Modularity in Java

With OSGi

Alex Blewitt
@alblue
Docklands.LJC
January 2016

Modularity in Java

Modularity is Easy?

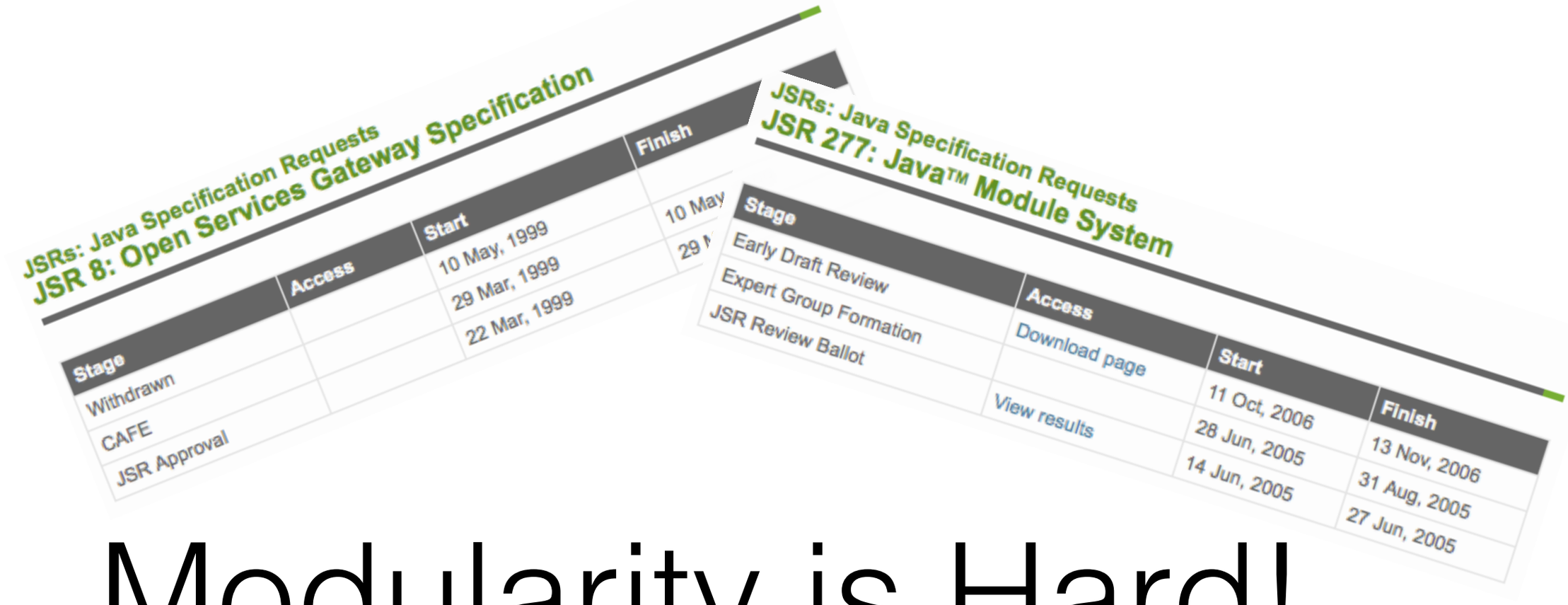
Modularity is Hard!

JSRs: Java Specification Requests

JSR 8: Open Services Gateway Specification

Stage	Access	Start	Finish
Withdrawn		10 May, 1999	10 May, 1999
CAFE		29 Mar, 1999	29 Mar, 1999
JSR Approval		22 Mar, 1999	

Modularity is Hard!



Modularity is Hard!

JSRs: Java Specification Requests

JSR 8: Open Services Gateway Specification

Stage	Access	Start	Finish
Withdrawn		10 May, 1999	10 May
		29 Mar, 1999	29 M
		22 Mar, 1999	

JSRs: Java Specification Requests

JSR 277: Java™ Module System

Stage	Access	Start	Finish
Early Draft Review	Download page	11 Oct, 2006	13 Nov, 2006
Expert Group Formation		28 Jun, 2005	31 Aug, 2005
JSR Review Ballot	View results	14 Jun, 2005	27 Jun, 2005

JSRs: Java Specification Requests

JSR 291: Dynamic Component Support for Java™ SE

Stage	Access	Start	Finish
Final Release	Download page	07 Aug, 2007	
Final Approval Ballot	View results	08 May, 2007	21 May, 2007
Proposed Final Draft	Download page	23 Mar, 2007	
Public Review Ballot	View results	16 Jan, 2007	22 Jan, 2007
Public Review	Download page	19 Dec, 2006	22 Jan, 2007
Early Draft Review	Download page	02 Aug, 2006	01 Sep, 2006
Expert Group Formation		14 Mar, 2006	20 Mar, 2006
JSR Review Ballot	View results	28 Feb, 2006	13 Mar, 2006

Hard!

JSRs: Java Specification Requests

JSR 8: Open Services Gateway Specification

Stage	Access	Start	Finish
Withdrawn		10 May, 1999	10 May
		29 Mar, 1999	29 M
		22 Mar, 1999	

JSRs: Java Specification Requests

JSR 277: Java™ Module System

Stage	Access	Start	Finish
Early Draft Review	Download page	11 Oct, 2006	13 Nov, 2006
Expert Group Formation		28 Jun, 2005	31 Aug, 2005
JSR Review Ballot	View results	14 Jun, 2005	27 Jun, 2005

JSRs: Java Specification Requests

JSR 291: Dynamic Component Support for Java™ SE

Stage	Access	Start	Finish
Final Release	Download page	07 Aug, 2007	
Final Approval Ballot	View results	08 May, 2007	21 May
Proposed Final Draft	Download page	23 Mar, 2007	
Public Review Ballot	View results	16 Jan, 2007	22 Jan, 2
Public Review	Download page	19 Dec, 2006	22 Jan, 20
Early Draft Review	Download page	02 Aug, 2006	01 Sep, 200
Expert Group Formation		14 Mar, 2006	20 Mar, 2006
JSR Review Ballot	View results	28 Feb, 2006	13 Mar, 2006

Hard!

JSRs: Java Specification Requests

JSR 294: Improved Modularity Support in the Java™ Programming Language

Stage	Access	Start	Finish
Early Draft Review	Download page	20 Nov, 2007	20 Dec, 2007
Expert Group Formation		09 May, 2006	
JSR Review Ballot	View results	25 Apr, 2006	08 May, 2006

JSRs: Java Specification Requests

JSR 8: Open Services Gateway Specification

Stage	Access	Start	Finish
Withdrawn		10 May, 1999	10 May
		29 Mar, 1999	29 M
		22 Mar, 1999	

JSRs: Java Specification Requests

JSR 277: Java™ Module System

Stage	Access	Start	Finish
Early Draft Review	Download page		13 Nov, 2006
Expert Group Formation			31 Aug, 2005
JSR Review Ballot			27 Jun, 2005

JSRs: Java Specification Requests

JSR 291: Dynamic Compone

Stage	Access	Start	Finish
Final Release	Download pa	08 May, 2007	21 May
Final Approval Ballot	View results	23 Mar, 2007	
Proposed Final Draft	Download page	16 Jan, 2007	22 Jan, 2
Public Review Ballot	View results	19 Dec, 2006	22 Jan, 20
Public Review	Download page	02 Aug, 2006	01 Sep, 200
Early Draft Review		14 Mar, 2006	20 Mar, 2006
Expert Group Formation	View results	28 Feb, 2006	13 Mar, 2006
JSR Review Ballot			

JSRs: Java Specification Requests

JSR 376: Java™ Platform Module System

Stage	Access	Start	Finish
JSR Renewal Ballot	View results	10 Nov, 2015	23 Nov, 2015
JSR Review Ballot	View results	09 Dec, 2014	22 Dec, 2014
JSR Review		25 Nov, 2014	08 Dec, 2014

JSRs: Java Specification Requests

JSR 294: Improved Modularity Support in the Java™ Programming Language

Stage	Access	Start	Finish
Early Draft Review	Download page	20 Nov, 2007	20 Dec, 2007
Expert Group Formation		09 May, 2006	
JSR Review Ballot	View results	25 Apr, 2006	08 May, 2006

JSRs: Java Specification Requests
JSR 8: Open Services Gateway Specification

Stage	Access	Start	Finish
Withdrawn		10 May, 1999	10 May
		29 Mar, 1999	29 M
		22 Mar, 1999	

JSRs: Java Specification Requests
JSR 277: Java™ Module System

Stage	Access	Start	Finish
Early Draft Review	Download page		13 Nov, 2006
Expert Group Formation			31 Aug, 2005
JSR Review Ballot			27 Jun, 2005

JSRs: Java Specification Requests
JSR 291: Dynamic Compone

Stage	Access	Start	Finish
Final Release	Download pa	08 May, 2007	21 May
Final Approval Ballot	View results	23 Mar, 2007	
Proposed Final Draft	Download page	16 Jan, 2007	22 Jan, 2
Public Review Ballot	View results	19 Dec, 2006	22 Jan, 20
Public Review	Download page	02 Aug, 2006	01 Sep, 200
Early Draft Review		14 Mar, 2006	20 Mar, 2006
Expert Group Formation	View results	28 Feb, 2006	13 Mar, 2006
JSR Review Ballot			

JSRs: Java Specification Requests
JSR 376: Java™ Platform Module System

Stage	Access	Start	Finish
JSR Renewal Ballot	View results	10 Nov, 2015	23 Nov, 2015
JSR Review Ballot	View results	09 Dec, 2014	22 Dec, 2014
JSR Review		25 Nov, 2014	08 Dec, 2014

JSRs: Java Specification Requests
JSR 294: Improved Modularity Support in the Java™ Programming Language

Stage	Access	Start	Finish
Early Draft Review	Download page	20 Nov, 2007	20 Dec, 2007
Expert Group Formation		09 May, 2006	
JSR Review Ballot	View results	25 Apr, 2006	08 May, 2006

Proposed schedule change for JDK 9

mark.reinhold at oracle.com mark.reinhold at oracle.com
Tue Dec 1 17:08:06 UTC 2015

JSRs: Java Specification Requests

JSR 8: Open Services Gateway Specification

Stage	Access	Start	Finish
Withdrawn		10 May, 1999	10 May
		29 Mar, 1999	29 M
		22 Mar, 1999	

JSRs: Java Specification Requests

JSR 277: Java™ Module System

Stage	Access	Start	Finish
Early Draft Review	Download page		13 Nov, 2006
Expert Group Formation			31 Aug, 2005
JSR Review Ballot			27 Jun, 2005

JSRs: Java Specification Requests

JSR 291: Dynamic Compone

Stage	Access	Start	Finish
Final Release	Download pa	08 May, 2007	21 May
Final Approval Ballot	View results	23 Mar, 2007	
Proposed Final Draft	Download page	16 Jan, 2007	22 Jan, 2
Public Review Ballot	View results	19 Dec, 2006	22 Jan, 20
Public Review	Download page	02 Aug, 2006	01 Sep, 200
Early Draft Review		14 Mar, 2006	20 Mar, 2006
Expert Group Formation	View results	28 Feb, 2006	13 Mar, 2006
JSR Review Ballot			

JSRs: Java Specification Requests

JSR 376: Java™ Platform Module System

Stage	Access	Start	Finish
JSR Renewal Ballot	View results	10 Nov, 2015	23 Nov, 2015
JSR Review Ballot	View results	09 Dec, 2014	22 Dec, 2014
JSR Review		25 Nov, 2014	08 Dec, 2014

JSRs: Java Specification Requests

JSR 294: Improved Modularity Support in the Java™ Programming Language

Stage	Access	Start	Finish
Early Draft Review	Download page	20 Nov, 2007	20 Dec, 2007
Expert Group Formation		09 May, 2006	
JSR Review Ballot	View results	25 Apr, 2006	08 May, 2006

Proposed schedule change for JDK 9

mark.reinhold at oracle.com mark.reinhold at oracle.com

Tue Dec 1 17:08:06 UTC 2015

```
Here are the proposed dates for the interim milestones:

 (2016/05/26  Feature Complete)
  2016/08/11  All Tests Run
  2016/09/01  Rampdown Start
  2016/10/20  Zero Bug Bounce
  2016/12/01  Rampdown Phase 2
  2017/01/26  Final Release Candidate
 (2017/03/23  General Availability)
```

JSRs: Java Specification Requests
JSR 8: Open Services Gateway Specification

Stage	Access	Start	Finish
Withdrawn		10 May, 1999	10 May
		22 Mar, 1999	29 M

JSRs: Java Specification Requests
JSR 277: Java™ Module System

Stage	Access	Start	Finish
Early Draft Review	Download page	11	13 Nov, 2006
Expert Group Formation			31 Aug, 2005
JSR Review Ballot			27 Jun, 2005

JSRs: Java Specification Requests
JSR 291: Dynamic Compone

Stage	Access	Start	Finish
Final Release	Download pa	08 May, 2007	21 May, 2007
Final Approval Ballot	View results	23 Mar, 2007	
Proposed Final Draft	Download page	16 Jan, 2007	22 Jan, 2
Public Review Ballot	View results	19 Dec, 2006	22 Jan, 20
Public Review	Download page	02 Aug, 2006	01 Sep, 200
Early Draft Review	Download page	14 Mar, 2006	20 Mar, 2006
Expert Group Formation		28 Feb, 2006	13 Mar, 2006
JSR Review Ballot	View results		

JSRs: Java Specification Requests
JSR 376: Java™ Platform Module System

Stage	Access	Start	Finish
JSR Renewal Ballot	View results	10 Nov, 2015	23 Nov, 2015
JSR Review Ballot	View results	09 Dec, 2014	22 Dec, 2014
JSR Review		25 Nov, 2014	08 Dec, 2014

JSRs: Java Specification Requests
JSR 294: Improved Modularity Support in the Java™ Programming Language

Stage	Access	Start	Finish
Early Draft Review	Download page	20 Nov, 2007	20 Dec, 2007
Expert Group Formation		09 May, 2006	
JSR Review Ballot	View results	25 Apr, 2006	08 May, 2006

Proposed schedule change for JDK 9

mark.reinhold at oracle.com mark.reinhold at oracle.com
Tue Dec 1 17:08:06 UTC 2015

```
Here are the proposed dates for the interim milestones:

(2016/05/26  Feature Complete)
2016/08/11  All Tests Run
2016/09/01  Rampdown Start
2016/10/20  Zero Bug Bounce
2016/12/01  Rampdown Phase 2
2017/01/26  Final Release Candidate
(2017/03/23  General Availability)
```

JSRs: Java Specification Requests
JSR 8: Open Services Gateway Specification

Stage	Access	Start	Finish
Withdrawn		10 May, 1999	10 May
		22 Mar, 1999	29 M

JSRs: Java Specification Requests
JSR 277: Java™ Module System

Stage	Access
Early Draft Review	Download page
Expert Group Formation	
JSR Review Ballot	

JSRs: Java Specification Requests
JSR 291: Dynamic Compone

Stage	Access
Final Release	Download pa
Final Approval Ballot	View results
Proposed Final Draft	Download page
Public Review Ballot	View results
Public Review	Download page
Early Draft Review	Download page
Expert Group Formation	
JSR Review Ballot	View results

		08 May, 2007	21 [illegible]
		23 Mar, 2007	
		16 Jan, 2007	22 Jan, 2
		19 Dec, 2006	22 Jan, 20
		02 Aug, 2006	01 Sep, 200
		14 Mar, 2006	20 Mar, 2006
		28 Feb, 2006	13 Mar, 2006

JSRs: Java Specification Requests
JSR 376: Java™ Platform Module System

Stage	Access	Start	Finish
JSR Renewal Ballot	View results	10 Nov, 2015	23 No
JSR Review Ballot	View results	09 Dec, 2014	22 De
JSR Review		25 Nov, 2014	08 Dec

JSRs: Java Specification Requests
JSR 294: Improved Modularity
Programming Language

Stage	Access
Early Draft Review	Download p
Expert Group Formation	
JSR Review Ballot	View result

Finish
0 Dec, 2007
08 May, 2006

Proposed schedule change for JDK 9

mark.reinhold at oracle.com mark.reinhold at oracle.com
Tue Dec 1 17:08:06 UTC 2015

Here are the proposed [illegible] erim milestones:

(2016/05/26 Feature
2016/08/11 All Tes
2016/09/01 Rampdow
/10/20 Zero Bu
/12/01 Rampdow
Final Re
(2017/03/23 General Ava

Modularity is Easy?

Modularity is Hard!

solutions are hard

solutions are complex

sufficiently advanced technology is indistinguishable from magic

Modularity is Hard!

OSGi is hard

OSGi is complex

sufficiently advanced technology is indistinguishable from magic

Modularity is Hard!

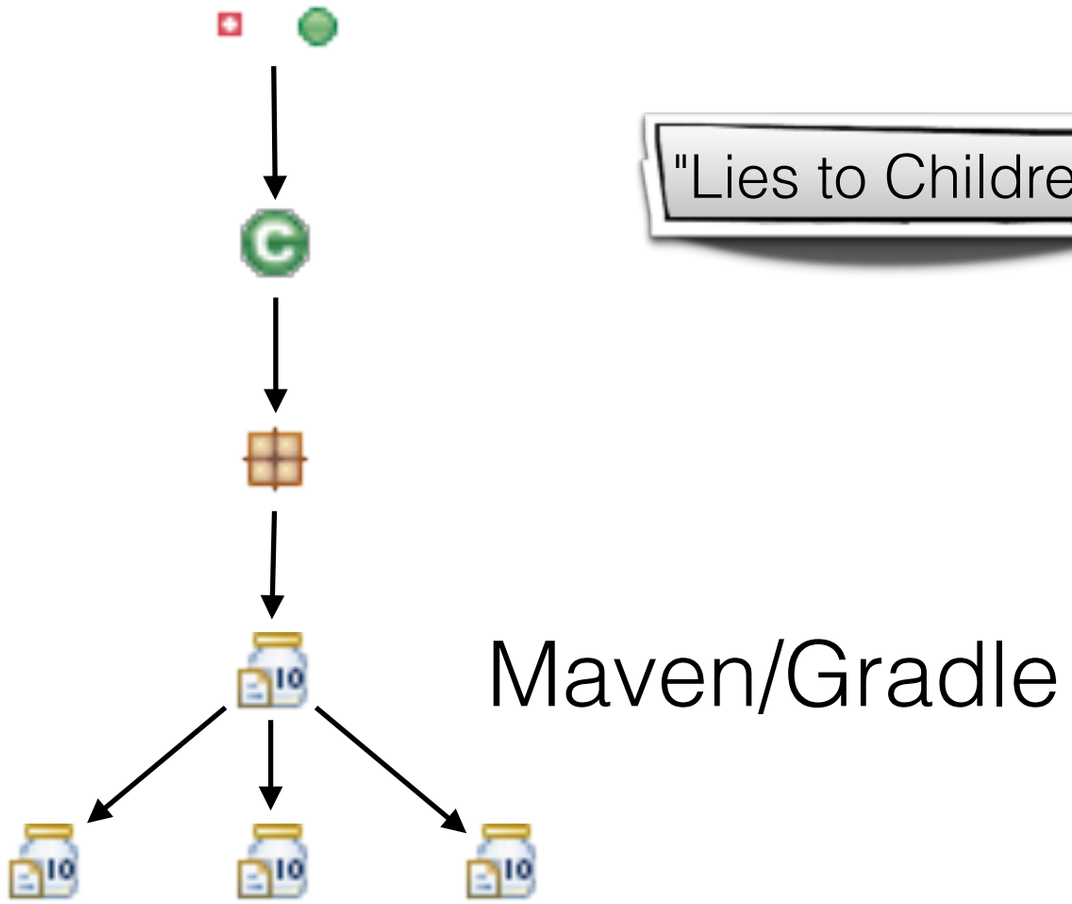
Jigsaw is hard

Jigsaw is complex

sufficiently advanced technology is indistinguishable from magic

Why do people think modularity is easy?

- Java's modular already, right?
  - Fields and Methods
  - Classes
  - Packages
  - JARs



"Lies to Children"*

- Terry Pratchett – Science of the Discworld

Any explanation of an observed phenomenon which, while not 100% scientifically accurate, is simple enough, and just accurate enough, to convey the beginnings of understanding to anyone who is new to the subject.

http://wiki.lspace.org/mediawiki/index.php/Lies-To-Children

**also known as Lies to Bosses*

"Lies to Developers"

- Fields are private
  - Apart from reflection
- Fields are final
  - Apart from setAccessible
- Methods are standalone
  - Apart from lambdas and inner classes

"Lies to Developers"

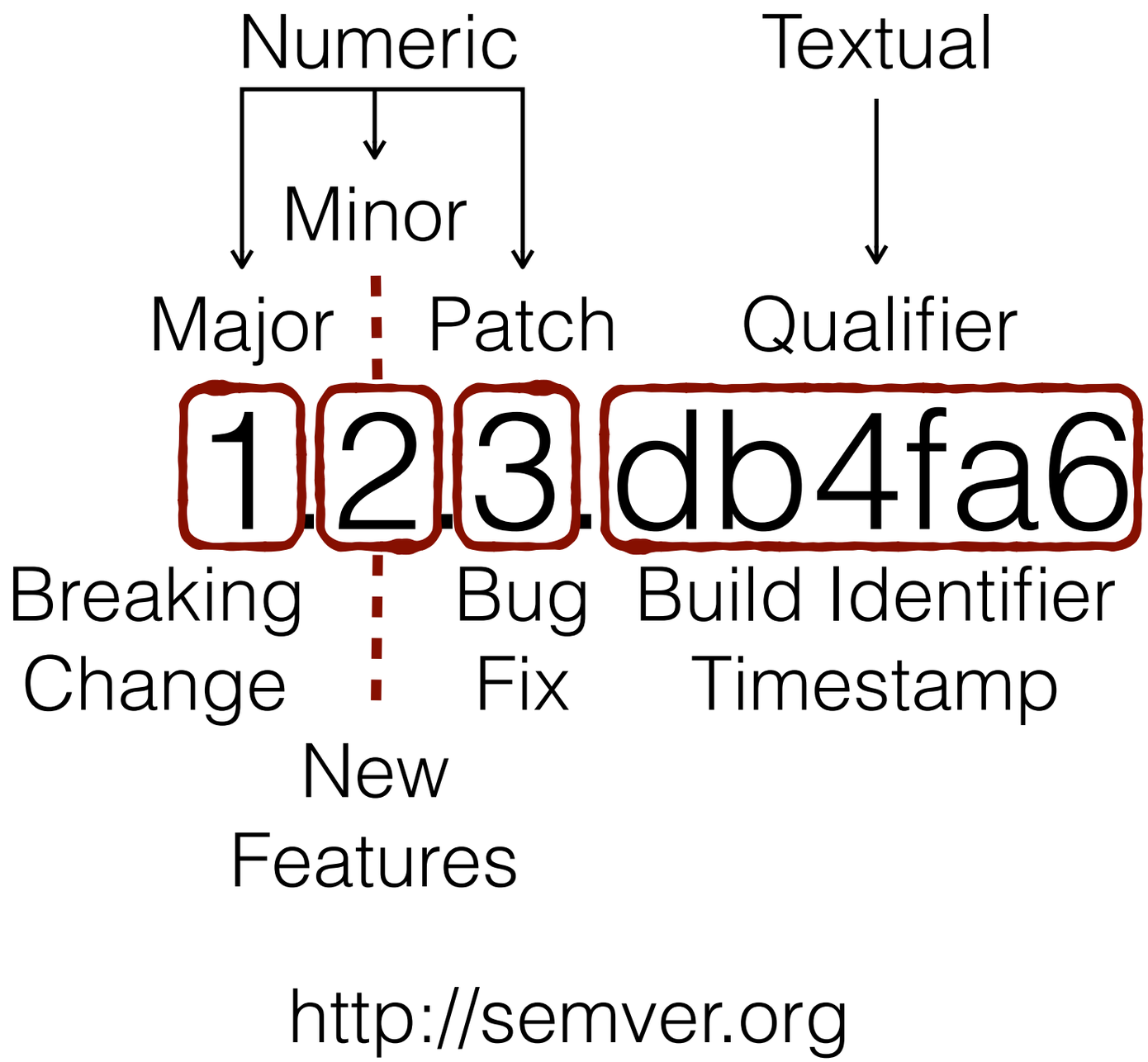
- Classes are encapsulated
    - Apart from dependent types for internal dependencies
- Packages are boundaries for classes
    - Except cyclic references between packages can easily occur
- JARs are unique elements of deployment
    - Except JARs can contain duplicate classes ("first one wins")

"Lies to Developers"

- Only one class/static per VM
  - Classes are unique per ClassLoader, not VM
- JARs are versioned
  - Except no-one agrees on version numbers
- Semantic versioning is important
  - Except when it isn't

https://github.com/vert-x3/wiki/wiki/3.1-Release-Notes

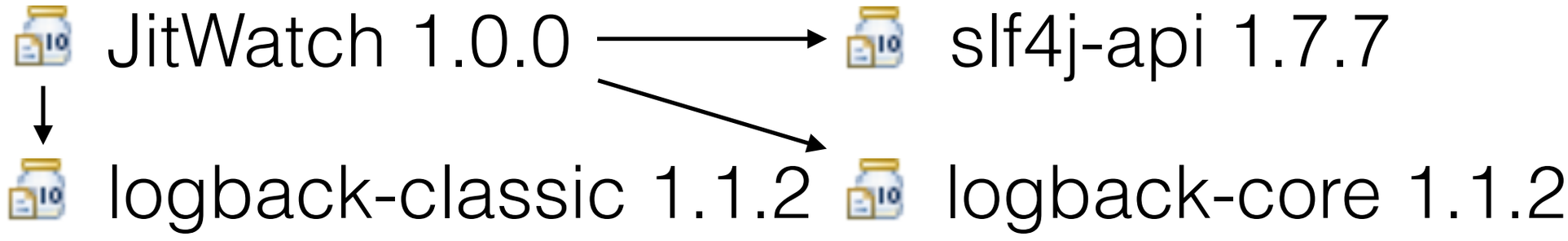
Semantic Versioning



http://semver.org

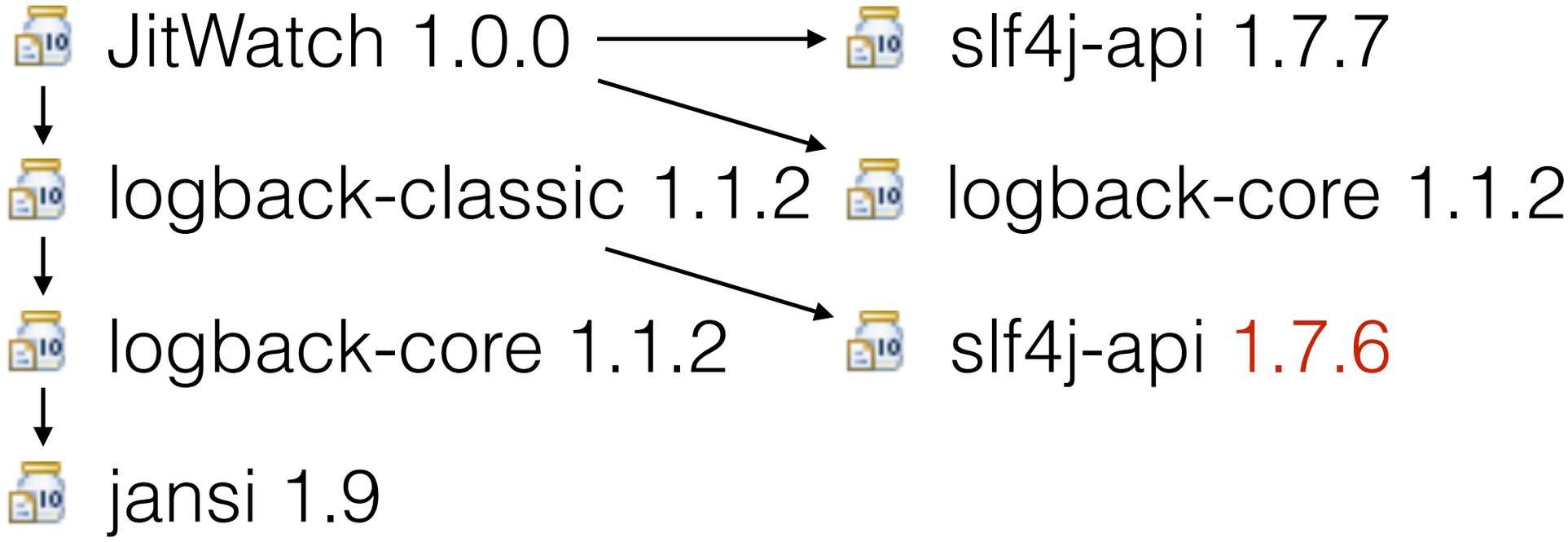
What Developers believe

- Dependencies are easy to manage



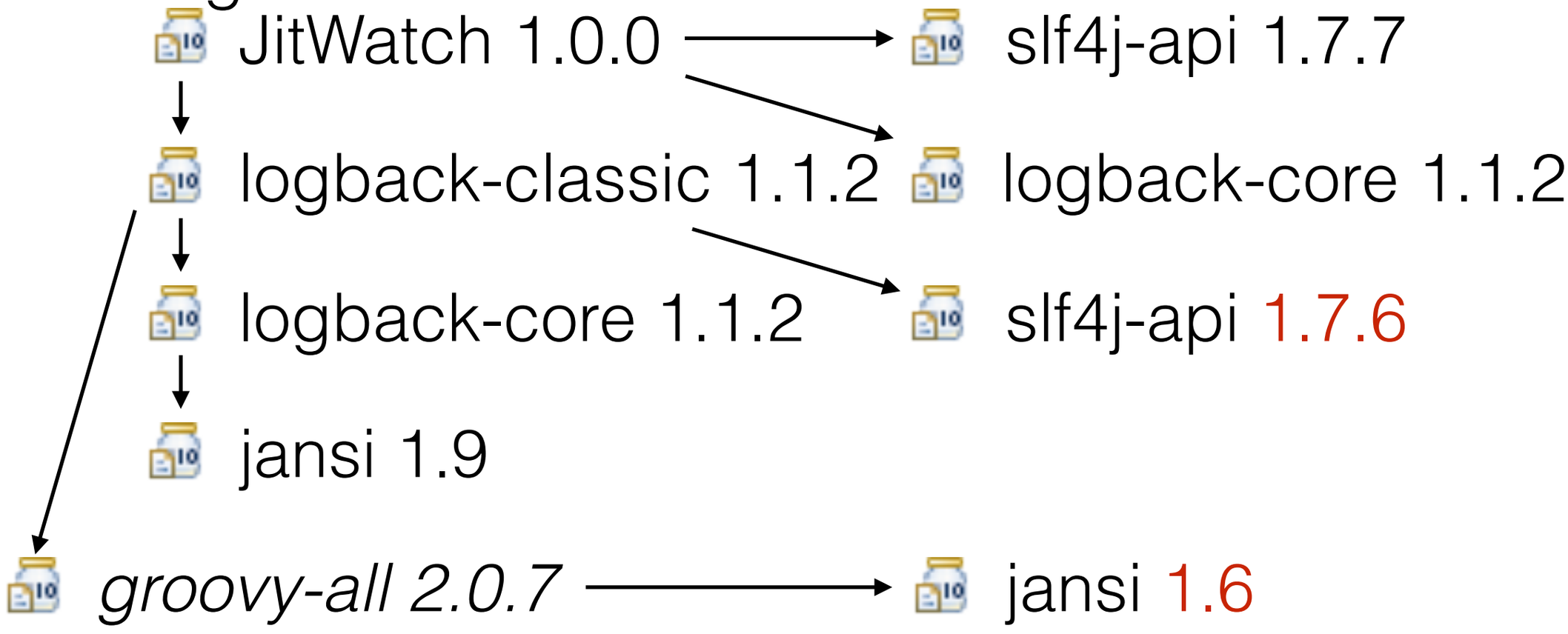
What Developers believe

- Transitive dependencies are easy to manage



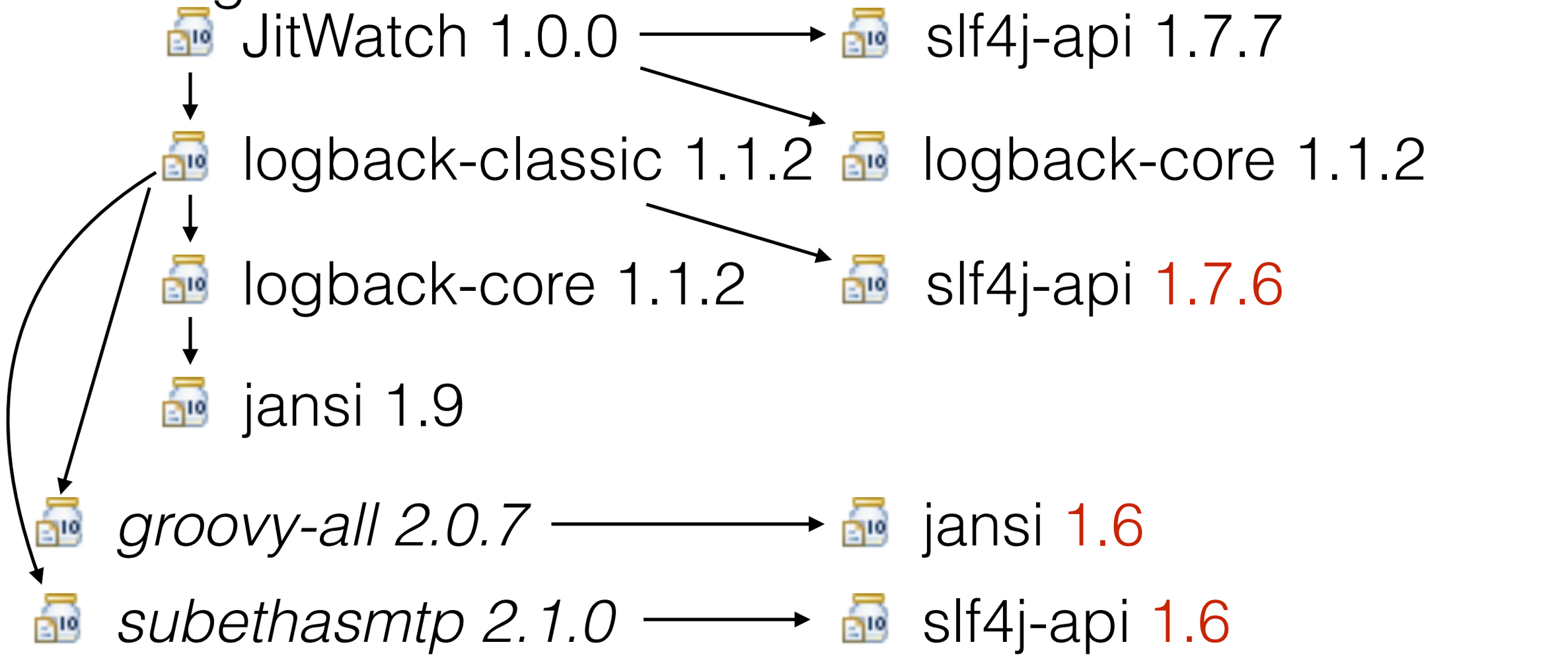
What Developers believe

- Optional transitive dependencies are easy to manage



What Developers believe

- Test optional transitive dependencies are easy to manage



** many dependencies not shown for brevity*

What Developers hope

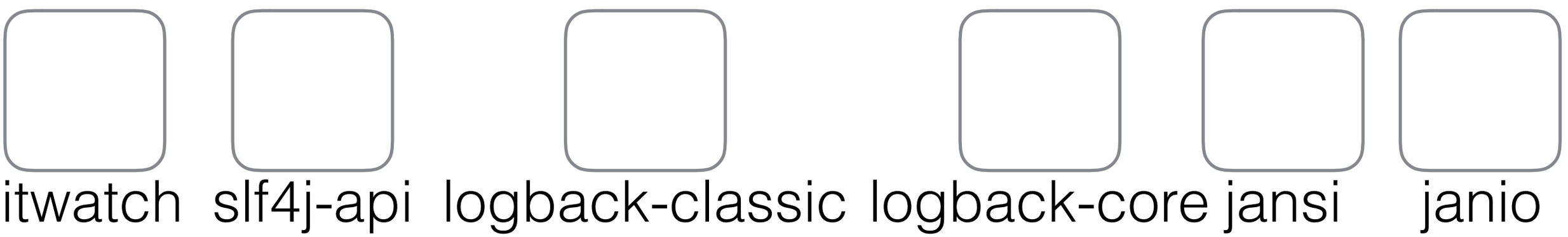
- It all just works

What the JVM sees

- Series of JARs loaded in a ClassLoader

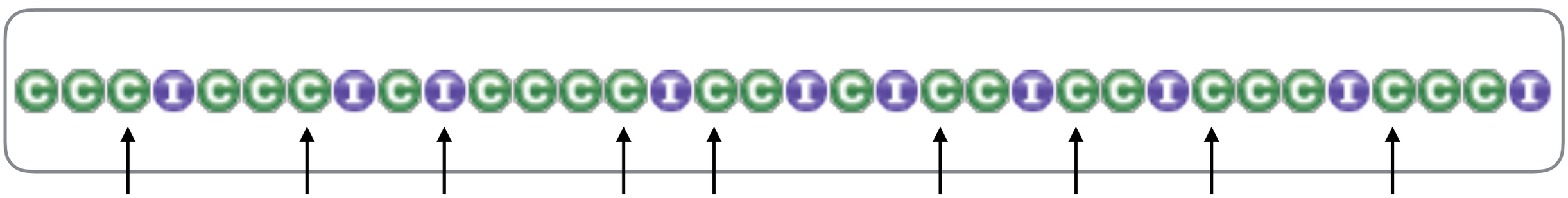
What the JVM sees

- JVM sees a one-dimensional list of classes



Looking up classes

- Resolving a class is stepping along to find it
- Packages are ignored
- No concept of modularity



Module busting

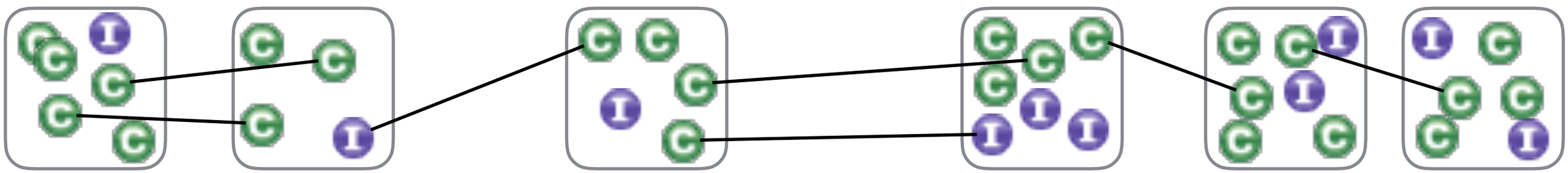
- Compile- & run- time dependencies may differ
- `Class.forName()` can bust through module barriers
- Dynamic instantiation may look up implementation
  - SLF4J – which logger to use
  - Hibernate – looking up database drivers
  - Annotation scanners walk the entire list of classes

Modularisation

- Only adds benefit once reaching a certain size
  - No-one needs a *Hello World* module
- Difficult to retro fit
  - (Just ask the Jigsaw team)
- Prevents accidental leakage between packages
  - `jetty-client` 6.1.23 -> `jetty` (server) 6.1.23

Packages are leaky

- Classes are oblivious to package boundaries



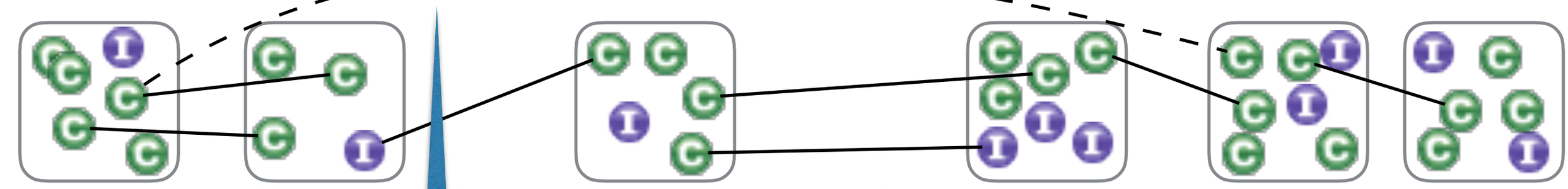
jitwatch slf4j-api logback-classic logback-core jansi janio

Friendly classes/interfaces are only available in the same package

However the same friendly package can be present in two or more JAR files

Packages are leaky

- Classes can follow transitive chain accidentally

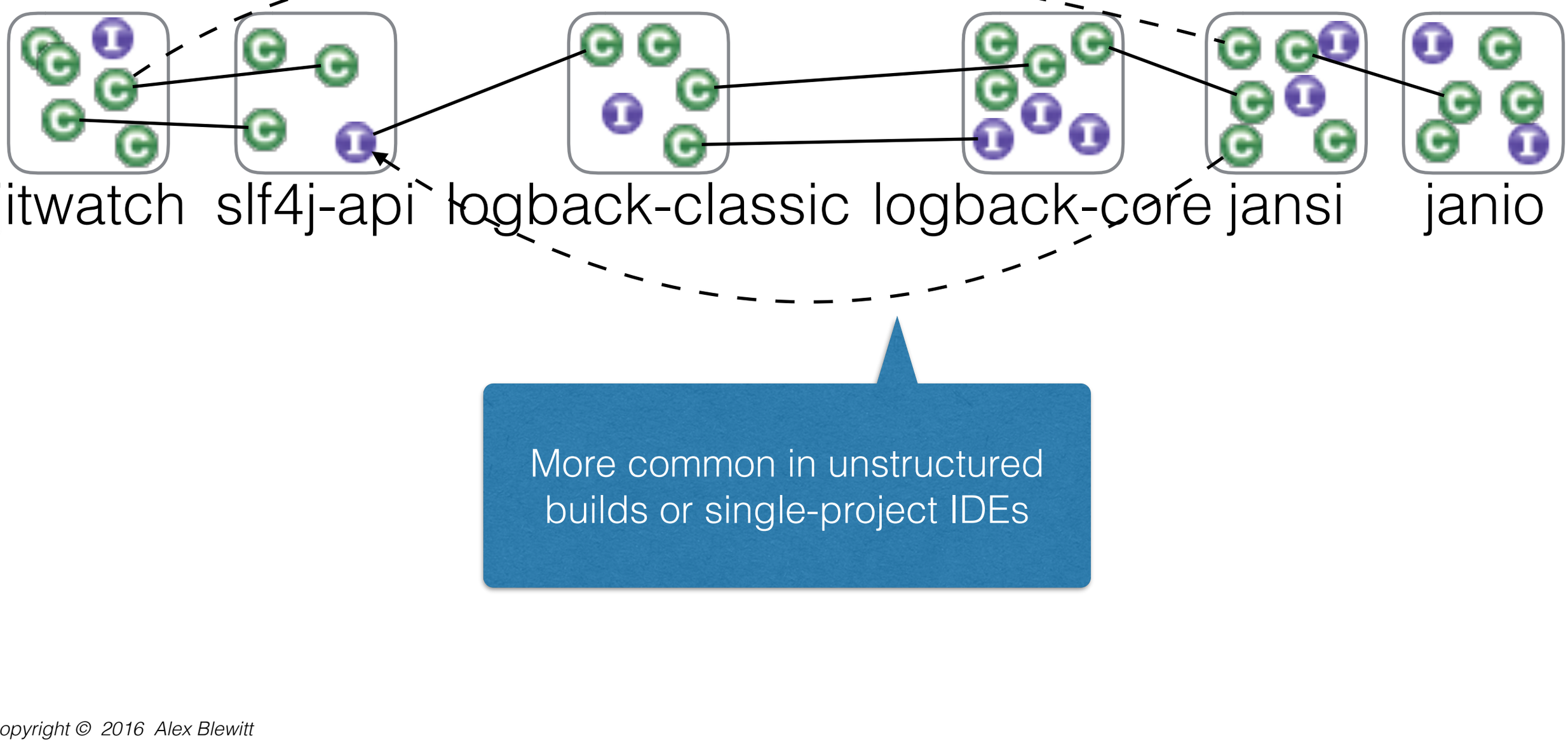


It is easy to accidentally depend on a class that comes from a transitive dependency without realising that it has happened

JIT watch does not do this; it's used as an example

Packages are leaky

- JARs can have cycles



Unstructured builds

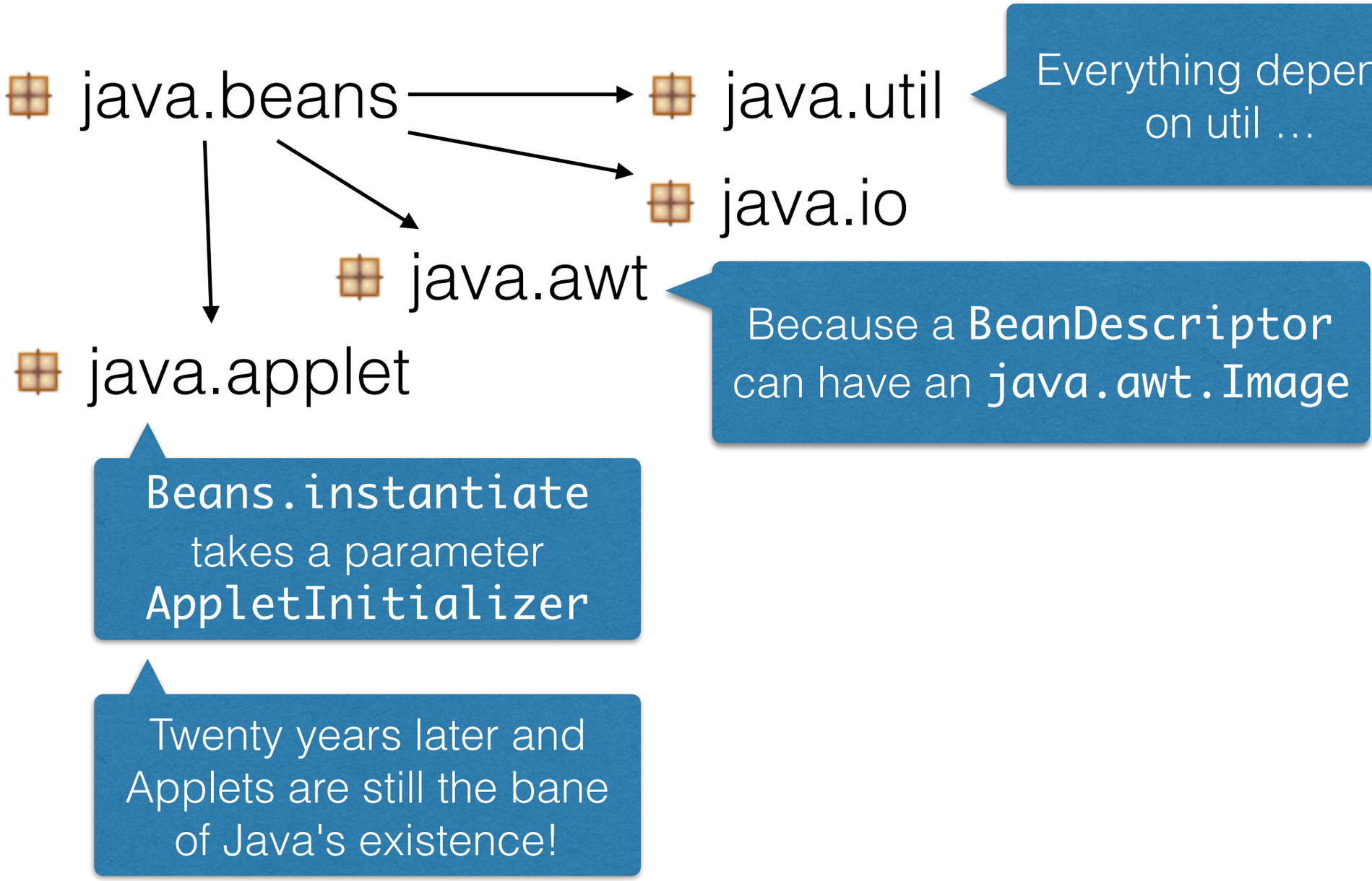
```
src/com/example/client/Client.java
src/com/example/server/Server.java
src/com/example/server/AnException.java
```

```
javac -d client com/example/client/*.java
javac -d server com/example/server/*.java
```

```
import com.example.server.AnException;
public class Client {
  void method() throws AnException {
  }
}
```

Client directory now contains client/com/example/server/AnException.class

Accidental dependencies



Easy come, hard go

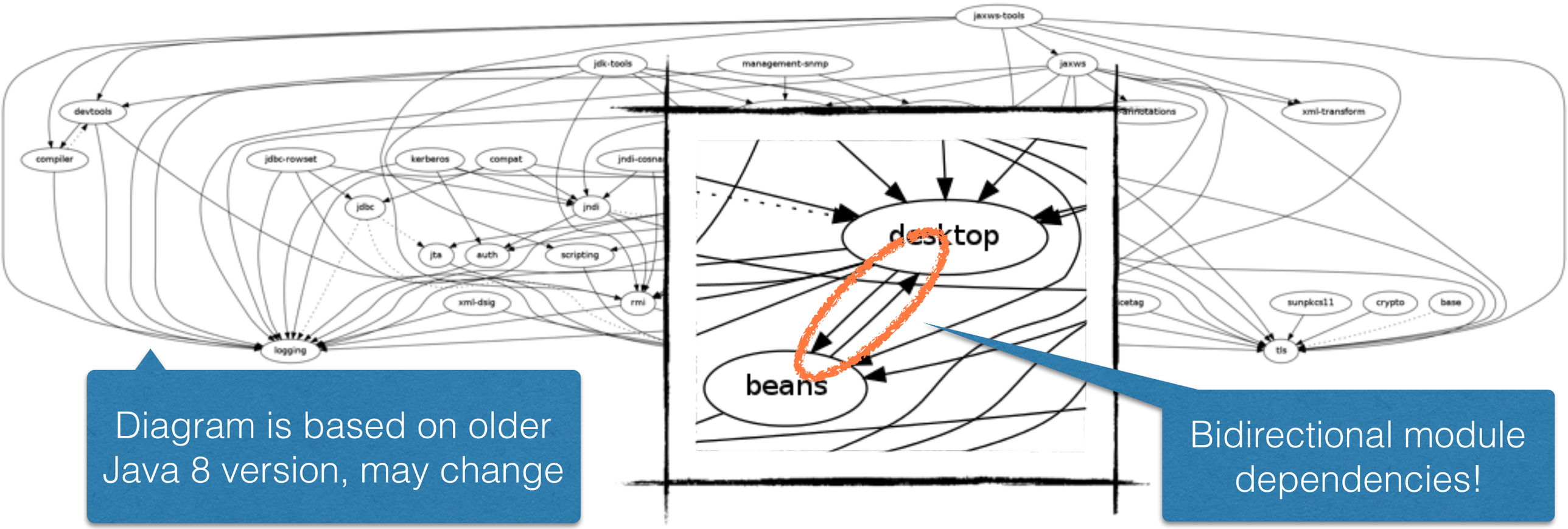


Diagram is based on older Java 8 version, may change

Bidirectional module dependencies!

http://cr.openjdk.java.net/~mchung/jigsaw/graphs/jdk8-b48.png

http://openjdk.java.net/projects/jigsaw/doc/jdk-modularization.html

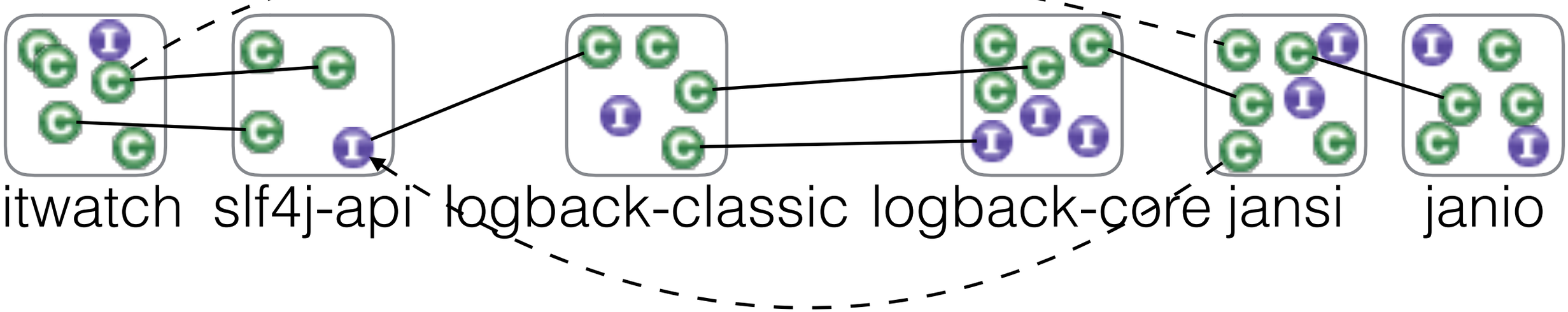
Beans and Desktop were merged in Java 9

It's amazing anything works at all …

How do we solve these problems?

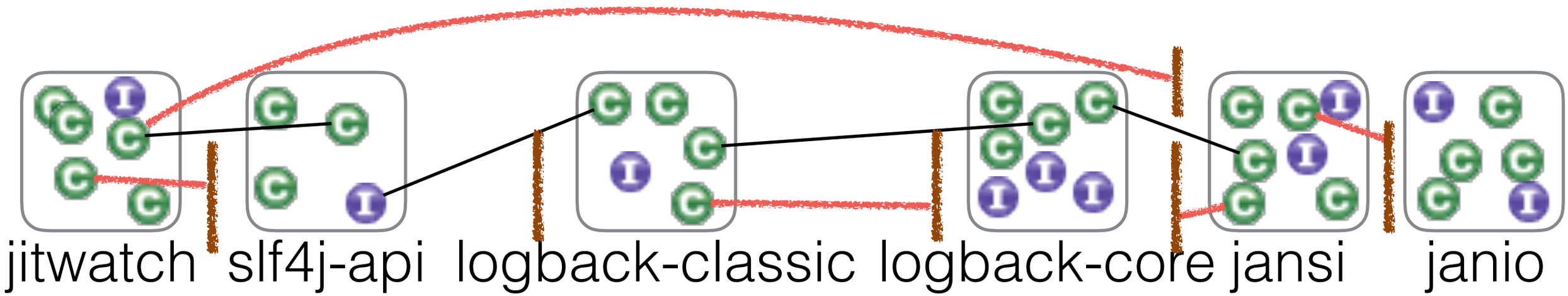
Module barriers

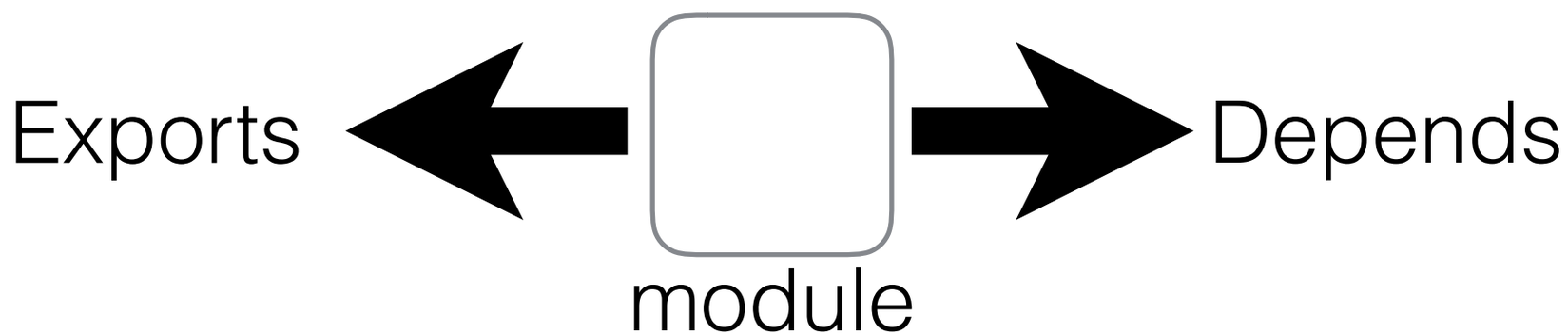
- Good fences make good neighbours



Module barriers

- Good fences make good neighbours





OSGi and Jigsaw

This is where they start to differ

OSGi

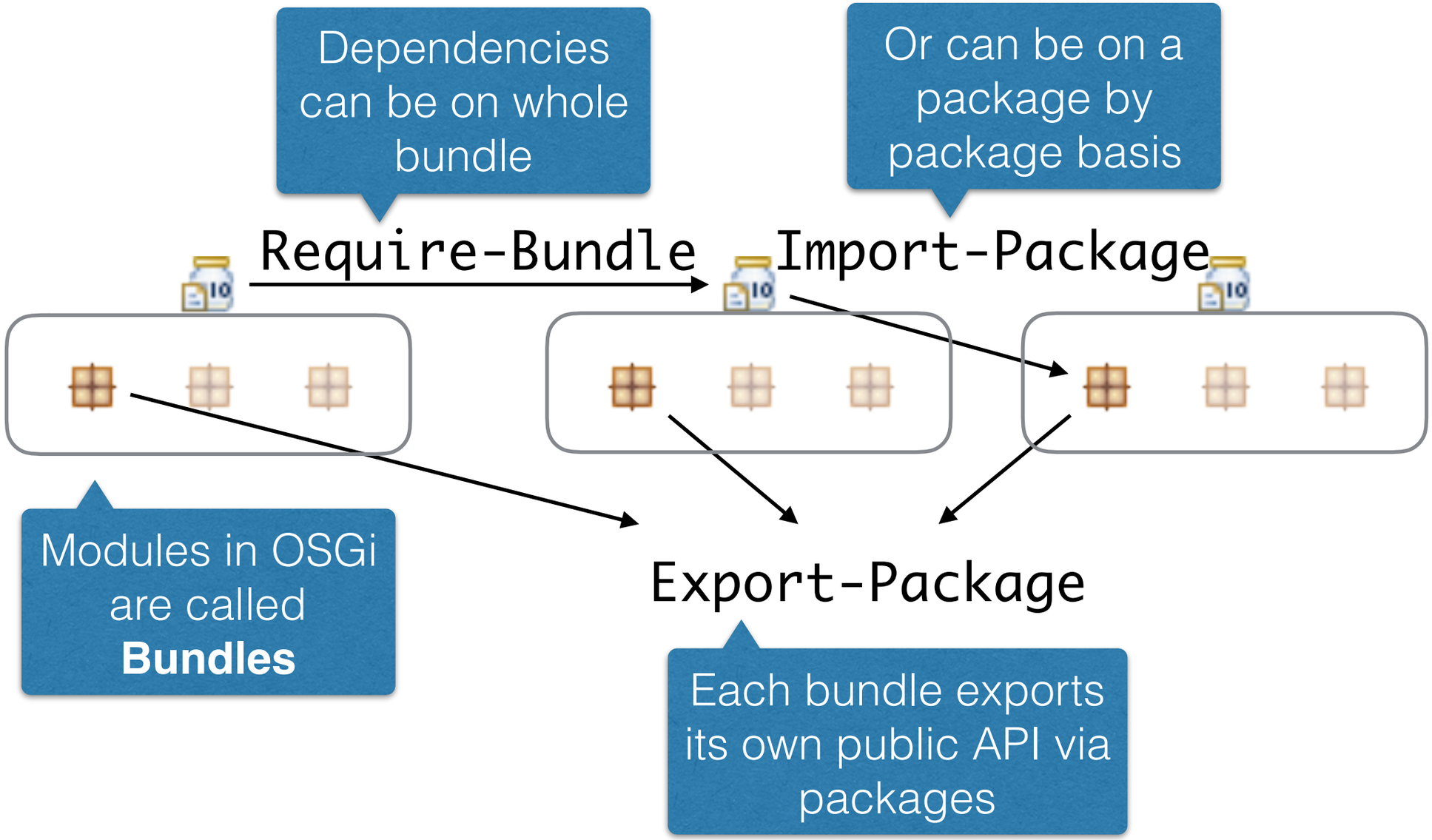
- Dynamic
- MANIFEST.MF
- Services
  - Export
    - Package
  - Import
    - Module*
    - Package
  - Versioned
    - Module
    - Package

Jigsaw

- Static
- module-info
- ServiceLoader
  - Export
    - Package
  - Import
    - Module*
  - No versioning

* Module dependencies may be declared as transitive

OSGi



MANIFEST.MF

```
Export-Package: com.example.ui.widgets
Import-Package: com.example.util
Require-Bundle: com.example.monolith
Bundle-SymbolicName: com.example.ui
Bundle-Version: 1.2.3
Bundle-ManifestVersion: 2
```

Manifest.MF chosen because it is first file in JAR and therefore easily accessible

How is the manifest used?

- Can be used by compiler to construct paths
- Can be used by runtime to ensure dependencies
- Can be used by IDEs to wire projects together
- Can be used to resolve dependencies from repo
- Can be used by GUIs to show content
- Can be used by humans for documentation

OSGi Frameworks

- Bundles are managed by a framework
  - Felix
  - Equinox
  - Knopflerfish
  - Prosyst

OSGi frameworks are like WebApp engines like Tomcat or Jetty

1. Start Tomcat
2. Drop in WAR file
3. Profit!

OSGi Frameworks

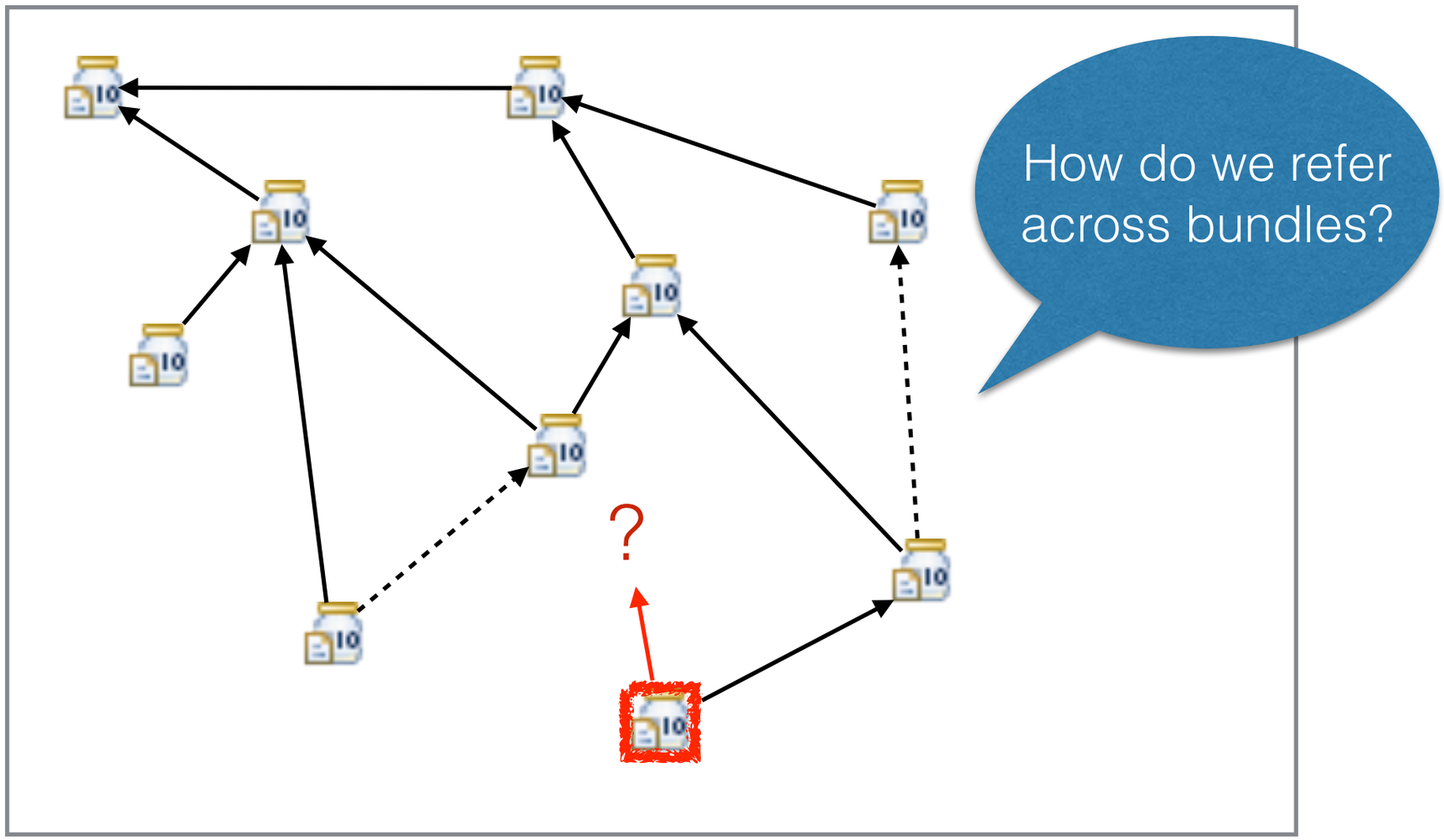
- Bundles are managed by a framework
  - Felix
  - Equinox
  - Knopflerfish
  - Prosyst

OSGi frameworks are like WebApp engines like Tomcat or Jetty

1. Start Tomcat
2. Drop in WAR file
3. Profit!

1. Start OSGi
2. Drop in JAR file
3. Profit!

OSGi Frameworks

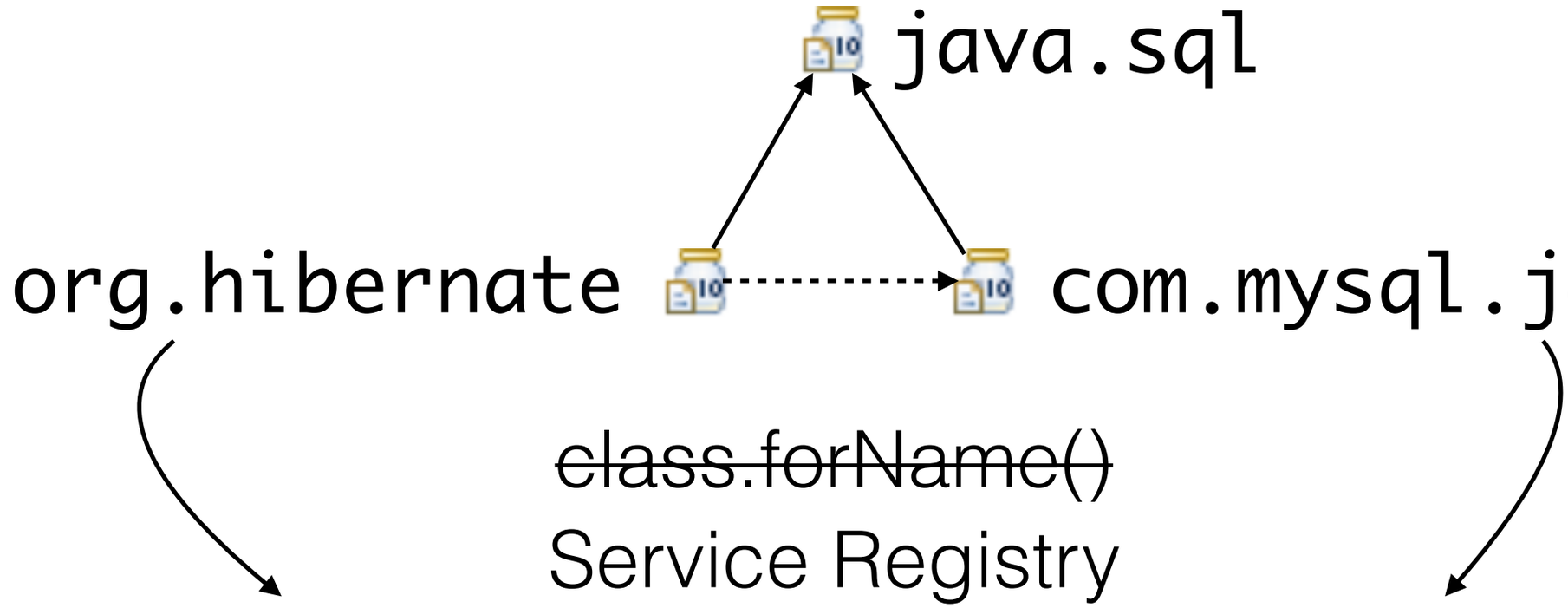


INSTALLED RESOLVED ACTIVE

OSGi Services

- Services provide a way of bundles to communicate
- Have a shared interface (e.g. `java.sql.Driver`)
- Bundles can provide service instances
- Bundles can require service instances
- Service registry stores service instances

OSGi Services



~~class.forName()~~

Service Registry

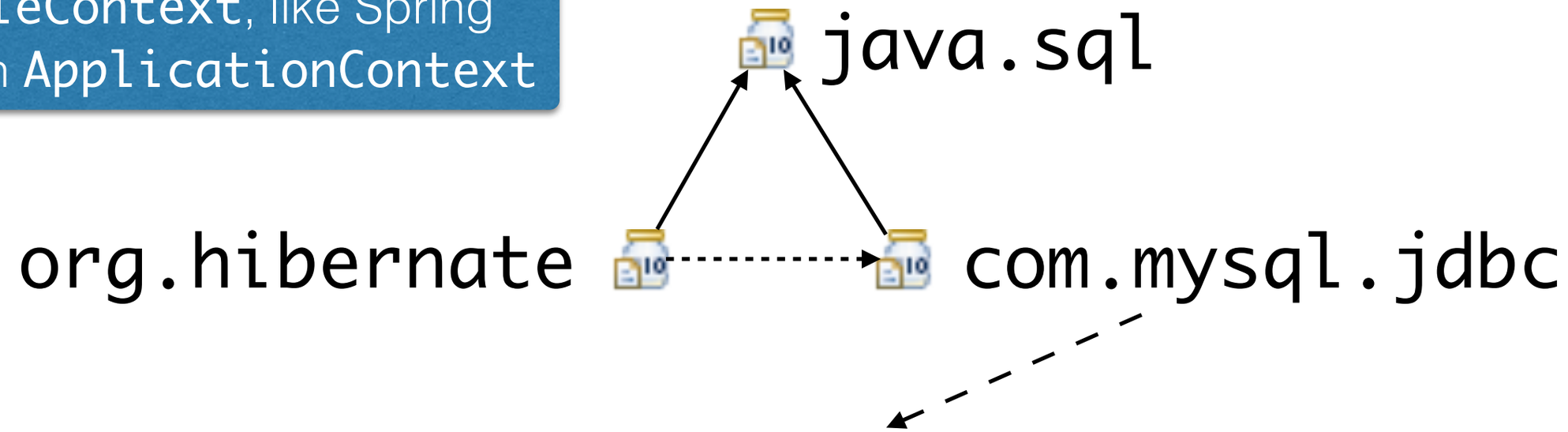
Interface Implementation

`java.sql.Driver com.mysql.jdbc.Driver`

Inversion of control

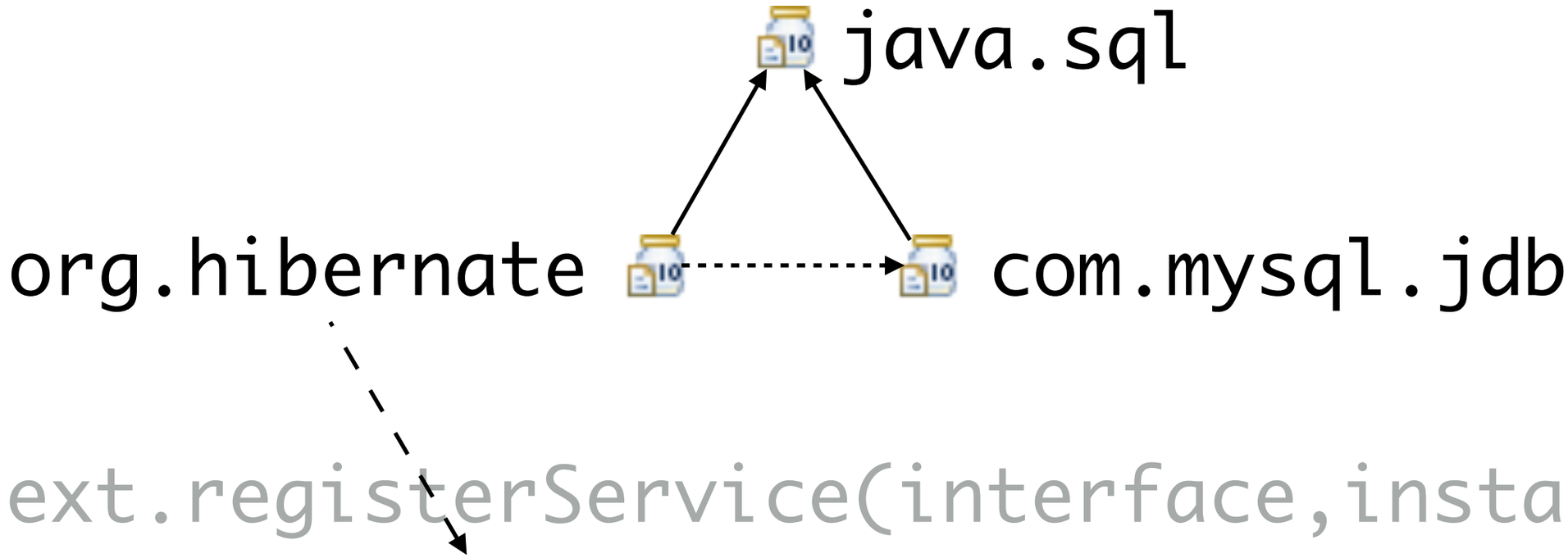
OSGi Services

The framework gives you the `BundleContext`, like Spring gives an `ApplicationContext`



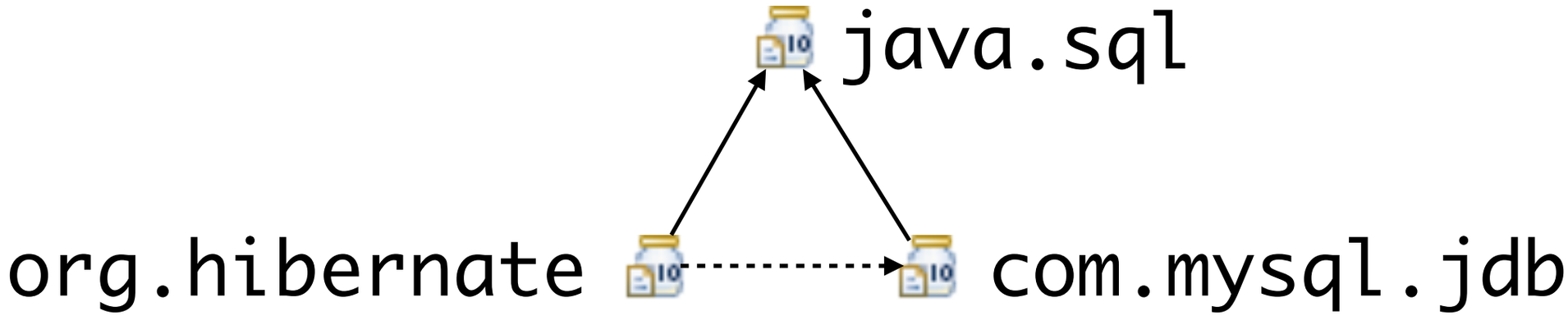

```
context.registerService(interface,instance)
```

OSGi Services



** actually it gets a Service from a ServiceReference*

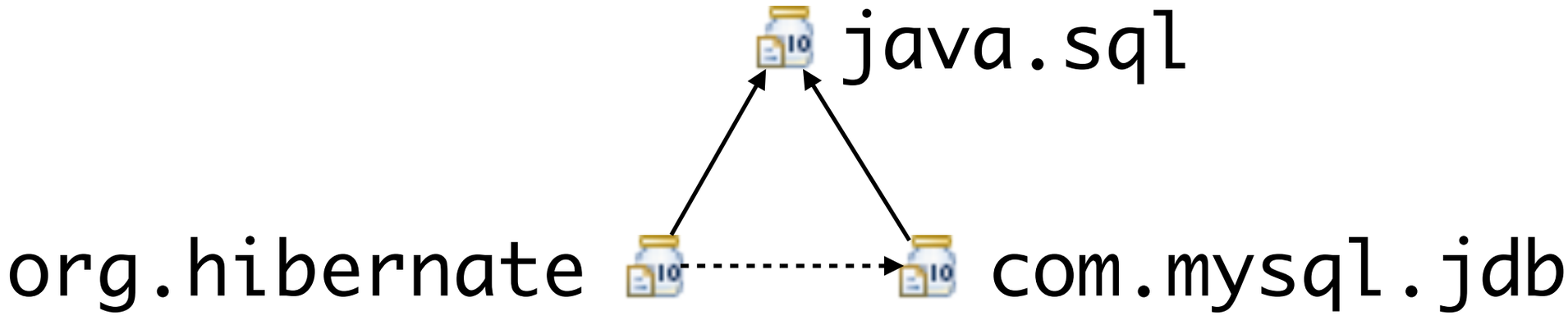
OSGi Services




```
context.registerService(interface,instance)
context.getService(interface)
```

Service users have to cope with the service not being present (null) and acting accordingly

OSGi Services



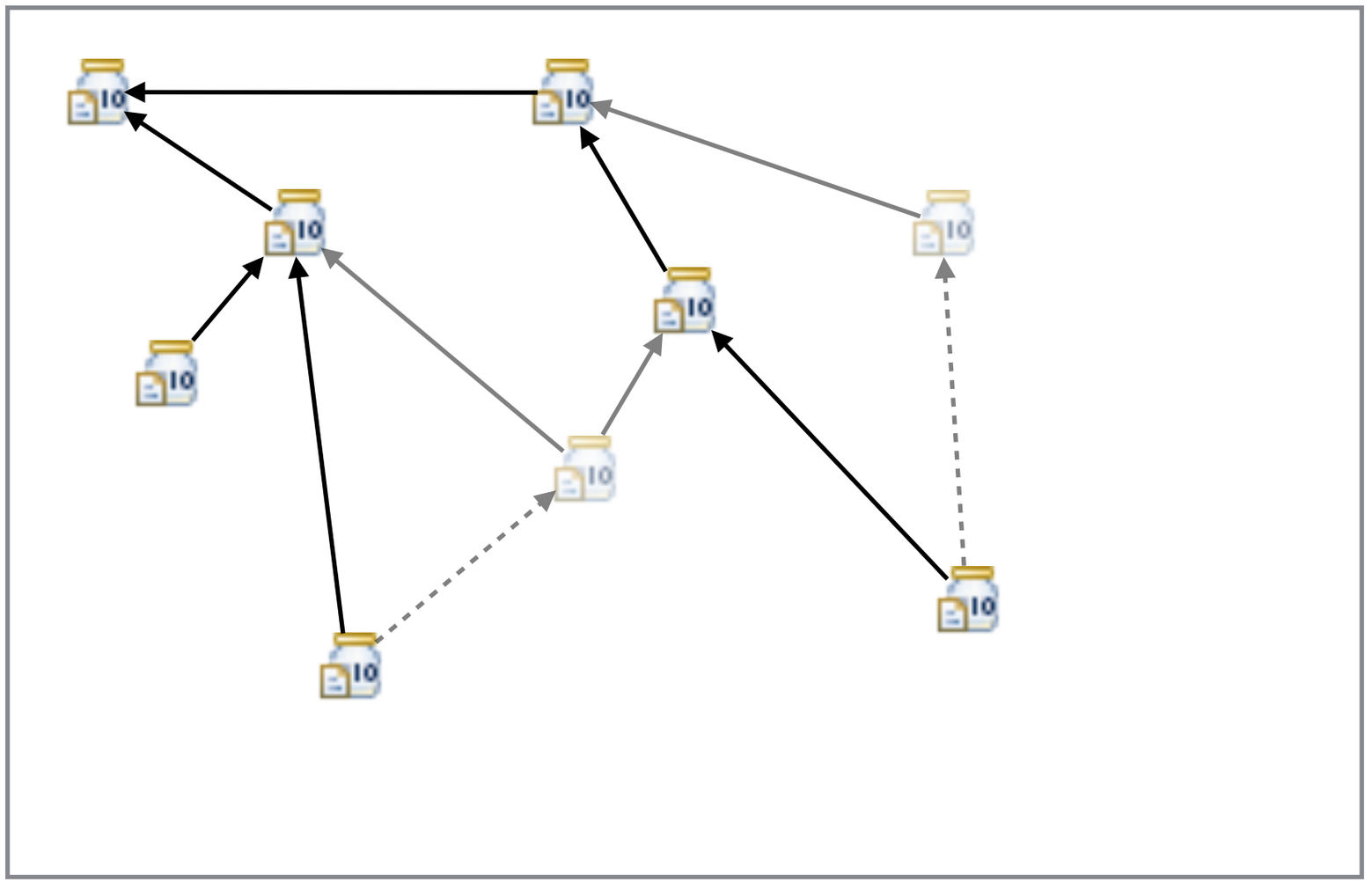
Declarative Services

I can need a `Driver` ⟷ I can provide a `Driver`

`<xml/>` `<xml/>`

Can be generated from annotations in code

Dynamic OSGi

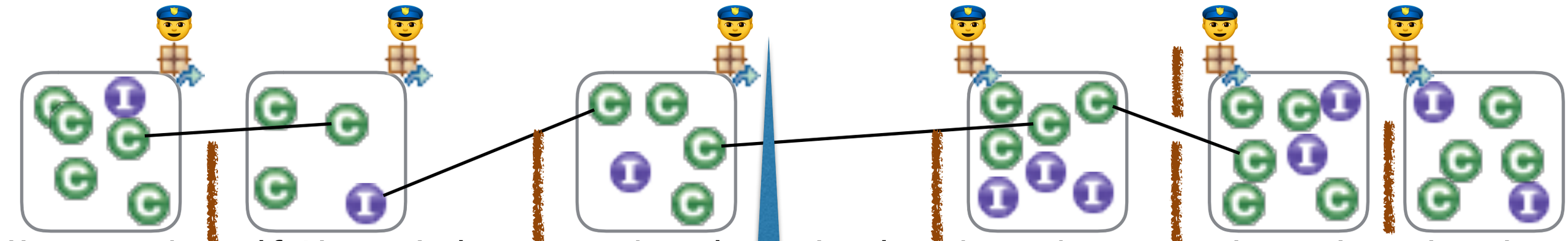


How does this work?

- Things can't come and go in a Java program!
  - Because classes are cached by the ClassLoader
- WebApps can come and go in a Tomcat server …
  - Each WebApp gets its own ClassLoader
  - When WebApp is removed, ClassLoader goes
  - Classes are recycled

Bundle barriers

- Each bundle has its own ClassLoader



Each module ClassLoader implements visibility rules

ClassLoader is the API police

When module is stopped, ClassLoader thrown away

Works in the same way as Tomcat and WebApps

ClassLoaders

- ClassLoaders are critical to the success of Java
- Allowed evolution of files -> JARs -> Jmods
- Propelled Java into enterprise with Servlets
- Popularised Java through AppletClassLoader

ClassLoaders are the guardians of the Java spirit

Getting Started with OSGi

1. Update existing build to generate OSGi metadata

   - Maven: `maven-bundle-plugin`
   - Gradle: `apply plugin: 'osgi'`
   - or: `apply plugin: 'biz.aQute.bnd'`

2. Install bundles into OSGi framework

3. Use OSGi console to inspect dependencies

Getting Started with OSGi

4. Use annotations to define service components

5. Decompose larger bundles into smaller ones

6. Review dependencies regularly

7. Use a tool to verify semantic versioning

OSGi and Jigsaw

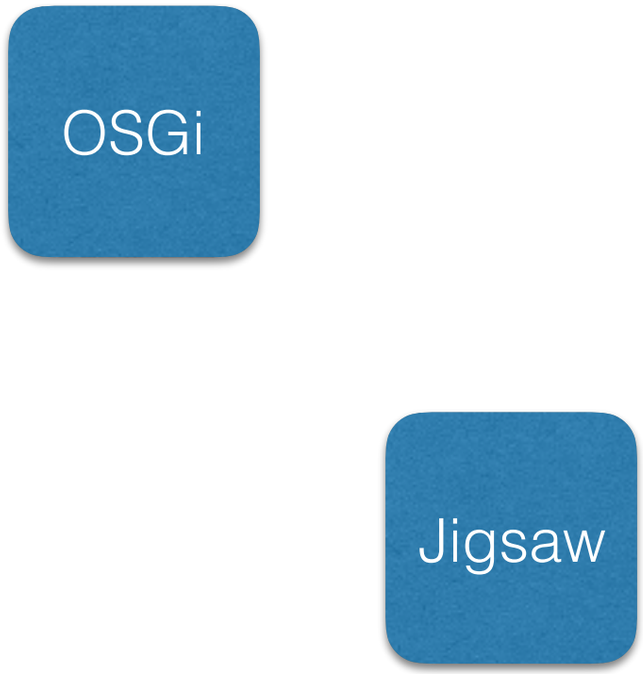
- OSGi and Jigsaw have different target markets
  - OSGi uses a dynamic application runtime
    - Jigsaw is about the modular JDK
- Both will encourage Java developers to modularise
- They share far more in common than differences

OSGi and Jigsaw

- Changes for Jigsaw will benefit OSGi & vice-versa
  - Fixes for `Class.forName()`
  - Proper segregation into modular boundaries
  - Using services to acquire implementations (IoC)
- OGSi and Jigsaw interoperability getting closer
  - "OSGi and Java 9 Modules Working Together" (Neil Bartlett)

    http://njbartlett.name/2015/11/13/osgi-jigsaw.html

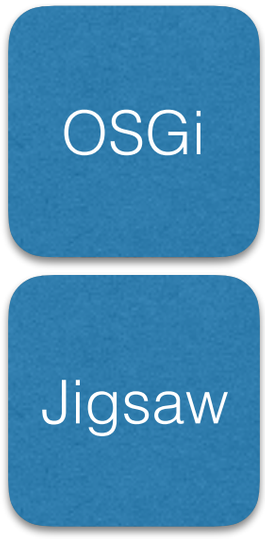
OSGi and Jigsaw



OSGi and Jigsaw

<u>Similarities</u>

- Module paths
- Strict separation
- Future of Java
- Services to separate



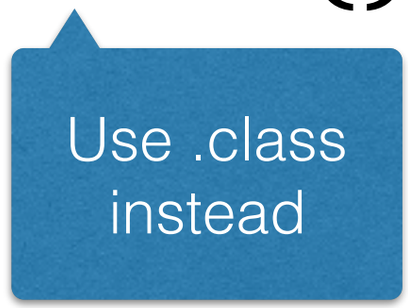
<u>Differences</u>

- Static vs Dynamic
- Package imports
- Service creation
- Versioning
- JARs vs Jmods
- Java Any vs Java 9

Can OSGi use Jigsaw modules or load Jmods?
Can Jigsaw modules use OSGi bundles?
Can Jigsaw services be created manually?

Future of Java

- The future of Java is modular
  - Will cause some pain
  - Will highlight less-than-perfect dependencies
  - Will cause problems for `Class.forName()` code
  - No "one true classpath"
  - Module paths are the path to success



Questions?

Modularity in Java with OSGi

Alex Blewitt
@alblue
Docklands.LJC
January 2016

http://docklandsljc.uk