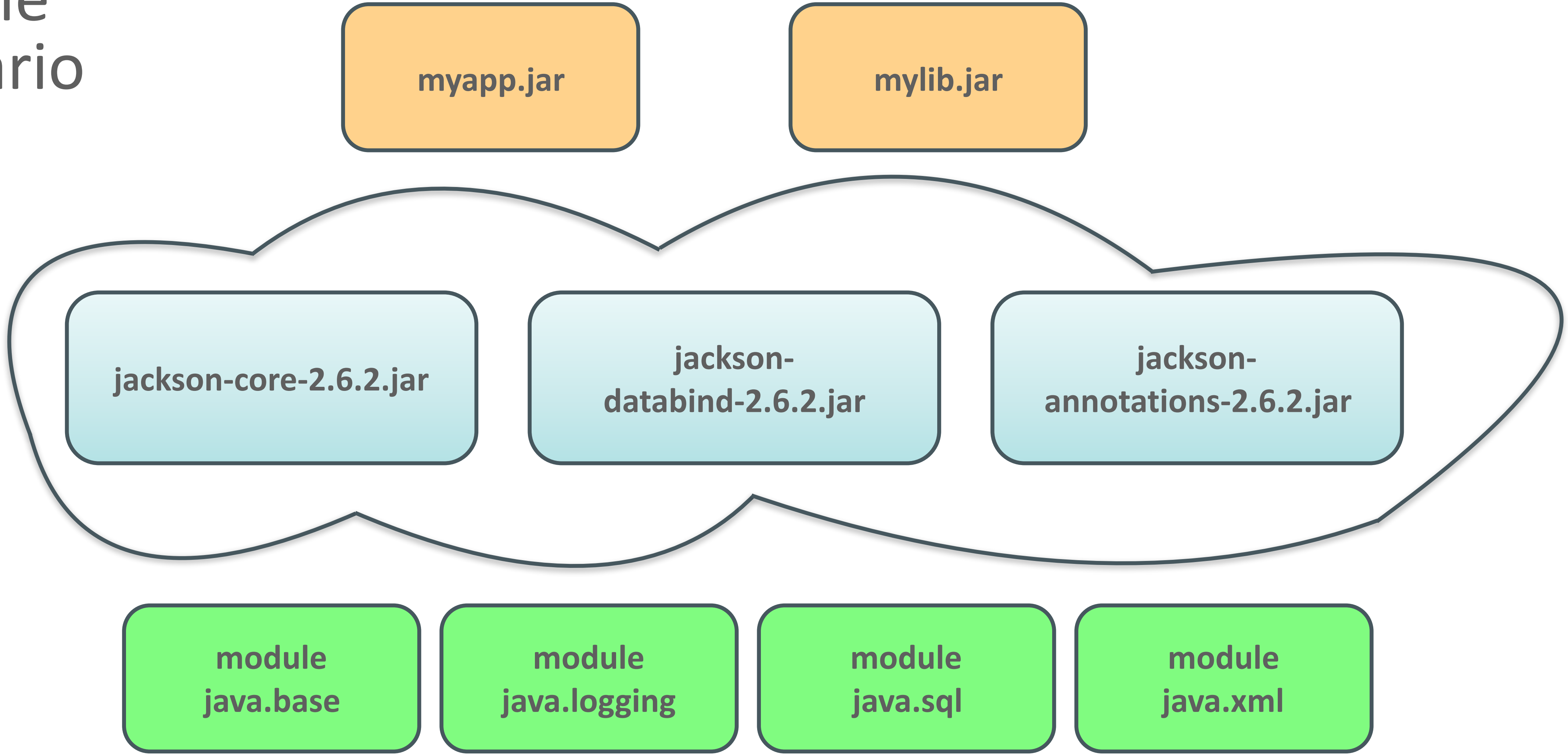
```
$ javac --module-path lib --module-source-path src -d mods …

            lib/jackson-core-2.6.2.jar
            lib/jackson-databind-2.6.2.jar
            lib/jackson-annotations-2.6.2.jar

            src/myapp/module-info.java
            src/myapp/...

            src/mylib/module-info.java
            src/mylib/...

$ jar --create --file mlib/mylib.jar -C mods/mylib .

$ jar --create --file mlib/myapp.jar -C mods/myapp . \
      --main-class myapp.Main
```

```
$ java -cp \
    lib/myapp.jar:\
    lib/mylib.jar:\
    lib/jackson-core-2.6.2.jar:\
    lib/jackson-databind-2.6.2.jar:\
    lib/jackson-annotations-2.6.2.jar \
    myapp.Main
```

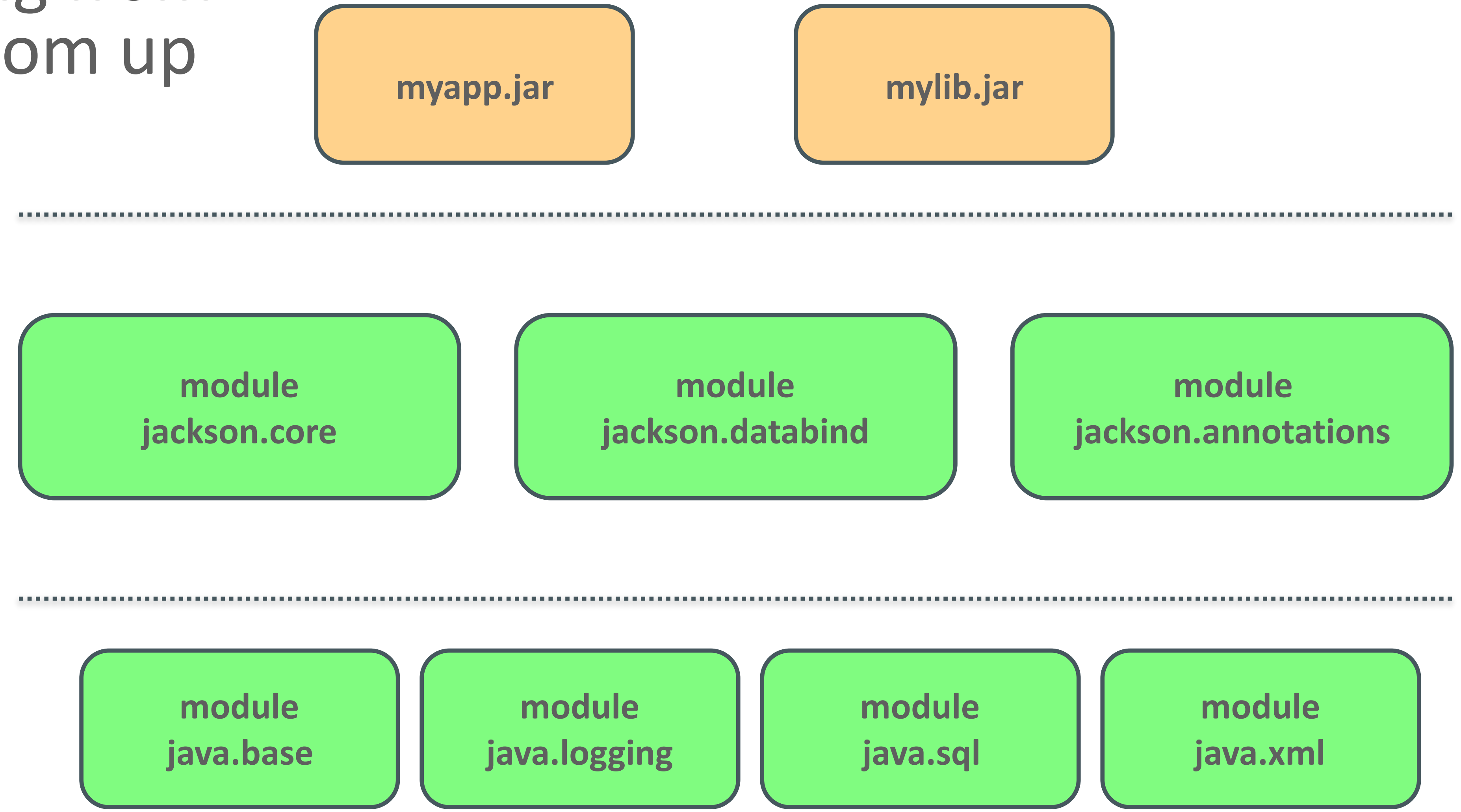
```
$ java --module-path mlib:lib –m myapp
```

Library Migration

Sample Scenario



Migrating from the bottom up

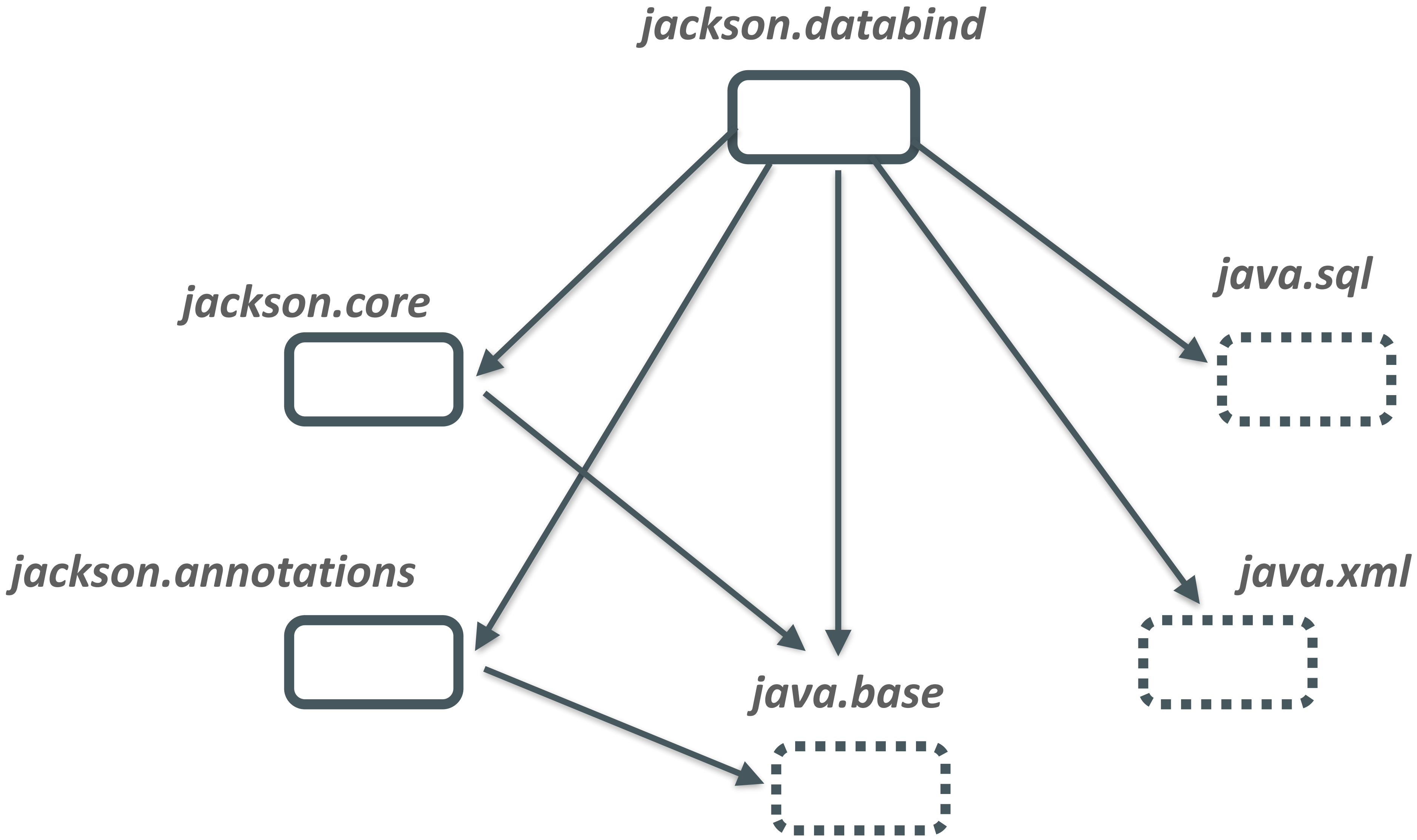


Migrating from the bottom

- module jackson.core requires?
- module jackson.core exports?
- module jackson.databind requires?
- module jackson.databind exports?
- module jackson.annotations requires?
- module jackson.annotations exports?

Requires?

```
$ jdeps -s lib/jackson*.jar
jackson-annotations-2.6.2.jar -> java.base
jackson-core-2.6.2.jar -> java.base
jackson-databind-2.6.2.jar -> lib/jackson-annotations-2.6.2.jar
jackson-databind-2.6.2.jar -> lib/jackson-core-2.6.2.jar
jackson-databind-2.6.2.jar -> java.base
jackson-databind-2.6.2.jar -> java.sql
jackson-databind-2.6.2.jar -> java.xml
```


jackson.databind
jackson.core
jackson.annotations
java.base
java.sql
java.xml

Creating the module-info.java for each module

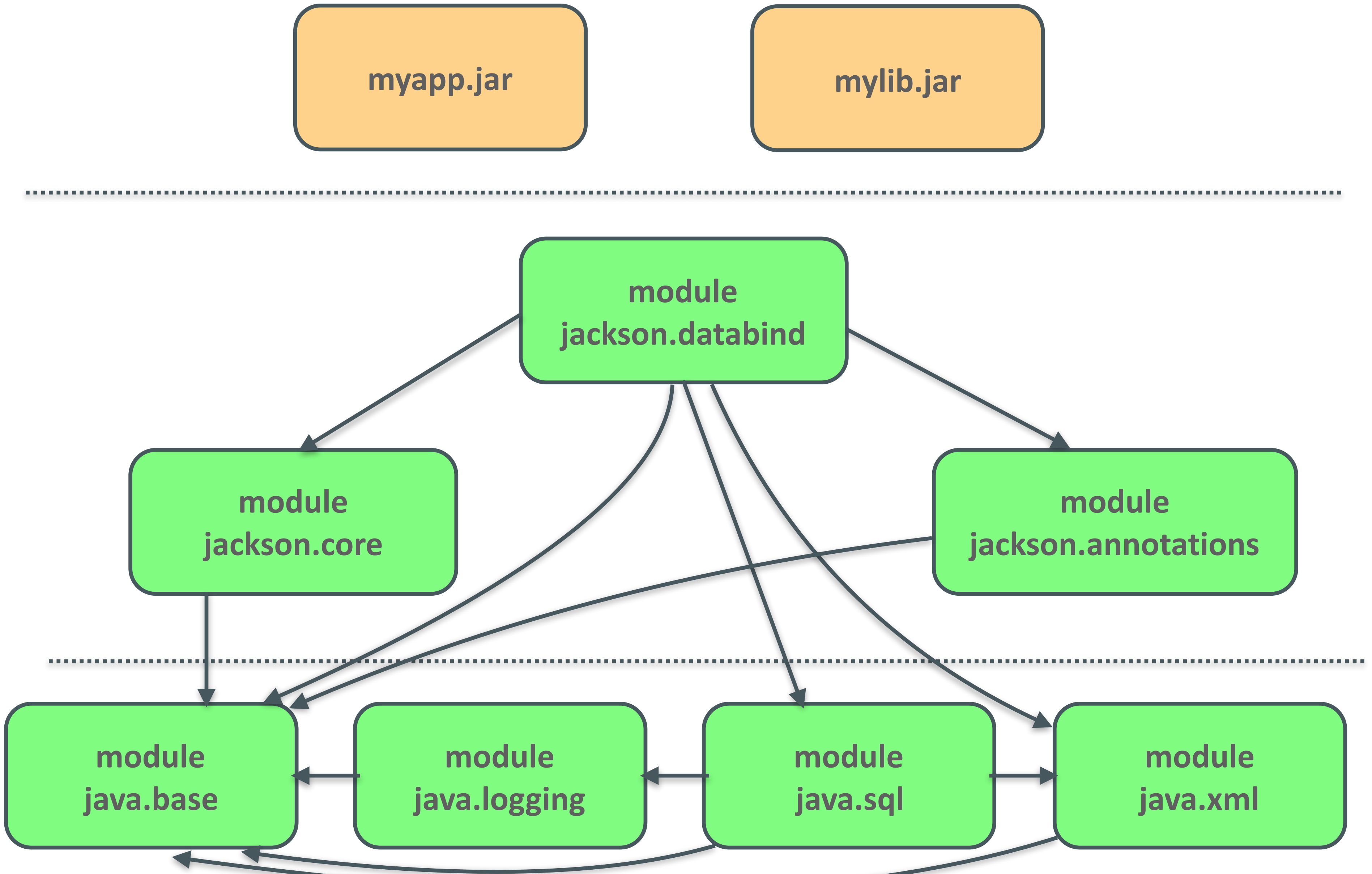
```
$ jdeps --gen-module-info src *.jar
writing to src/jackson.annotations/module-info.java
writing to src/jackson.databind/module-info.java
writing to src/jackson.core/module-info.java
```

```
// src/jackson.databind/module-info.java
module jackson.databind {
    requires transitive jackson.annotations;
    requires transitive jackson.core;
    requires java.logging;
    requires transitive java.sql;
    requires transitive java.xml;
    exports com.fasterxml.jackson.databind;
    exports com.fasterxml.jackson.databind.annotation;
    exports com.fasterxml.jackson.databind.cfg;
    exports com.fasterxml.jackson.databind.deser;
    exports com.fasterxml.jackson.databind.deser.impl;
    exports com.fasterxml.jackson.databind.jsontype;
    exports com.fasterxml.jackson.databind.jsontype.impl;
    exports com.fasterxml.jackson.databind.module;
    exports com.fasterxml.jackson.databind.node;
    exports com.fasterxml.jackson.databind.ser;
    exports com.fasterxml.jackson.databind.ser.impl;
    exports com.fasterxml.jackson.databind.ser.std;
    exports com.fasterxml.jackson.databind.type;
    exports com.fasterxml.jackson.databind.util;
}
```

```java
// src/jackson.databind/module-info.java
module jackson.databind {
    requires transitive jackson.annotations;
    requires transitive jackson.core;
    requires java.logging;
    requires transitive java.sql;
    requires transitive java.xml;
    exports com.fasterxml.jackson.databind;
    exports com.fasterxml.jackson.databind.annotation;
    exports com.fasterxml.jackson.databind.cfg;
    exports com.fasterxml.jackson.databind.deser;
    ~~exports com.fasterxml.jackson.databind.deser.impl;~~
    exports com.fasterxml.jackson.databind.jsontype;
    ~~exports com.fasterxml.jackson.databind.jsontype.impl;~~
    exports com.fasterxml.jackson.databind.module;
    exports com.fasterxml.jackson.databind.node;
    exports com.fasterxml.jackson.databind.ser;
    ~~exports com.fasterxml.jackson.databind.ser.impl;~~
    exports com.fasterxml.jackson.databind.ser.std;
    exports com.fasterxml.jackson.databind.type;
    exports com.fasterxml.jackson.databind.util;
}
```

```
src/jackson.core/module-info.java
src/jackson.core/...

src/jackson.databind/module-info.java
src/jackson.databind/...

src/jackson.annotations/module-info.java
src/jackson.annotations/...
```

```
$ javac --module-source-path src -d mods $(find src -name “*.java”)
$ javac --module-source-path src -d mods ...

$ jar --create --file mlib/jackson-core-2.6.2.jar \
   --module-version 2.6.2 -C mods/jackson.core .

$ jar --create --file mlib/jackson-databind-2.6.2.jar \
   --module-version 2.6.2 -C mods/jackson.databind .

$ jar --create --file mlib/jackson-annotations-2.6.2.jar \
   --module-version 2.6.2 -C mods/jackson.annotations .
```

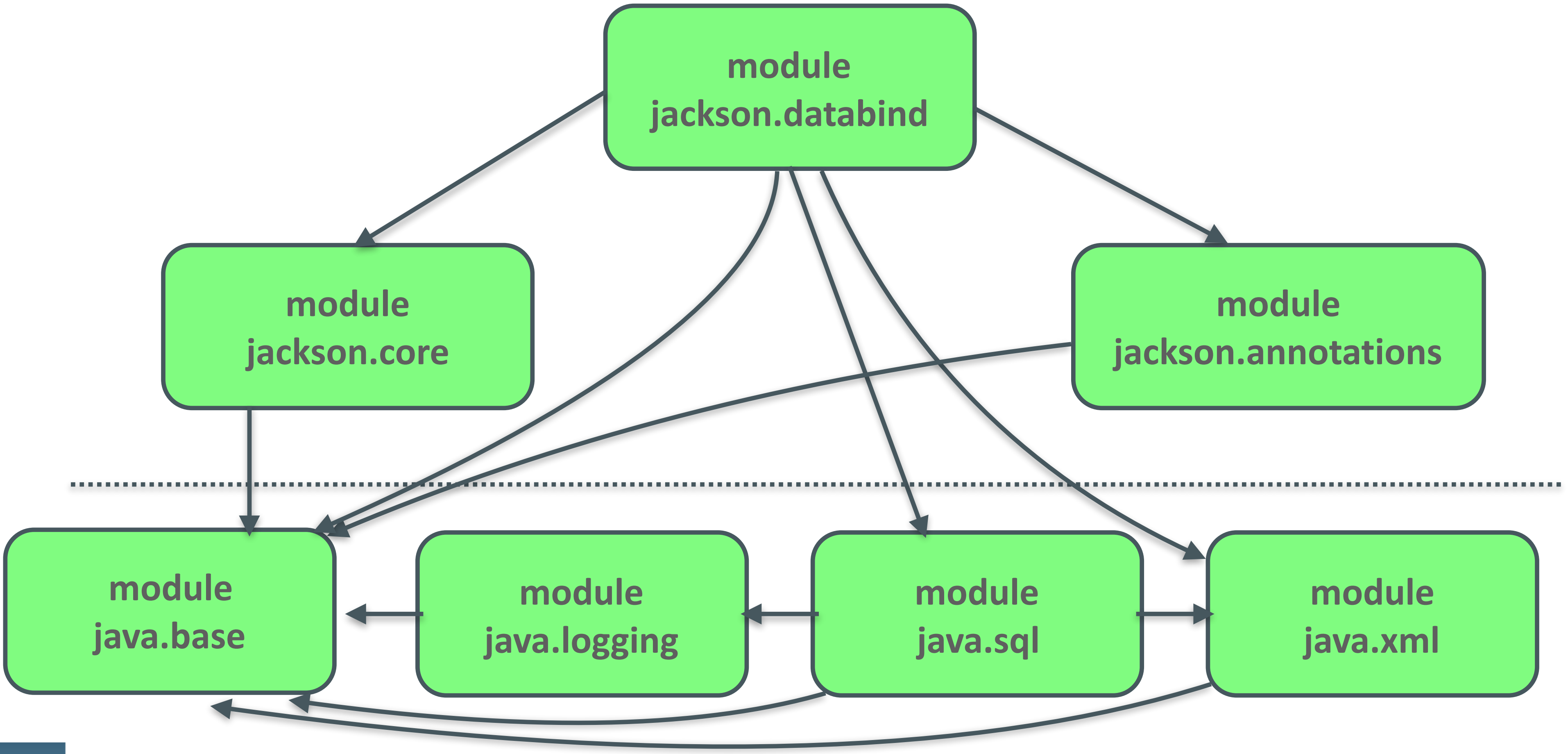

myapp.jar
mylib.jar
module jackson.databind
module jackson.core
module jackson.annotations
module java.base
module java.logging
module java.sql
module java.xml

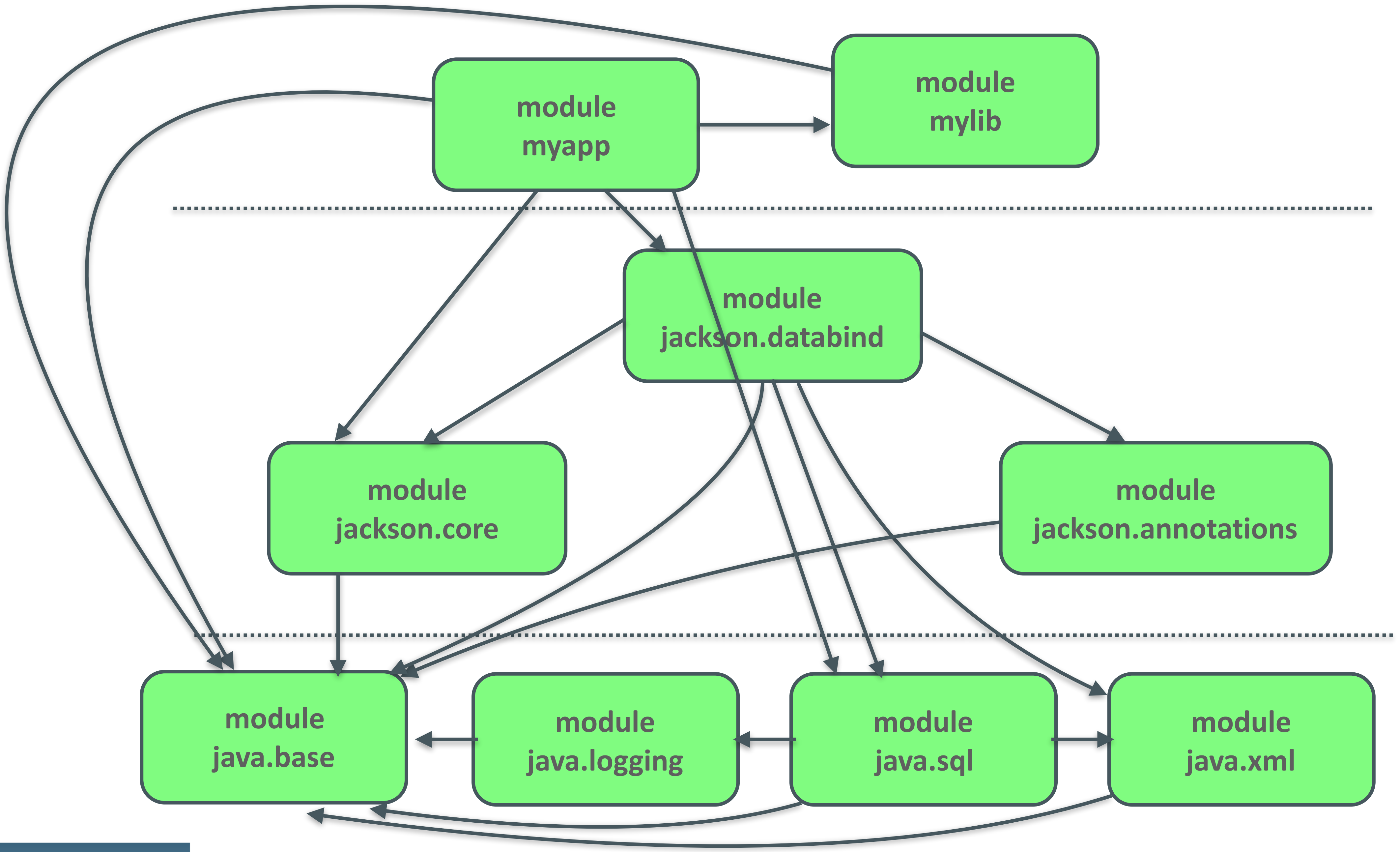
```
$ java -cp \
    lib/myapp.jar:\
    lib/mylib.jar:\
    lib/jackson-core-2.6.2.jar:\
    lib/jackson-databind-2.6.2.jar:\
    lib/jackson-annotations-2.6.2.jar \
    myapp.Main
```

```
$ java -cp \
    lib/myapp.jar:\
    lib/mylib.jar:\
    lib/jackson-core-2.6.2.jar:\
    lib/jackson-databind-2.6.2.jar:\
    lib/jackson-annotations-2.6.2.jar \
    myapp.Main
```

```
$ java --module-path mlib \
       --add-modules jackson.databind \
       -cp lib/myapp.jar:lib/mylib.jar
   myapp.Main
```

Putting it all together


module jackson.databind
module jackson.core
module jackson.annotations
module java.base
module java.logging
module java.sql
module java.xml


module myapp
module mylib
module jackson.databind
module jackson.core
module jackson.annotations
module java.base
module java.logging
module java.sql
module java.xml

```
$ ls -1 mlib
myapp.jar
mylib.jar
jackson-core-2.6.2.jar
jackson-databind-2.6.2.jar
jackson-annotations-2.6.2.jar

$ java --module-path mlib -m myapp
```

Linking

```
$ jlink --module-path $JDKMODS:mlib --add-modules myapp \
    --compress=2 --strip-debug --output myimage

$ du -ks myimage
26000   myimage

$ ls myimage/bin
myapp    java    keytool

$ myimage/bin/java --list-modules
myapp
mylib
jackson.annotations@2.6.2
jackson.core@2.6.2
jackson.databind@2.6.2
java.base@9
java.logging@9
java.sql@9
java.xml@9
```

```
$ myimage/bin/java –m myapp
Greetings from myapp, here’s some json!
```

```
$ myimage/bin/myapp
Greetings from myapp, here’s some json!
```

Summary

- Freedom to adopt modules at your own pace
  - Modularize your application before its libraries
  - Modularize libraries independently
- Different kinds of modules
  - Explicit vs. automatic
  - Strong vs. weak

Go forth and modularize!

Other sessions, mostly this room

- Prepare for JDK 9: Tues @ 2.30pm
- Introduction to Modular Development: Wed @ 3pm
- Modules and Services: Tues @ 11.00am, Thur @ 2.30pm
- Project Jigsaw: Under The Hood: Tues @ 4pm
- Project Jigsaw Hack Session: Wed @ 8.30am

More Information

OpenJDK Project Jigsaw

http://openjdk.java.net/projects/jigsaw/

mailto:jigsaw-dev@openjdk.java.net

Early Access Builds

https://jdk9.java.net/jigsaw/

Safe Harbor Statement

ORACLE®

JavaOne™

ORACLE®