
Java™
ORACLE

Working through Modularity

Vaibhav Choudhary (@vaibhav_c)
Java Platforms Team, Oracle
https://blogs.oracle.com/vaibhav

Java Your Next (Cloud)

Safe Harbor Statement

The following 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.

Agenda of the day …

1. What JIGSAW brings to the system
2. Change in accessibility and Readability
3. Types of Module brings the Migration Plan
4. Final Thoughts

What is Modularity ?

- Degree to which a system's component can be separated or can be recombined.
- Success story of modularity exists in every part of your life.
- What a good module will look like
  - Hidden implementation details (encapsulation)
  - Understandable interface
  - Autonomous unit of deployment (loose coupling)
  - Requirements can be easily discoverable

So Java Modularity for ?

- Strong encapsulation
- As-per-required binaries
- Small devices
- Compile time resolution

- and many more …

JIGSAW JEP's in a nutshell.

- JEP 200 : The modular JDK
- JEP 261 : Module System
- JEP 220 : Modular run-time images
- JEP 260: Encapsulate Most Internal API's
- JEP 282: jlink: The java linker

- and many more …

Change in accessibility and Readability

Accessibility in Java < Java 9

- public
- protected
- <package>
- private

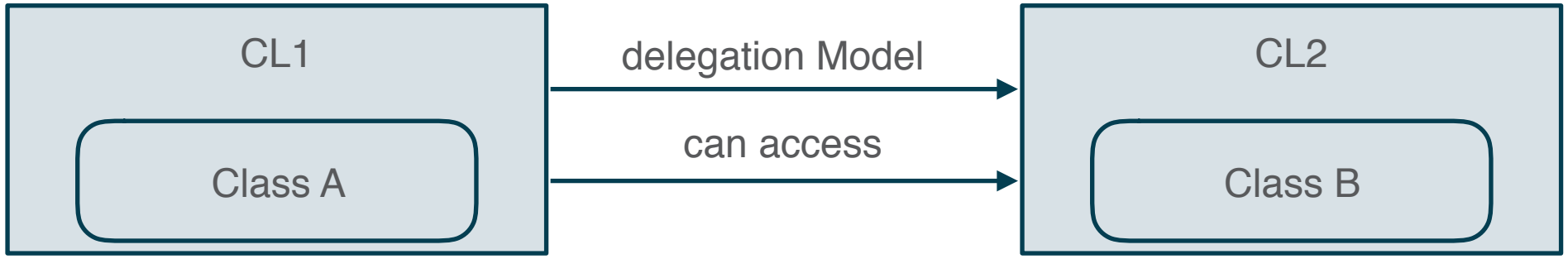
Accessibility in Java >= Java 9

- **public to everyone**
- **public but only to specific module**
- **public within a module**
- protected
- <package>
- private

```
public is no longer means accessible
```

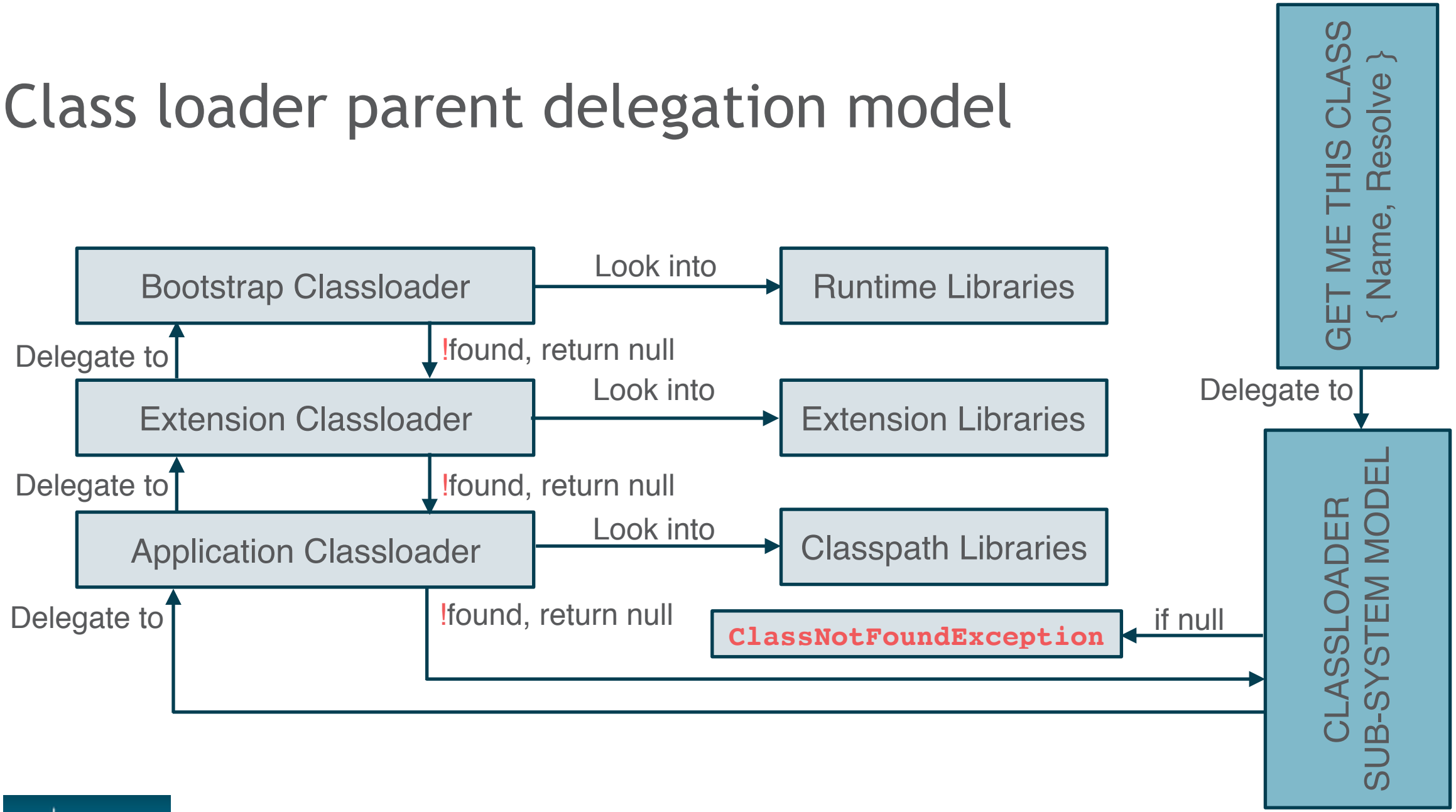
Accessibility in Java - Classloader view

- **Same class concept**
  - (C1, P1, K1) != (C1, P1, K2)



- **Ideally a class loader can set the accessibility in Java.**
- **No enforcement, too many classloader, boot loader.**

Class loader parent delegation model

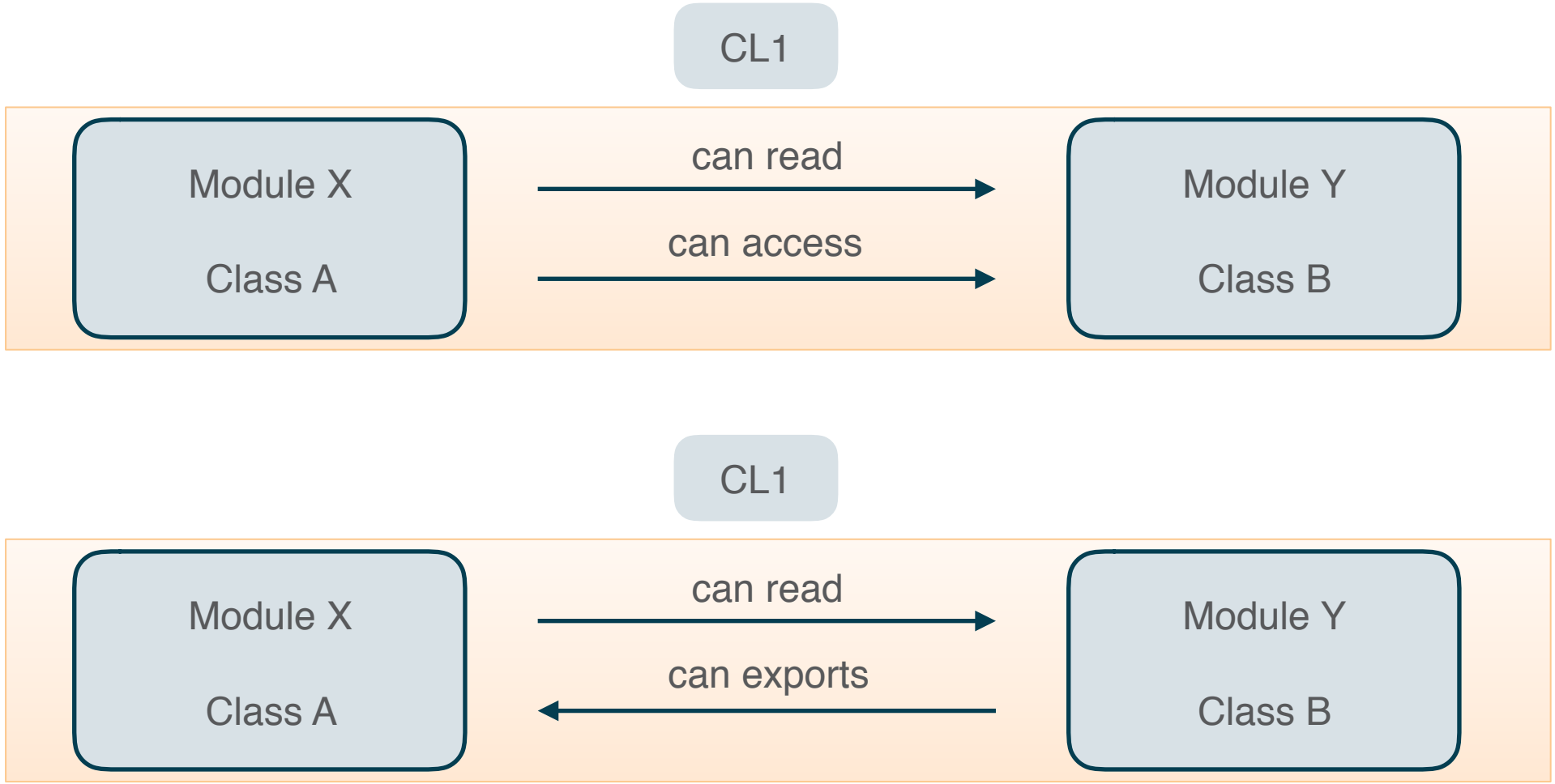


In short, this is how it works

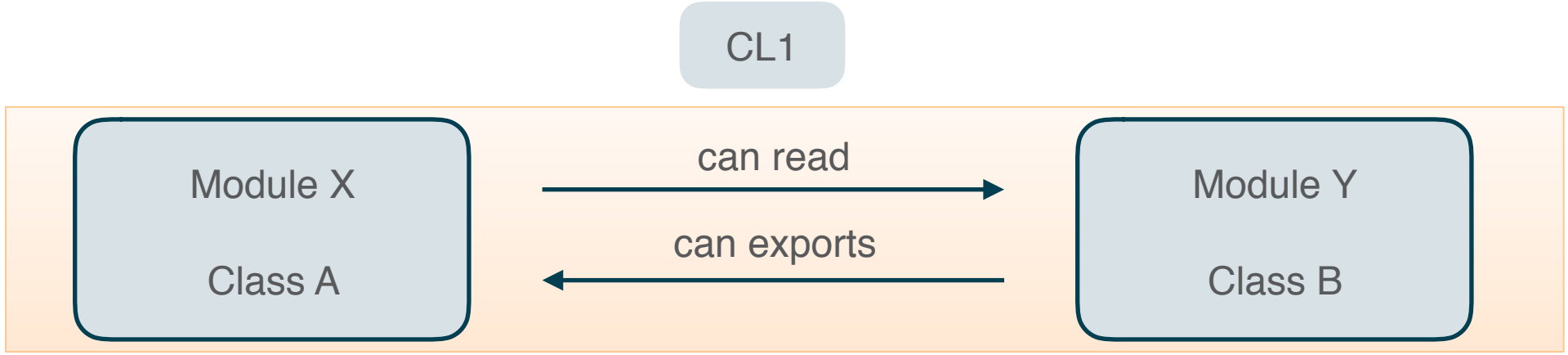
```
protected synchronized Class<?> loadClass
    (String name, boolean resolved)
    throws ClassNotFoundException {

    // First check if the class is already loaded
    Class c = findLoadedClass(name);
    if (c == null) {
        try {
            if (parent != null) {
                c = parent.loadClass(name, false);  // parent delegation
            } else {
                    c = findBootstrapClass0(name);
            }
        } catch (ClassNotFoundException e) {
            c = findClass(name);                    // if still not found, invoke findClass
        }
    }
    if(resolved) {
           resolveClass(c);
    }
    return c;
}
```

Accessibility in Java - Role of Readability



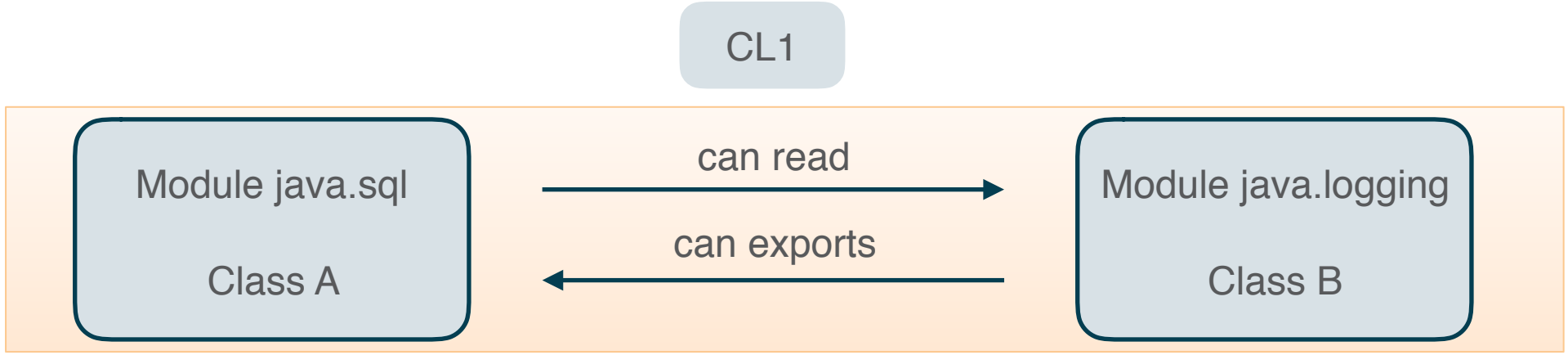
Accessibility in Java - Role of Readability




```
module X {
    requires Y;
}
```

```
module Y {
    exports P;
}
```

Readability in the SE dependency graph




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

```
module java.logging {
    exports java.util.logging;
}
```

Core reflection + Internal API’s

- At runtime, we need to respect accessibility and readability.
- For example : setAccessible(true) - No more a good choice

- JDK internal API’s are meant to use internally.
- A major portion of that will not be accessible now.
  - some are accessible
  - some tells the replacement
  - some are not

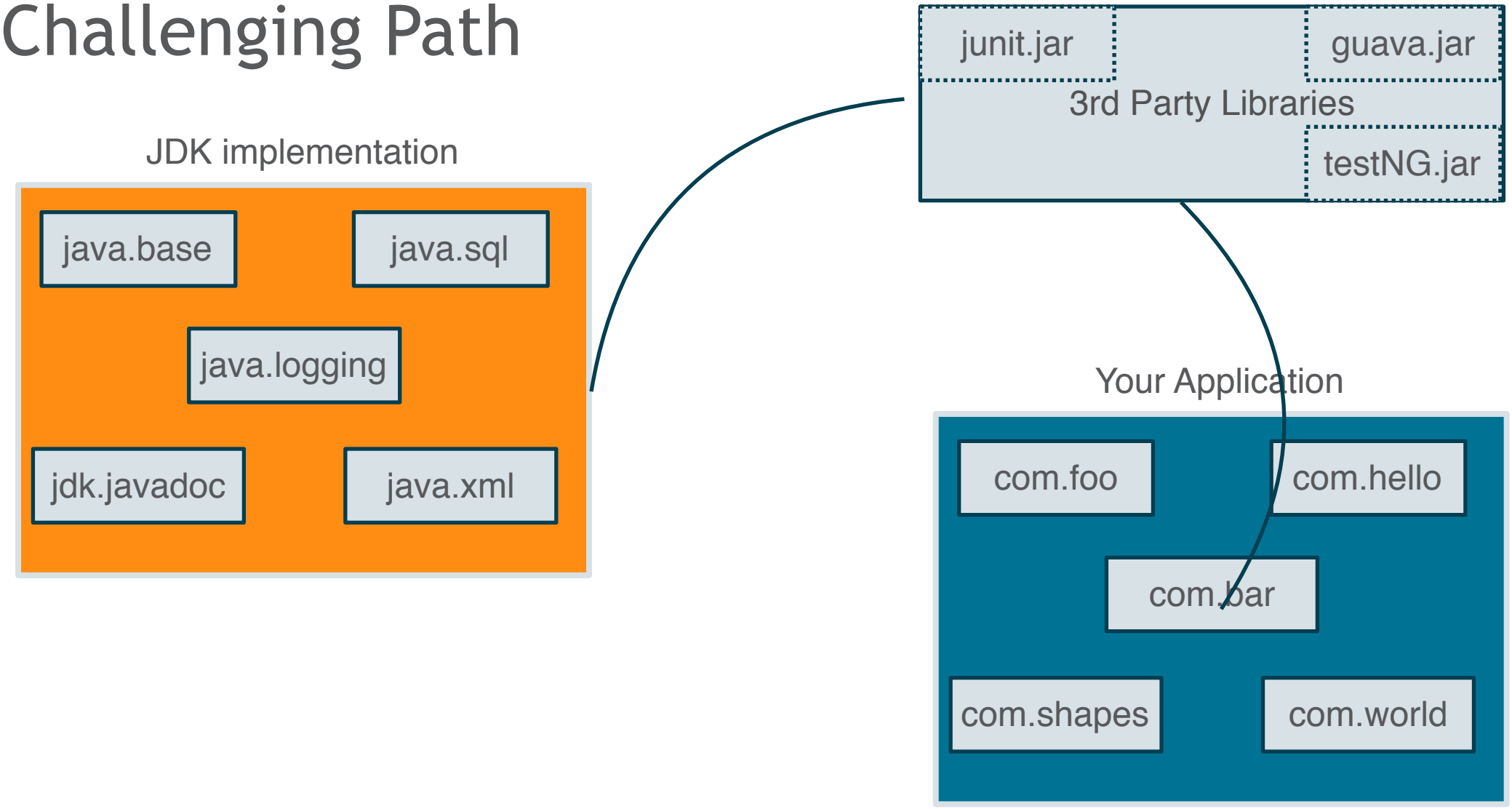
Summary

- Accessibility is “enforced” by the module system
- Like all module system, dependency discover at compile-time
- module-info.java manages exports, requires and many more
- Readability can be direct or can be implied.

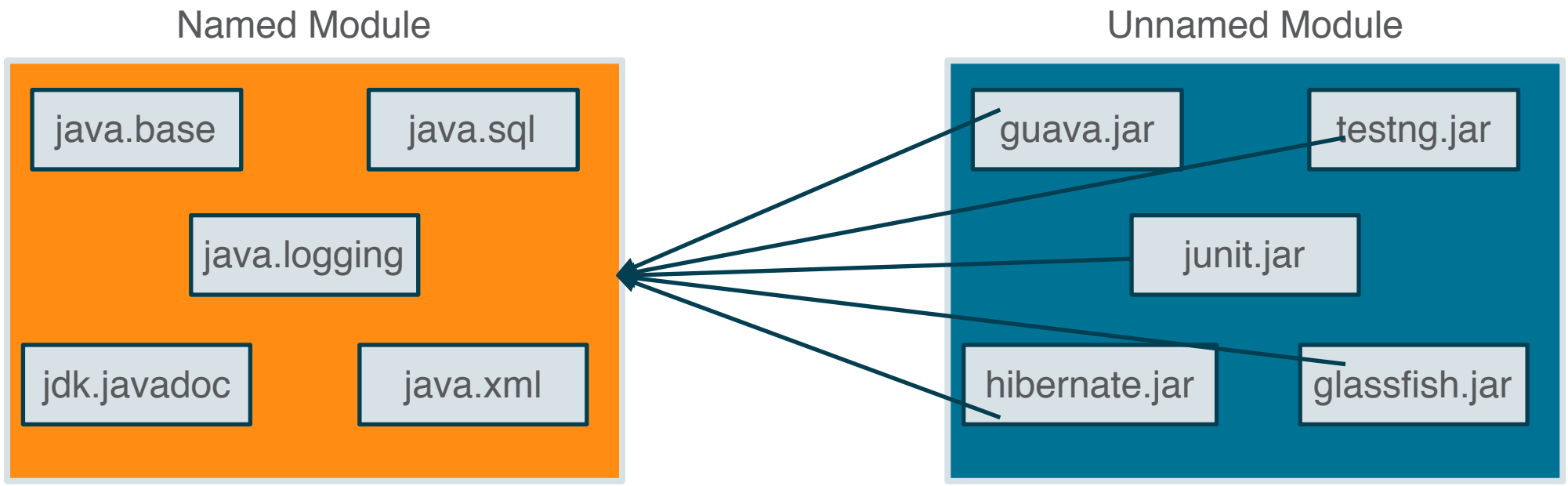
- Core reflection needs to respect accessibility and readability.
- Be aware of internal API’s usages

Types of Modules

Challenging Path

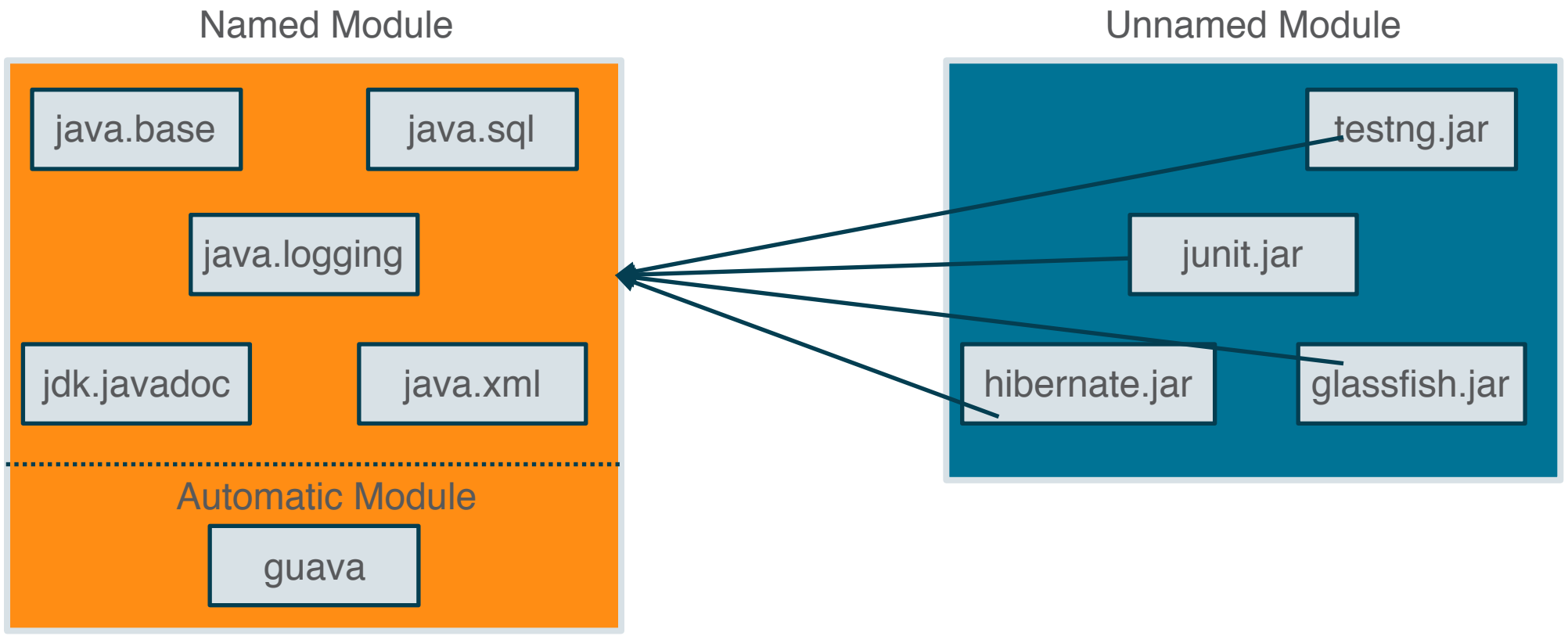


What if not a “Named Module”



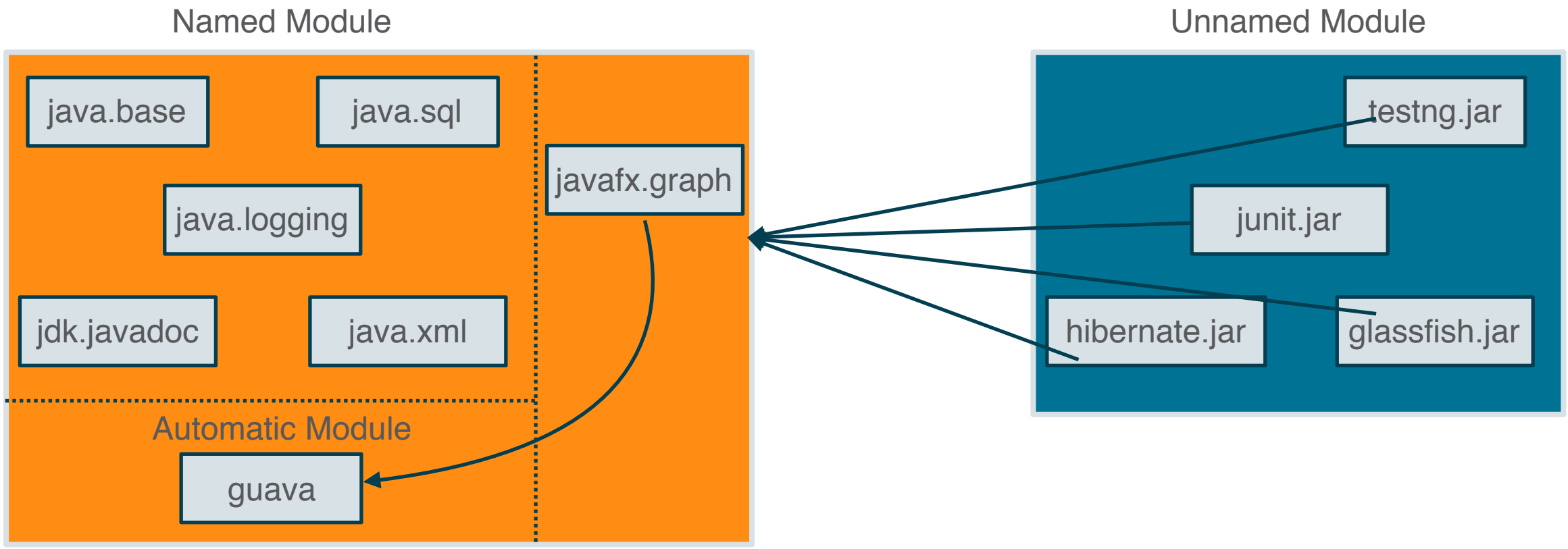
- Ideally unnamed module gives everything
- But is it a good idea for the named module to read everything of unnamed module ? (Something like reads class-path or reads “unnamed”)

Welcome to “Automatic Module”



- change from class-path to module-path will do the magic

Welcome to “Automatic Module”



- requires guava (or can say requires **public** guava)

Summary

- Named Module - is the new way to go
- Automatic Module - will help in migration
- Unnamed module - will live a old life in class-path

- “Migration can be done as lot of readability is FREE”

What we can see

- Demo
  - Create some modules
  - Add some dependency, understand the simple dependencies
  - See the dependency graph
  - Create your own binary
  - Check out the size of your binary
  - Further compression with the binary, if possible.

Important References

- State of Module - http://openjdk.java.net/projects/jigsaw/spec/sotms/
- Mark Reinhold Talk - https://www.youtube.com/watch?v=UKC0uC7QUkI
- Project Penrose - http://openjdk.java.net/projects/penrose/
- Project Jigsaw - http://openjdk.java.net/projects/jigsaw/

Thank you

For any query :-

Vaibhav Choudhary (@vaibhav_c)
Java Platforms Team, Oracle
https://blogs.oracle.com/vaibhav

Java
Your
Next
(Cloud)