

# JDK 9 New Features

杨晓峰, Java平台组, Oracle  
Apr, 2017

Java  
Your  
(Next)

# 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.

# JDK 9基本信息

- 目前建议的发布时间: GA July 2017, 每周更新:
  - Early access binaries + docs: <https://jdk9.java.net/>
  - Early access binaries with cutting edge Jigsaw: <https://jdk9.java.net/jigsaw/>
- OpenJDK:
  - 所有项目: <http://openjdk.java.net/projects/jdk9/>
  - 邮件列表: <http://mail.openjdk.java.net/mailman/listinfo/jdk9-dev>
  - 源代码: <http://hg.openjdk.java.net/jdk9/dev/>
  - [JEPs](#) (JDK Enhancement Proposals) 用来跟踪JDK新特性
- Adoption:
  - 欢迎大家参与 <http://mail.openjdk.java.net/pipermail/adoption-discuss/>  
<https://wiki.openjdk.java.net/display/quality/Quality+Outreach/>

# Java Platform Module System (JPMS)

# Jigsaw项目

- JSR 376: Java Platform Module System
  - 200: The Modular JDK
  - 201: Modular Source Code
  - 220: Modular Run-Time Images
  - 260: Encapsulate Most Internal APIs
  - 261: Module System
  - 282: jlink: The Java Linker

# 相信大家一定不陌生

```
common/hadoop-common-3.0.0-SNAPSHOT.jar:common/hadoop-nfs-3.0.0-SNAPSHOT.jar:common/lib/activation-1.1.jar:common/lib/apacheds-i18n-2.0.0-M15.jar:common/lib/apacheds-kerberos-codec-2.0.0-M15.jar:common/lib/api-asn1-api-1.0.0-M20.jar:common/lib/api-util-1.0.0-M20.jar:common/lib/asm-3.2.jar:common/lib/avro-1.7.4.jar:common/lib/commons-beanutils-1.7.0.jar:common/lib/commons-beanutils-core-1.8.0.jar:common/lib/commons-cli-1.2.jar:common/lib/commons-codec-1.4.jar:common/lib/commons-collections-3.2.1.jar:common/lib/commons-compress-1.4.1.jar:common/lib/commons-configuration-1.6.jar:common/lib/commons-digester-1.8.jar:common/lib/commons-httpclient-3.1.jar:common/lib/commons-io-2.4.jar:common/lib/commons-lang-2.6.jar:common/lib/commons-logging-1.1.3.jar:common/lib/commons-math3-3.1.1.jar:common/lib/commons-net-3.1.jar:common/lib/curator-client-2.7.1.jar:common/lib/curator-framework-2.7.1.jar:common/lib/curator-recipes-2.7.1.jar:common/lib/gson-2.2.4.jar:common/lib/guava-11.0.2.jar:common/lib/hadoop-annotations-3.0.0-SNAPSHOT.jar:common/lib/hadoop-auth-3.0.0-SNAPSHOT.jar:common/lib/hamcrest-core-1.3.jar:common/lib/htrace-core4-4.0.1-incubating.jar:common/lib/httpclient-4.2.5.jar:common/lib/httpcore-4.2.5.jar:common/lib/jackson-core-asl-1.9.13.jar:common/lib/jackson-jaxrs-1.9.13.jar:common/lib/jackson-mapper-asl-1.9.13.jar:common/lib/jackson-xc-1.9.13.jar:common/lib/java-xmlbuilder-0.4.jar:common/lib/jaxb-api-2.2.2.jar:common/lib/jaxb-impl-2.2.3-1.jar:common/lib/jcip-annotations-1.0.jar:common/lib/jersey-core-1.9.jar:common/lib/jersey-json-1.9.jar:common/lib/jersey-server-1.9.jar:common/lib/jets3t-0.9.0.jar:common/lib/jettison-1.1.jar:common/lib/jetty-6.1.26.jar:common/lib/jetty-util-6.1.26.jar:common/lib/jsch-0.1.51.jar:common/lib/json-smart-1.1.1.jar:common/lib/jsp-api-2.1.jar:common/lib/jsr305-3.0.0.jar:common/lib/junit-4.11.jar:common/lib/log4j-1.2.17.jar:common/lib/mockito-all-1.8.5.jar:common/lib/netty-3.6.2.Final.jar:common/lib/nimbus-jose-jwt-3.9.jar:common/lib/paranamer-2.3.jar:common/lib/protobuf-java-2.5.0.jar:common/lib/servlet-api-2.5.jar:common/lib/slf4j-api-1.7.10.jar:common/lib/slf4j-log4j12-1.7.10.jar:common/lib/snappy-java-1.0.4.1.jar:common/lib/stax-api-1.0-2.jar:common/lib/xmlenc-0.52.jar:common/lib/xz-1.0.jar:common/lib/zookeeper-3.4.6.jar:hdfs/hadoop-hdfs-3.0.0-SNAPSHOT.jar:hdfs/hadoop-hdfs-nfs-3.0.0-SNAPSHOT.jar:hdfs/lib/commons-daemon-1.0.13.jar:hdfs/lib/hadoop-hdfs-client-3.0.0-SNAPSHOT.jar:hdfs/lib/hpack-0.11.0.jar:hdfs/lib/leveldbjni-all-1.8.jar:hdfs/lib/netty-all-4.1.0.Beta5.jar:hdfs/lib/okhttp-2.4.0.jar:hdfs/lib/okio-1.4.0.jar:hdfs/lib/xercesImpl-2.9.1.jar:mapreduce/hadoop-mapreduce-client-app-3.0.0-SNAPSHOT.jar:mapreduce/hadoop-mapreduce-client-common-3.0.0-SNAPSHOT.jar:mapreduce/hadoop-mapreduce-client-core-3.0.0-SNAPSHOT.jar:mapreduce/hadoop-mapreduce-client-hs-3.0.0-SNAPSHOT.jar:mapreduce/hadoop-mapreduce-client-hs-plugins-3.0.0-SNAPSHOT.jar:mapreduce/hadoop-mapreduce-client-jobclient-3.0.0-SNAPSHOT.jar:mapreduce/hadoop-mapreduce-client-nativeclient-3.0.0-SNAPSHOT.jar:mapreduce/hadoop-mapreduce-client-shuffle-3.0.0-SNAPSHOT.jar:mapreduce/hadoop-mapreduce-examples-3.0.0-SNAPSHOT.jar:yarn/hadoop-yarn-api-3.0.0-SNAPSHOT.jar:yarn/hadoop-yarn-applications-distributedshell-3.0.0-SNAPSHOT.jar:yarn/hadoop-yarn-applications-unmanaged-am-launcher-3.0.0-SNAPSHOT.jar:yarn/hadoop-yarn-client-3.0.0-SNAPSHOT.jar:yarn/hadoop-yarn-common-3.0.0-SNAPSHOT.jar:yarn/hadoop-yarn-registry-3.0.0-SNAPSHOT.jar:yarn/hadoop-yarn-server-applicationhistoryservice-3.0.0-SNAPSHOT.jar:yarn/hadoop-yarn-server-common-3.0.0-SNAPSHOT.jar:yarn/hadoop-yarn-server-nodemanager-3.0.0-SNAPSHOT.jar:yarn/hadoop-yarn-server-resourcemanager-3.0.0-SNAPSHOT.jar:yarn/hadoop-yarn-server-sharedcachemanager-3.0.0-SNAPSHOT.jar:yarn/hadoop-yarn-server-web-proxy-3.0.0-SNAPSHOT.jar:yarn/lib/aopalliance-1.0.jar:yarn/lib/commons-math-2.2.jar:yarn/lib/curator-test-2.7.1.jar:yarn/lib/fst-2.24.jar:yarn/lib/guice-3.0.jar:yarn/lib/guice-servlet-3.0.jar:yarn/lib/javassist-3.18.1-GA.jar:yarn/lib/javax.inject-1.jar:yarn/lib/jersey-client-1.9.jar:yarn/lib/jersey-guice-1.9.jar:yarn/lib/objenesis-2.1.jar
```

# Jigsaw的目标

- 提供两方面的基本能力:
  - 可靠的配置，以替换目前被证明很脆弱、易出错的class-path机制(See: Jar Hell)
  - 强封装性(encapsulation)
- 带来的好处:
  - 可扩展性+可维护性+安全

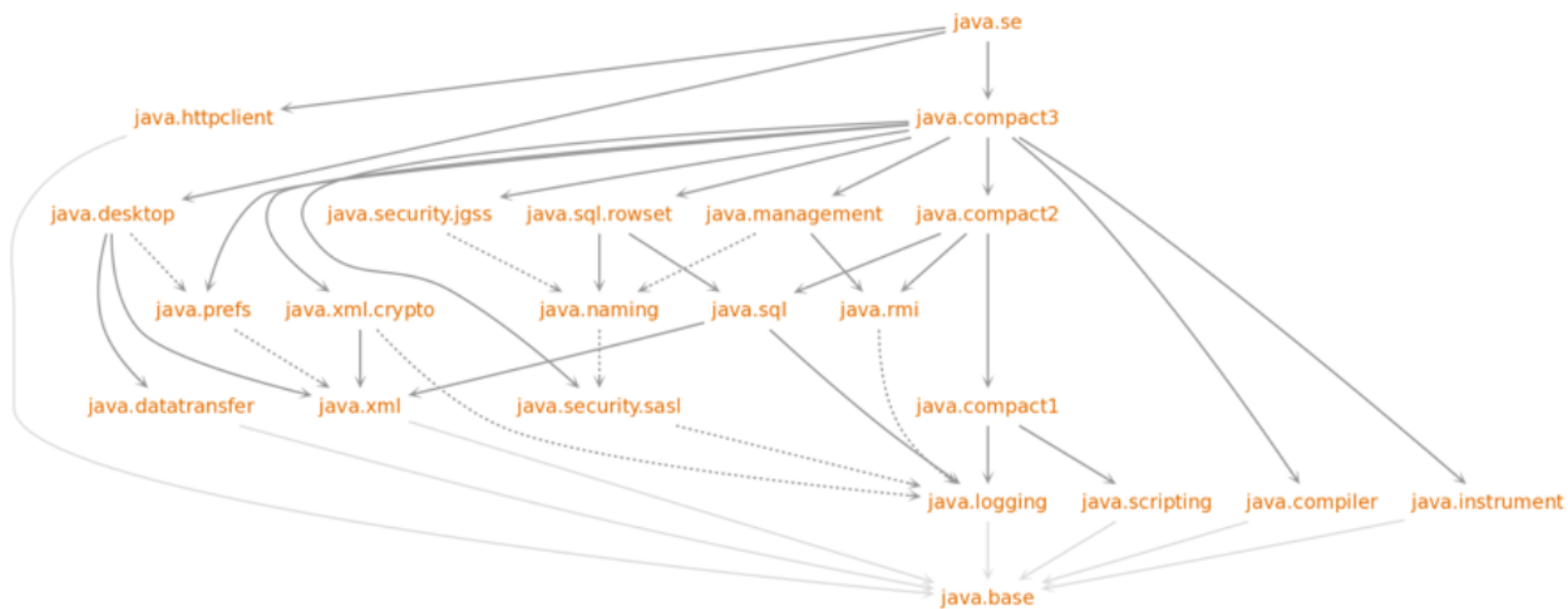
# 模块系统组成

- 新增或修改工具和API，以在编译、链接和运行时支持模块
- Javac/java:
  - 增加对模块的支持，比如module path
  - 各种针对模块的选项
- jlink工具
  - Java新增了一个可选的链接阶段（linking Phase）
  - 方便的创建最小依赖关系的Java运行时



# 模块化JDK

- JDK自身的代码被划分为一组模块



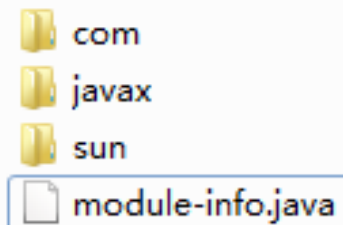
# Java模块

***stats.core***

```
com.acme.stats.core.clustering.Cluster  
com.acme.stats.core.regression.Linear  
:
```

Consisted by:

- Module descriptor (module-info.java)
- Source codes



```
com  
javax  
sun  
module-info.java
```

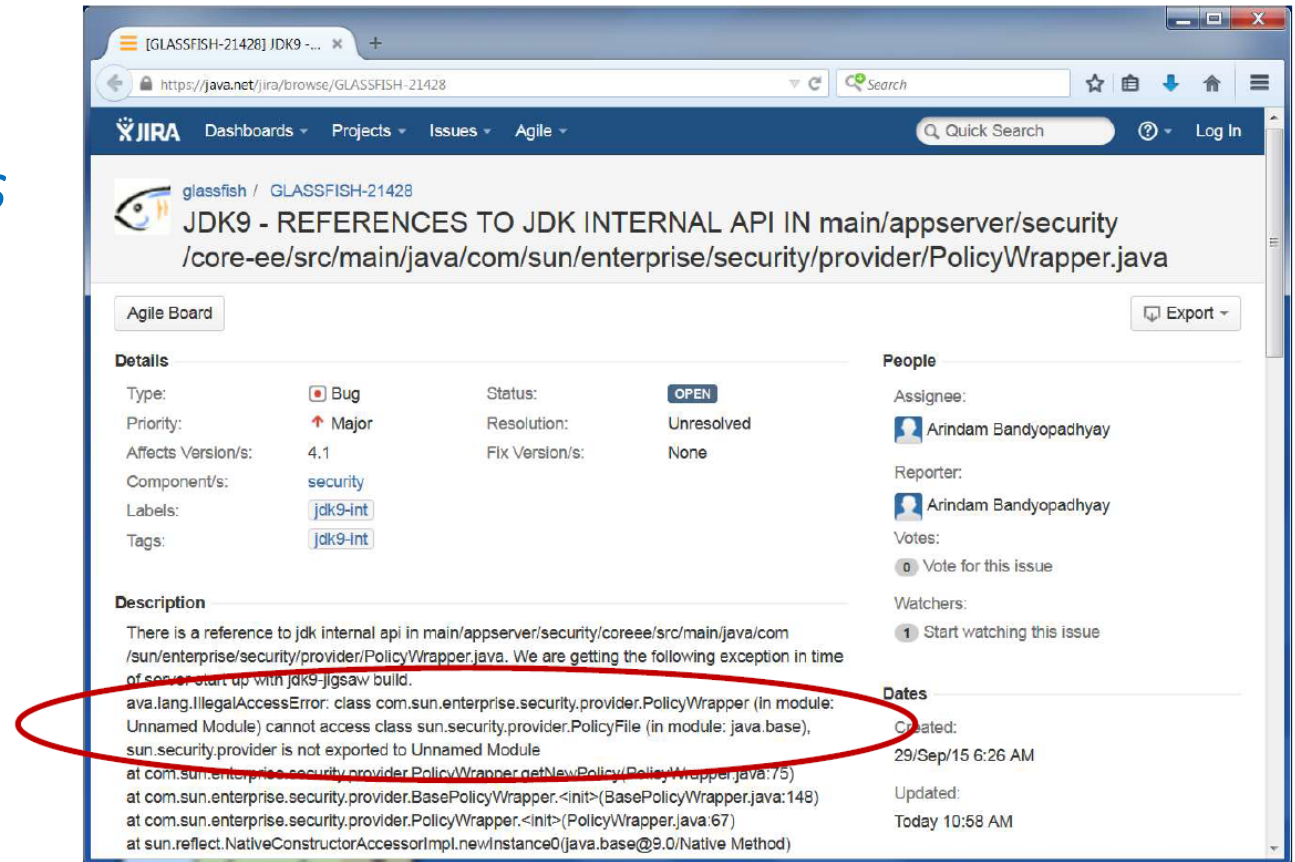
# 典型的Module Descriptor

```
module java.naming {  
    requires java.security.sasl;  
    exports javax.naming;  
    exports javax.naming.directory;  
    exports javax.naming.event;  
    exports javax.naming.ldap;  
    exports javax.naming.spi;  
    exports com.sun.jndi.toolkit.ctx to jdk.naming.dns;  
    exports com.sun.jndi.toolkit.url to java.corba, jdk.naming.dns, jdk.naming.rmi;  
    uses javax.naming.ldap.StartTlsResponse;  
    uses javax.naming.spi.InitialContextFactory;  
    provides java.security.Provider with sun.security.provider.certpath.ldap.JdkLDAP;  
}
```

# 深入分析JPMS

# Accessibility (JDK 9)

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



access-control boundaries can be workarounded with

`--add-exports <source-module>/<package>=<target-module>(<target-module>)*`

# Readability – direct or implied

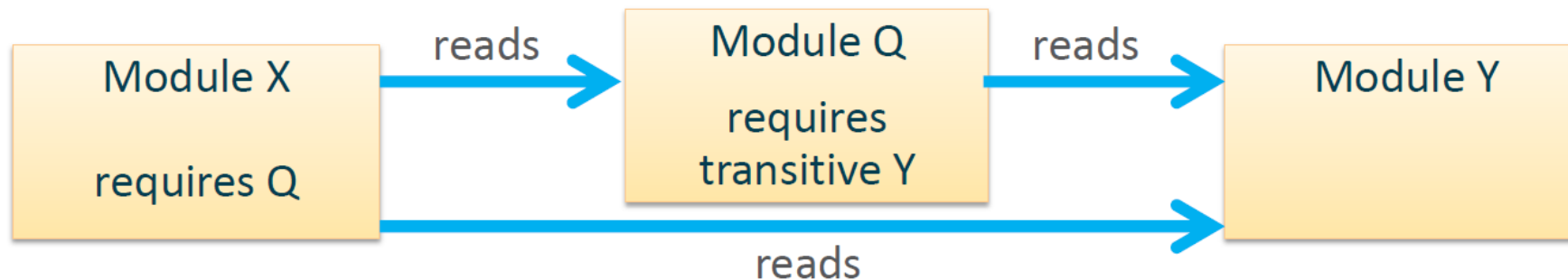
X reads Y if:

- X requires Y



or

- X reads Q, and Q requires transitive Y



Readability can be increased with

**--add-reads <source-module>=<target-module>**

# 强封装性对反射机制的影响(Reflection)

- Restrict deep reflection to access non-public elements
  - Badly affect proxies with 'setAccessible()'
- To allow all of the non-public elements of a package:
  - A new concept of open modules
  - Introduce the new per-package directive “open”
- Command-line equivalences
  - “--add-opens”

# Java模块的分类

- Explicit named modules
- Automatic named modules – put jars to module path
- Unnamed module – those loaded by class path
- behaves differently:
  - Unnamed modules:
    - Reads all modules
    - Exports all its packages, but named modules cannot access types in unnamed modules
  - Automatic named modules:
    - Reads all modules
    - Exports all its packages



# 模块方面的API

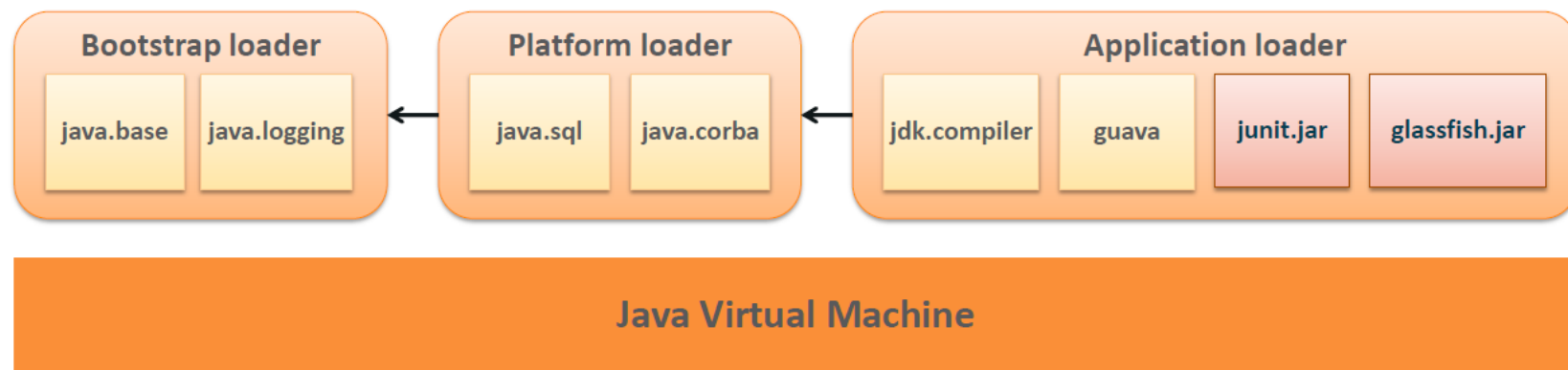
- New/extended APIs:
  - `java.lang.module.*`
  - `Java.lang.Module/ModuleLayer`
  - Extends: `Java.lang.Class::getModule()`
- Layers control the relationship between modules and class loaders
  - Hierarchical
  - Boot layer created at JVM startup
  - A container application can create a new layer

# Samples to instance Module/Layer

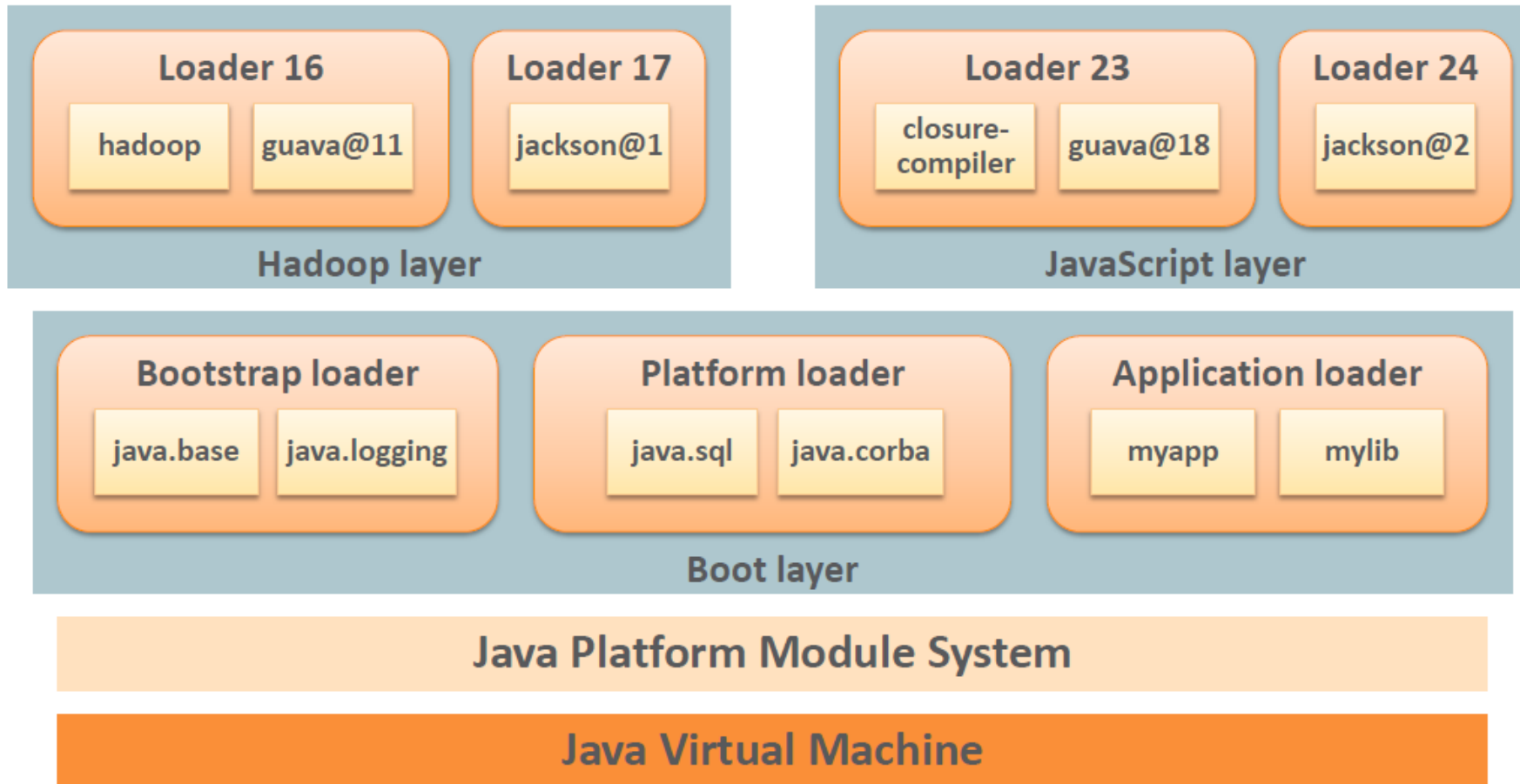
```
Path dir = Paths.get("path_to_your_modules");  
ModuleFinder finder = ModuleFinder.of(dir);  
// Get boot layer  
ModuleLayer bootLayer = ModuleLayer.boot();  
Configuration parent = bootLayer.configuration();  
// Create a new configuration  
Configuration cf = parent.resolve(finder, ModuleFinder.of(), Set.of(modules...));  
ClassLoader scl = ClassLoader.getSystemClassLoader();  
// Create a new ModuleLayer  
ModuleLayer layer = bootLayer.defineModulesWithOneLoader(cf, scl);
```

# ClassLoader演进

- Bootstrap class loader
- Platform class loader
  - was extension classloader
  - Drop extension mechanism
  - Modules de-privileged for security improvement
    - Side effect: NOT all Java SE types are visible to boot loader
- Application class loader
- Above two are no longer URLClassLoader



# Layers, classloaders and modules



# JPMS应用举例

# 基本用例

- 查看所有或特定模块

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

```
java --list-modules java.se
```

- 编译模块

```
javac --module-path mod_dir \  
      --add-modules java.desktop \  
      --module-source-path src_dir java_files
```

- 运行模块化应用

```
java --module-path mod_dir -m my_mod/MainClass
```

- 根据依赖关系定制java运行环境

```
$ jlink --module-path jmods/ --add-modules java.desktop --output myimage
```

```
$ myimage/bin/java --list-modules
```

```
java.base@9
```

```
java.datatransfer@9
```

```
java.desktop@9
```

```
java.prefs@9
```

```
java.xml@9
```

```
$ myimage/bin/java -jar jedit.jar
```

# 模块化现有应用

- 如果仍然按照classpath形式执行
  - 实际是作为 unnamed modules
  - 建议使用JDK 9编译和运行以测试兼容性
- 尝试进行模块化:
  - 把Jar文件放到 module path, Java会将其转化为automatic modules:
  - 当然也可以重构成named modules



# 典型思路

- 分析依赖关系:

```
$ jdeps -s lib/myapp.jar lib/mylib.jar
```

```
myapp.jar -> lib/jackson-core-2.6.2.jar
```

```
myapp.jar -> lib/jackson-databind-2.6.2.jar
```

```
myapp.jar -> mylib.jar
```

```
myapp.jar -> java.base
```

```
myapp.jar -> java.sql
```

```
mylib.jar -> java.base
```

- 生成 module descriptor, 进行相关重构

```
$ jdeps --gen-module-info 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
```

# 常见问题

- 初步定义的模块边界很可能是不完善的，比如：
  - circling references, “split packages”等问题
- 使用了JDK internal APIs:
  - 可以参考文档替换相应API:  
<https://wiki.openjdk.java.net/display/JDK8/Java+Dependency+Analysis+Tool>
  - 也可以临时规避模块封装导致的可见性问题，：  
`--add-exports java.base/jdk.internal.misc=Your_mod`  
(or use ALL-UNNAMED for unnamed module)
- 其他JDK 9引入的兼容性问题，比如：
  - 对tools.jar或者rt.jar的依赖（已经删除）

# Troubleshooting选项

- 模块功能的常用诊断选项:
  - `-Xdiag:resolver` expose detail of constructing the initial module graph.
  - `-Dsun.reflect.debugModuleAccessChecks` thread dump to help investigate accessibility issues in reflection
  - `-Xlog:modules=[debug|trace]` verbose logging for modules

# JVM新特性

# Ahead-of-Time Compilation

## JEP 295

- 通过提供类库风格的机制(library-like mechanism)以降低启动开销
- 新的编译工具: jaotc
- 示例:
  - Compile:  
`jaotc --output libHelloWorld.so HelloWorld.class`
  - Run:  
`java -XX:AOTLibrary=./libHelloWorld.so HelloWorld`

# G1作为默认GC

## JEP 248

- 在通用场景下，通常认为gc延迟的重要性高于吞吐量
- G1是一个非常健壮并经过充分测试的收集器：
  - 直接设定延迟目标，能够达到延迟 SLAs
  - 最坏场景的延迟表现优于CMS (设计原理导致碎片化问题)

# CMS的未来?

- 移除Incremental CMS (iCMS, deprecated in 8)
- 请注意是 incremental mode of CMS
- 建议Deprecate CMS (JEP 291)

# Remove GC Combinations Deprecated in JDK 8

## JEP 214

- 相应的命令行选择不再允许
- 请注意不是Warning而是error

“Unrecognized option: xxx

Error: Could not create the Java Virtual Machine... fatal error... Program will exit.”

Flags	GC Configuration
-XX:-UseParNewGC -XX:+UseConcMarkSweepGC	DefNew + CMS
-XX:+UseParNewGC	ParNew + SerialOld
-Xincgc	ParNew + iCMS
-XX:+CMSIncrementalMode -XX:+UseConcMarkSweepGC	ParNew + iCMS
-XX:+CMSIncrementalMode -XX:+UseConcMarkSweepGC -XX:-UseParNewGC	DefNew + iCMS



# Unified logging for JVM and gc

[JEP 158](#) and [JEP 271](#)

- Fine-grained, easy to configure JVM logging
- -Xlog:gc 基本类似于“-XX:PrintGC”
- 示例:
  - Xlog:help
  - Xlog:disable
  - Xlog:gc
  - Xlog:gc=trace:file=gctrace.txt:updatetimeillis,pids:filecount=5,filesize=1024

# 其他类库、语言和新特性

# Process API Updates

## JEP 102

- On `java.lang.Process` new methods to get the PID, direct children, and all descendants
- New `java.lang.ProcessHandle` interface

```
ProcessHandle current = ProcessHandle.current();  
current.info().totalCpuDuration().ifPresent(d -> {  
    System.out.println("Total cpu duration :" + d);  
});  
current.children().forEach(p -> System.out.println("Pid:" + p.getPid()));
```

# Convenience Factory Methods for Collections

## JEP 269

是否已经对下面的代码感到厌倦：

```
Set<String> set = new HashSet<>();  
set.add("a");  
set.add("b");  
set.add("c");  
set = Collections.unmodifiableSet(set);
```

or

```
Set<String> set =  
Collections.unmodifiableSet(new HashSet<>(  
Arrays.asList("a", "b", "c"))));
```

- Now:

```
Set<String> set = Set.of("a", "b", "c");
```

- 新的静态工厂方法: “of” on Set, List, and Map.
- Return unmodifiable collections.
- Randomized iteration for set and maps.
- Up to 10 methods, though varargs is supported

```
S of() : Map<K,V> - Map
S of(K k1, V v1, K k2, V v2, K k3, V v3, K k4, V v4, K k5, V v5) : Map<K,V> - Map
S of(K k1, V v1, K k2, V v2, K k3, V v3, K k4, V v4, K k5, V v5, K k6, V v6, K k7, V v7, K k8, V v8, K k9, V v9, K k10, V v10) : Map<K,V> - Map
S of(K k1, V v1) : Map<K,V> - Map
S of(K k1, V v1, K k2, V v2, K k3, V v3, K k4, V v4, K k5, V v5, K k6, V v6) : Map<K,V> - Map
S of(K k1, V v1, K k2, V v2) : Map<K,V> - Map
S of(K k1, V v1, K k2, V v2, K k3, V v3, K k4, V v4, K k5, V v5, K k6, V v6, K k7, V v7) : Map<K,V> - Map
S of(K k1, V v1, K k2, V v2, K k3, V v3) : Map<K,V> - Map
S of(K k1, V v1, K k2, V v2, K k3, V v3, K k4, V v4, K k5, V v5, K k6, V v6, K k7, V v7, K k8, V v8) : Map<K,V> - Map
S of(K k1, V v1, K k2, V v2, K k3, V v3, K k4, V v4) : Map<K,V> - Map
S of(K k1, V v1, K k2, V v2, K k3, V v3, K k4, V v4, K k5, V v5, K k6, V v6, K k7, V v7, K k8, V v8, K k9, V v9) : Map<K,V> - Map
S ofEntries(Entry<? extends K,? extends V>... entries) : Map<K,V> - Map
```

# 字符串优化

## JEP 254: Compact Strings

- replace String-internal `char[]` representation (16 bits/char) with a `byte[]` array plus encoding field
  - Based on contents of string, either 1-byte per character or 2-bytes per character representation
  - Transparent to users; same API with better memory density

## JEP 280: Indify String Concatenation

- `Javac` compiles string concatenation down to `StringBuilder` calls. Call pattern may not be optimal for a given JVM implementation.
  - New stable library entry point `java.lang.invoke.StringConcatFactory`
  - Compilers use this facility; dynamically linked at runtime, optimized by JVM

# Variable Handles

## JEP 193

- A standard means to invoke the equivalents of `java.util.concurrent.atomic` and `sun.misc.Unsafe` operations upon object fields and array elements.
- A standard set of fence operations for fine-grained control of memory ordering
- A standard reachability fence operation ensuring a referenced object remains strongly-reachable.

# HTTP/2 Client

## JEP 110

- 支持HTTP/2 and WebSocket协议
- 支持sync and async工作模式
- Reactive style programming API
- High performance but much light weight API comparing with netty or Apache HttpClient
- Delivered as an incubator module



# Brief samples

```
HttpClient client = HttpClient.newBuilder()  
    .executor(exec)  
    .sslContext(sslContext)  
    .version(HTTP_2)  
    .build();
```

```
HttpRequest req = HttpRequest.newBuilder(uri)  
    .POST()  
    .build();  
client.sendAsync(req, abodyhandler)  
    .thenApply(...);
```

# jshell

## OpenJDK Project Kulla/ JEP 222: jshell: The Java Shell (Read-Eval-Print Loop)

- 可以非常方便的使用API或者语言特性:
- 打开 command-line, 然后:

```
jdk-9\bin\jshell
```

```
Jshell>/help
```

```
jshell> ProcessHandle ph = ProcessHandle.current();
```

```
jshell> ph.getPid();
```

```
jshell> ph.info().command();
```

# Private interface method

## JDK-8071453: Allow interface methods to be private

- JDK 8开始支持在接口中定义default method
- Java 9接口可以定义private methods :
  - static methods
  - instance methods
- 典型的使用场景是作为default method的helper method

ORACLE®