

# Advanced Modular Development

CON6821

Mark Reinhold, Alex Buckley, Alan Bateman  
Java Platform Group, Oracle  
October 2015

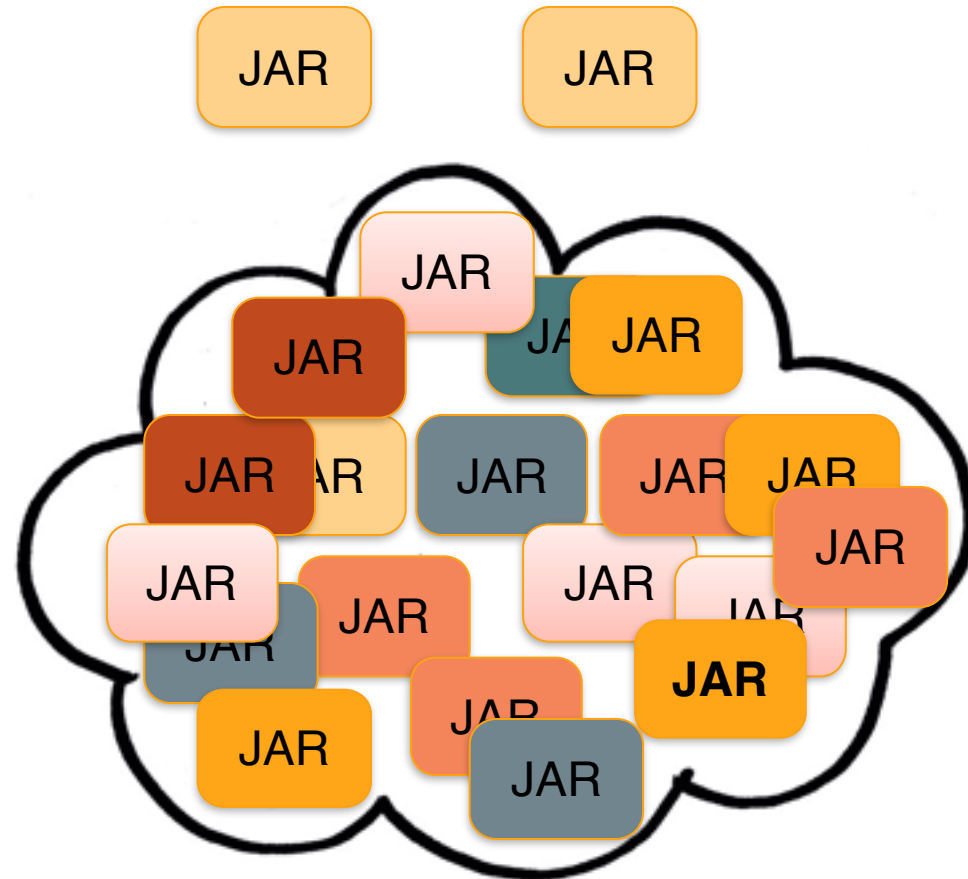


# Sessions

- 1 Prepare for JDK 9
- 2 Introduction to Modular Development
- 3 Advanced Modular Development**
- 4 Project Jigsaw: Under the Hood
- 5 Project Jigsaw Hack Session

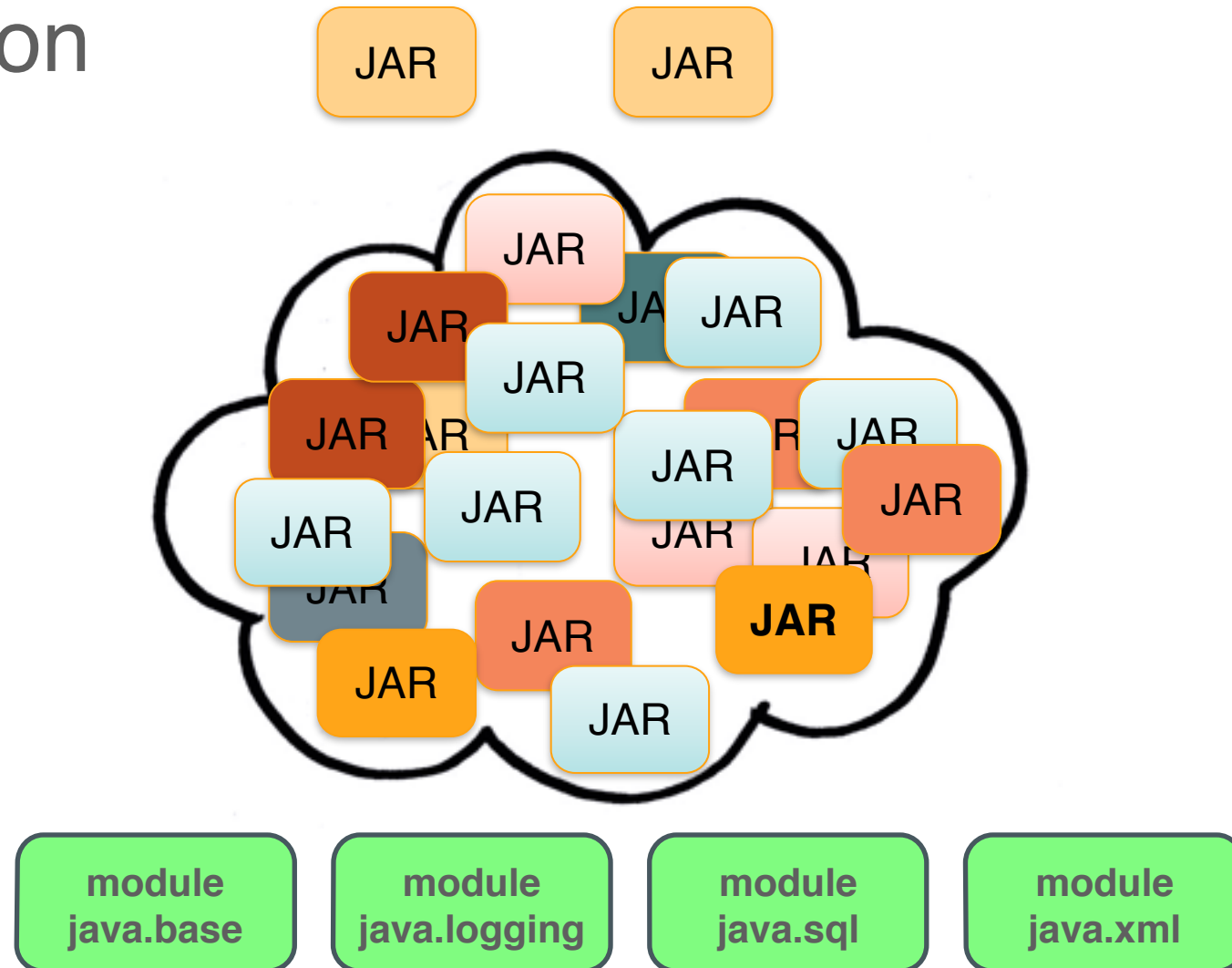
# Application Migration

# Typical application

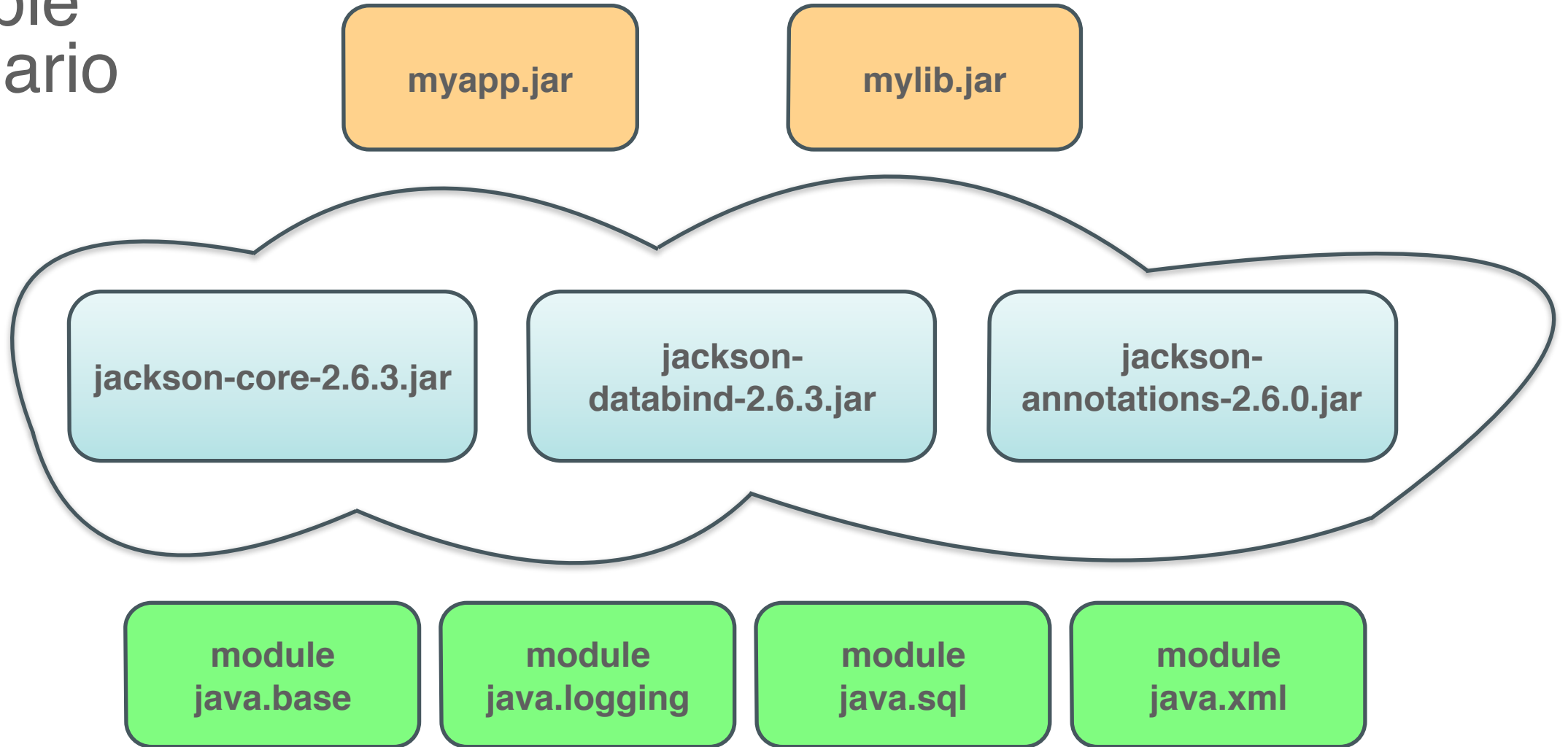


**JDK**

# Typical application



# Sample Scenario



# Running my application

```
$ java -cp \  
lib/myapp.jar:\  
lib/mylib.jar:\  
lib/jackson-core-2.6.3.jar:\  
lib/jackson-databind-2.6.3.jar:\  
lib/jackson-annotations-2.6.0.jar\  
myapp.Main
```

# Migrating from the top down

module  
myapp

module  
mylib

jackson-core-2.6.3.jar

jackson-  
databind-2.6.3.jar

jackson-  
annotations-2.6.0.jar

module  
java.base

module  
java.logging

module  
java.sql

module  
java.xml



# Migrating from the top down

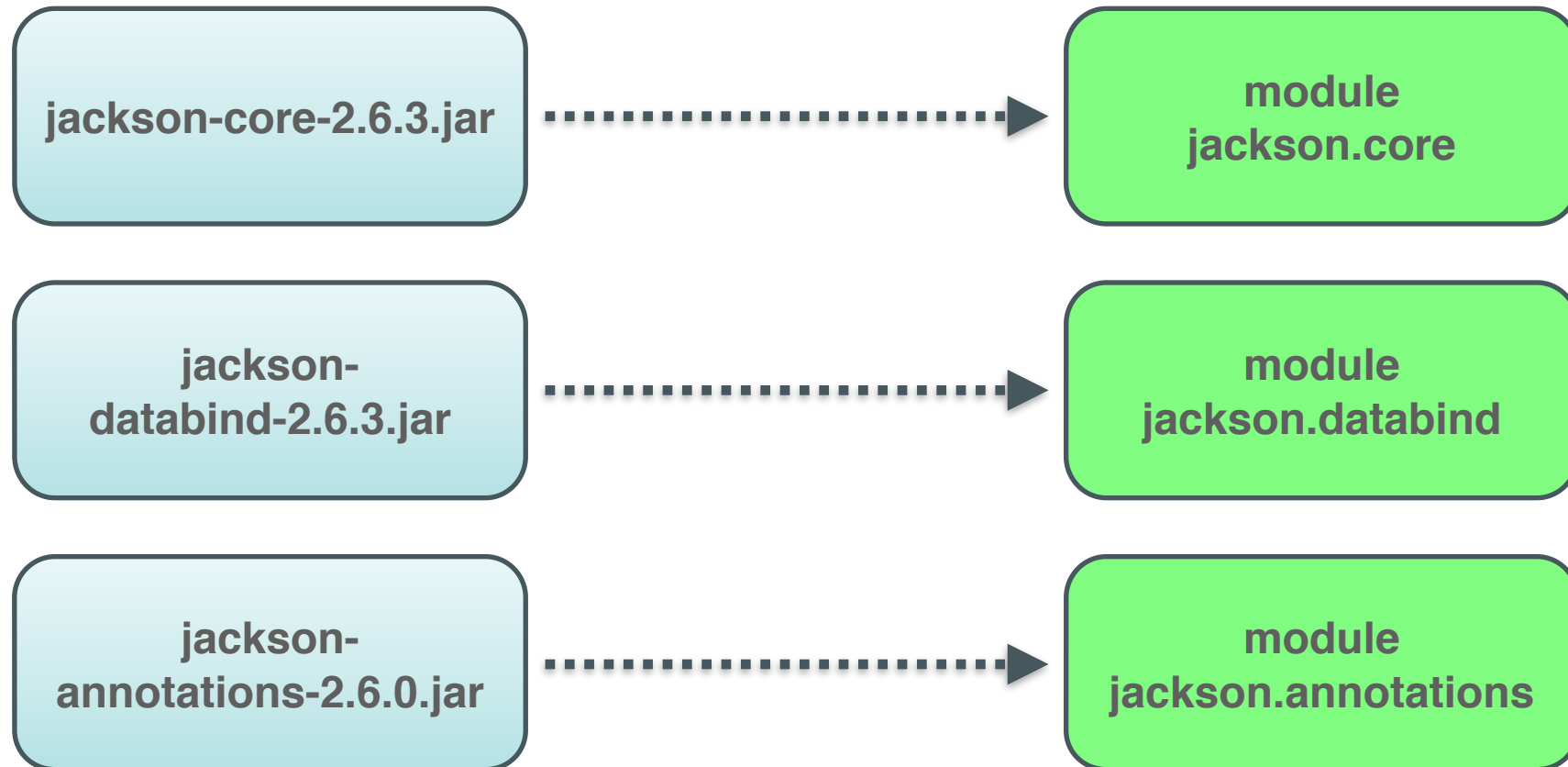
- module myapp requires?
- module myapp exports?
  
- module mylib requires?
- module mylib exports?

```
$ jdeps -s lib/myapp.jar lib/mylib.jar  
myapp.jar -> lib/jackson-core-2.6.3.jar  
myapp.jar -> lib/jackson-databind-2.6.3.jar  
myapp.jar -> mylib.jar  
myapp.jar -> java.base  
myapp.jar -> java.sql  
mylib.jar -> java.base
```

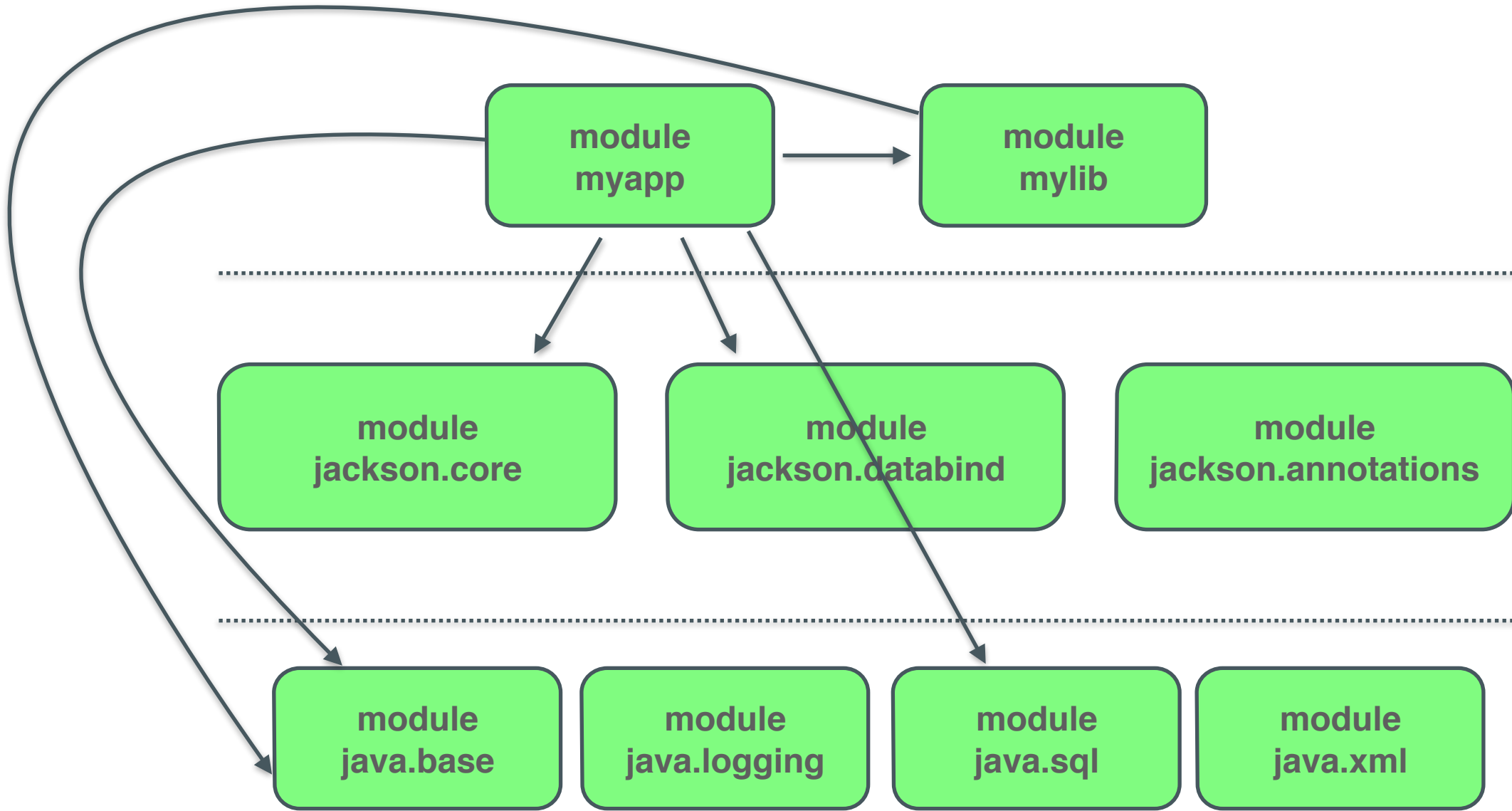
```
// src/mylib/module-info.java
module mylib {
    requires java.base;
    exports com.myapp.lib.util to myapp;
}
```

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

# If only ...



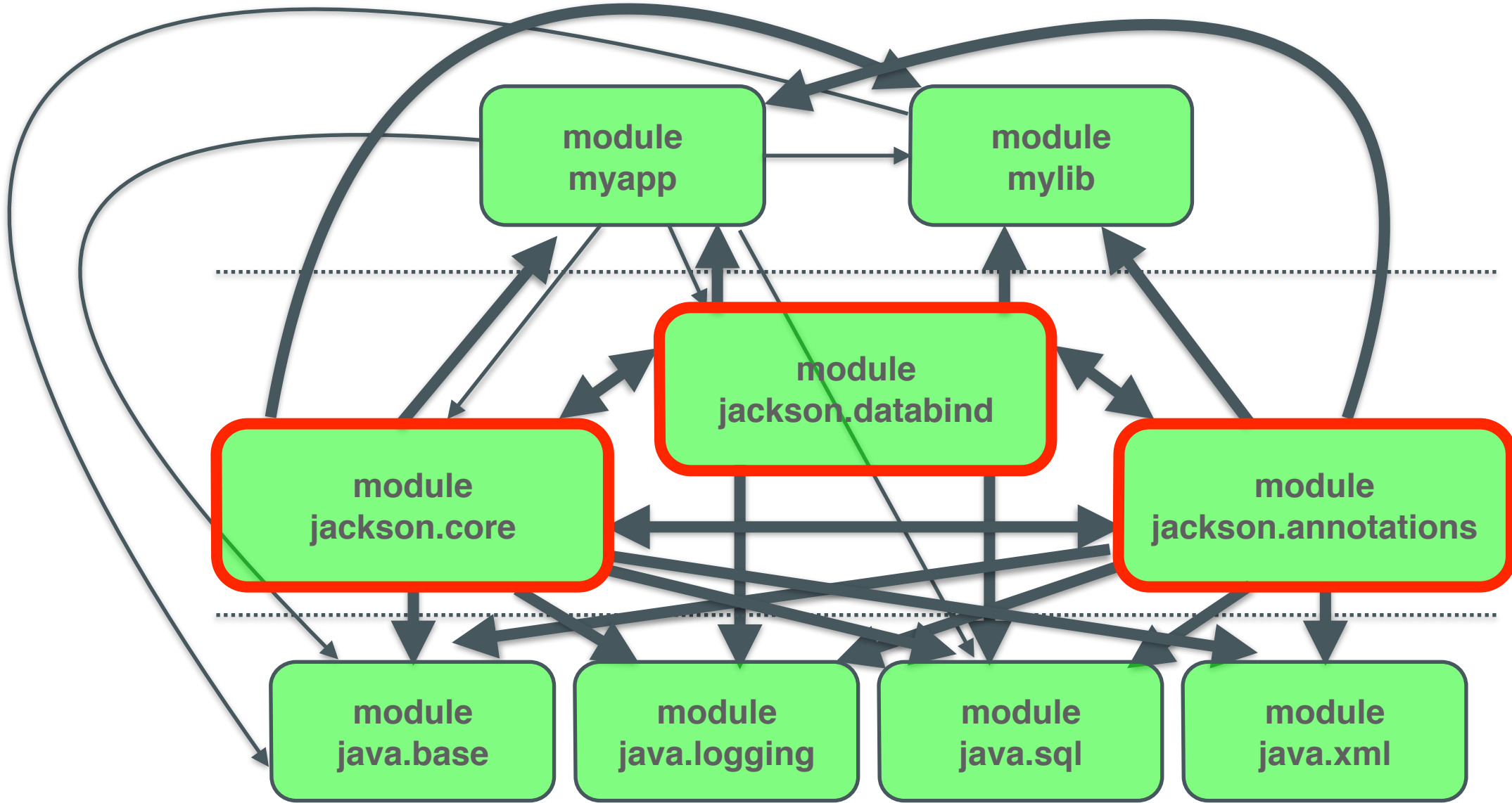
```
// src/myapp/module-info.java
module myapp {
    requires mylib;
    requires java.base;
    requires java.sql;
    requires jackson.core;
    requires jackson.databind;
}
```

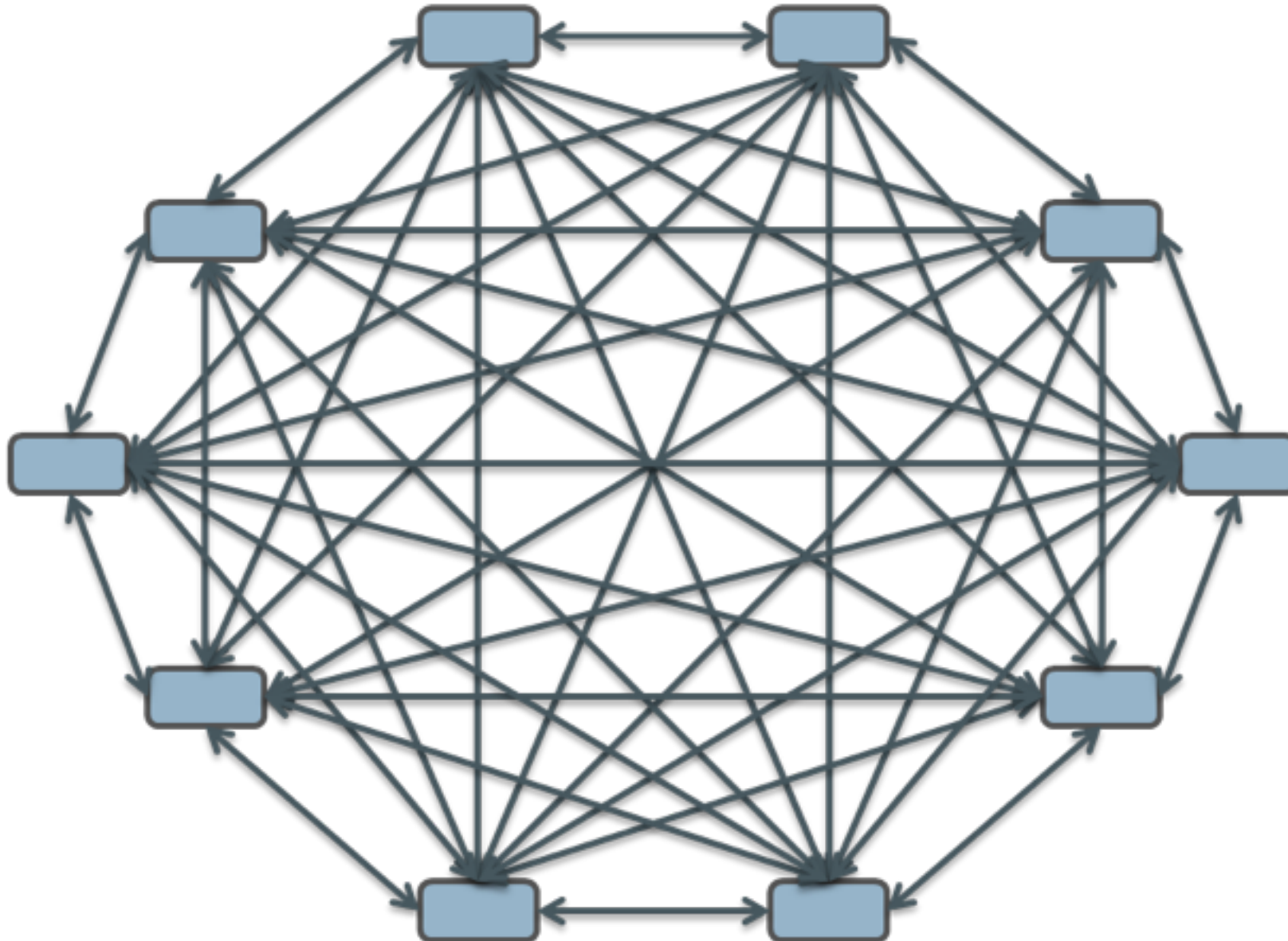


# Automatic modules

- “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







```
lib/jackson-core-2.6.3.jar  
lib/jackson-databind-2.6.3.jar  
lib/jackson-annotations-2.6.0.jar
```

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

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

```
$ javac -modulesourcepath src -mp lib -d mods ...
```

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

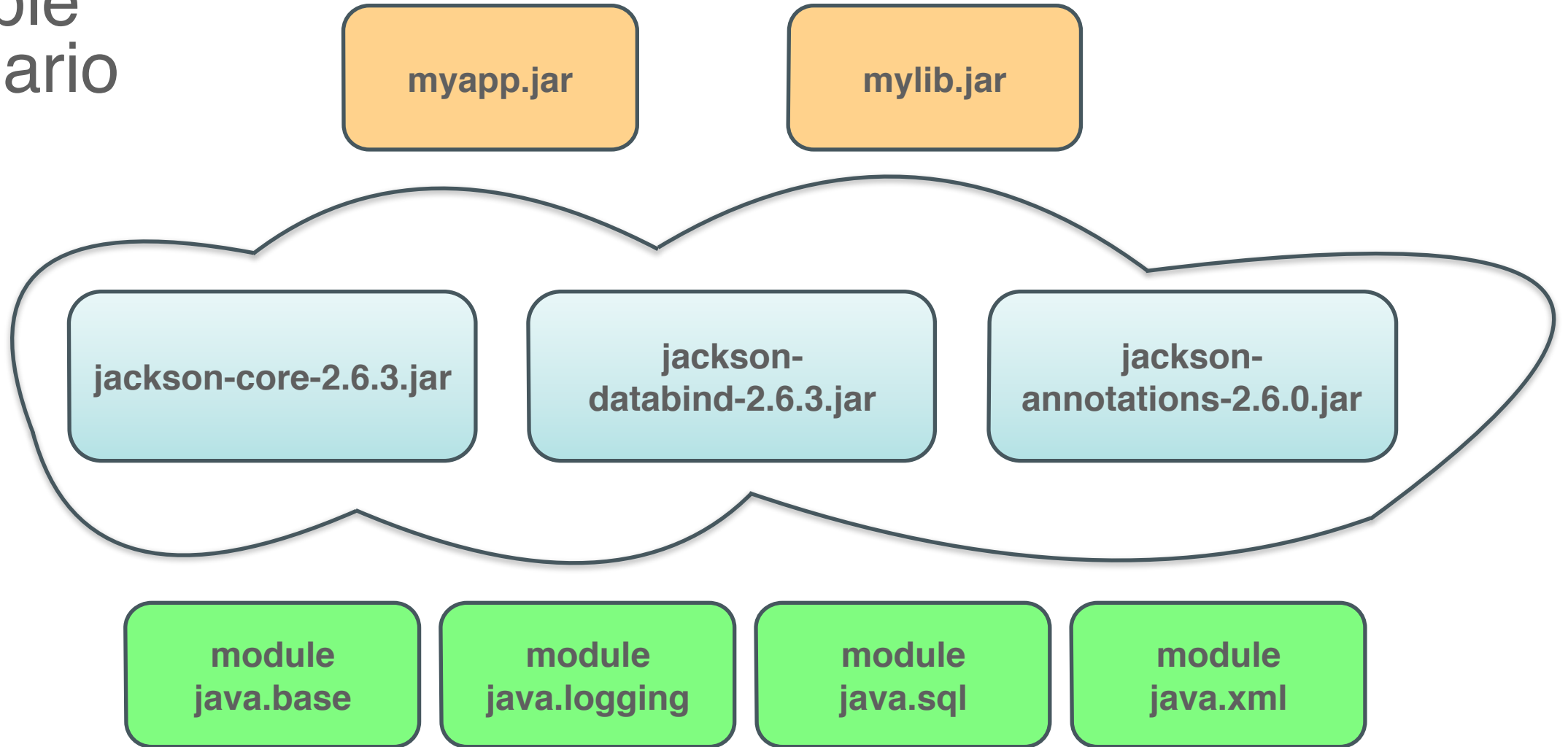
```
$ jar --create --file mlib/myapp.jar -C mods/myapp . \  
--main-class myapp.Main
```

```
$ java -cp \  
lib/myapp.jar:\  
lib/mylib.jar:\  
lib/jackson-core-2.6.3.jar:\  
lib/jackson-databind-2.6.3.jar:\  
lib/jackson-annotations-2.6.0.jar \  
myapp.Main
```

```
$ java -mp mlib:lib -m myapp
```

# Library Migration

# Sample Scenario



# Migrating from the bottom up

myapp.jar

mylib.jar

module  
jackson.core

module  
jackson.databind

module  
jackson.annotations

module  
java.base

module  
java.logging

module  
java.sql

module  
java.xml

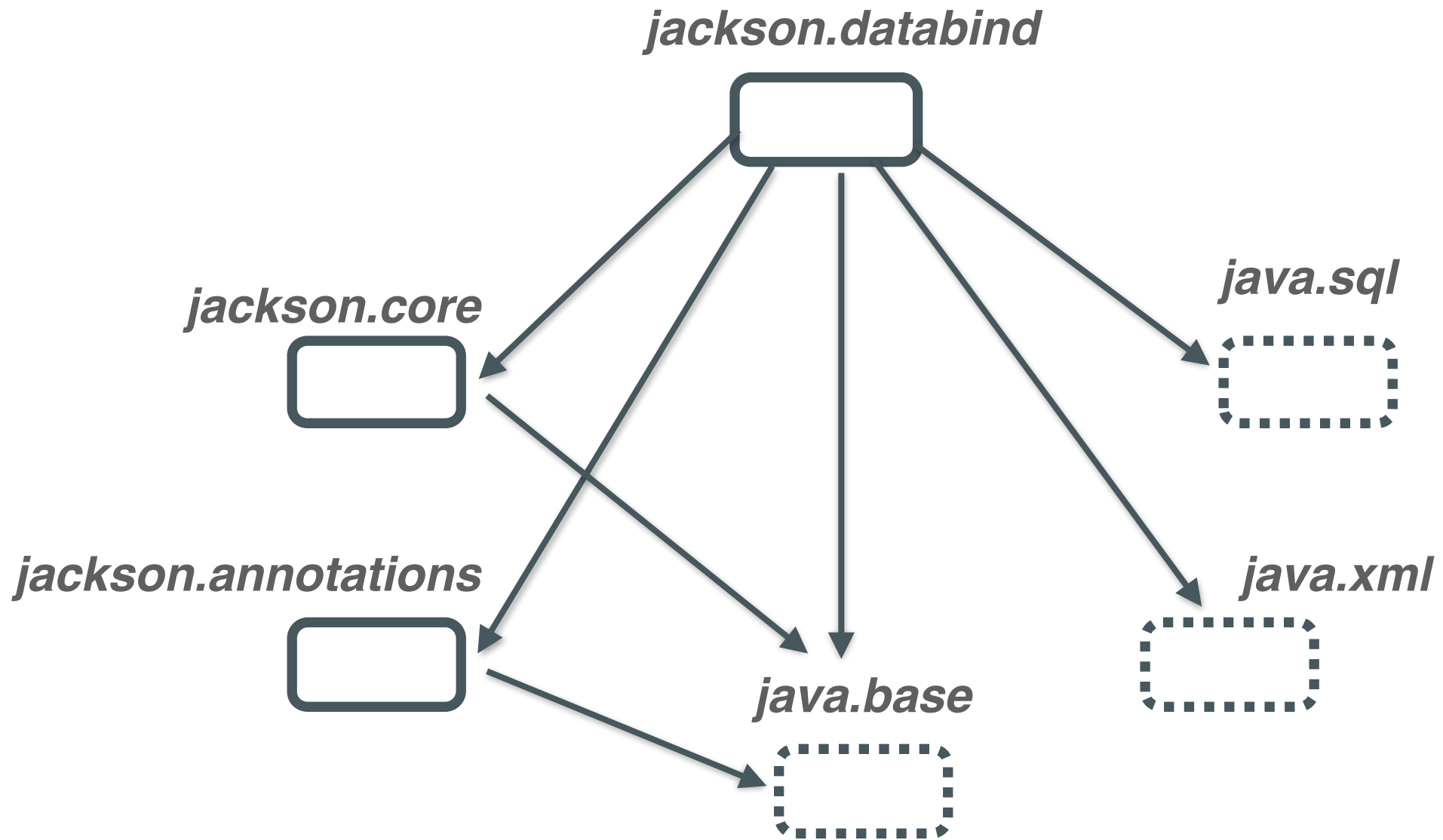
# Migrating from the bottom

- 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-2.6.0.jar -> java.base  
jackson-core-2.6.3.jar -> java.base  
jackson-databind-2.6.3.jar -> lib/jackson-annotations-2.6.0.jar  
jackson-databind-2.6.3.jar -> lib/jackson-core-2.6.3.jar  
jackson-databind-2.6.3.jar -> java.base  
jackson-databind-2.6.3.jar -> java.sql  
jackson-databind-2.6.3.jar -> java.xml
```



# Creating the module-info.java for each module

```
$ jdeps -genmoduleinfo 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 public java.sql;
    requires public java.xml;
    requires public jackson.annotations;
    requires public jackson.core;
    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.databind/module-info.java
module jackson.databind {
    requires public java.sql;
    requires public java.xml;
    requires public jackson.annotations;
    requires public jackson.core;
    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 -modulesourcepath src -d mods ...
```

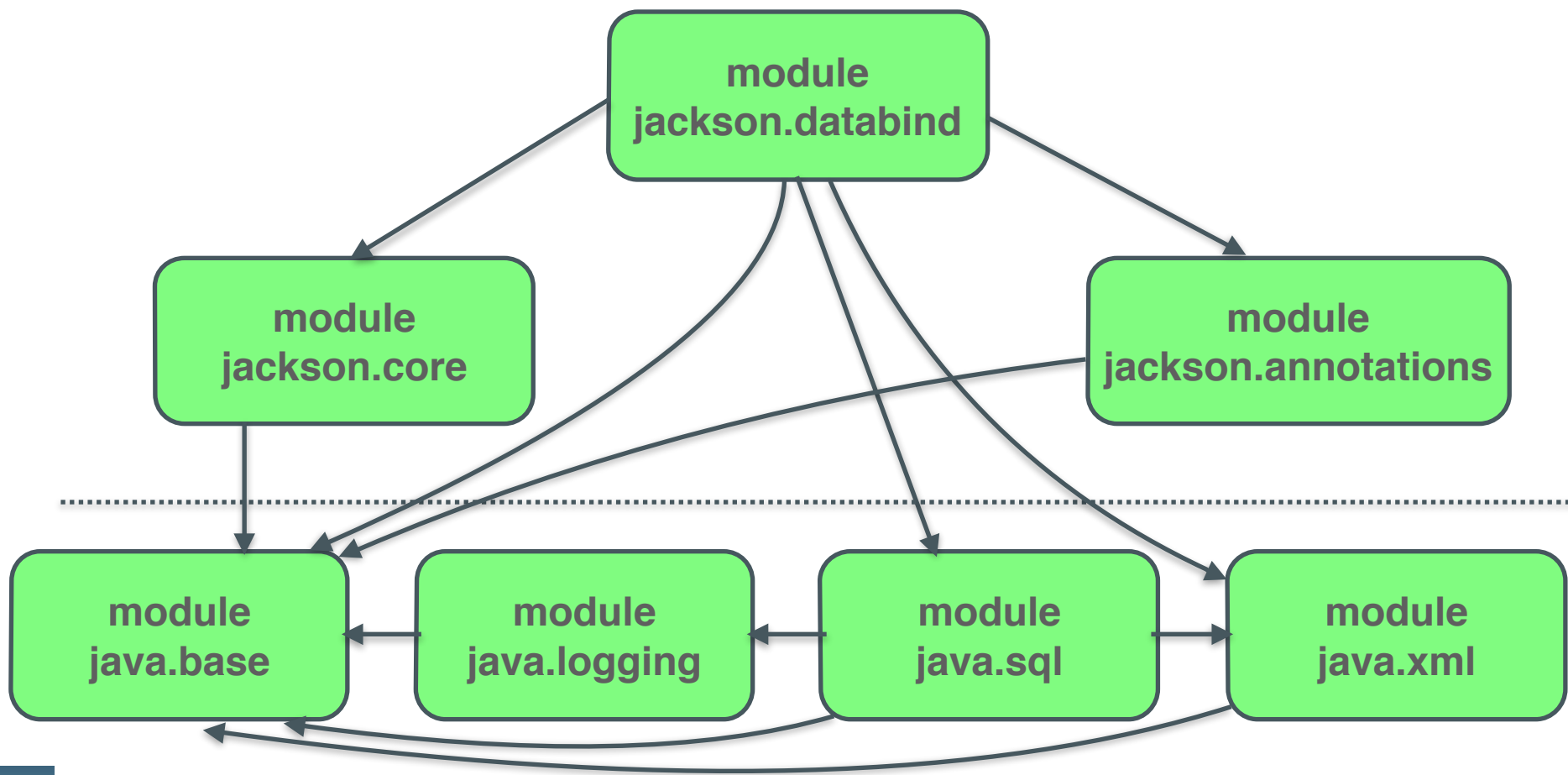
```
$ jar --create --file mlib/jackson.core-2.6.3.jar \  
  --module-version 2.6.3 -C mods/jackson.core .
```

```
$ jar --create --file mlib/jackson.databind-2.6.3.jar \  
  --module-version 2.6.3 -C mods/jackson.databind .
```

```
$ jar --create --file mlib/jackson.annotations-2.6.0.jar \  
  --module-version 2.6.0 -C mods/jackson.annotations .
```

myapp.jar

mylib.jar



```
$ java -cp \  
lib/myapp.jar:\  
lib/mylib.jar:\  
lib/jackson-core-2.6.3.jar:\  
lib/jackson-databind-2.6.3.jar:\  
lib/jackson-annotations-2.6.0.jar \  
myapp.Main
```

```
$ java -mp mlib -cp lib/myapp.jar:lib/mylib.jar  
-addmods jackson.databind \  
myapp.Main
```

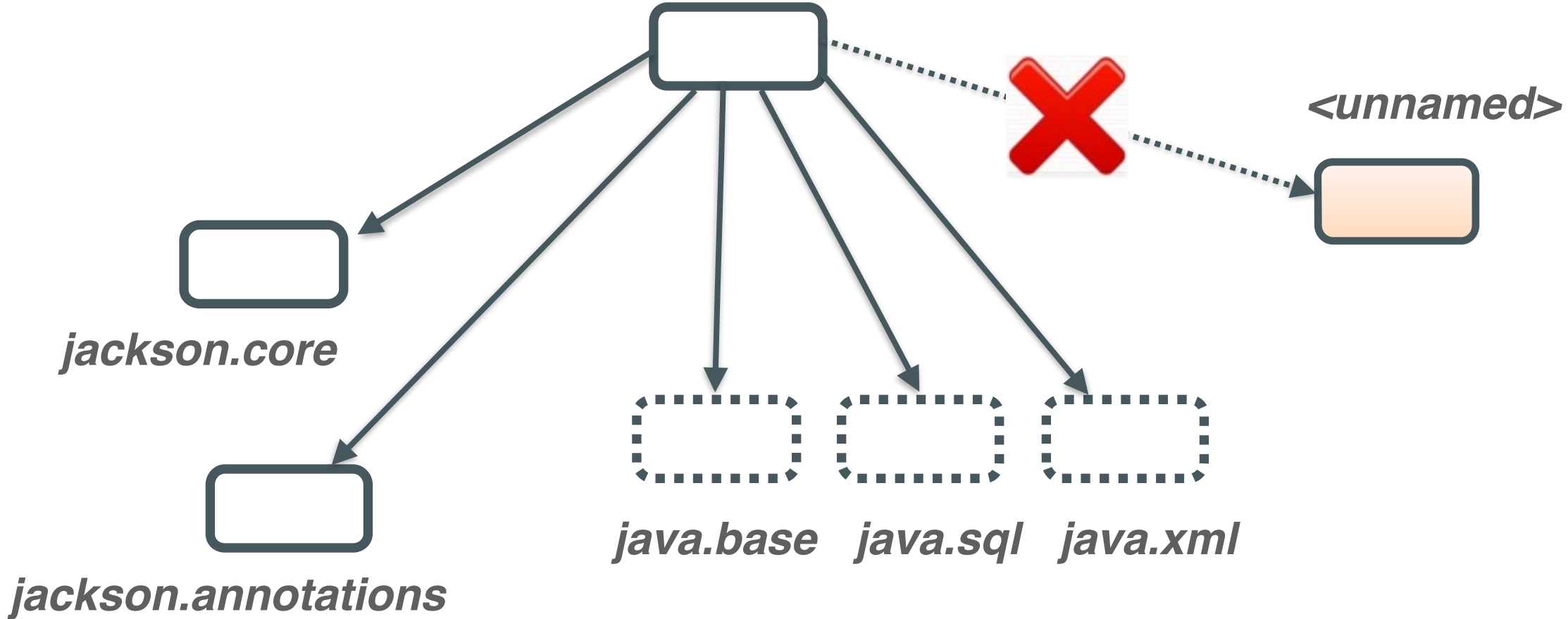


```
$ java -mp mlib -cp lib/myapp.jar:lib/mylib.jar \  
-addmods jackson.databind myapp.Main
```

**Exception in thread "main" java.lang.reflect.InaccessibleObjectException: Unable to make member of class app.MyValue accessible: module jackson.databind does not read <unnamed module @6536e911>**

```
at sun.reflect.Reflection.throwInaccessibleObjectException(java.base@9.0/Reflection.java:464)  
at java.lang.reflect.AccessibleObject.checkCanSetAccessible(java.base@9.0/AccessibleObject.java:175)  
at java.lang.reflect.Constructor.checkCanSetAccessible(java.base@9.0/Constructor.java:174)  
at java.lang.reflect.Constructor.setAccessible(java.base@9.0/Constructor.java:167)  
at com.fasterxml.jackson.databind.util.ClassUtil.checkAndFixAccess(jackson.databind/ClassUtil.java:505)  
at com.fasterxml.jackson.databind.deser.impl.CreatorCollector._fixAccess(jackson.databind/  
CreatorCollector.java:280)  
at com.fasterxml.jackson.databind.deser.impl.CreatorCollector.setDefaultCreator(jackson.databind/  
CreatorCollector.java:155)  
:
```

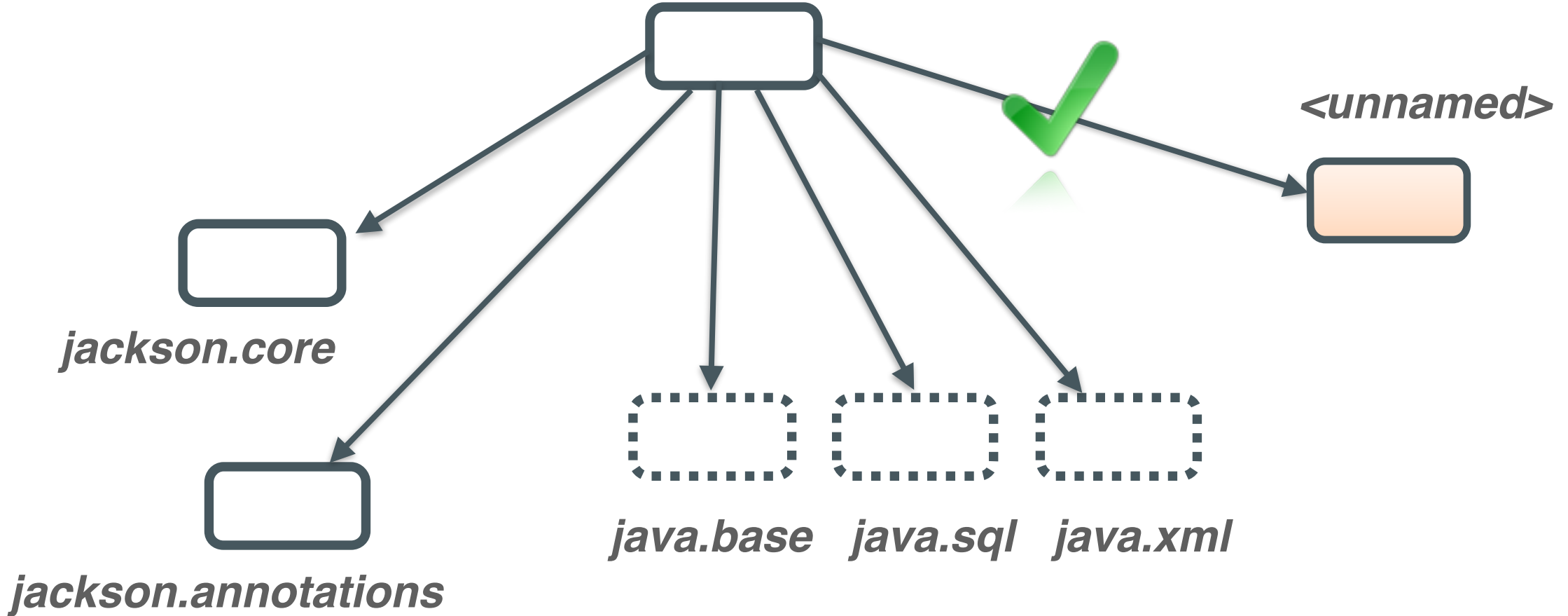
*jackson.databind*



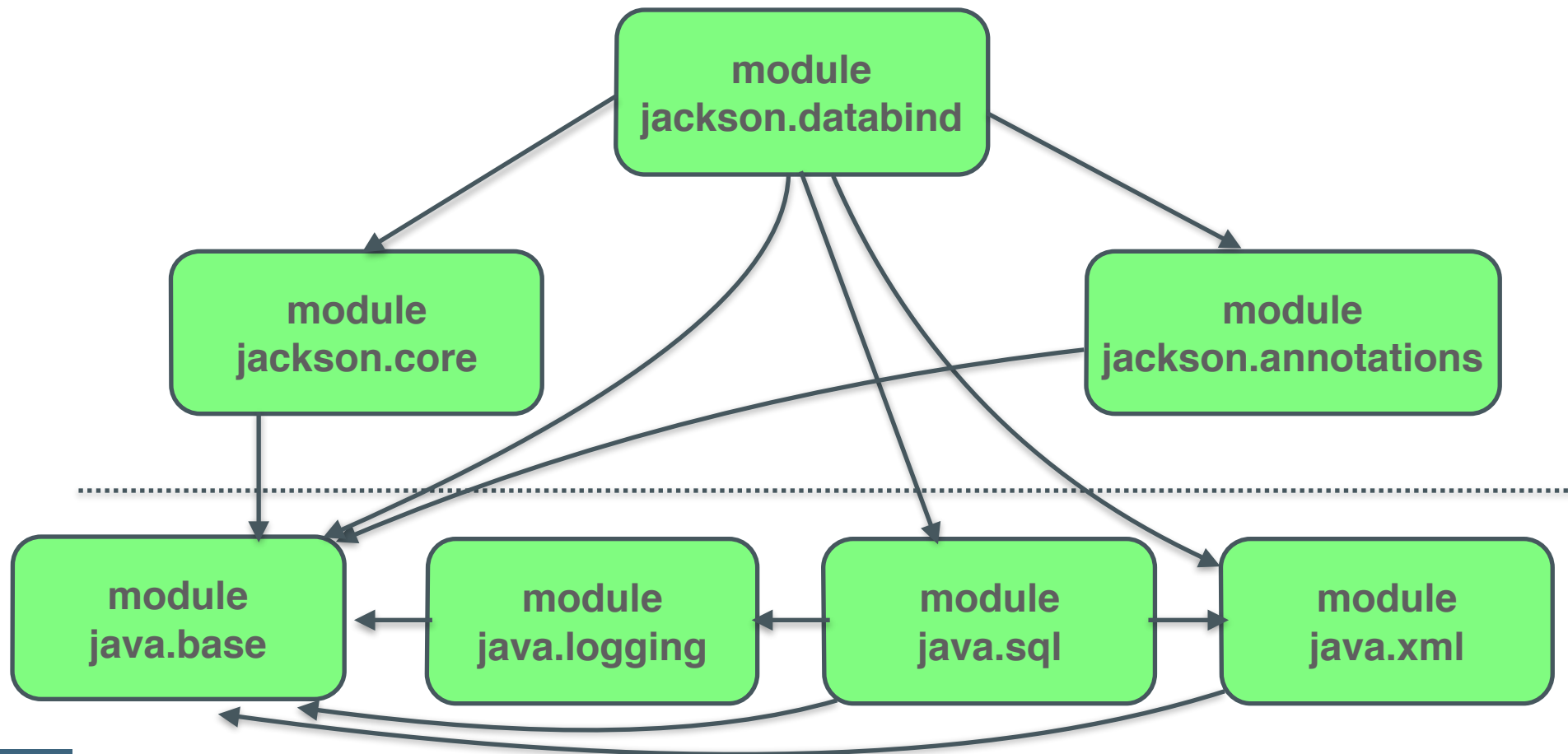
```
package java.lang.reflect;  
  
public final class Module {  
  
    public String getName();  
  
    public boolean canRead(Module source);  
    public Module addReads(Module source);  
  
    ...  
  
}
```

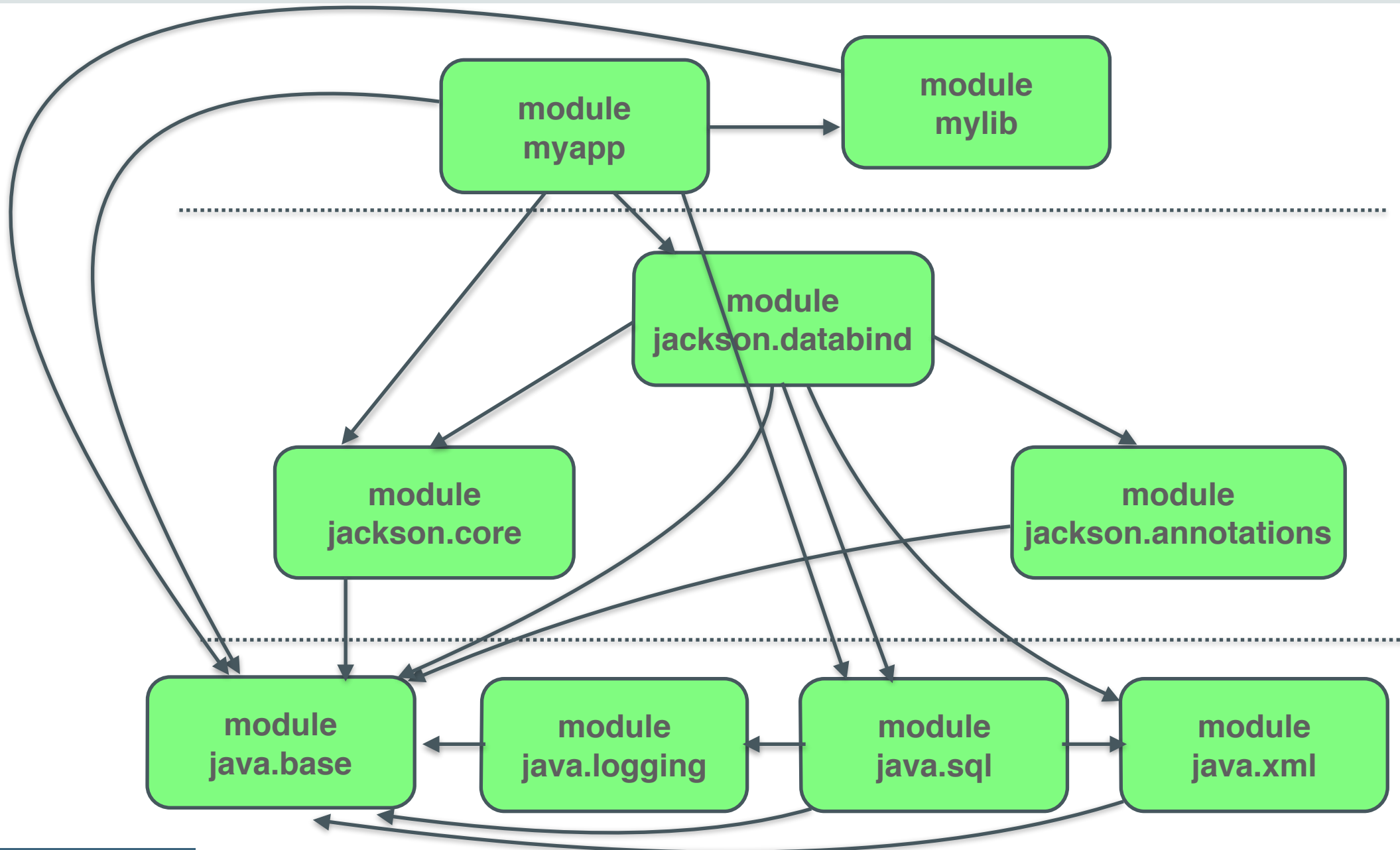
```
Module source = clazz.getModule();  
this.getClass().getModule().addReads(source);
```

*jackson.databind*



# Putting it all together







```
$ java -mp mlib -m myapp
```

# Linking

```
$ jlink -mp $JDKMODS:mlib -addmods myapp -output myimage
```

```
$ ls myimage
```

```
bin    conf    lib     release
```

```
$ ls myimage/bin
```

```
myapp  java    keytool
```

```
$ myimage/bin/java -listmods
```

```
myapp
```

```
mylib
```

```
jackson.annotations@2.6.0
```

```
jackson.core@2.6.3
```

```
jackson.databind@2.6.3
```

```
java.base@9.0
```

```
java.logging@9.0
```

```
java.sql@9.0
```

```
java.xml@9.0
```

```
$ 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
- Some libraries will need changes to work as modules

# Go forth and modularize!

# Other sessions, this room

- Project Jigsaw: Under the Hood @ 5.30pm
- Project Jigsaw Hack Session @ Tuesday 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

The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.



JavaOne™

ORACLE®