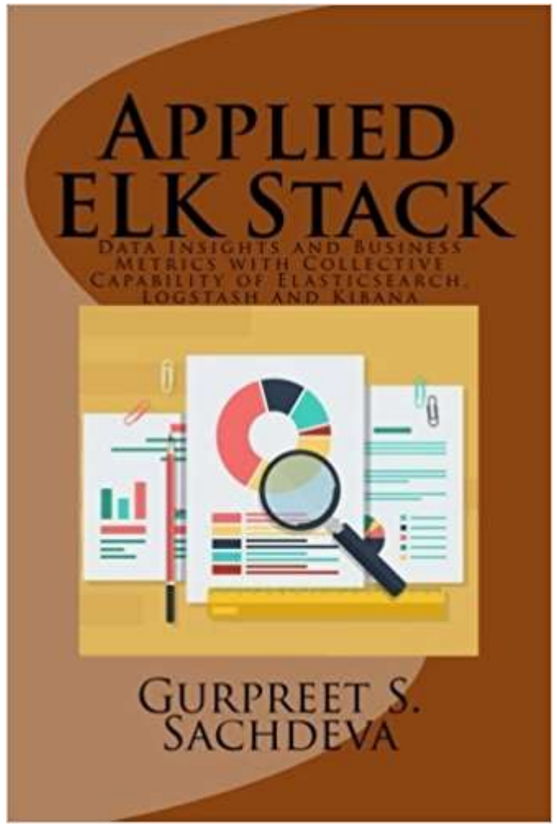
# Java 9 Modularity

Gurpreet S. Sachdeva
AVP - Technology
Aricent, Gurgaon (India)

## Who **AM I?**

- DevOps / Cloud Technologies enthusiast
- Assistant Vice President – Technology @ Aricent, Gurgaon
- Co-Founder Delhi/NCR JUG
- I blog @ www.thistechnologylife.com
- Author of Applied ELK Stack
- @ gssachdeva
- https://www.linkedin.com/in/gurpreets


APPLIED ELK STACK
DATA INSIGHTS AND BUSINESS METRICS WITH COLLECTIVE CAPABILITY OF ELASTICSEARCH, LOGSTASH AND KIBANA
GURPREET S. SACHDEVA

## Introduction

# Project Jigsaw & Modules

Make Java more Scalable & Flexible

Improve Security, Maintainability & Performance

Simplify Construction, Deployment & Maintenance of Large Scale applications

Strong Encapsulation – Hide Platform internals

## Specifications

### Java Platform Module System

- JSR 376: Part of JDK 9

### Java SE 9: Separate JSR

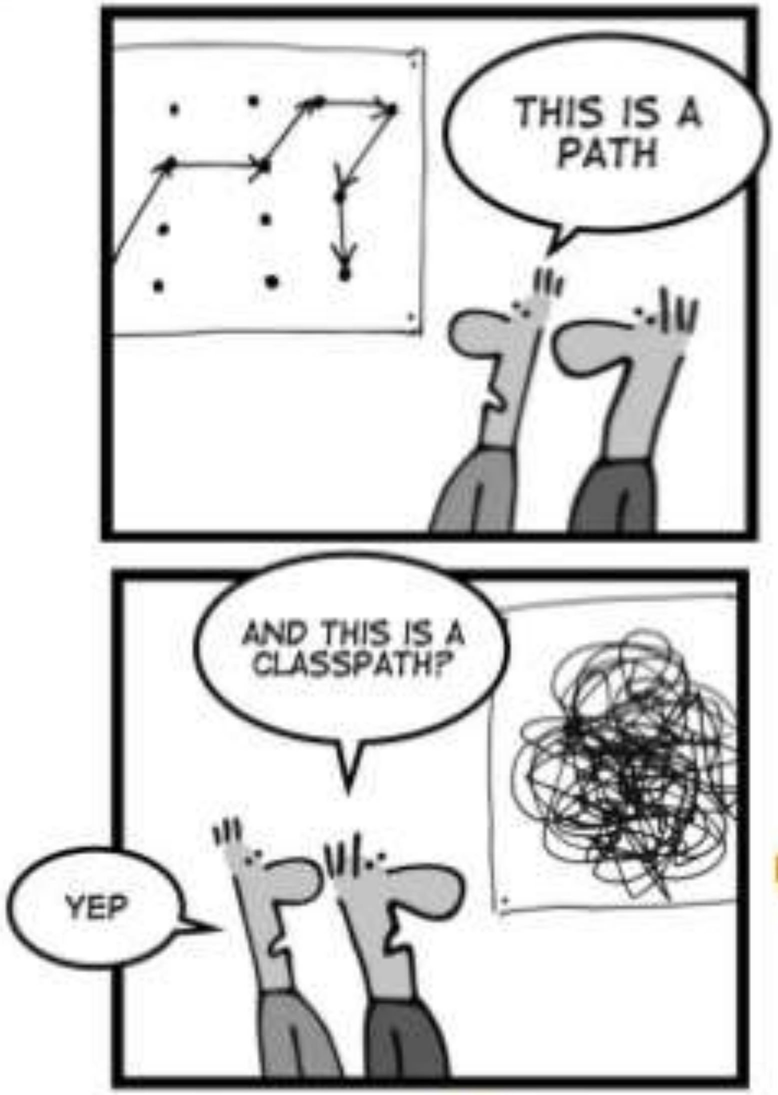
### OpenJDK Project Jigsaw

- Reference Implementation for JSR 376
- JEP 200: The modular JDK
- JEP 201: Modular Source Code
- JEP 220: Modular run-time images
- JEP 260: Encapsulate most internal APIs
- JEP 261: Module System

# Before Java 9

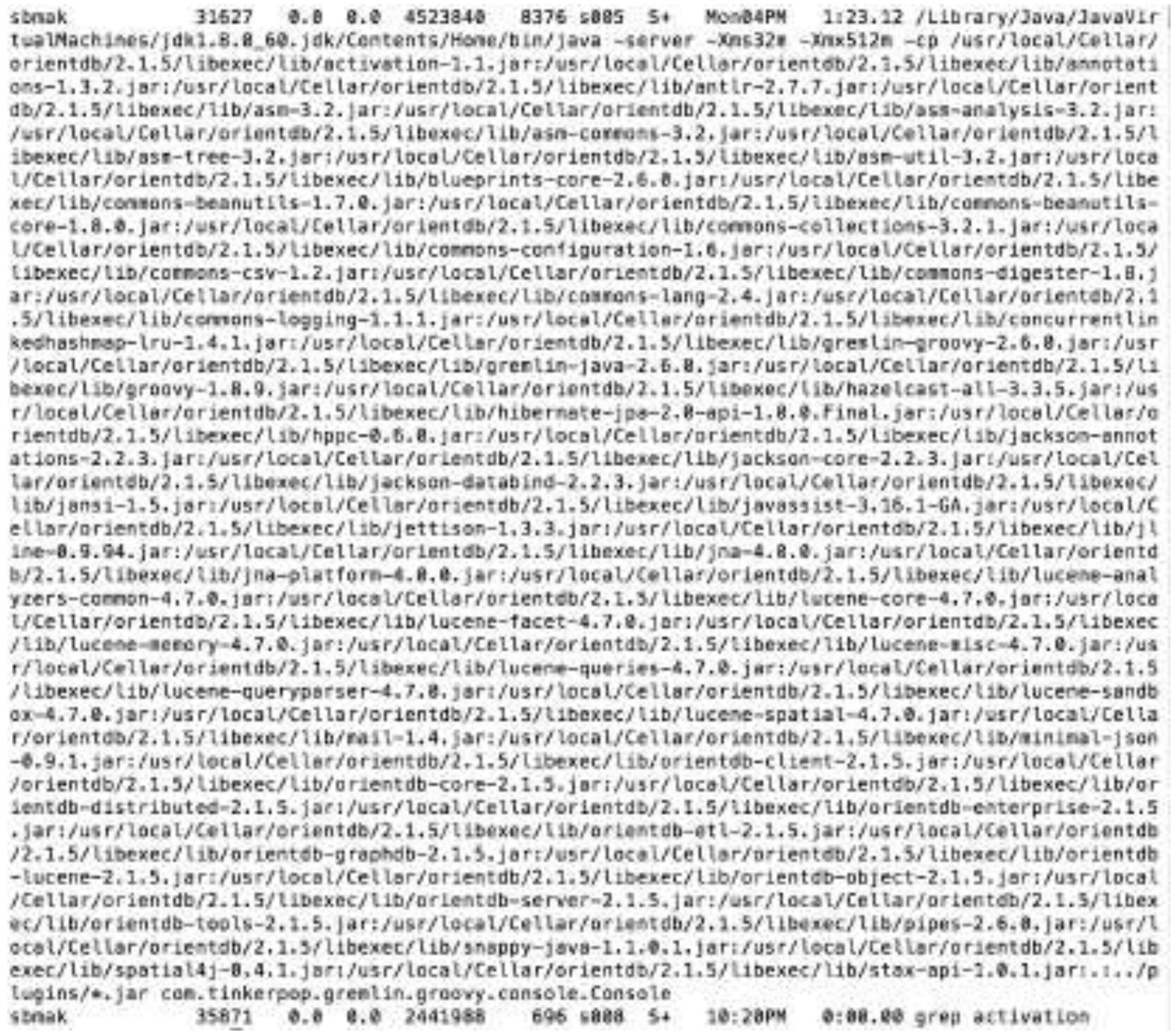
→Classpath

```
java.lang.Object
java.lang.String
...
sun.misc.BASE64Encoder
sun.misc.Unsafe
...
javax.crypto.Cypher
javax.crypto.SecretKey
...
com.myapp.Main
...
com.google.common.base.Joiner
...
com.google.common.base.internal.Joiner
```



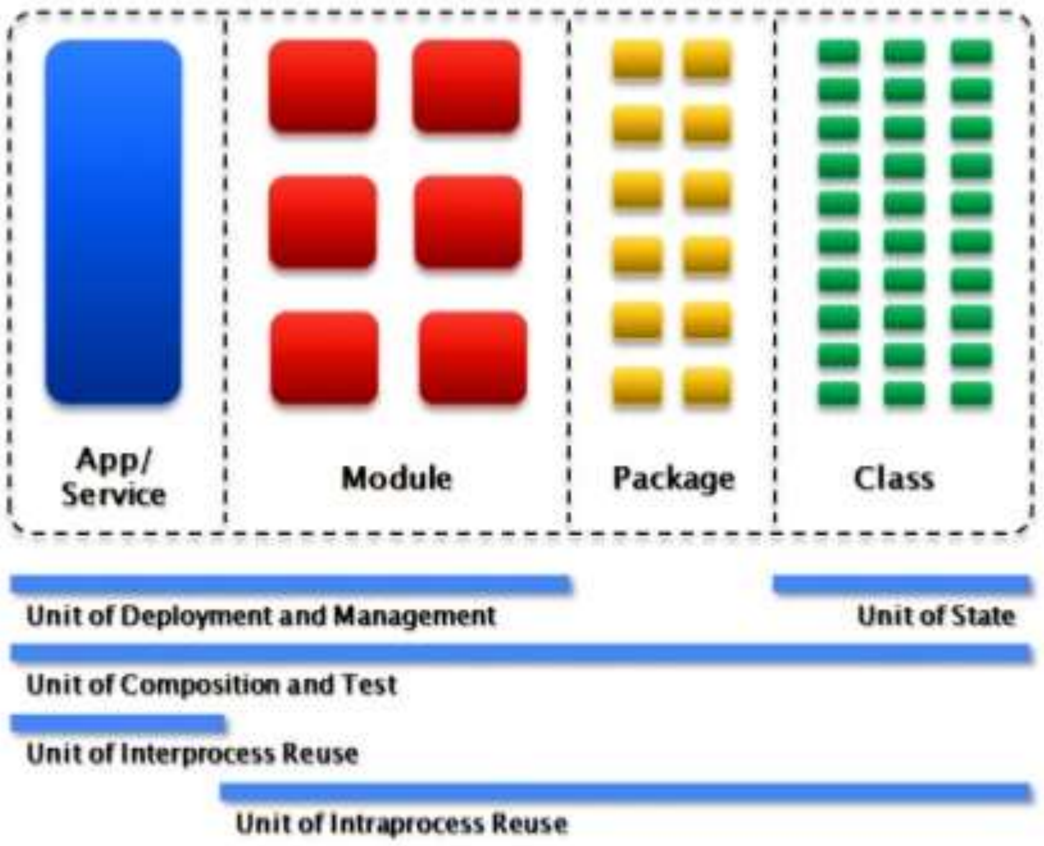
# Before Java 9

## Complex Classpaths

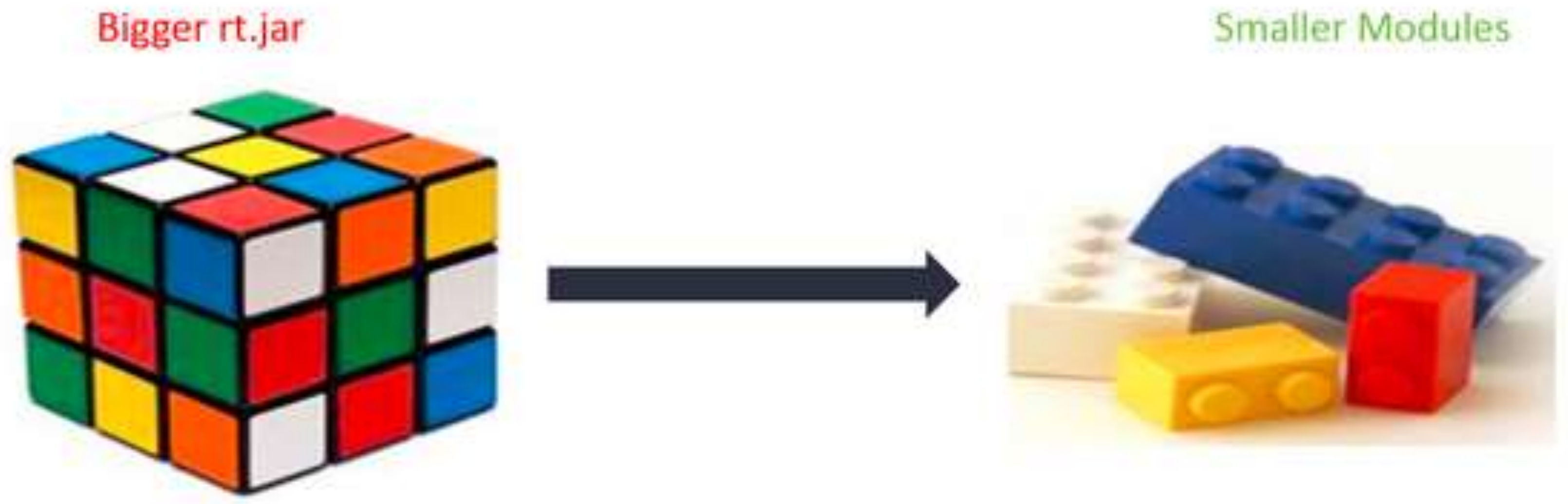


```
sbmak            31627   0.0  0.0  4523840   8376 s005  S+   Mon04PM   1:23.12 /Library/Java/JavaVir
tualMachines/jdk1.8.0_60.jdk/Contents/Home/bin/java -server -Xms32m -Xmx512m -cp /usr/local/Cellar/
orientdb/2.1.5/libexec/lib/activation-1.1.jar:/usr/local/Cellar/orientdb/2.1.5/libexec/lib/annotati
ons-1.3.2.jar:/usr/local/Cellar/orientdb/2.1.5/libexec/lib/antlr-2.7.7.jar:/usr/local/Cellar/orient
db/2.1.5/libexec/lib/asm-3.2.jar:/usr/local/Cellar/orientdb/2.1.5/libexec/lib/asm-analysis-3.2.jar:
/usr/local/Cellar/orientdb/2.1.5/libexec/lib/asm-commons-3.2.jar:/usr/local/Cellar/orientdb/2.1.5/l
ibexec/lib/asm-tree-3.2.jar:/usr/local/Cellar/orientdb/2.1.5/libexec/lib/asm-util-3.2.jar:/usr/loca
l/Cellar/orientdb/2.1.5/libexec/lib/blueprints-core-2.6.0.jar:/usr/local/Cellar/orientdb/2.1.5/libe
xec/lib/commons-beanutils-1.7.0.jar:/usr/local/Cellar/orientdb/2.1.5/libexec/lib/commons-beanutils-
core-1.8.0.jar:/usr/local/Cellar/orientdb/2.1.5/libexec/lib/commons-collections-3.2.1.jar:/usr/loca
l/Cellar/orientdb/2.1.5/libexec/lib/commons-configuration-1.6.jar:/usr/local/Cellar/orientdb/2.1.5/
libexec/lib/commons-csv-1.2.jar:/usr/local/Cellar/orientdb/2.1.5/libexec/lib/commons-digester-1.8.j
ar:/usr/local/Cellar/orientdb/2.1.5/libexec/lib/commons-lang-2.4.jar:/usr/local/Cellar/orientdb/2.1
.5/libexec/lib/commons-logging-1.1.1.jar:/usr/local/Cellar/orientdb/2.1.5/libexec/lib/concurrentlin
kedhashmap-lru-1.4.1.jar:/usr/local/Cellar/orientdb/2.1.5/libexec/lib/gremlin-groovy-2.6.0.jar:/usr
/local/Cellar/orientdb/2.1.5/libexec/lib/gremlin-java-2.6.0.jar:/usr/local/Cellar/orientdb/2.1.5/li
bexec/lib/groovy-1.8.9.jar:/usr/local/Cellar/orientdb/2.1.5/libexec/lib/hazelcast-all-3.3.5.jar:/us
r/local/Cellar/orientdb/2.1.5/libexec/lib/hibernate-jpa-2.0-api-1.0.0.Final.jar:/usr/local/Cellar/o
rientdb/2.1.5/libexec/lib/hppc-0.6.0.jar:/usr/local/Cellar/orientdb/2.1.5/libexec/lib/jackson-annot
ations-2.2.3.jar:/usr/local/Cellar/orientdb/2.1.5/libexec/lib/jackson-core-2.2.3.jar:/usr/local/Cel
lar/orientdb/2.1.5/libexec/lib/jackson-databind-2.2.3.jar:/usr/local/Cellar/orientdb/2.1.5/libexec/
lib/jansi-1.5.jar:/usr/local/Cellar/orientdb/2.1.5/libexec/lib/javassist-3.16.1-GA.jar:/usr/local/C
ellar/orientdb/2.1.5/libexec/lib/jettison-1.3.3.jar:/usr/local/Cellar/orientdb/2.1.5/libexec/lib/jl
ine-0.9.94.jar:/usr/local/Cellar/orientdb/2.1.5/libexec/lib/jna-4.0.0.jar:/usr/local/Cellar/orientd
b/2.1.5/libexec/lib/jna-platform-4.0.0.jar:/usr/local/Cellar/orientdb/2.1.5/libexec/lib/lucene-anal
yzers-common-4.7.0.jar:/usr/local/Cellar/orientdb/2.1.5/libexec/lib/lucene-core-4.7.0.jar:/usr/loca
l/Cellar/orientdb/2.1.5/libexec/lib/lucene-facet-4.7.0.jar:/usr/local/Cellar/orientdb/2.1.5/libexec
/lib/lucene-memory-4.7.0.jar:/usr/local/Cellar/orientdb/2.1.5/libexec/lib/lucene-misc-4.7.0.jar:/us
r/local/Cellar/orientdb/2.1.5/libexec/lib/lucene-queries-4.7.0.jar:/usr/local/Cellar/orientdb/2.1.5
/libexec/lib/lucene-queryparser-4.7.0.jar:/usr/local/Cellar/orientdb/2.1.5/libexec/lib/lucene-sandb
ox-4.7.0.jar:/usr/local/Cellar/orientdb/2.1.5/libexec/lib/lucene-spatial-4.7.0.jar:/usr/local/Cella
r/orientdb/2.1.5/libexec/lib/mail-1.4.jar:/usr/local/Cellar/orientdb/2.1.5/libexec/lib/minimal-json
-0.9.1.jar:/usr/local/Cellar/orientdb/2.1.5/libexec/lib/orientdb-client-2.1.5.jar:/usr/local/Cellar
/orientdb/2.1.5/libexec/lib/orientdb-core-2.1.5.jar:/usr/local/Cellar/orientdb/2.1.5/libexec/lib/or
ientdb-distributed-2.1.5.jar:/usr/local/Cellar/orientdb/2.1.5/libexec/lib/orientdb-enterprise-2.1.5
.jar:/usr/local/Cellar/orientdb/2.1.5/libexec/lib/orientdb-etl-2.1.5.jar:/usr/local/Cellar/orientdb
/2.1.5/libexec/lib/orientdb-graphdb-2.1.5.jar:/usr/local/Cellar/orientdb/2.1.5/libexec/lib/orientdb
-lucene-2.1.5.jar:/usr/local/Cellar/orientdb/2.1.5/libexec/lib/orientdb-object-2.1.5.jar:/usr/local
/Cellar/orientdb/2.1.5/libexec/lib/orientdb-server-2.1.5.jar:/usr/local/Cellar/orientdb/2.1.5/libex
ec/lib/orientdb-tools-2.1.5.jar:/usr/local/Cellar/orientdb/2.1.5/libexec/lib/pipes-2.6.0.jar:/usr/l
ocal/Cellar/orientdb/2.1.5/libexec/lib/snappy-java-1.1.0.1.jar:/usr/local/Cellar/orientdb/2.1.5/lib
exec/lib/spatial4j-0.4.1.jar:/usr/local/Cellar/orientdb/2.1.5/libexec/lib/stax-api-1.0.1.jar:.:../p
lugins/*.jar com.tinkerpop.gremlin.groovy.console.Console
sbmak            35871   0.0  0.0  2441988    696 s008  S+   10:20PM   0:00.00 grep activation
```

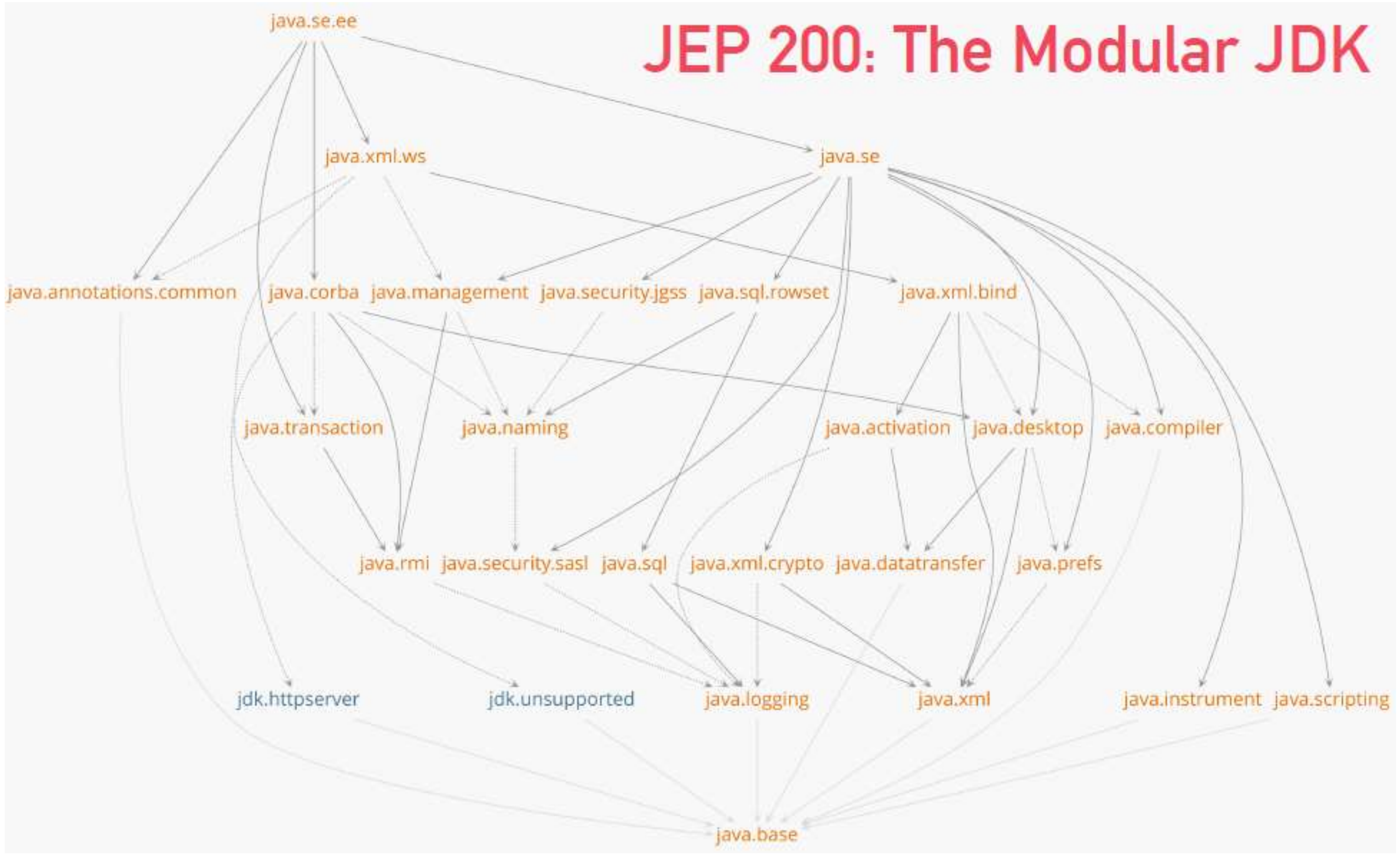
## Enter Modularity


App/ Service
Module
Package
Class
Unit of Deployment and Management
Unit of State
Unit of Composition and Test
Unit of Interprocess Reuse
Unit of Intraprocess Reuse

# Enter Modularity



# Enter Modularity

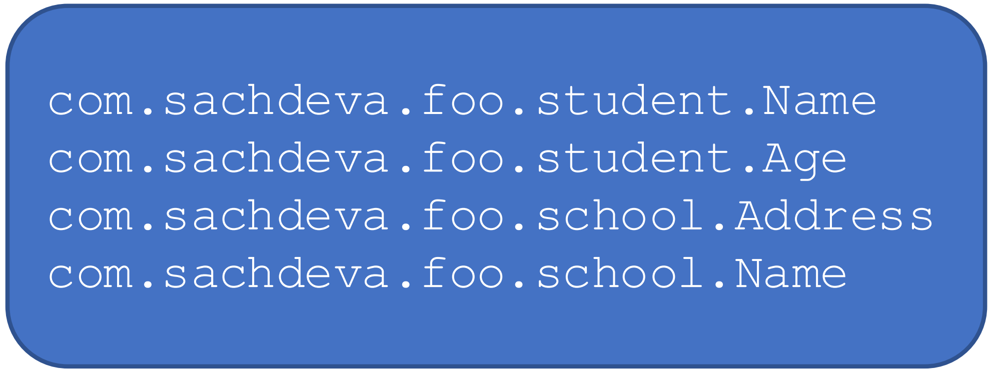


## OSGi was doing fine …

- OSGi built on top of JVM
  - Can’t be used to modularize the JVM itself
  - OSGi has never seen mass adoption
  - Java 9 aims to make modularity popular
- OSGi supports split packages, using Require-Bundle
  - But there is an assumption of single classloader
  - Package-private accessibility requires whole packages in a single classloader
  - We want packages split across modules and not across classloaders

# Module Fundamentals

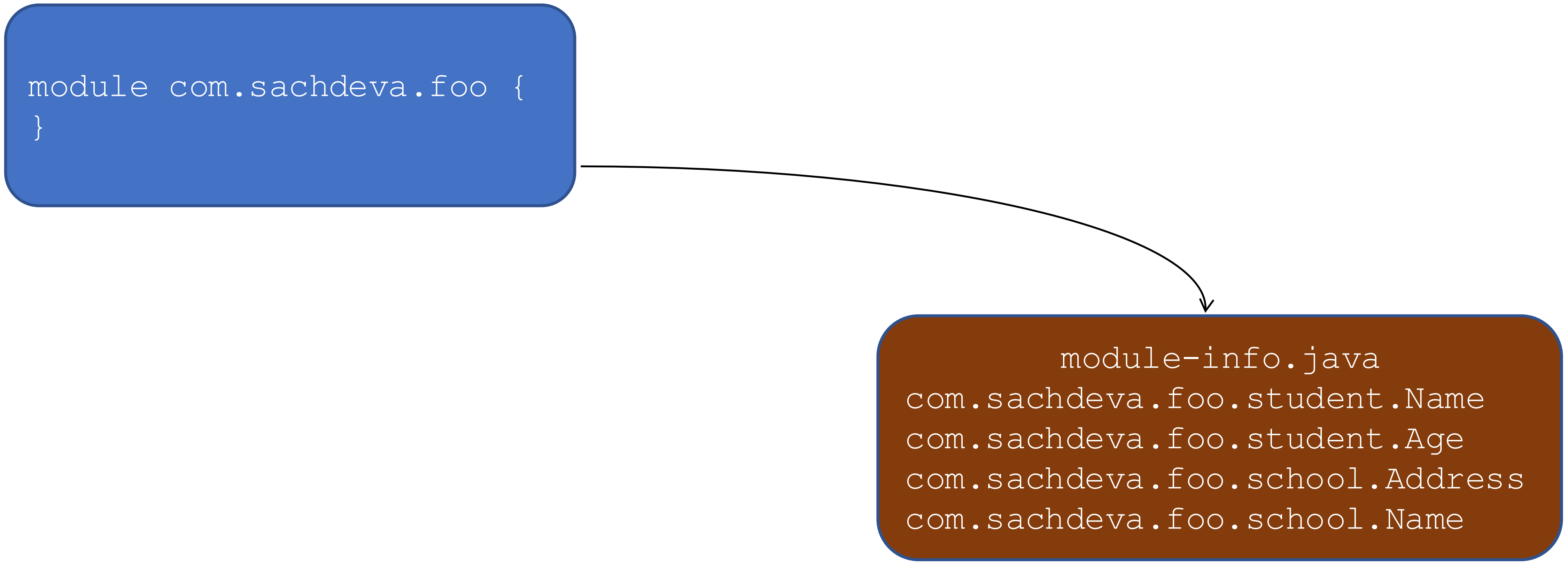
- Module is a grouping of code
  - For Java this is a collection of packages
- Module can contain other things
  - Native code
  - Resources
  - Configuration Data



```
com.sachdeva.foo.student.Name
com.sachdeva.foo.student.Age
com.sachdeva.foo.school.Address
com.sachdeva.foo.school.Name
```

`com.sachdeva.foo`

# Module Declaration



```
module com.sachdeva.foo {
}
```

```
        module-info.java
com.sachdeva.foo.student.Name
com.sachdeva.foo.student.Age
com.sachdeva.foo.school.Address
com.sachdeva.foo.school.Name
```

## Module Descriptor

Name

Dependencies

Public Packages

Services Offered

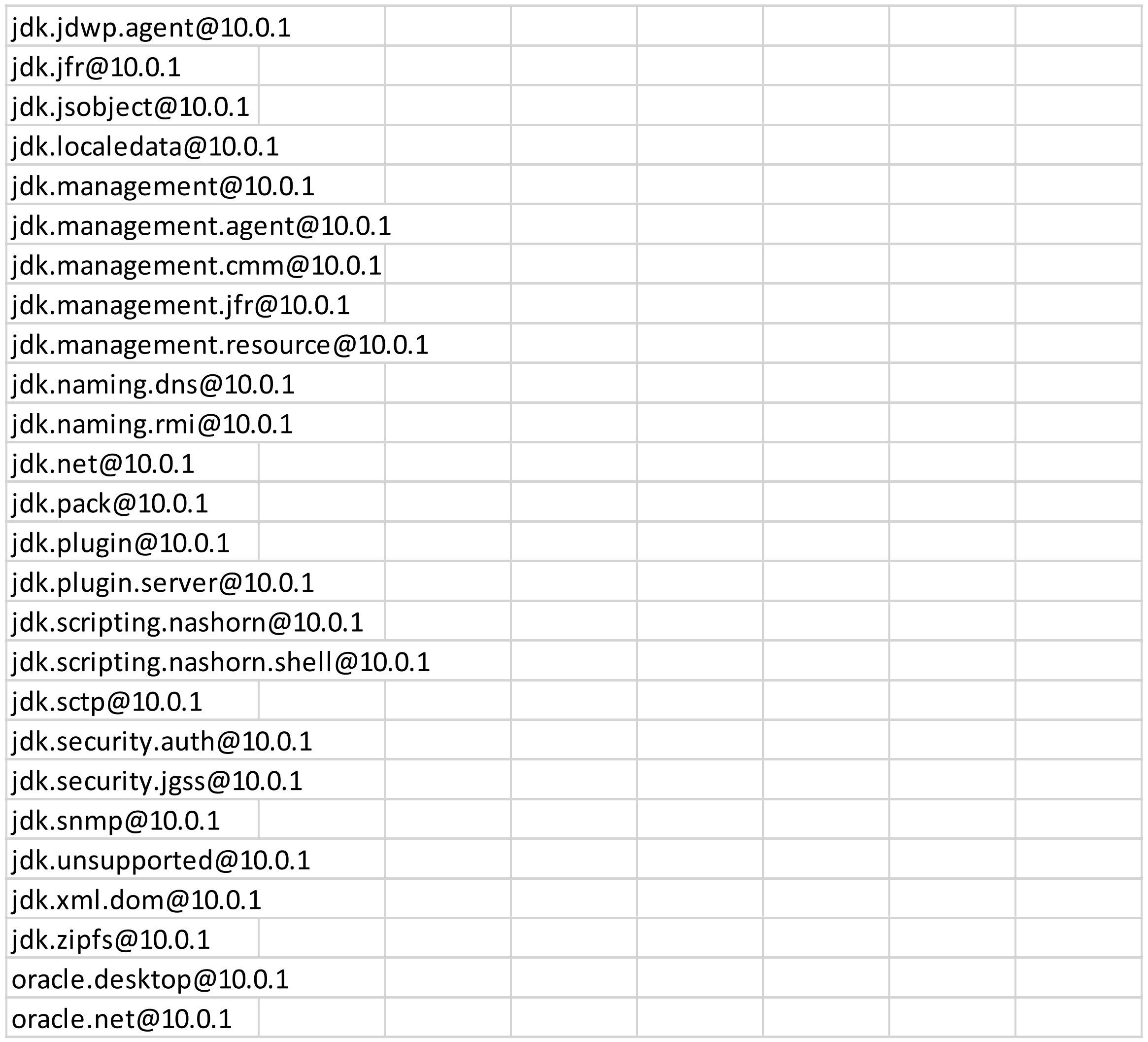
Services Consumed

Reflection Permissions

# Module Types

- System Modules – Java SE and JDK modules.
- Application Modules – User defined modules, named and defined in the compiled module-info.class file included in the assembled JAR
- Automatic Modules – Unofficial modules included by adding existing JAR files to the module path
- Unnamed Module – Class or JAR loaded onto the classpath, but not the module path

```
java --list-modules
```

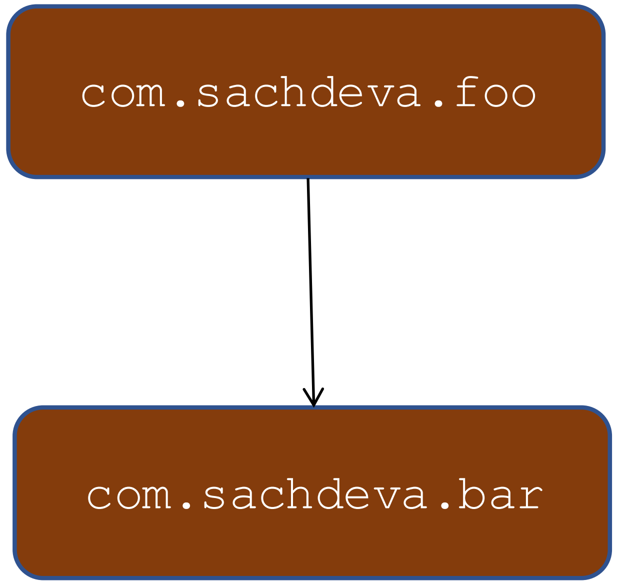


```
jdk.jdwp.agent@10.0.1
jdk.jfr@10.0.1
jdk.jsobject@10.0.1
jdk.localedata@10.0.1
jdk.management@10.0.1
jdk.management.agent@10.0.1
jdk.management.cmm@10.0.1
jdk.management.jfr@10.0.1
jdk.management.resource@10.0.1
jdk.naming.dns@10.0.1
jdk.naming.rmi@10.0.1
jdk.net@10.0.1
jdk.pack@10.0.1
jdk.plugin@10.0.1
jdk.plugin.server@10.0.1
jdk.scripting.nashorn@10.0.1
jdk.scripting.nashorn.shell@10.0.1
jdk.sctp@10.0.1
jdk.security.auth@10.0.1
jdk.security.jgss@10.0.1
jdk.snmp@10.0.1
jdk.unsupported@10.0.1
jdk.xml.dom@10.0.1
jdk.zipfs@10.0.1
oracle.desktop@10.0.1
oracle.net@10.0.1
```

Lists 78 modules for Java 10.0.1 on Windows

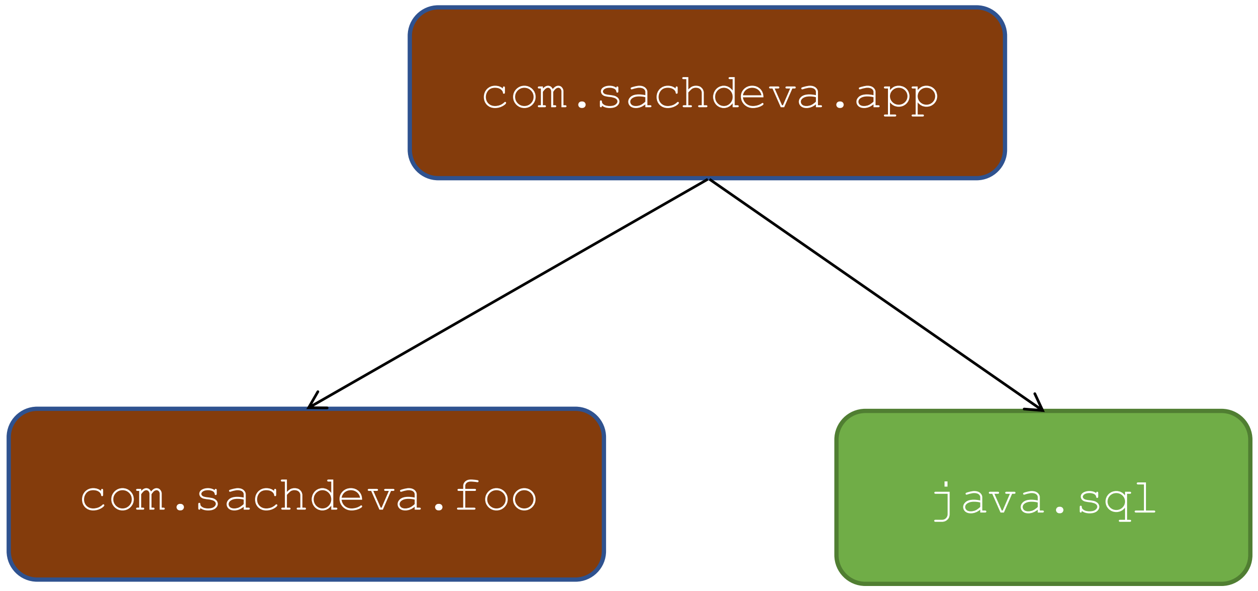
## Module Dependencies

```
module com.sachdeva.foo {
    requires com.sachdeva.bar;
}
```

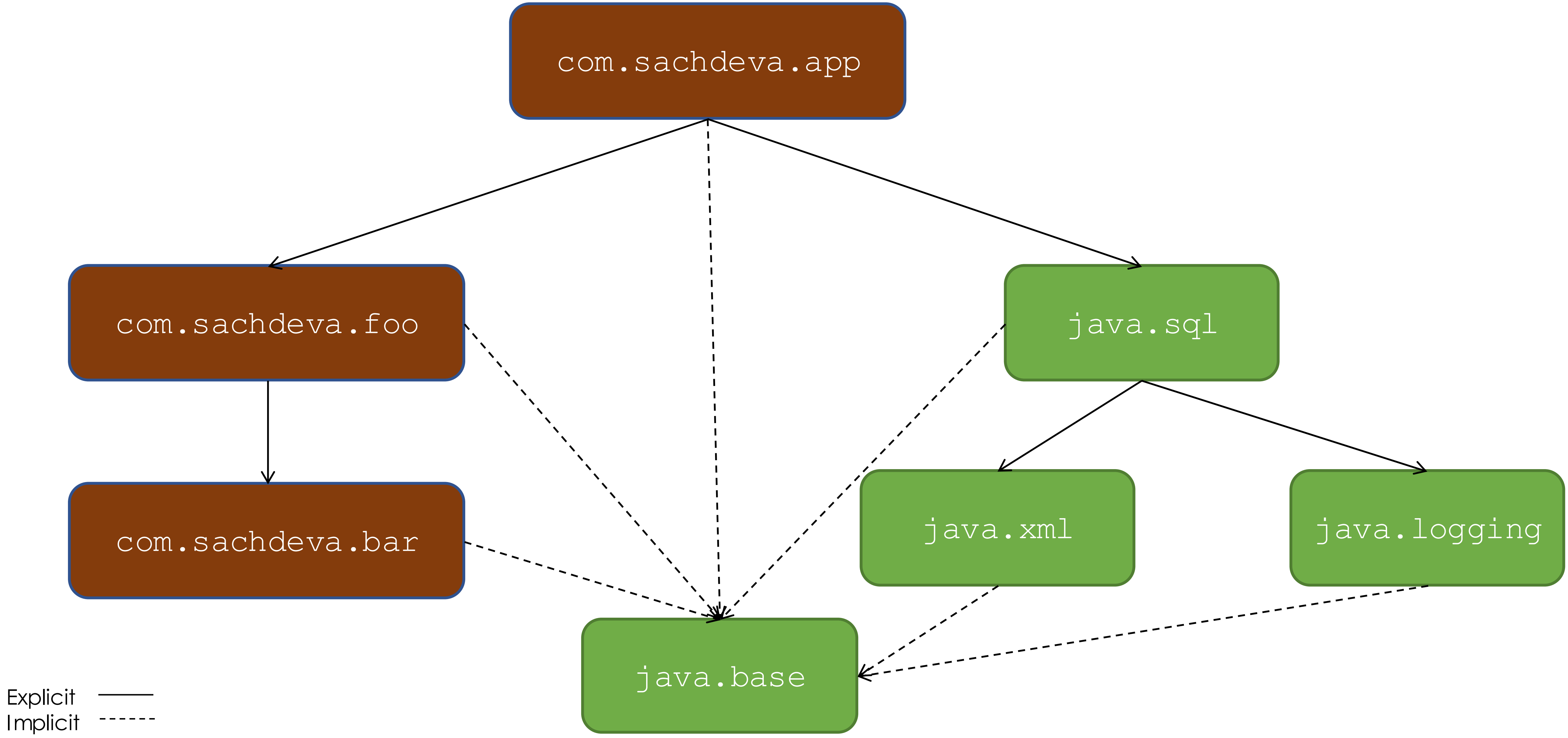


## Module Dependencies

```
module com.sachdeva.app {
    requires com.sachdeva.foo;
    requires java.sql;
}
```

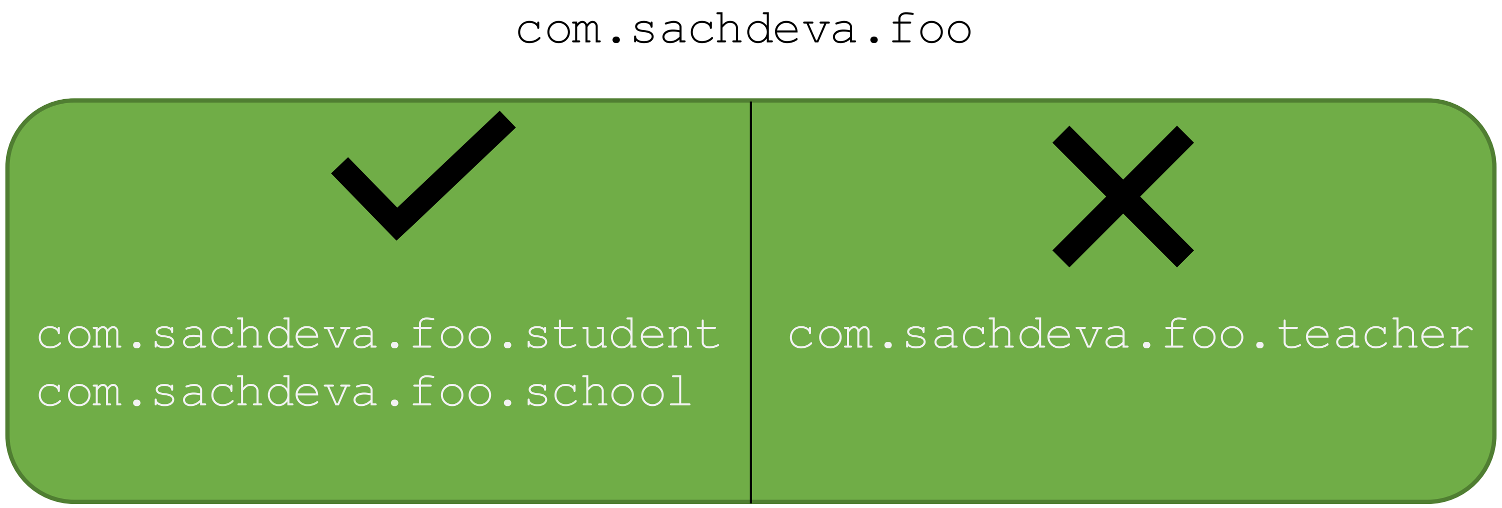

$$\begin{tikzcd}
 & \text{com.sachdeva.app} \arrow[dl] \arrow[dr] & \\
\text{com.sachdeva.foo} & & \text{java.sql}
\end{tikzcd}$$

# Module Dependency Graph



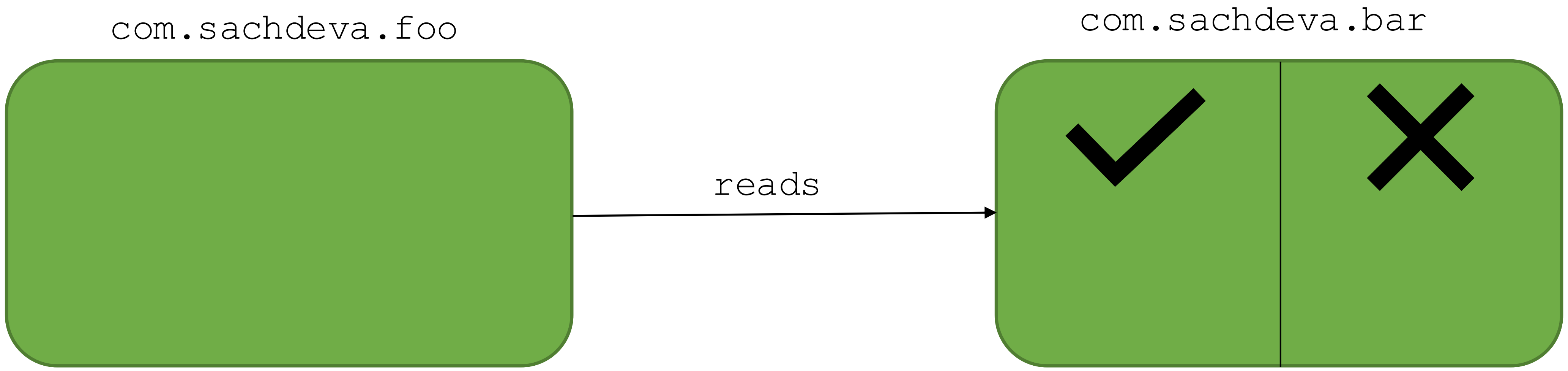
# Package Visibility

```
module com.sachdeva.foo {
    requires com.sachdeva.bar;
    exports com.sachdeva.foo.student;
    exports com.sachdeva.foo.school;
}
```



# Package Accessibility

- For a Package to be Visible
  - Package must be exported by the containing module
  - Containing module must be read by the using module
- Public types from those packages can then be used



# Java Accessibility

- Pre - JDK 9
  - public
  - protected
  - <package>
  - private

- JDK 9
  - public to everyone
  - public, but only to specific modules
  - public only within a module
  - protected
  - <package>
  - private

Public **IS NOT** accessible, by **default**
(fundamental change to Java)

## How to code with modules

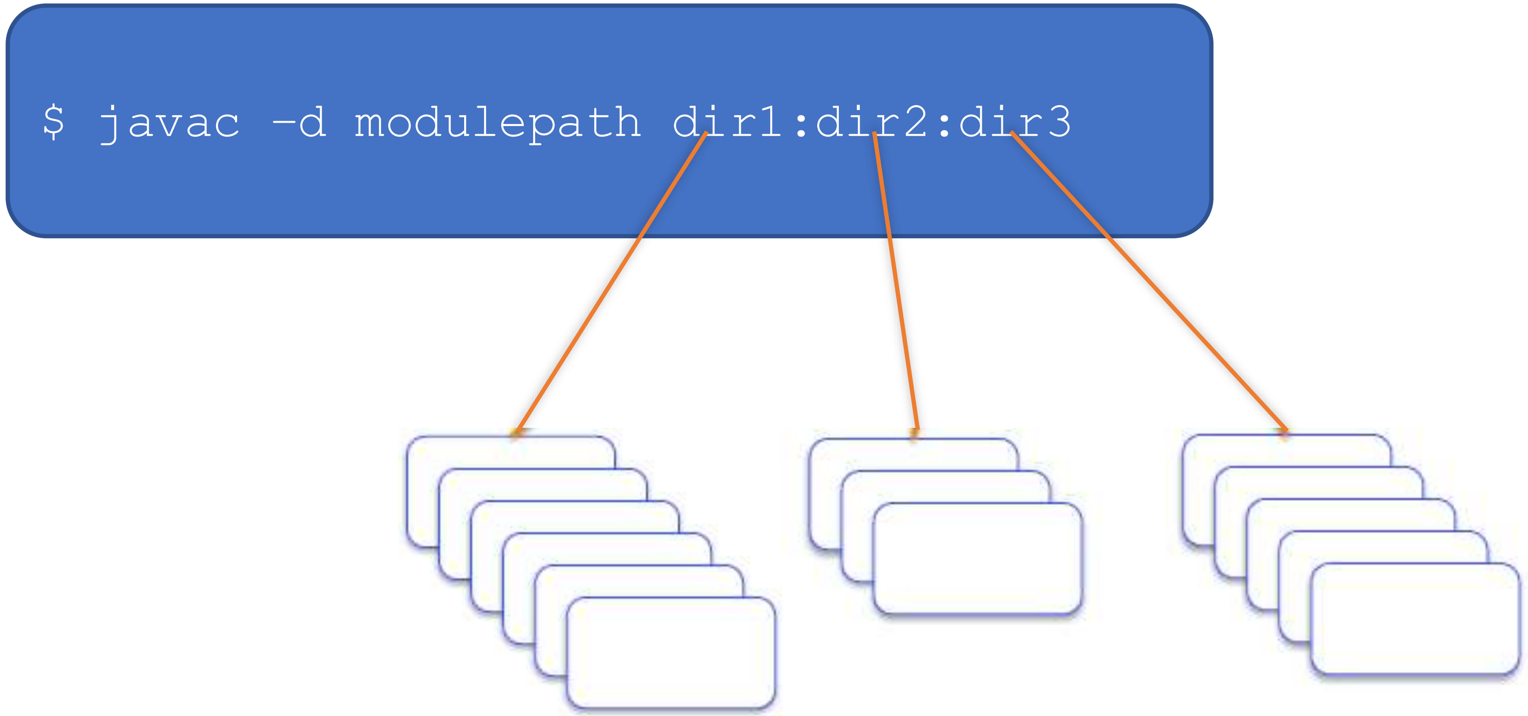
# Compilation

```
$ javac -d mods \
  src/foo/module-info.java \
  src/foo/com/sachdeva/foo/Student.java
```

```
src/foo/module-info.java
src/foo/com/sachdeva/foo/Student.java
```

```
mods/foo/module-info.class
mods/foo/com/sachdeva/foo/Student.class
```

## Compilation



```
$ javac -d modulepath dir1:dir2:dir3
```

# Compilation with Module Path

```
$ javac -modulepath mods -d mods \
  src/foo/module-info.java \
  src/foo/com/sachdeva/foo/Student.java
```

```
src/foo/module-info.java
src/foo/com/sachdeva/foo/Student.java
```

```
mods/foo/module-info.class
mods/foo/com/sachdeva/foo/Student.class
```

# Application Execution

module name          main class

**Application initialized**

```
$ javac -mp mods -m com.sachdeva.app/com.sachdeva.app.Main
```

**Diagnostics**

```
$ java -Xdiag:resolver -mp mods -m com.sachdeva.app/com.sachdeva.app.Main
```

## Packaging

```
mods/foo/module-info.java.class
mods/foo/com/sachdeva/foo/app/Main.class
```

**app.jar**

```
module-info.class
com/sachdeva/foo/Main.class
```

```
$  jar --create -file myLib/app.jar \
  --main-class com.sachdeva.foo.Main \
  -C mods .
```

# Jar Files & Module Information

```
$  jar -file myLib/app.jar -p
Name:
    com.sachdeva.foo
Requires:
    com.sachdeva.bar
    java.base [MANDATED]
    java.sql
Main.class:
    com.sachdeva.foo.Main
```

## Application Execution (JAR)

```
$  jar -mp myLib:mods -m com.sachdeva.foo.Main
```

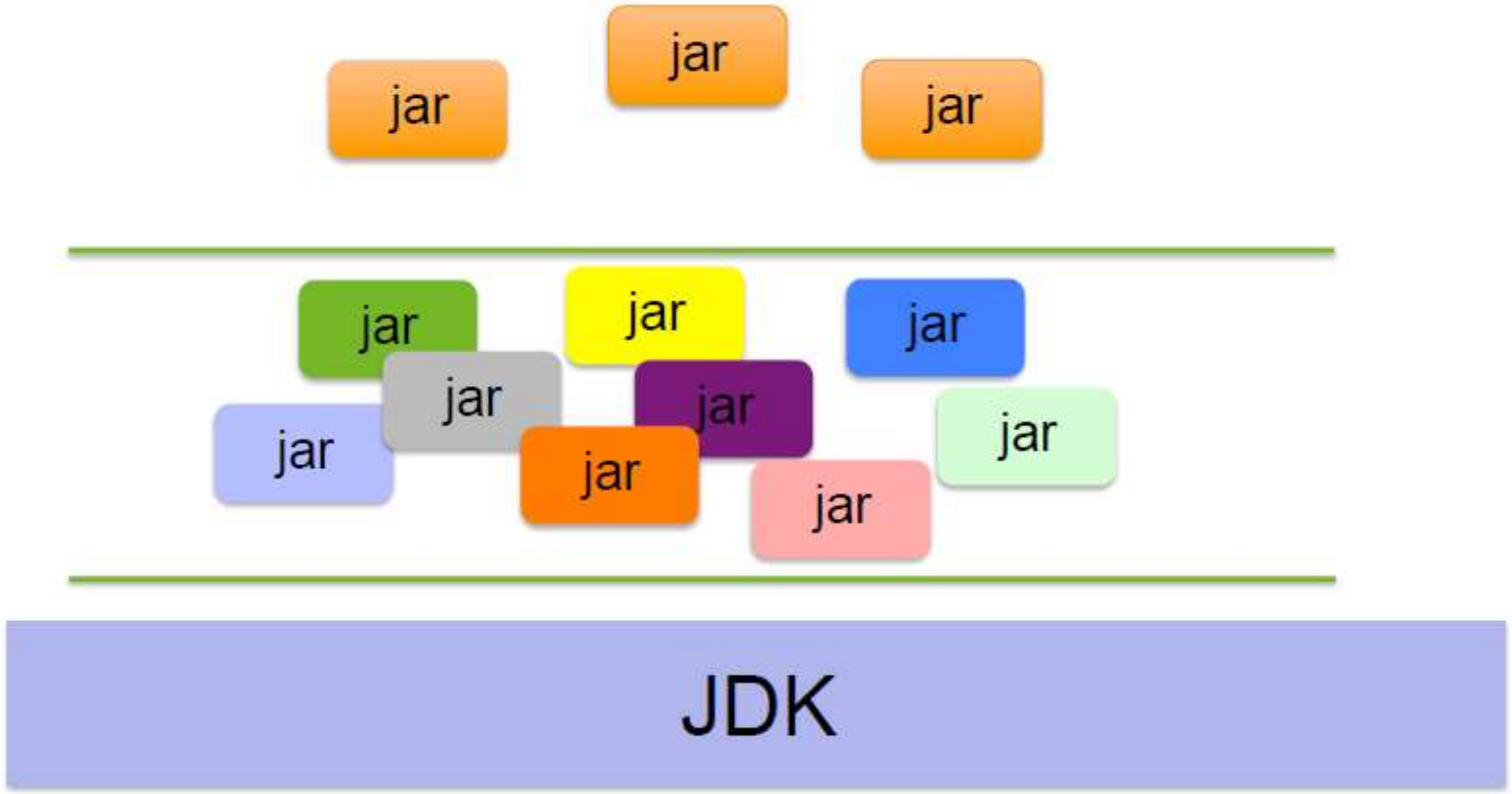
## Linking an Application

```
$  jlink --modulepath $JDKMODS:$MYMODS \
   --addmods com.sachdeva.app -output myimage
```

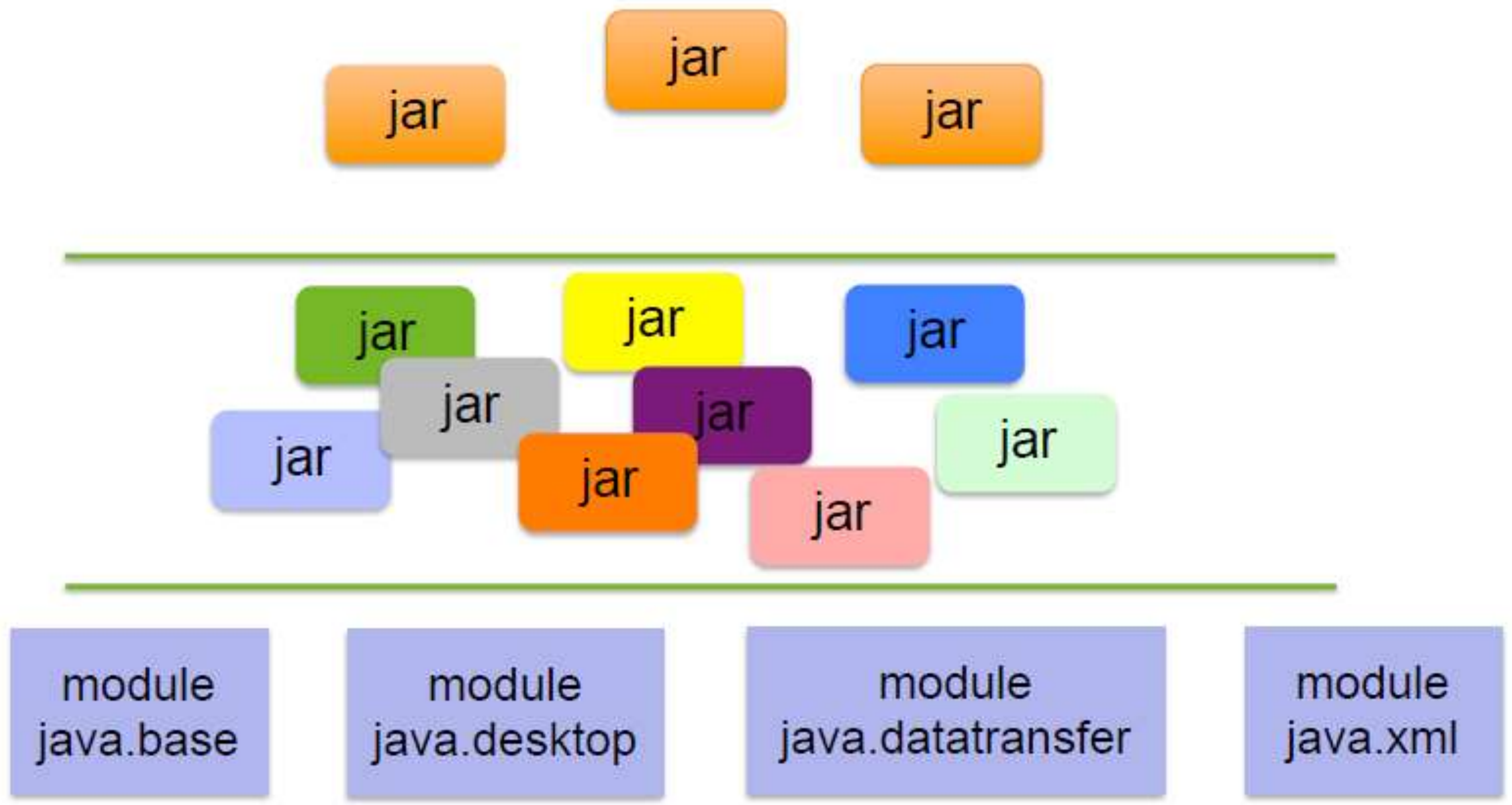
```
$  myimage/bin/java -listmods
java.base@9.0
java.logging@9.0
java.sql@9.0
java.xml@9.0
com.sachdeva.app@1.0
com.sachdeva.foo@1.0
com.sachdeva.bar@1.0
```

## Migrating Applications

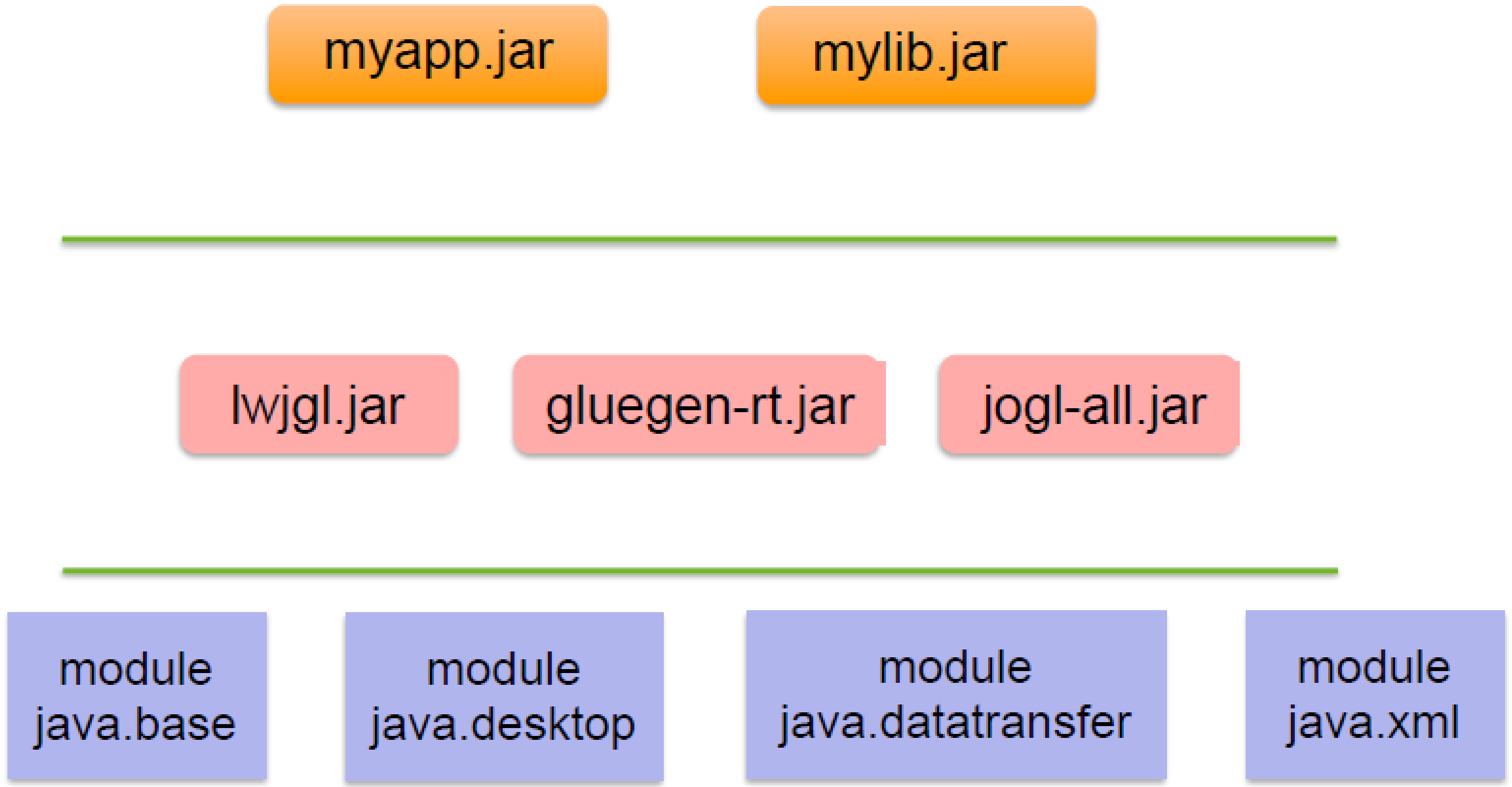
## Typical Application – Pre Java 9



# Typical Application –Java 9



## Sample Application



## Preparation

- Use jdeps to audit your code
- Escape hatch:

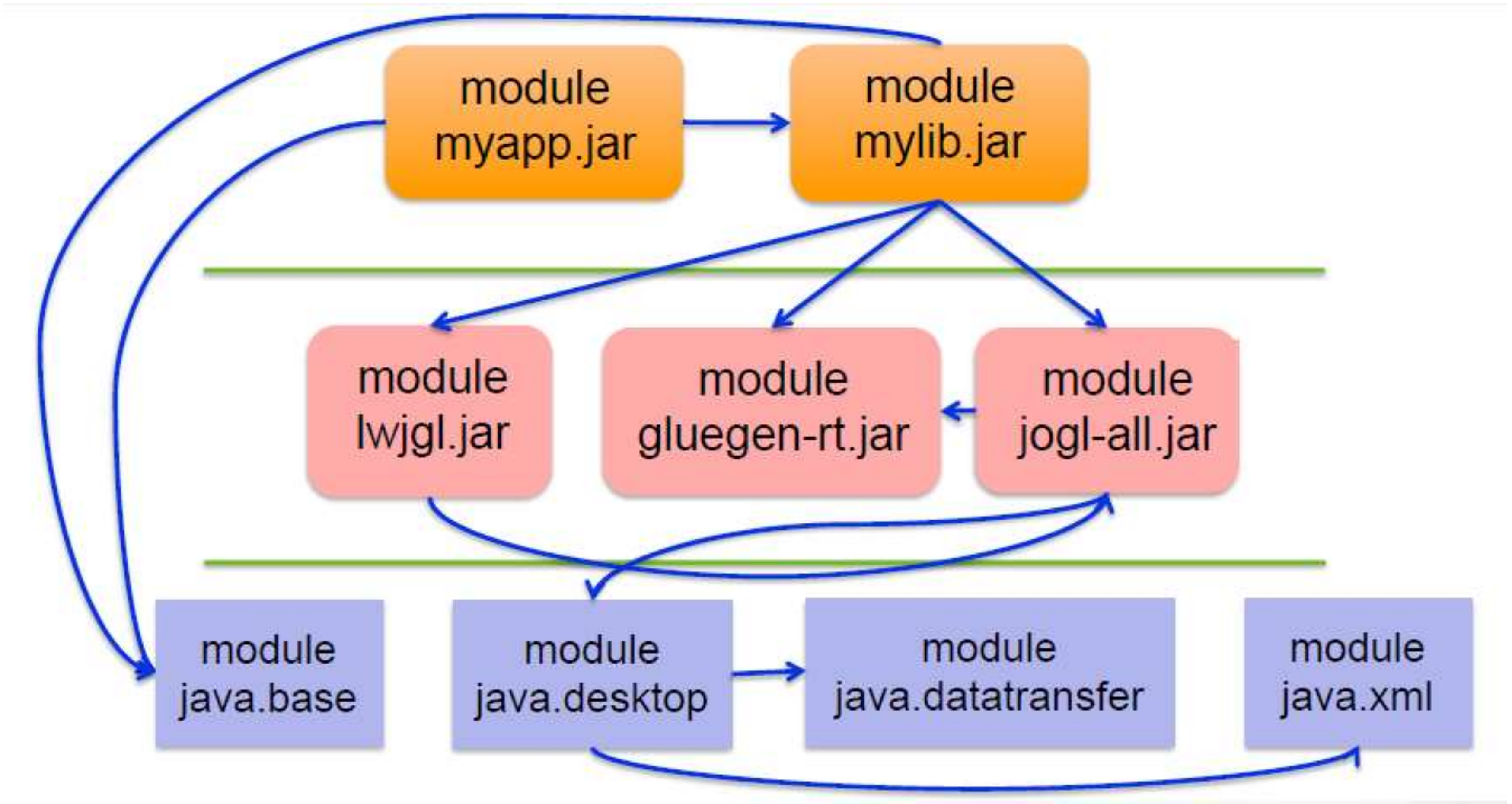
--add-exports java.base/javax.security.auth.x500=mymod

- Gradual Migration can be done
  - Mix claspath and modulepath
  - Automatic modules

## Application module-info.java

```
module myApp {
        requires mylib;
        requires java.base;
        requires java.sql;
        requires lwjgl;               ???
        requires gluegen-rt;          ???
        requires jogl-all;            ???
}
```

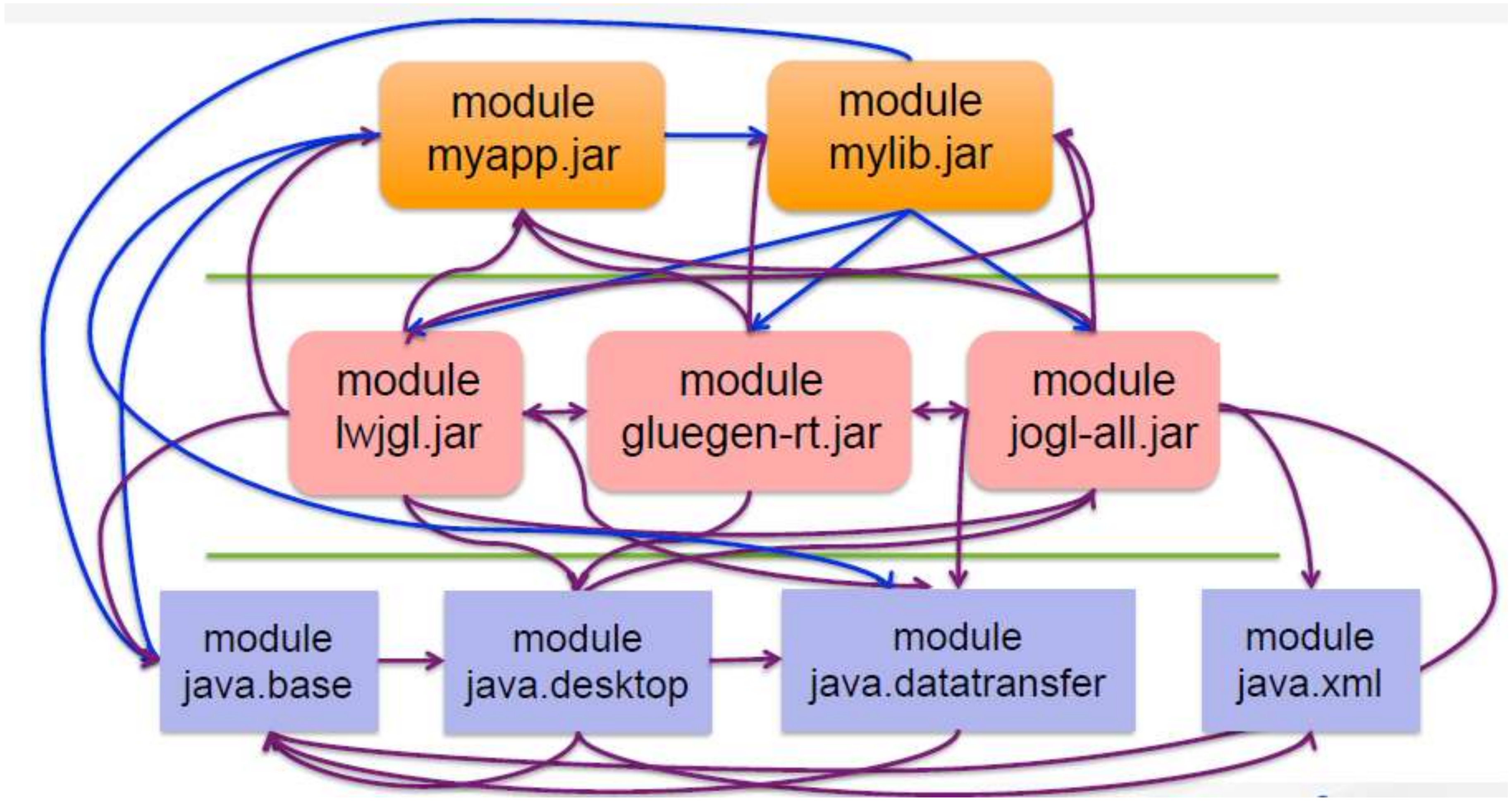
## Sample Application



## Automatic Modules

- Real modules
- Reuse and existing JAR without change
- Module name derived from JAR file name
- Exports all its packages
  - No selectivity
- Requires all modules accessible from the module path

# Sample Application



## Smooth Running with Modules

```
$ ~~java -classpath \~~
~~   lib/myapp.jar: \~~
~~   lib/mylib.jar: \~~
~~   lib/liblwjgl.jar: \~~
~~   lib/gluegen-rt.jar: \~~
~~   lib/jog-all.jar: \~~
~~   myApp.Main~~
```

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

## Migration – Best Practices

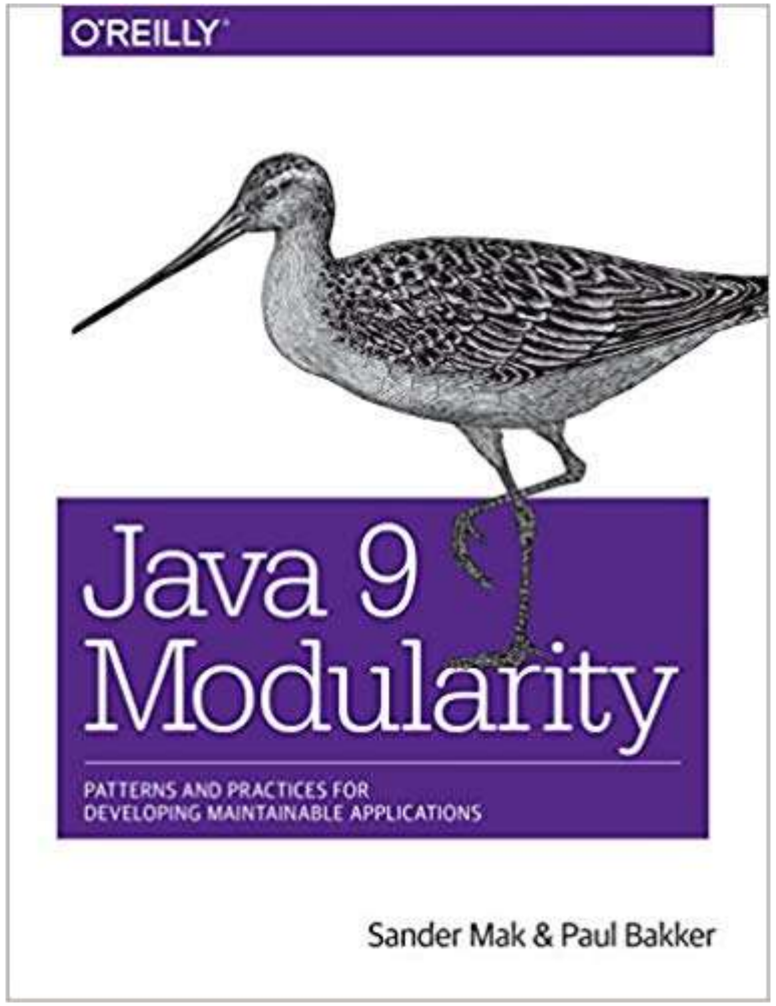
- Migrate to Java 9 using classpath only (run with –-illegal-access=deny)
- Create a module around your whole application
- Modularize your application
- Encourage library creators to produce Java 9 modules

## Recap

- Modularisation is a fundamental change for Java
  - JVM/JRE rather than language / APIs
- Disruptive change where APIs are exposed publicly
- Some learning curve to become comfortable with modularity

## References

- openjdk.java.net
- openjdk.java.net/jeps
- openjdk.java.net/projects/jigsaw
- jcp.org


O'REILLY
Java 9 Modularity
PATTERNS AND PRACTICES FOR DEVELOPING MAINTAINABLE APPLICATIONS
Sander Mak & Paul Bakker

## Thank You