Project Jigsaw in JDK 9: Modularity Comes To Java

Simon Ritter
Deputy CTO, Azul Systems
@speakjava | azul.com

Agenda

- API structure changes
- Introduction to Jigsaw
- Developing code with modules
- Application migration
- Advanced stuff
- Summary & Further Information

API Structure Changes

API Classification

- Supported, intended for public use
  - JCP specified: java.*, javax.*
  - JDK specific: some com.sun.*, some jdk.*
- Unsupported, not intended for public use
  - Mostly sun.*
  - Most infamous is sun.misc.Unsafe

General Java Compatability Policy

- If an application uses only supported APIs on version N of Java it *should* work on version N+1, even without recompilation
- Supported APIs can be removed, but only with advanced notice
- To date 23 classes, 18 interfaces and 379 methods have been deprecated
  - None have been removed

JDK 9: Incompatible Changes

- Encapsulate most JDK internal APIs
- Remove a small number of supported APIs
  - 6 in total, all add/remove PropertyChangeListener
  - Already flagged in JSR 337 (Java SE 8), JEP 162
- Change the binary structure of the JRE and JDK
- New version string format
- A single underscore will no longer be allowed as an identifier in source code

Removed In JDK 9

- Endorsed standard API override mechanism
- Extension mechanism

- No longer required now we have a module system

JDK Internal API Classification

- Non-critical
  - Little or no use outside the JDK
  - Used only for convenience (alternatives exist)
- Critical
  - Functionality that would be difficult, if not impossible to implement outside the JDK

JEP 260 Proposal

- Encapsulate all non-critical JDK-internal APIs
- Encapsulate all critical JDK-internal APIs, for which supported replacements exist in JDK 8
- Do *not* encapsulate other critical JDK-internal APIs
  - Deprecate these in JDK 9
  - Plan to encapsulate or remove them in JDK 10
  - Provide command-line option to access encapsulated critical APIs

JEP 260 Accessible Critical APIs

- sun.misc.Unsafe
- sun.misc.Signal
- sun.misc.SignalHandler
- sun.misc.Cleaner
- sun.reflect.Reflection.getCallerClass
- sun.reflect.ReflectionFactory

Reviewing Your Own Code

- jdeps tool
  - Introduced in JDK 8, improved in JDK 9
  - Maven jdeps plugin

```
jdeps –jdkinternals path/myapp.jar
```

```
path/myapp.jar -> /opt/jdk1.8.0/jre/lib/rt.jar
   <unnamed> (myapp.jar)
      -> java.awt
      -> java.awt.event
      -> java.beans
      -> java.io
      ...
```

Introduction To Jigsaw And Modules

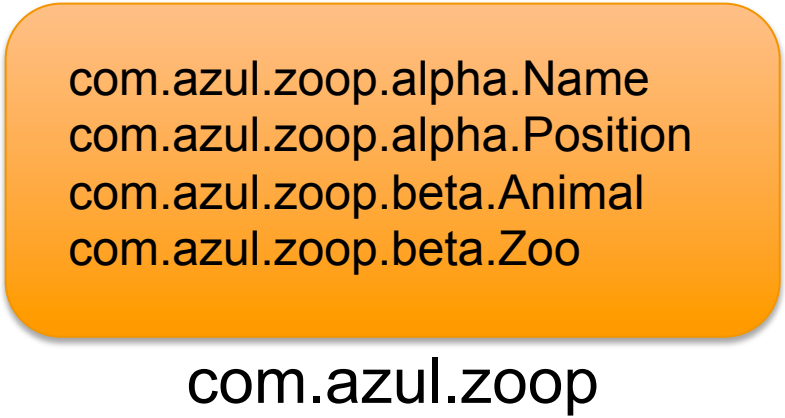
Goals For Project Jigsaw

- Make Java SE more scalable and flexible
  - IoT devices need more compact Java runtimes
- Improve security, maintainability and performance

- Simplify construction, deployment and maintenance of large scale applications
- Eliminate classpath hell

Module Fundamentals

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



Module Declaration

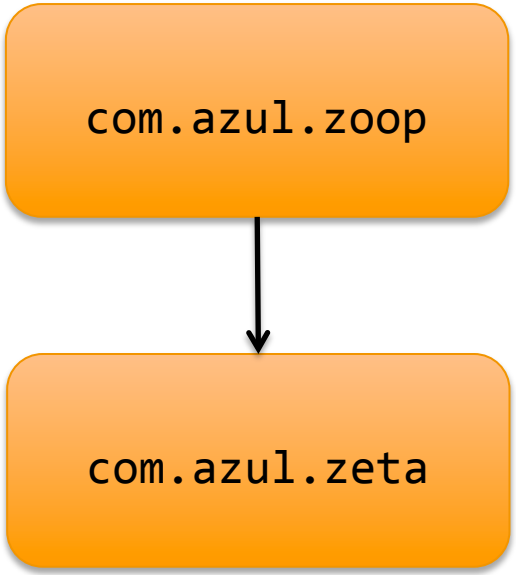
```
module com.azul.zoop {
}
```

module-info.java

```
com/azul/zoop/alpha/Name.java
com/azul/zoop/alpha/Position.java
com/azul/zoop/beta/Animal.java
com/azul/zoop/beta/Zoo.java
```

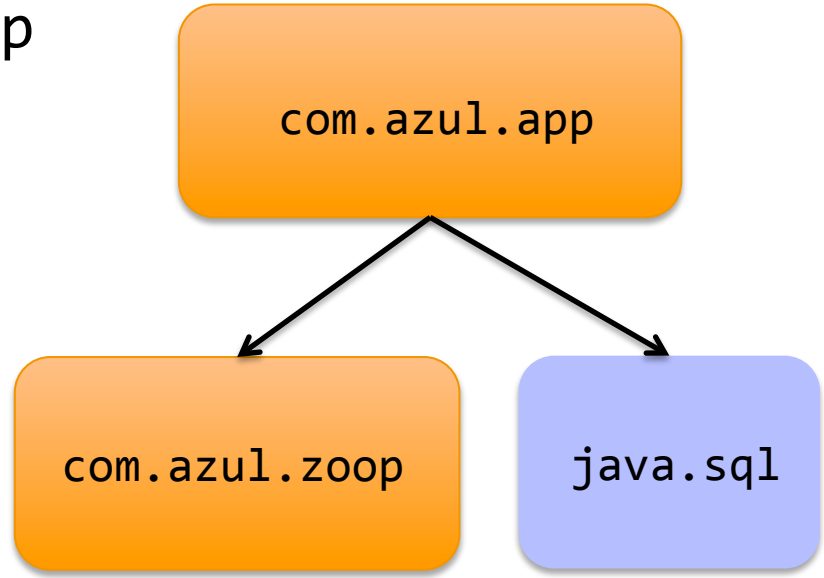
Module Dependencies

```
module com.azul.zoop {
  requires com.azul.zeta;
}
```

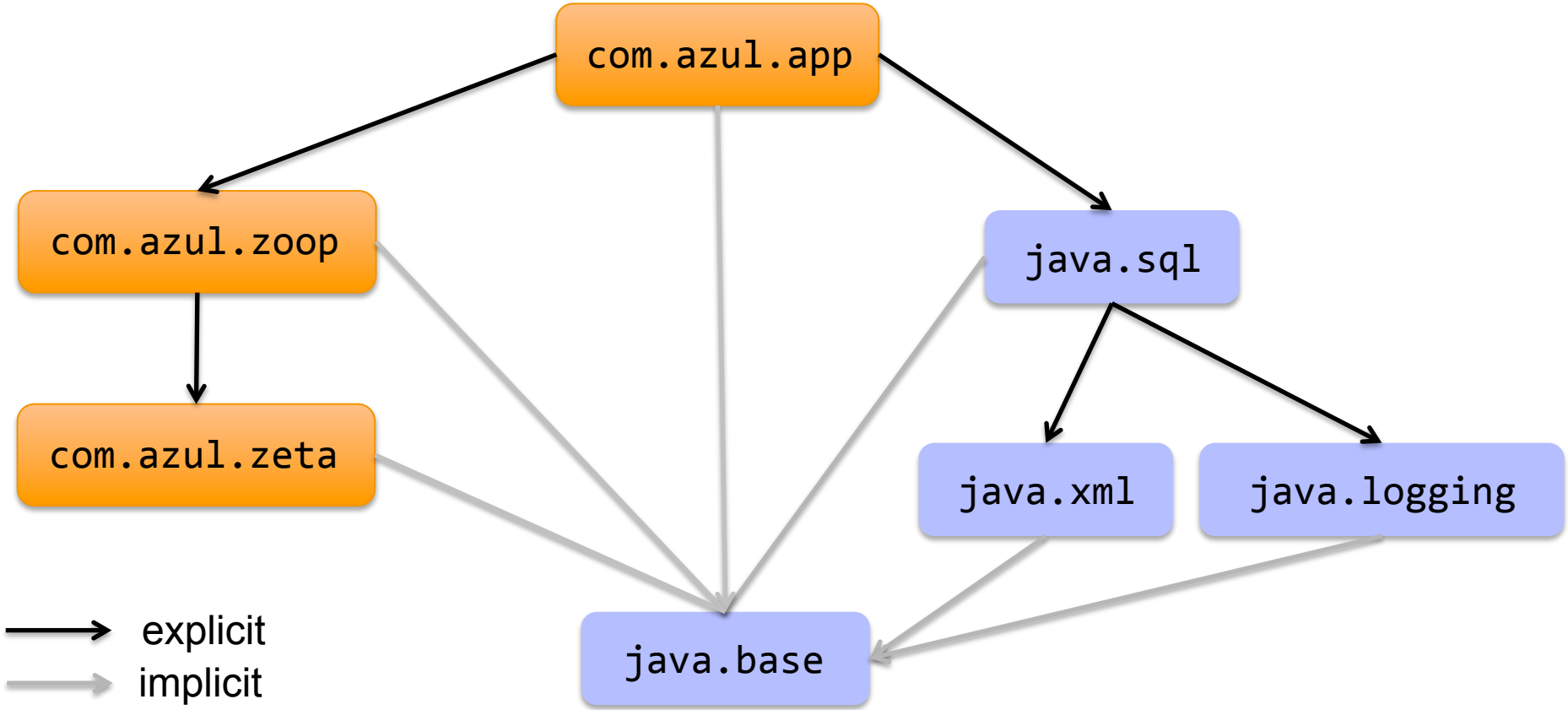


Module Dependencies

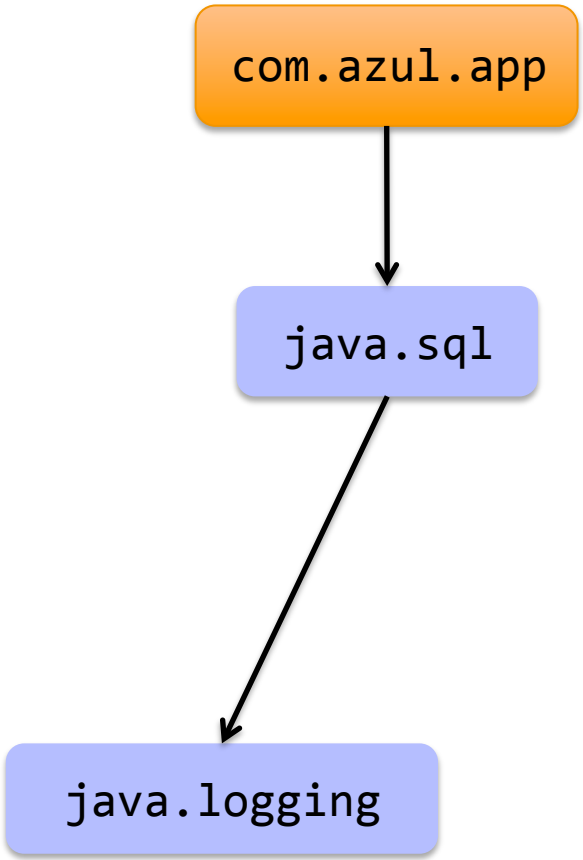
```
module com.azul.app {
  requires com.azul.zoop
  requires java.sql
}
```



Module Dependency Graph



Readability v. Dependency

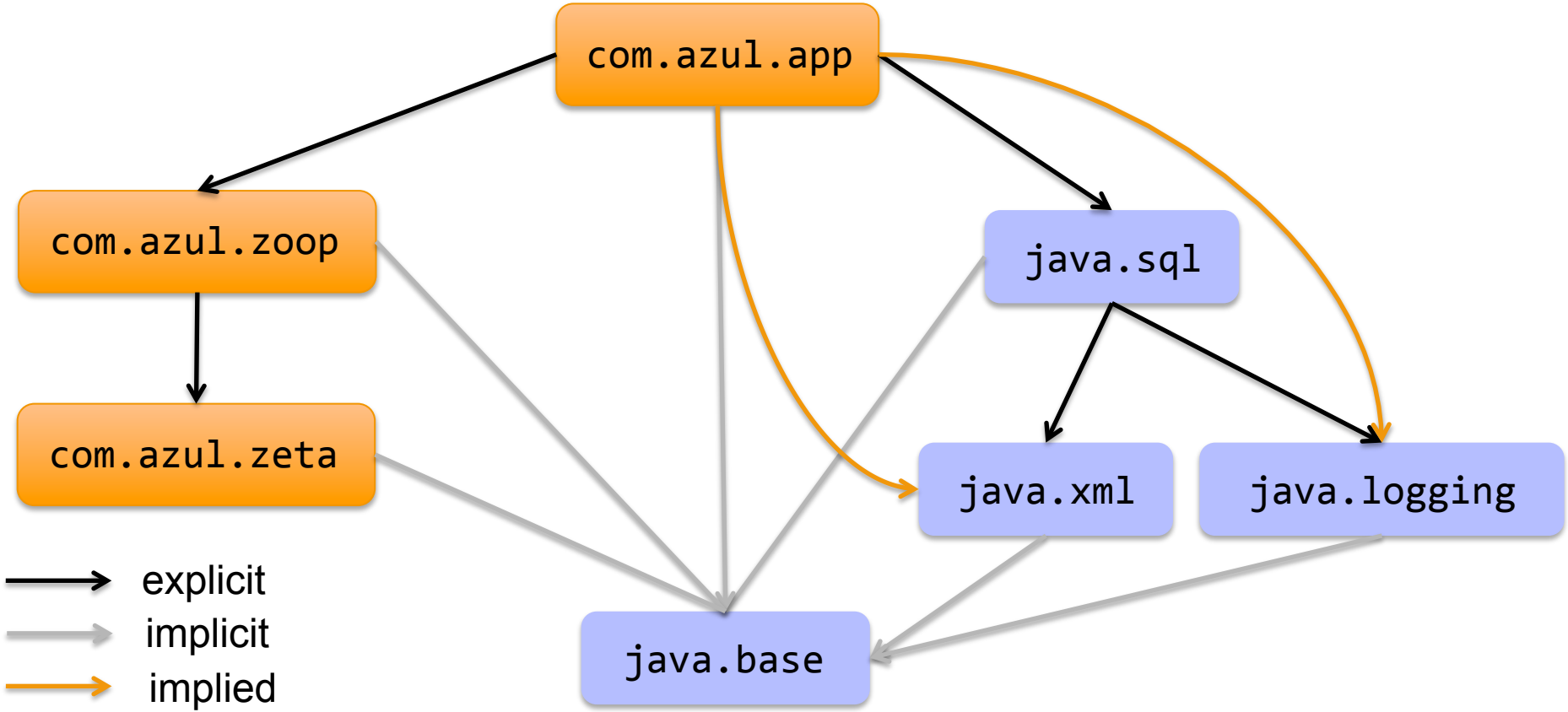



```
Driver d = …
Logger l = d.getParentLogger();
l.log(“azul’);
```

```
module java.sql {
  requires public java.logging;
}
```

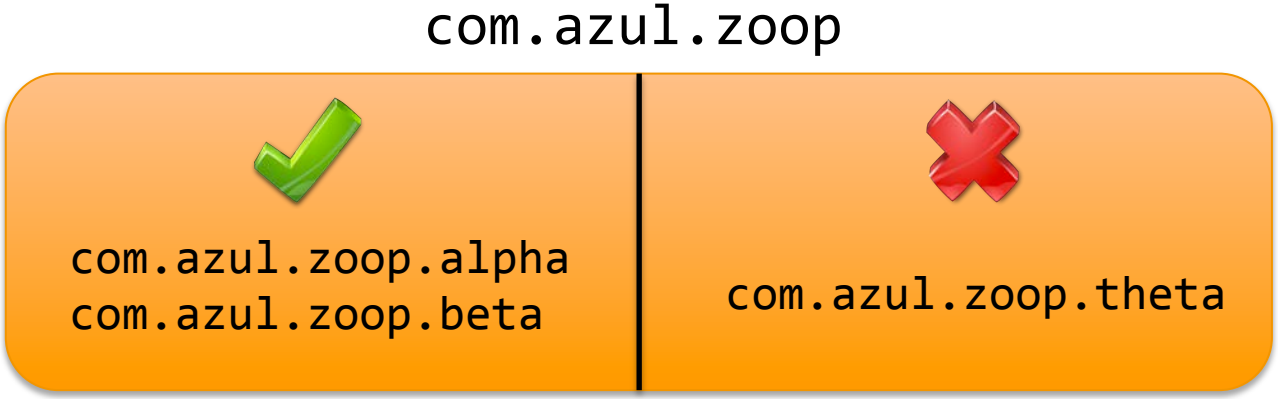
Implied readability

Module Implied Readability Graph



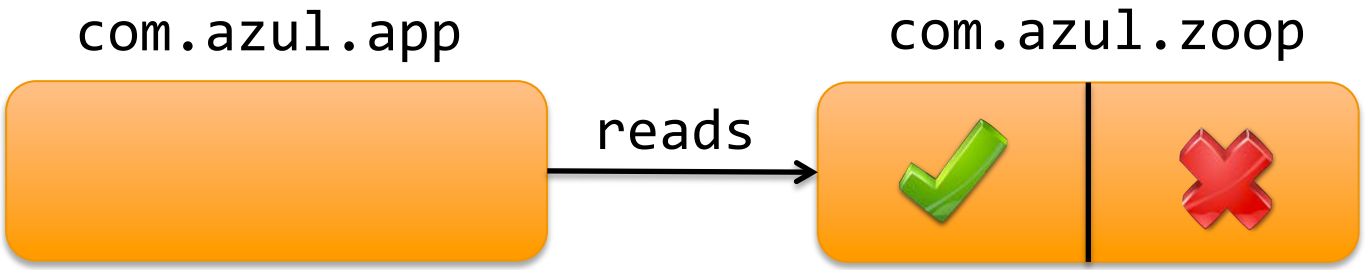
Package Visibility

```
module com.azul.zoop {
  requires com.azul.zeta;
  exports com.azul.zoop.alpha;
  exports com.azul.zoop.beta;
}
```



Accessibility

- For a package to be visible
  - The package must be exported by the containing module
  - The containing module must be read by the using module
- Public types from those packages can then be used



Java Accessibility (pre-JDK 9)

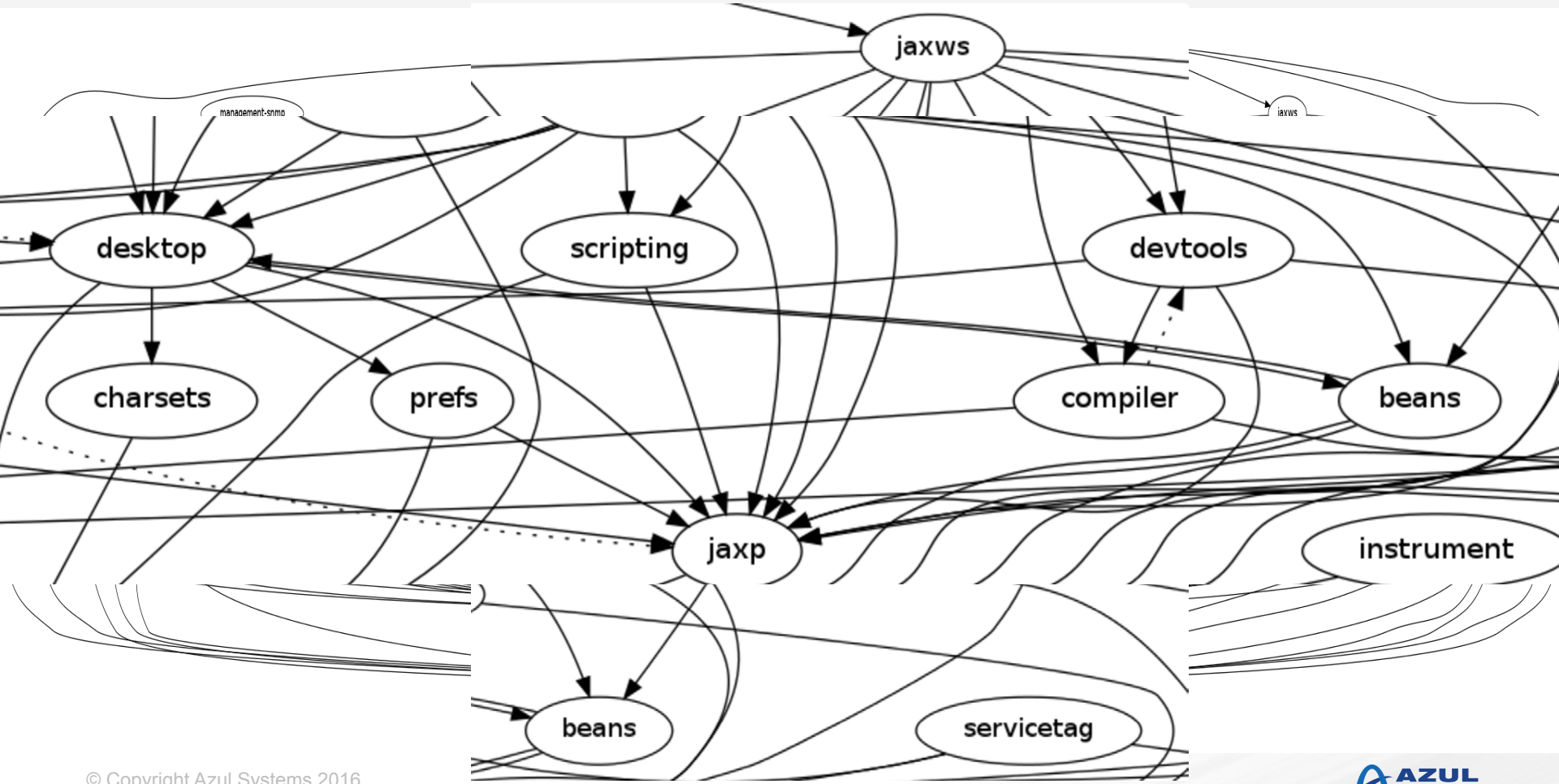
```
public
protected
<package>
private
```

Java Accessibility (JDK 9)

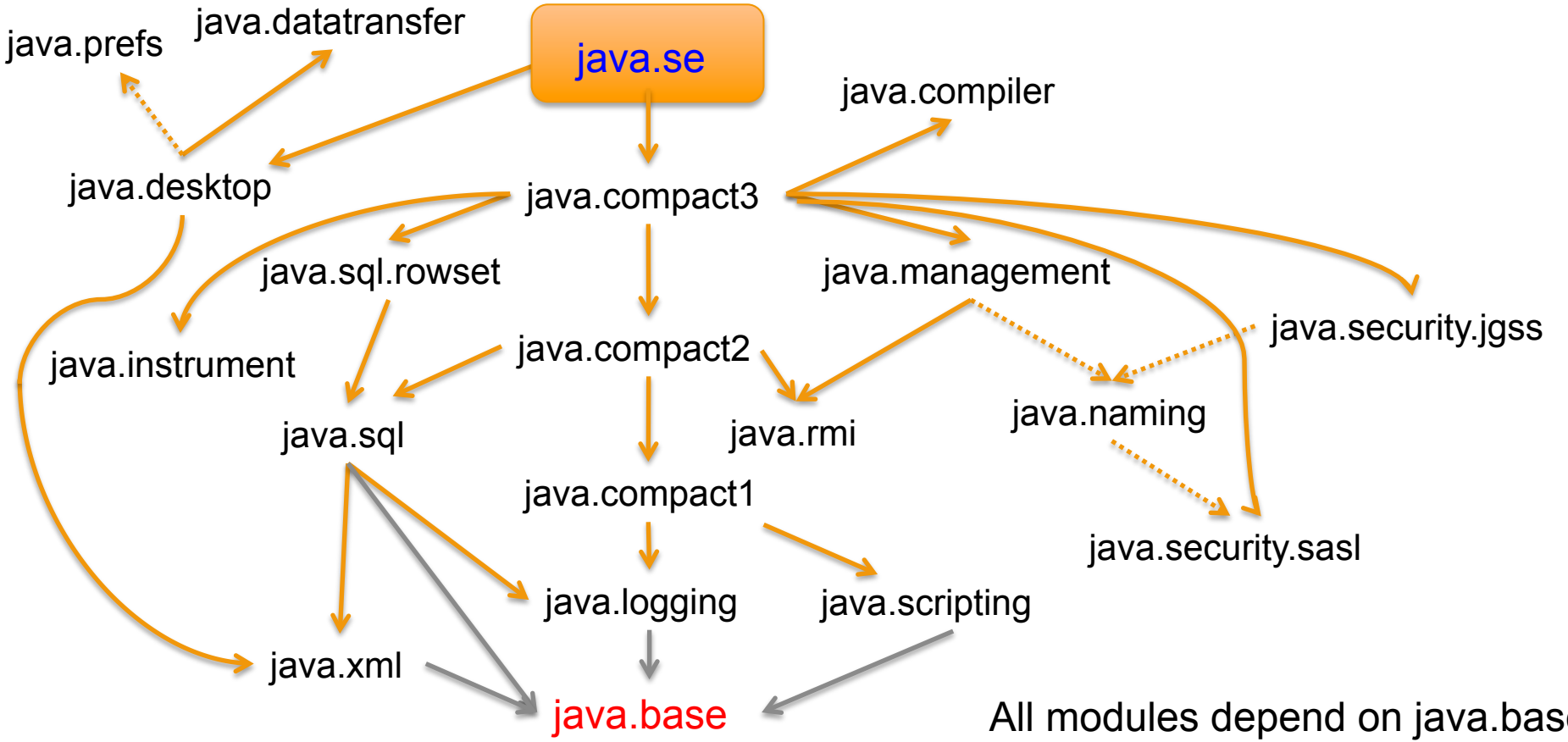
public to everyone
public, but only to specific modules
public only within a module
protected
<package>
private

public ≠ accessible (fundamental change to Java)

JDK 8 Dependencies

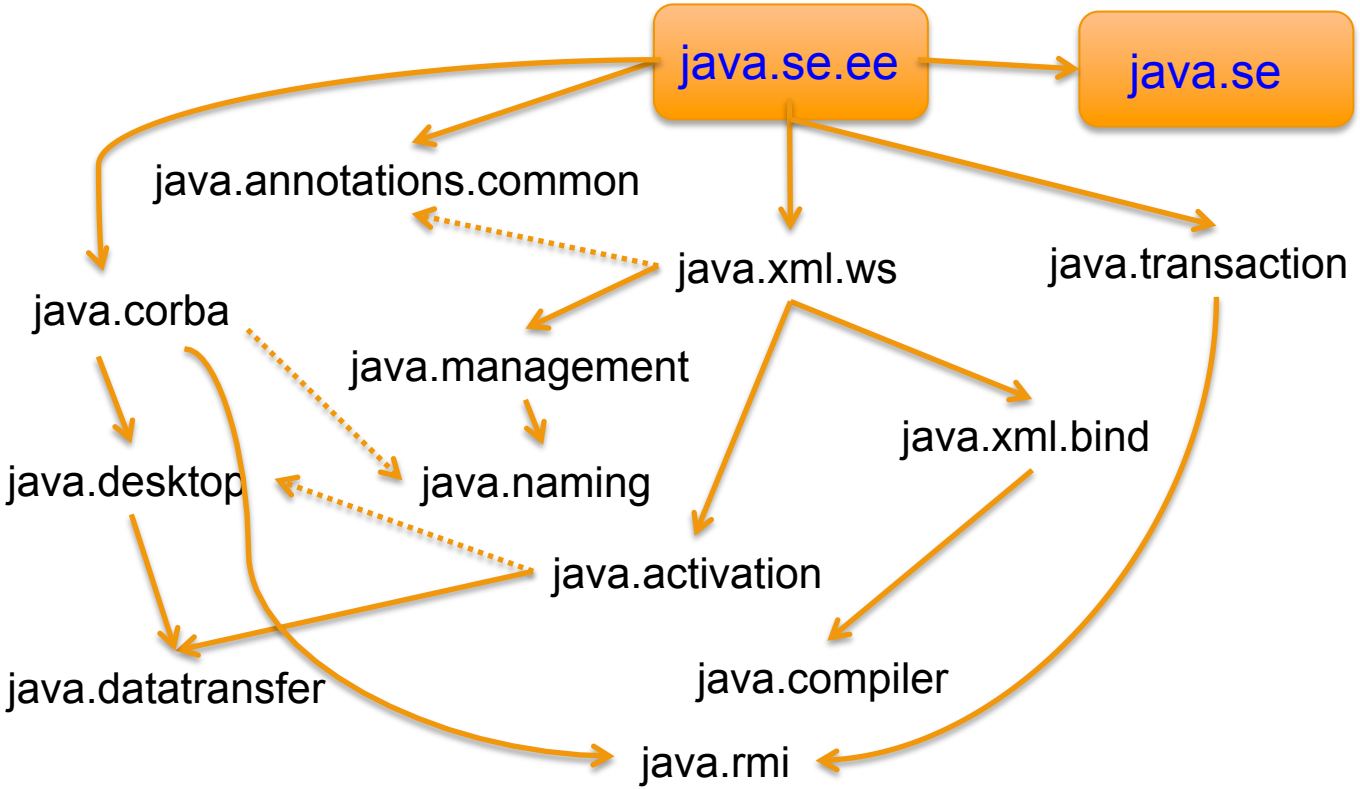


JDK 9 Platform Modules



All modules depend on java.base

JDK 9 Platform Modules



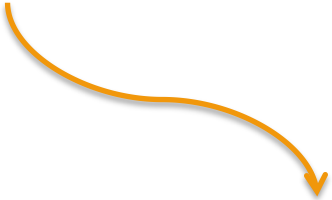
All modules depend on java.base

Developing Code With Modules

Compilation

```
$ javac -d mods \
  src/zeta/module-info.java \
  src/zeta/com/azul/zeta/Vehicle.java
```

```
src/zeta/module-info.java
src/zeta/com/azul/zeta/Vehicle.java
```



```
mods/zeta/module-info.class
mods/zeta/com/azul/zeta/Vehicle.class
```

Module Path

```
$ javac –modulepath dir1:dir2:dir3
```