JDK 9 - Das neue Java Platform Module System

Wolfgang Weigend
Sen. Leitender Systemberater
Java Technology and Architecture

JavaOne
ORACLE

Safe Harbor Statement

The preceding 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.

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Agenda

1. JDK 9 Status
2. Migration Guide
3. Modularity
4. Jigsaw und die Werkzeuge
5. Ausblick und Zusammenfassung

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

JDK 9 Status

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

JDK 9 Status – 91 JEP's targeted to JDK 9

http://openjdk.java.net/projects/jdk9/

Behind the scenes

- Store Interned Strings in CDS Archives
- Improve Contended Locking
- Compact Strings
- Improve Secure Application Performance
- Leverage CPU Instructions for GHASH and RSA
- Tiered Attribution for javac
- Javadoc Search
- Marlin Graphics Renderer
- HiDPI Graphics on Windows and Linux
- Enable GTK 3 on Linux
- Update JavaFX/Media to Newer Version of GStreamer

New functionality

- **Jigsaw – Modularize JDK**
- Enhanced Deprecation
- Stack-Walking API
- Convenience Factory Methods for Collections
- Platform Logging API and Service
- jshell: The Java Shell (Read-Eval-Print Loop)
- Compile for Older Platform Versions
- Multi-Release JAR Files
- Platform-Specific Desktop Features
- TIFF Image I/O\
- Multi-Resolution Images

Specialized

- Process API Updates
- Variable Handles
- Spin-Wait Hints
- Dynamic Linking of Language-Defined Object Models
- Enhanced Method Handles
- More Concurrency Updates
- Compiler Control

New standards

- HTTP 2 Client
- Unicode 8.0
- UTF-8 Property Files
- Datagram Transport Layer Security (DTLS)
- OCSP Stapling for TLS
- TLS Application-Layer Protocol Negotiation Extension
- SHA-3 Hash Algorithms
- DRBG-Based SecureRandom Implementations
- Create PKCS12 Keystores by Default
- Merge Selected Xerces 2.11.0 Updates into JAXP
- XML Catalogs
- HarfBuzz Font-Layout Engine
- HTML5 Javadoc

Housekeeping

- Parser API for Nashorn
- Prepare JavaFX UI Controls & CSS APIs for Modularization
- Modular Java Application Packaging
- New Version-String Scheme
- Reserved Stack Areas for Critical Sections
- Segmented Code Cache
- Indify String Concatenation
- Unified JVM Logging
- Unified GC Logging
- Make G1 the Default Garbage Collector
- Use CLDR Locale Data by Default
- Validate JVM Command-Line Flag Arguments
- Java-Level JVM Compiler Interface
- Disable SHA-1 Certificates
- Simplified Doclet API
- Deprecate the Applet API
- Process Import Statements Correctly
- Annotations Pipeline 2.0
- Elide Deprecation Warnings on Import Statements
- Milling Project Coin

Gone

- Remove GC Combinations Deprecated in JDK 8
- Remove Launch-Time JRE Version Selection
- Remove the JVM TI hprof Agent
- Remove the jhat Tool

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

JEP 223: New Version-String Scheme (1)

Revise the JDK's version-string scheme: Project Verona

- It's long past time for a simpler, more intuitive versioning scheme.
- A *version number* is a non-empty sequence of non-negative integer numerals, without leading zeroes, separated by period characters
  - [1-9][0-9]*(\.(0|[1-9][0-9]*))*
- **$MAJOR.$MINOR.$SECURITY**
- A *version string* consists of a version number $VNUM, as described above, optionally followed by pre-release and build information
- The version-string drops the initial 1 element from JDK version numbers.
  - First release of JDK 9 will have the version number 9.0.0 rather than 1.9.0.0.

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

JEP 223: New Version-String Scheme (2)

New version-string format $MAJOR.$MINOR.$SECURITY

Release Type	Old long	Old short	New long	New short
Early Access	1.9.0-ea-b19	9-ea	9-ea+19	9-ea
Major	1.9.0-b100	9	9+100	9
Security #1	1.9.0_5-b20	9u5	9.0.1+20	9.0.1
Security #2	1.9.0_11-b12	9u11	9.0.2+12	9.0.2
Minor #1	1.9.0_20-b62	9u20	9.1.2+62	9.1.2
Security #3	1.9.0_25-b15	9u25	9.1.3+15	9.1.3
Security #4	1.9.0_31-b08	9u31	9.1.4+8	9.1.4
Minor #2	1.9.0_40-b45	9u40	9.2.4+45	9.2.4

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

JEP 223: New Version-String Scheme (3)

A simple JDK-specific Java API to parse, validate, and compare version strings

```
package jdk;

import java.util.Optional;

public class Version
    implements Comparable<Version>
{

    public static Version parse(String);
    public static Version current();

    public int major();
    public int minor();
    public int security();

    public List<Integer> version();
    public Optional<String> pre();
    public Optional<Integer> build();
    public Optional<String> optional();

    public int compareTo(Version o);
    public int compareToIgnoreOpt(Version o);

    public boolean equals(Object o);
    public boolean equalsIgnoreOpt(Object o);

    public String toString();
    public int hashCode();
}
```

$MAJOR.$MINOR.$SECURITY

```
Release Type        Versions
----------------    -----------
Major  GA           jdk-9+181
Minor  #1           jdk-9.1.2+27
Security #1         jdk-9.0.1+3
```

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

JDK Version numbers with specific proposal

- **$FEATURE.$INTERIM.$UPDATE.$EMERG vs. $MAJOR.$MINOR.$SECURITY**
- Mark Reinhold asked two questions:

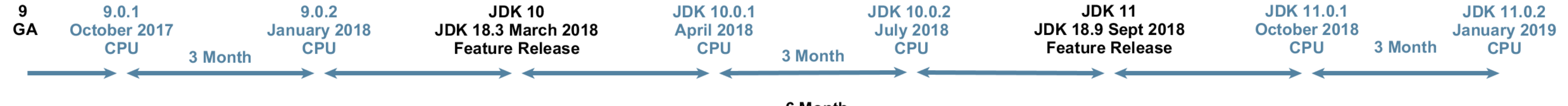
➢ **(1) Bearing in mind that no version-string scheme is ideal, is this scheme acceptable?**

➢ **(2) If this scheme is not acceptable then please explain why, and identify exactly what you would change.**

> “Ground rules, as before: I'll give much greater weight to your first reply to this message than to any other, I'll ignore replies-to-replies, and I'll heavily discount replies that quote more text than add new text of their own.
> **I'll summarize relevant replies in about a week, and then draft a JEP. – Mark Reinhold**”

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

JDK Version numbers & Java Critical Patch Updates



- Rules for Java CPU's
  - Main release for security vulnerabilities
  - Covers all JDK families (9, 8, 7, 6)
  - CPU release triggers Auto-update
  - Dates published 12 months in advance
  - Security Alerts are released as necessary
  - Based off the previous (non-CPU) release
  - Released simultaneously on java.com and OTN

- JDK 9.0.1 - Security Baselines

JRE Family Version	JRE Security Baseline (Full Version String)
9	9.0.1+11
8	1.8.0_151-b12
7	1.7.0_161-b13
6	1.6.0_171-b13

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Migrating to Oracle JDK 9 - Migration Guide (1)

https://docs.oracle.com/javase/9/migrate/

- **How to proceed as you migrate your existing Java application to JDK 9**
  - Every new Java SE release introduces some binary, source and behavioral incompatibilities with previous releases
  - The modularization of the Java SE Platform brings many benefits but also many changes
  - Code that uses only official Java SE Platform APIs and supported JDK-specific APIs should continue to work without change
  - Code that uses certain features or JDK-internal APIs may not run or may give different results
- **Prepare for Migration**
  - Get the JDK 9 Build
  - Run Your Program Before Recompiling
  - Update Third-Party Libraries
  - Compile Your Application
  - Run jdeps on Your Code

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Migrating to Oracle JDK 9 - Migration Guide (2)

https://docs.oracle.com/javase/9/migrate/

- **Beware of changes that you may encounter as you run your application**
  - Changes to the Installed JDK/JRE Image
  - Removed APIs
  - Deployment
  - Changes to Garbage Collection
  - Removed Tools
  - Removed macOS-specific Features

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Migrating to Oracle JDK 9 - Migration Guide (3)

Removed Tools

- **JavaDB**, which was a rebranding of Apache Derby, is not included in JDK 9
- **JVM Tools Interface hprof agent** library (libhprof.so) has been removed
  - The hprof agent was written as demonstration code for the **JVM Tool Interface** and not intended to be a production tool. The useful features of the hprof agent have been superseded by better tools in the JDK
- The **jhat tool** was an experimental, unsupported heap visualization tool added in JDK 6. Superior heap visualizers and analyzers have been available for many years
- The **launchers java-rmi.exe** from Windows and **java-rmi.cgi** from Linux and Solaris have been removed
- The **IIOP transport** support from the **JMX RMI Connector** along with its supporting classes have been removed in JDK 9
- **Windows 32 Client VM is dropped** and only a server VM is offered in JDK 9
- **Visual VM** removed
  - Visual VM is a tool that provides information about code running on a Java Virtual Machine. It was provided with JDK 6, JDK 7, and JDK 8
  - Visual VM is not bundled with JDK 9. If you would like to use Visual VM with JDK 9, you can get it from the Visual VM open source project site

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Modularity

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

JSR 376: Java Platform Module System (1)

Public Review Reconsideration Ballot from 2017-06-13 to 2017-06-26

- **These are the final results of the Public Review Reconsideration Ballot for JSR #376.**
  - **The EC has approved this ballot. The votes are below:**

EC

Member	Vote	Member	Vote	Member	Vote	Member	Vote
ARM Limited	✓	Azul Systems, Inc.	✓	Credit Suisse	✓	Eclipse Foundation, Inc	✓
Fujitsu Limited	✓	Gemalto M2M GmbH	✓	Goldman Sachs & Co.	✓	Grimstad, Ivar	✓
Hazelcast	✓	Hewlett Packard Enterprise	✓	IBM	✓	Intel Corp.	✓
JetBrains s.r.o.	✓	Keil, Werner	✓	London Java Community	✓	MicroDoc	✓
NXP Semiconductors	✓	Oracle	✓	Red Hat	■	SAP SE	✓
Software AG	✓	SouJava	✓	Tomitribe	✓	Twitter, Inc.	✓
V2COM	✓						

On 2017-06-13 **IBM voted Yes with the following comment:**
IBM supports the revised JPMS specification moving to Proposed Final Draft, with credit due to Oracle as the specification leader and those in the JSR 376 Expert Group who dedicated their time to reaching this milestone.

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Modular Development with JDK 9

Modular Applications

Modular Libraries

Modular JDK

Module-
Aware Tools

Java Language & JVM

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Goals of the Java SE 9 Module System (1)

- Reliable configuration with better maintenance
  - **replace the brittle, error-prone class-path mechanism with a means for program components to declare explicit dependences upon one another**
  - **Each modul exists once**
  - **Verification of all necessary modules exist at start time**
- Strong encapsulation
  - **allow a component to declare which of its public types are accessible to other components, and which are not. Prevents access to non-public classes and API's**
  - **module can declare an API to other modules**
  - **packages not on the API are hidden**
- Addressing these goals would enable further benefits:
  - **A scalable platform** for deployment of small applications and tiny runtime
  - **Greater platform integrity** and **Improved performance**

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Non-Goals of the Java SE 9 Module System (2)

- The Java Platform Module System does not replace OSGi
- The Java Platform Module System does not support versioning of modules

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

JSR 376: Java Platform Module System (2)

An approachable yet scalable module system for the Java Platform

- Provide a means for developers and libraries to define their own modules
- Reflection API's for module information
- Integration with developer tools (Maven, Gradle, IDE's)
- Integration with existing package managers (e.g., RPM)
- Dynamic configuration of module graph (e.g., for Java EE containers)
- Current documents, code, & builds
  - Requirements
  - The State of the Module System (design overview)
  - Initial draft JLS and JVMS changes
  - Draft API specification (diffs relative to JDK 9)
    - java.lang.Class
    - java.lang.ClassLoader
    - java.lang.reflect.Module
    - java.lang.module
  - Issue summary
  - RI prototype: Source, binary

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Projekt Jigsaw

JDK Enhancement Proposal's (JEP's)

- JSR 376 Java Platform Module System
- 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
- JEP 282: jlink - The Java Linker
- JDK 9 GA – JDK 9+181
- JDK 9.0.1 General-Availability Release is here: http://jdk.java.net/9/

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

JEP 200: The Modular JDK (1)

Goal: Define a modular structure for the JDK

- Make minimal assumptions about the module system that will be used to implement that structure.
- **Divide the JDK into a set of modules** that can be combined at **compile time, build time, install time, or run time** into a variety of configurations including, **but not limited to**:
  - Configurations corresponding to the full Java SE Platform, the full JRE, and the full JDK;
  - Configurations roughly equivalent in content to each of the the Compact Profiles defined in Java SE 8; and
  - Custom configurations which contain only a specified set of modules and the modules transitively required by those modules.

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

JEP 200: The Modular JDK (2)

Module System Assumptions: A module …

- can contain class files, resources, and related native and configuration files.
- **has a name**.
- can depend, by module name, upon one or more other modules.
- **can export all of the public types in one or more of the API packages that it contains, making them available to code in other modules depending on it**
- can restrict, by module name, the set of modules to which the public types in one or more of its API packages are exported. (sharing internal interface)
- **can re-export all of the public types that are exported by one or more of the modules upon which it depends**. (support refactoring & aggregation)
  - ***A module is a set of packages with classes & interfaces***
  - ***The module metadata is in module-info.class***

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

JEP 200: The Modular JDK (3)

Design Principles

- **Standard modules**, whose specifications are governed by the JCP, must have names starting with the string "**java.**".
- **All other modules** are merely part of the JDK, and must have names starting with the string "**jdk.**".
- **If a module exports a type** that contains a public or protected member that, in turn, refers to a type from some other module **then the first module must re-export the public types of the second.** This ensures that method-invocation chaining works in the obvious way.
- Additional principles in JEP 200 text to ensure that code which depends only upon Java SE modules will depend only upon standard Java SE types.

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

JEP 200: The Modular JDK (4)

Module Graph

java.se
jdk.xml.ws
jdk.rmic
jdk.hotspot.agent
java.xml.ws
jdk.xml.bind
jdk.javadoc
jdk.jcmd
jdk.jdi
jdk.jconsole
jdk.compact3
java.xml.soap
java.annotations.common
jdk.compiler
jdk.attach
jdk.sctp
java.smartcardio
jdk.security.auth
jdk.httpserver
java.compact3
java.xml.bind
jdk.jvmstat
java.sql.rowset
java.compact2
java.security.jgss
java.activation
java.corba
java.management
java.compiler
jdk.naming.dns
jdk.naming.rmi
java.security.acl
java.sql
java.desktop
java.naming
java.rmi
java.prefs
java.xml.crypto
java.compact1
java.security.sasl
jdk.scripting.nashorn
jdk.crypto.pkcs11
jdk.hprof.agent
jdk.zipfs
jdk.localedata
jdk.jdwp.agent
java.instrument
java.xml
jdk.charsets
java.scripting
java.logging
jdk.crypto.ec
java.base

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Modules

A fundamental new kind of Java component

- **A module is a named, self-describing collection of code & data**
  - **Code is organized as a set of packages containing types**
- It **declares** which other **modules** it ***requires*** in order to be compiled and run
- It **declares** which of its **packages** it ***exports***.
- Module system locates modules
  - **Ensures code in a module can only refer to types in modules upon which it depends**
  - **The access-control mechanisms of the Java language and the Java virtual machine prevent code from accessing types in packages that are not exported by their defining modules.**

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Module descriptors

module-info.class advantages

- class files already have a <u>precisely-defined and extensible format</u>
- consider **module-info.class file** as ***module descriptor***
  - includes the compiled forms of source-level module declarations
  - may include additional kinds of information recorded in class-file attributes
    - inserted after the declaration is initially compiled.
- **An IDE can insert class file attributes containing documentary information**
  - **module version, title, description, and license.**
- **This information can be read at compile time and run time**
  - for use in documentation, diagnosis, and debugging

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Platform modules

Modules all the way down to the base module: java.base

- The **only module known** specifically to the **module system** is **java.base**.
  - The base module is always present. Every other module depends implicitly upon the base module, while the base module depends upon no other modules
- **The base module defines and exports all of the platform's core packages, including the module system itself:**

```
module java.base {
    exports java.io;
    exports java.lang;
    exports java.lang.module;
... }
```

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Module Paths

Where do modules fulfilling dependences come from?

- **module system can select a module to resolve a dependence**
  - built-in to the compile-time or run-time environment or
  - a module defined in an artifact
    - the module system locates artifacts on one or more *module paths* defined by the host system.
- **A module path is a sequence of directories containing module artifacts**
  - searched, in order, for the first artifact that defines a suitable module.
- **Module paths are materially different from class paths, and more robust:**
  - A class path is a means to locate individual types in all the artifacts on the path.
  - A **module path** is a means to **locate whole modules** rather than individual types.
    - If a particular **dependence can not be fulfilled then resolution will fail** with an error message

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Packages and Modules - Module Declarations

Java Language Specification, Java SE 9 Edition

```
ModuleDirective:
    requires {RequiresModifier} ModuleName ;
    exports PackageName [to ModuleName {, ModuleName}] ;
    opens PackageName [to ModuleName {, ModuleName}] ;
    uses TypeName ;
    provides TypeName with TypeName {, TypeName} ;
```

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Services (1)

Loose coupling

- Our com.foo.app **module extended** to use a MySQL database
  - a MySQL JDBC driver implementing java.sql.Driver is provided in a module:

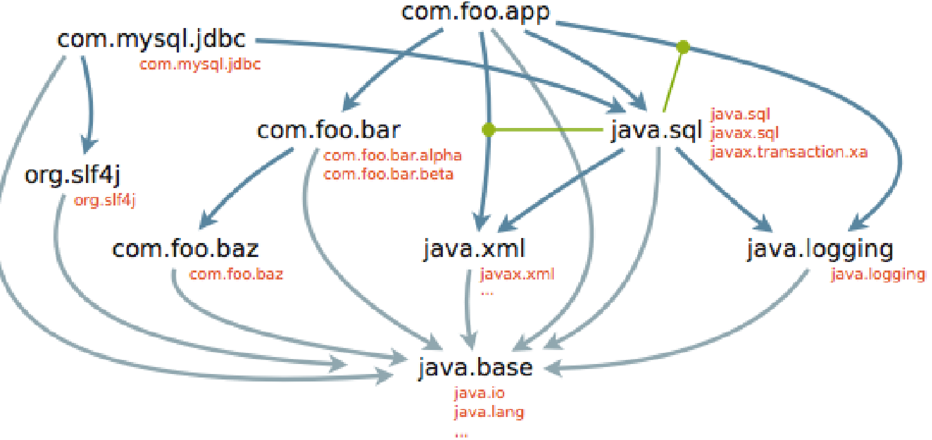
```
module com.mysql.jdbc {
        requires java.sql;
        requires org.slf4j;
        exports com.mysql.jdbc;
}
```

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Services (2)

Loose coupling

- In order for the java.sql module to make use of this driver we must
  - **add the driver module to the run-time module graph**
  - **resolve its dependences**
- ***java.util.ServiceLoader*** **class can instantiate the driver class via reflection**



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Services (3)

Loose coupling

- Module system must be able to locate service providers.
- **Services provided are declared** with a **provides** clause:

```
module com.mysql.jdbc {
    requires java.sql;
    requires org.slf4j;
    exports com.mysql.jdbc;
    provides java.sql.Driver with com.mysql.jdbc.Driver;
}
```

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Services (4)

Loose coupling

- Module system must be able to locate service users.
- Services **used** are declared with a **uses** clause:

```
module java.sql {
    requires public java.logging;
    requires public java.xml;
    exports java.sql;
    exports javax.sql;
    exports javax.transaction.xa;
    uses java.sql.Driver;
}
```

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Services (5)

Advantages of using module declations to declare service relationships

- Clarity
- **Service declarations can be interpreted at compile time**
  - to ensure that the **service interface is accessible**
  - to ensure that providers actually **do implement their declared service interfaces**
  - to ensure that **observable providers are appropriately compiled and linked prior to run time**
- Catching runtime problems at compile time!

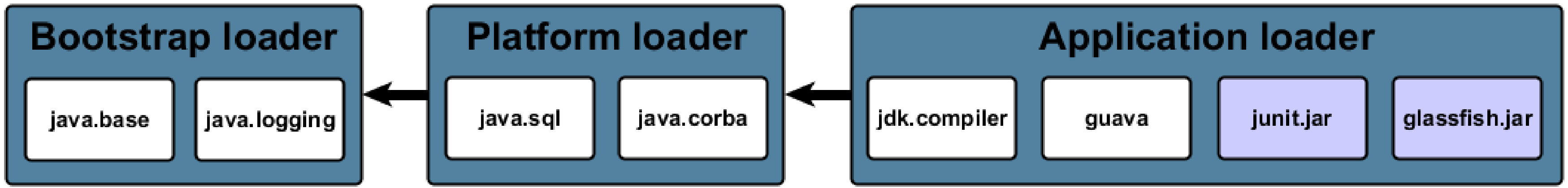
Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Class Loaders

- **Few restrictions on the relationships between modules and class loaders:**
  - **A class loader can load types from one module or from many modules**
    - **as long the modules do not interfere with each other and**
    - **the types in any particular module are loaded by just one loader**
- **Critical to compatibility**
  - **retains the existing hierarchy of built-in class loaders.**
- Easier to modularize existing applications with complex class loaders
  - **class loaders can be upgraded to load types in modules**
  - **without necessarily changing their delegation patterns**

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Modular Class Loading in JDK 9

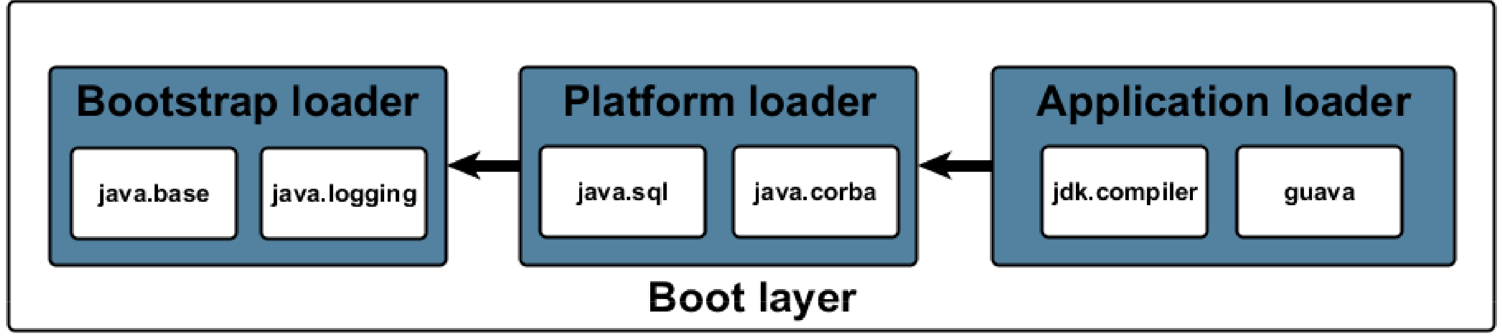


Java Platform Module System

Java Virtual Machine

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

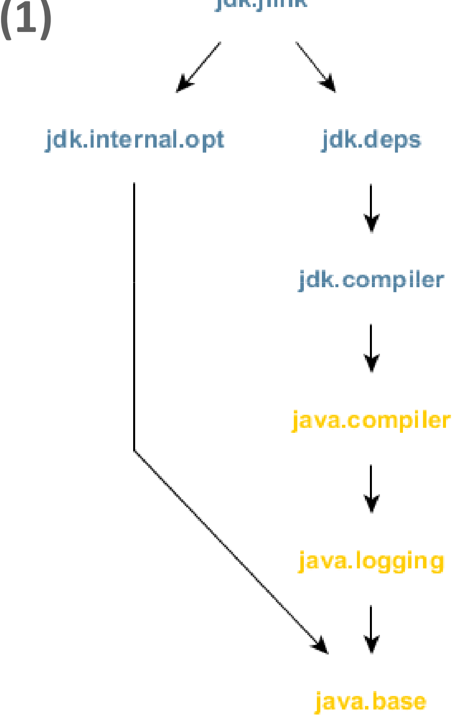
Layers



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Layer creation

(1)



(2)

```
String moduleName -> {
   switch (moduleName) {
           case "java.base":
           case "java.logging":
                   return BOOTSTRAP_LDR;
           default:
                   return APP_LDR;
   }
}
```

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Unnamed Modules

Backwards compatibility: Loading types from the class path

- **Every class loader has a unique *unnamed module***
  - returned by the new ClassLoader::getUnnamedModule method
- **A class loader loads a type not defined in a named module**
  - that type is considered to be in the unnamed module
- **An unnamed module**
  - **reads every other module**
  - **exports all of its packages to every other module**
- Existing class-path applications using only standard APIs can keep working

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Während der Übergangszeit bleibt diese Hintertür offen

At your own risk: java launcher and javac option, as part of JEP 261: Module System

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

  where <source-module> and <target-module> are module names and <package> is the name of a package

- `--add-exports java.management/com.sun.jmx.remote.internal=jmx.wbtest`
- `--add-exports java.management/sun.management=ALL-UNNAMED`
  - ***The --add-exports option must be used with great care. You can use it to gain access to an internal API of a library module, or even of the JDK itself, but you do so at your own risk: If that internal API changes or is removed then your library or application will fail.***

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Jigsaw und die Werkzeuge

jimage

jdeps

jlink

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

jimage – Modulverzeichnis-Kommando (1)

Tools should never read jimage files, directly or via code. It's an JVM-internal format ..

```
C:\jdk-9> java -version
java version "9"
Java(TM) SE Runtime Environment (build 9+181)
Java HotSpot(TM) Server VM (build 9+181, mixed mode, emulated-client)

C:\jdk-9\lib> jimage info modules

C:\jdk-9\lib> jimage list modules
       /* list all JDK 9 *.class files from the modules file */


C:\jdk-9\lib> jimage extract --dir=C:\jdk-9\mydir modules
       /* extract all JDK 9 *.class files from the lib\modules file */


C:\jdk-9> java --list-modules
       /* list the JDK 9 modules */
```

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

jimage – Modulverzeichnis-Kommando (2)

Tools should never read jimage files, directly or via code. It's an JVM-internal format ..

- **JIMAGE Format**
  - Schneller Zugriff auf die im JDK 9 enthaltene Klassen
  - Kein langsames Durchsuchen von ZIP-Einträgen
  - JIMAGE für beschleunigtes Klassenladen innerhalb vom JDK
- **JMOD Format**
  - Basiert auf dem ZIP-Format, wie das JAR-Format
  - Für modulspezifische Metadaten und plattformspezifische Bibliotheken (DLL's oder SO-Files)
  - JMOD Format soll künftig Ersatz für das JAR-Format werden, damit komplette Java-Anwendungen als Modul ausgeliefert werden können, inklusive allen Metadaten von Abhängigkeiten und exportierten API's
  - Anwendungs-Rollout mit abgespeckter JRE, nur mit den benötigten Modulen

Q: Without it, how can org.reflections and scannotations efficiently find all classes that have specific annotation?
A: Tools should use the jrt filesystem to scan classes in the image. Details in JEP 220: http://openjdk.java.net/jeps/220

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

jdeps - Java-Class-Dependency-Analyzer

```
C:\mlib> c:\jdk-9.0.1\bin\jdeps -profile com.greetings.jar
com.greetings
 [file:///C:/mlib/com.greetings.jar]
   requires mandated java.base (@9-ea)
com.greetings -> java.base (compact1)
   com.greetings                         -> java.io                          compact1
   com.greetings                         -> java.lang                        compact1


C:\mlib> c:\jdk-9.0.1\bin\jdeps -v com.greetings.jar
com.greetings
 [file:///C:/mlib/com.greetings.jar]
   requires mandated java.base (@9-ea)
com.greetings -> java.base
   com.greetings.Main                    -> java.io.PrintStream              java.base
   com.greetings.Main                    -> java.lang.Object                 java.base
   com.greetings.Main                    -> java.lang.String                 java.base
   com.greetings.Main                    -> java.lang.System                 java. base
```

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

jdeps - Java-Class-Dependency-Analyzer JDK 9.0.1 (1)

```
C:\jdk-9.0.1\JavaFXApplication3\dist> jdeps -profile JavaFXApplication3.jar
JavaFXApplication3.jar -> C:\Program Files (x86)\Java\jdk1.8.0_45\jre\lib\ext\jfxrt.jar
JavaFXApplication3.jar -> C:\Program Files (x86)\Java\jdk1.8.0_45\jre\lib\rt.jar (compact1)
   javafxapplication3 (JavaFXApplication3.jar)
      -> java.io                                          compact1
      -> java.lang                                        compact1
      -> javafx.application                               jfxrt.jar
      -> javafx.collections                               jfxrt.jar
      -> javafx.event                                     jfxrt.jar
      -> javafx.scene                                     jfxrt.jar
      -> javafx.scene.control                             jfxrt.jar
      -> javafx.scene.layout                              jfxrt.jar
      -> javafx.stage                                     jfxrt.jar

C:\jdk-9.0.1\JavaFXApplication3\dist> jdeps -v JavaFXApplication3.jar

C:\jdk-9.0.1\bin> jdeps --generate-module-info C:\jdk-9.0.1\JavaFXApplication3\dist c:\jdk-
9.0.1\JavaFXApplication3\dist\JavaFXApplication3.jar
writing to C:\jdk-9.0.1\JavaFXApplication3\dist\JavaFXApplication3\module-info.java
```

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

jdeps - Java-Class-Dependency-Analyzer JDK 9.0.1 (2)

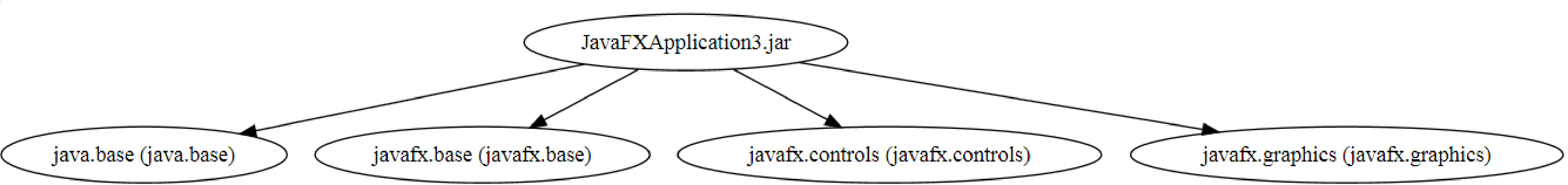
```
C:\jdk-9.0.1\bin> jdeps --generate-module-info C:\jdk-9.0.1\JavaFXApplication3\dist c:\jdk-
9.0.1\JavaFXApplication3\dist\JavaFXApplication3.jar
writing to C:\jdk-9.0.1\JavaFXApplication3\dist\JavaFXApplication3\module-info.java


C:\jdk-9.0.1\JavaFXApplication3\dist\JavaFXApplication3> dir
24.10.2017  21:33               171 module-info.java


C:\jdk-9.0.1\JavaFXApplication3\dist\JavaFXApplication3> type module-info.java
module JavaFXApplication3 {
    requires javafx.base;
    requires javafx.controls;

    requires transitive javafx.graphics;

    exports javafxapplication3;

}
```

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

jdeps - Java-Class-Dependency-Analyzer JDK 9.0.1 (3)

```
C:\jdk-9.0.1\bin> jdeps --module-path C:\jdk-9.0.1\JavaFXApplication3\dist -s -dotoutput c:\jdk-9.0.1\JavaFXApplication3\dist c:\jdk-9.0.1\JavaFXApplication3\dist\JavaFXApplication3.jar

C:\jdk-9.0.1\JavaFXApplication3\dist> type summary.dot
digraph "summary" {
  "JavaFXApplication3.jar"                          -> "java.base (java.base)";
  "JavaFXApplication3.jar"                          -> "javafx.base (javafx.base)";
  "JavaFXApplication3.jar"                          -> "javafx.controls (javafx.controls)";
  "JavaFXApplication3.jar"                          -> "javafx.graphics (javafx.graphics)";
}
```

http://www.webgraphviz.com/



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Additional diagnostic options supported by the launcher include

java --show-module-resolution to show module resolution output during startup, and causes the module system to describe its activities as it constructs the initial module graph

```
C:\jdk-9> java --show-module-resolution|more
root jdk.jdi jrt:/jdk.jdi
root javafx.web jrt:/javafx.web
root jdk.xml.dom jrt:/jdk.xml.dom
root jdk.jfr jrt:/jdk.jfr
root jdk.packager.services jrt:/jdk.packager.services
root jdk.httpserver jrt:/jdk.httpserver
root javafx.base jrt:/javafx.base
root jdk.net jrt:/jdk.net
root javafx.controls jrt:/javafx.controls
root java.se jrt:/java.se
root jdk.compiler jrt:/jdk.compiler
root jdk.jconsole jrt:/jdk.jconsole
root jdk.plugin.dom jrt:/jdk.plugin.dom
root jdk.attach jrt:/jdk.attach
root jdk.javadoc jrt:/jdk.javadoc
root jdk.jshell jrt:/jdk.jshell
root oracle.desktop jrt:/oracle.desktop
root jdk.sctp jrt:/jdk.sctp
root jdk.jsobject jrt:/jdk.jsobject
root javafx.swing jrt:/javafx.swing
root jdk.packager jrt:/jdk.packager
```

Since JDK 9 ea build 166:

"Also since early builds, `-Xdiag:resolver` was the option to print resolver diagnostic messages. This really odd option has now being replaced with `--show-module-resolution` to show resolution during startup.

The output has been cleaned up to make it easier to read and search."

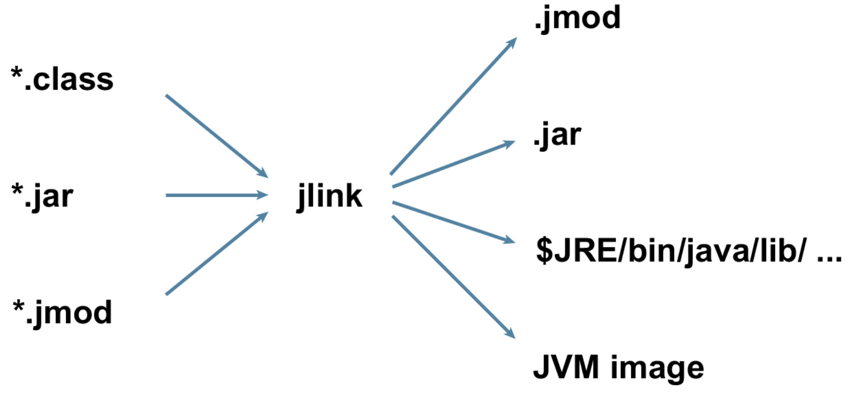
Java ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

jlink - generiert JRE und Applikations-Images (1)

- Platzsparende Runtime, inklusive eigener Anwendungsmodule im frei wählbaren Verzeichnis

```
jlink <options> --module-path <modulepath> --output <path>

jlink --module-path $JDKMODS:mlib --add-modules myapp --output myimage

C:\> jlink --module-path C:\jdk-9.0.1\jmods;mlib --add-modules com.greetings --compress=2 --
verbose --output greetingsapplication
        com.greetings file:///C:/mlib/com.greetings.jar
        java.base file:///C:/jdk-9.0.1/jmods/java.base.jmod
        Providers: java.base provides java.nio.file.spi.FileSystemProvider used by java.base
```



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

jlink - generiert JRE und Applikations-Images (2)

- Image-Verzeichnis C:\greetingsapplication 24 MB

```
C:\greetingsapplication> dir

Directory of C:\greetingsapplication
02.11.2017  19:04    <DIR>          .
02.11.2017  19:04    <DIR>          ..
02.11.2017  19:04    <DIR>          bin
02.11.2017  19:04    <DIR>          conf
02.11.2017  19:04    <DIR>          include
02.11.2017  19:04    <DIR>          legal
02.11.2017  19:04    <DIR>          lib
02.11.2017  19:04                57 release
```

- Datei release „9.0.1“

```
JAVA_VERSION="9.0.1"
MODULES="java.base com.greetings"
```

- Datei release mit früherem „build 9-ea+142-jigsaw-nightly-h5677-20161102”

```
#Thu Mar 09 22:11:23 CET 2017
OS_NAME="Windows"
MODULES="java.base com.greetings"
OS_VERSION="5.1"
OS_ARCH="i586"
JAVA_VERSION="9"
JAVA_FULL_VERSION="9-ea"
```

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

jlink - generiert JRE und Applikations-Images (3)

- Image-Verzeichnis C:\greetingsapplication 24 MB

```
C:\greetingsapplication\bin> java -m com.greetings/com.greetings.Main
Greetings!

C:\greetingsapplication\bin> dir
02.11.2017  19:04           143.360 java.dll
02.11.2017  19:04           225.280 java.exe
02.11.2017  19:04           225.792 javaw.exe
02.11.2017  19:04            19.456 jimage.dll
02.11.2017  19:04           195.072 jli.dll
02.11.2017  19:04            10.752 keytool.exe
02.11.2017  19:04           660.128 msvcp120.dll
02.11.2017  19:04           963.232 msvcr120.dll
02.11.2017  19:04            89.600 net.dll
02.11.2017  19:04            53.760 nio.dll
02.11.2017  19:04    <DIR>          server
02.11.2017  19:04            43.008 verify.dll
02.11.2017  19:04            70.144 zip.dll
```

```
C:\mlib> java -jar com.greetings.jar
Greetings!
```

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

jlink - generiert JRE und Applikations-Images (4)

- Image-Verzeichnis C:\greetingsapplication 24 MB

```
C:\greetingsapplication\bin> java -m com.greetings/com.greetings.Main
Greetings!

C:\greetingsapplication\bin> java --show-module-resolution -m com.greetings/com.greetings.Main
root com.greetings jrt:/com.greetings
Greetings!

C:\greetingsapplication\bin> java --list-modules -m com.greetings/com.greetings.Main
com.greetings
java.base@9.0.1

C:\greetingsapplication\bin> java -verbose -m com.greetings/com.greetings.Main

C:\greetingsapplication\bin>
java --add-exports java.base/jdk.internal.ref=ALL-UNNAMED -m com.greetings/com.greetings.Main
Greetings!
```

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

JSR 376: Java Platform Module System – Gradle (1)

Integration with developer tools (Maven, Gradle, IDE's)

- Sources per JDK
  - `service`
    - `java ..`
    - `java-8 ..`
    - `java-9`
      - `org.gradle.example.service`
        - `Service9`

- Dependencies per JDK

```
sources   {
java8     ..
java9       {
          dependencies {
                  library `org.apache.httpcomponents:httpclient:4.5.1`
      }
   }
}
```

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

JSR 376: Java Platform Module System – Gradle (2)

Integration with developer tools (Maven, Gradle, IDE‘s)

- **JEP 238: Multi-Release JAR**

```
jar root
  – A.class
  – B.class
  – C.class
  – D.class
  – META-INF
    – versions
      – 8
        – A.class
        – B.class
      – 9
        – A.class
      – 10
        – A.class
```

- A multi-release JAR "MRJAR" will contain additional directories for classes and resources specific to particular Java platform releases.
- A JAR for a typical library might look like this:

```
jar root
  – A.class
  – B.class
  – C.class
  – D.class
```

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

JSR 376: Java Platform Module System – NetBeans IDE (1)

Integration with developer tools (Maven, Gradle, IDE's)

JigsawSample5 - NetBeans IDE Dev 201703140002

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Projects: JigsawSample4; JigsawSample5 › Source Packages › <default package> › module-info.java; jigsawsample5 › JigsawSample5.java; Libraries › JDK 9 › java.base, java.datatransfer, java.desktop, java.logging, java.xml, jdk.accessibility

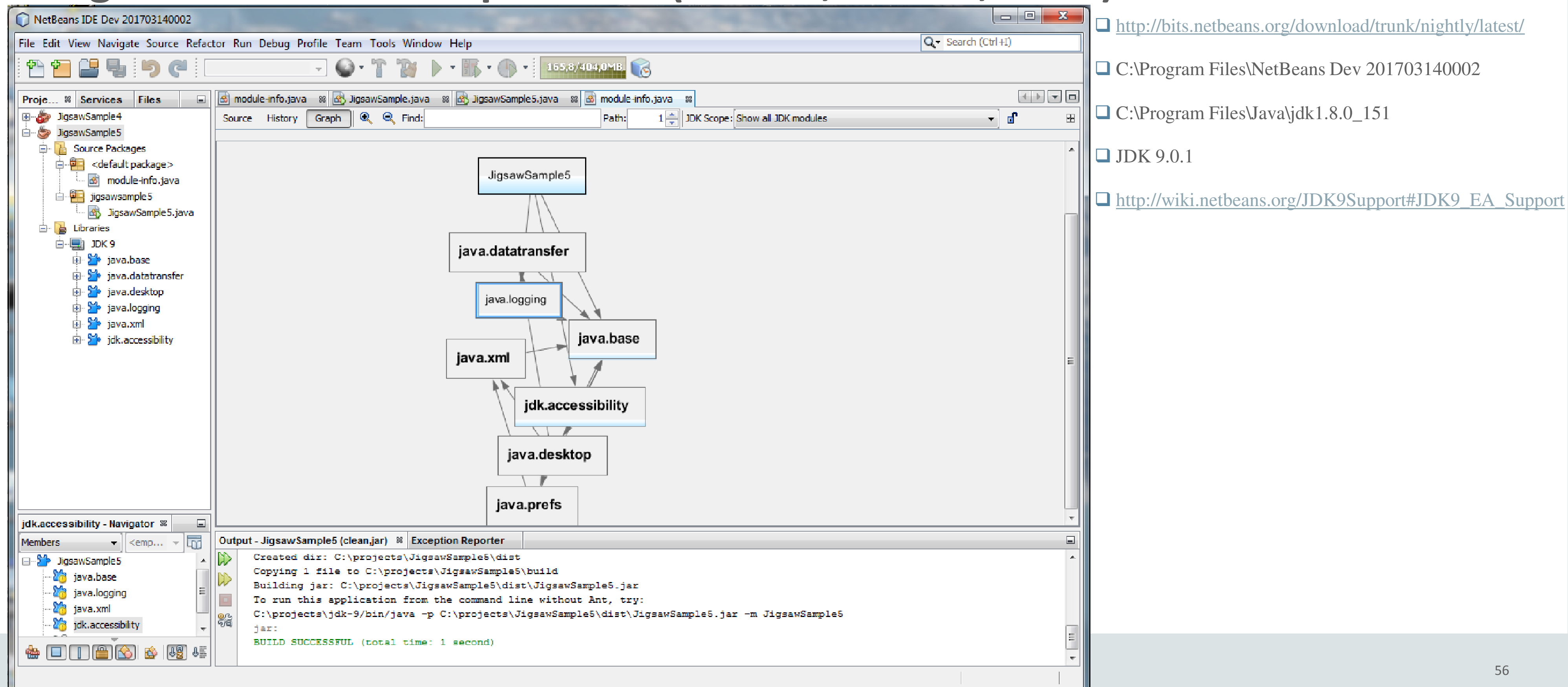
module-info.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

module JigsawSample5 {
    requires java.base;
    requires java.logging;
    requires java.xml;
    requires jdk.accessibility;

    uses java.lang.Readable;

}
```

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

JSR 376: Java Platform Module System – NetBeans IDE (2)

Integration with developer tools (Maven, Gradle, IDE‘s)



- http://bits.netbeans.org/download/trunk/nightly/latest/
- C:\Program Files\NetBeans Dev 201703140002
- C:\Program Files\Java\jdk1.8.0_151
- JDK 9.0.1
- http://wiki.netbeans.org/JDK9Support#JDK9_EA_Support

Summary of modularity impact in JDK 9

- **Modules for programming in the large**
- **Modules bundle together one or more packages for reuse and can offer stronger encapsulation than jars**
- **Supporting changes throughout the platform:**
  - **`module-info.java`** files to declare dependencies between modules
  - Changes to **`javac`** command line to find types in modules as well as in jars
  - Corresponding updates to **`java`** command line and HotSpot JVM runtime
  - New reflective API's to model modules (core reflection, **`javax.lang.model`**, etc.)

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Zusammenfassung

- **Die Modularisierung der Java SE Plattform im JDK 9 bringt viele Vorteile, aber auch größere Änderungen**

- Existierender Anwendungs-Code, der nur offizielle Java SE Plattform-API‘s mit den unterstützten JDK-spezifischen API’s verwendet, soll auch weiterhin ohne Änderungen ablauffähig sein
  - Abwärtskompatibilität
  - Dennoch ist es wahrscheinlich, wenn weiterhin veraltete Funktionalität oder JDK-interne API’s verwendet werden, dass der Code unverträglich sein kann

- Entwickler sollten sich frühzeitig damit vertraut machen, wie existierende Bibliotheken & Anwendungen auf JDK 9 anzupassen sind, sie modularisiert werden, welche Designfragen zu klären sind und wie man vorhandenen Anwendungs-Code trotzdem mit JDK 9 zum Laufen bekommt, auch wenn man diesen nicht verändern kann

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

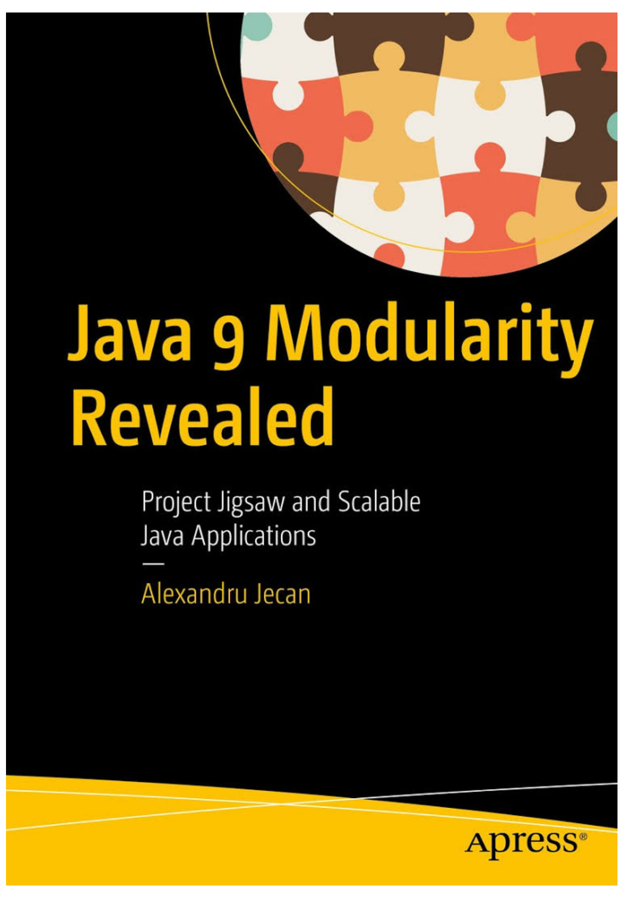
Danke!

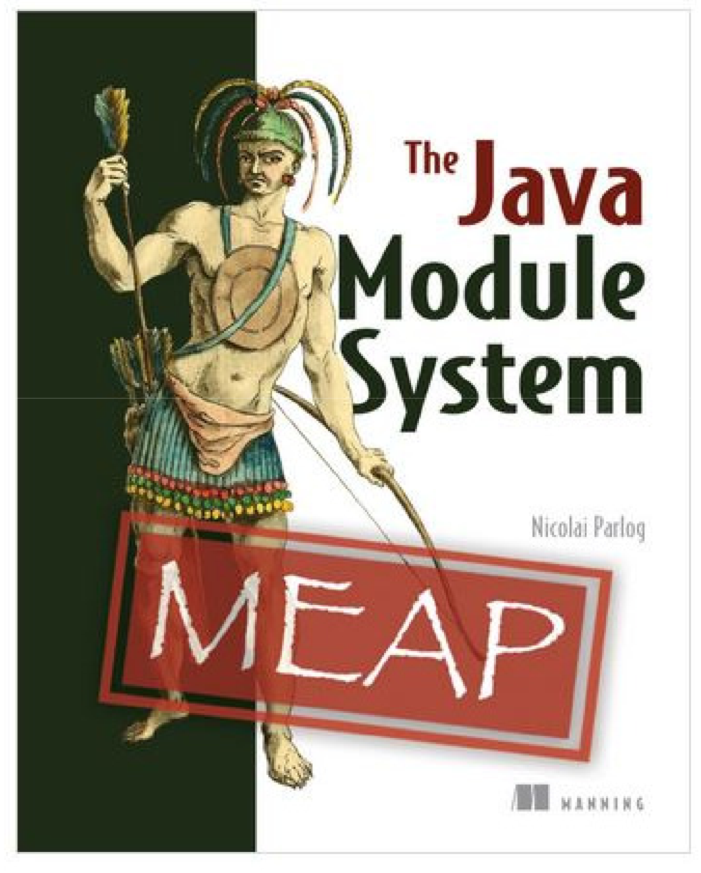
Wolfgang.Weigend@oracle.com

Twitter: @wolflook

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Bücher zum JDK 9 Java Module System





Copyright © 2017, Oracle and/or its affiliates. All rights reserved. Oracle Confidential – Internal/Restricted/Highly Restricted