ORACLE®

What is cool in Java 8 and new in 9

杨晓峰（Felix Yang）
Java核心类库中国团队Leader
Java Platform, Oracle

ORACLE CODE

developer.oracle.com

Live for the Code

ORACLE

免责声明

以下内容旨在概述产品的总体发展方向。该内容仅供参考，不可纳入任何合同。本演示不承诺提供任何材料、代码或功能，也不应将其作为购买决策的依据。此处所述有关 Oracle 产品的任何特性或功能的开发、发布和时间安排均由 Oracle 自行决定。

Java 8

“One of the biggest updates ever to a major language”

Andrew Binstock

Former Editor in Chief, Dr.Dobbs , now with Java Magazine

Java 8 议程

1. Lambda 表达式
2. 默认方法（Default Methods）
3. 方法引用（Method References）
4. Date Time APIs - JSR 310

Java 8 议程

1 Lambda 表达式

2 默认方法（Default Methods）

3 方法引用（Method References）

4 Date Time APIs - JSR 310

从行为进行抽象

“命题： 如何从人员信息的集合里删除所有超过18岁的人？”

```
Collection<Person> peoples = ...;
Iterator<Person> it = peoples.iterator();
while (it.hasNext()) {
    Person p = it.next();
    if (p.getAge() > 18)
        it.remove();
}
```

进一步抽象

The API Designer
Could create methods in the collection

```
interface Predicate<T> {
    boolean test(T t);
}

class Collections {
    public static<T>
        void removeMatching(Collection<T> coll,
                            Predicate<T> pred) {
            ...
        }
}
```

Not a real API

修改后的实现

But the code to use the new method would be bloated

```
Collection<Person> peoples = ...;

Collections.removeMatching(peoples,
    new Predicate<Person>() {
        public boolean test(Person p) {
            return p.getAge() > 18;
        }
    }
});
```

使用Java SE 8

- **Interface Collection<E>**
  - **removeIf(Predicate<? super E> filter)**

```
// 简化版
Collection<Person> peoples = ...;

peoples.removeIf(p -> p.getAge() > 18);
```

Real API in Java SE 8

ORACLE

聚合计算（Aggregate operations）

```
Collection<Person> peoples = ...;
int highestWeight =
        peoples.stream()
            .filter(p -> p.getGender() == MALE)
            .mapToInt(p -> p.getWeight())
            .max();
```

并行计算（Parallelism）

```
class MaxProblem {

  final List<Person> peoples;
  final int size;

  MaxProblem(List<Person> ps) {
    this.peoples = ps;
    size = ps.size();
  }

  public int solveSequentially() {
    int max = 0;
    for (Person p : peoples) {
      if (p.getGender() == MALE)
          max = Math.max(max, p.getWeight());
    }
    return max;
  }

  public MaxProblem subproblem(int start, int end) {
    return new MaxProblem(peoples.subList(start, end));
  }

}
```

```
class MaxFinder extends RecursiveAction {

  private final MaxProblem problem;
  int max;

  protected void compute() {
    if (problem.size < THRESHOLD)
      sum = problem.solveSequentially();
    else {
      int m = problem.size / 2;
      MaxFinder left, right;
      left = new MaxFinder(problem.subproblem(0, m))
      right = new MaxFinder(problem.subproblem(m, problem.size));
      forkJoin(left, right);
      max = Math.max(left.max, right.max);
    }
  }

}

ForkJoinExecutor pool = new ForkJoinPool(nThreads);
MaxFinder finder = new MaxFinder(problem);
pool.invoke(finder);
```

并行计算（Parallelism）

```
Collection<Person> peoples = ...;

int highestWeight =
   peoples.parallelStream()
         .filter(p -> p.getGender() == MALE)
         .mapToInt(p -> p.getWeight())
         .max();
```

默认方法

```
Collection<Person> people = ...;

int highestWeight =
   people.stream()
         ...

interface Collection<T> {
    ...
    default Stream<T> stream() {
        ...
    }
}
```

在接口中声明静态方法（ Static Methods）

- 静态隐含着非抽象，所以在Java 8之前是非法的
- 使用**@FunctionalInterface**是一个良好实践

```
static <T> Predicate<T> isEqual(Object target) {
       return (null == target)
              ? Objects::isNull
              : object -> target.equals(object);
}
```

ORACLE

方法引用

- `list.replaceAll(s -> s.toUpperCase());`
- `list.replaceAll(String::toUpperCase);`

- `list.sort(Comparator.comparing(p -> p.getName()));`
- `list.sort(Comparator.comparing(Person::getName));`

Lambda让代码更像问题本身的描述，清晰、简洁、便于维护

Java 8 议程

1 Lambda 表达式

2 默认方法（Default Methods）

3 方法引用（Method References）

4 Date Time APIs - JSR 310

ORACLE®

新的Date Time API – JSR 310

- 替换 java.util.Date, Calendar, TimeZone, DateFormat
- Immutable, Thread Safe
  - 解决了原有API令人诟病的线程安全问题
- 清晰、易用
  - Fluent
  - Factory Pattern, Strategy Pattern
  - 类设计基于相似的方法声明，触类旁通
  - 区分人类可读的日期时间和机器时间
- 良好的可扩展性
  - 基于ISO-8601 calendar system
  - 可以扩展到非ISO calendar system

ORACLE

示例

LocalDate	2016-12-03
LocalTime	11:05:30
LocalDateTime	2016-12-03T11:05:30
ZonedDateTime	2016-12-03T11:05:30+01:00 Europe/Paris
Instant	*2576458258.266 seconds after 1970-01-01*
Duration	PT30S *(30 seconds)*
Period	P1Y6M *(1 year and 6 months)*

```
// 操作本地时间
LocalTime current = LocalTime.now();
LocalTime time = LocalTime.of(13,30);
time = time.plusHours(4).minusMinutes(1).withNano(0);
```

JDK 8

Innovation

- **Lambda aka Closures**
- Language Interop
- Nashorn

Core Libraries

- Parallel operations for core collections APIs
- Improvements in functionality
- Improved type inference

Security

- Limited doPrivilege
- NSA Suite B algorithm support
- SNI Server Side support
- DSA updated to FIPS186-3
- AEAD JSSE CipherSuites

Java for Everyone

- Profiles for constrained devices
- JSR 310-Date & Time APIs
- Non-Gregorian calendars
- Unicode 6.2
- ResourceBundle
- BCP47 locale matching
- Globalization & Accessibility

Client

- Deployment enhancements
- JavaFX 8
- Public UI Control API
- Java SE Embedded support
- Enhanced HTML5 support
- 3D shapes and attributes
- Printing

Tools

- JSR 308-Annotations on Java Type
- Native app bundling
- App Store Bundling tools
- jdeps

General Goodness

- JVM enhancements
- No PermGen limitations
- Performance improvements

Enterprise

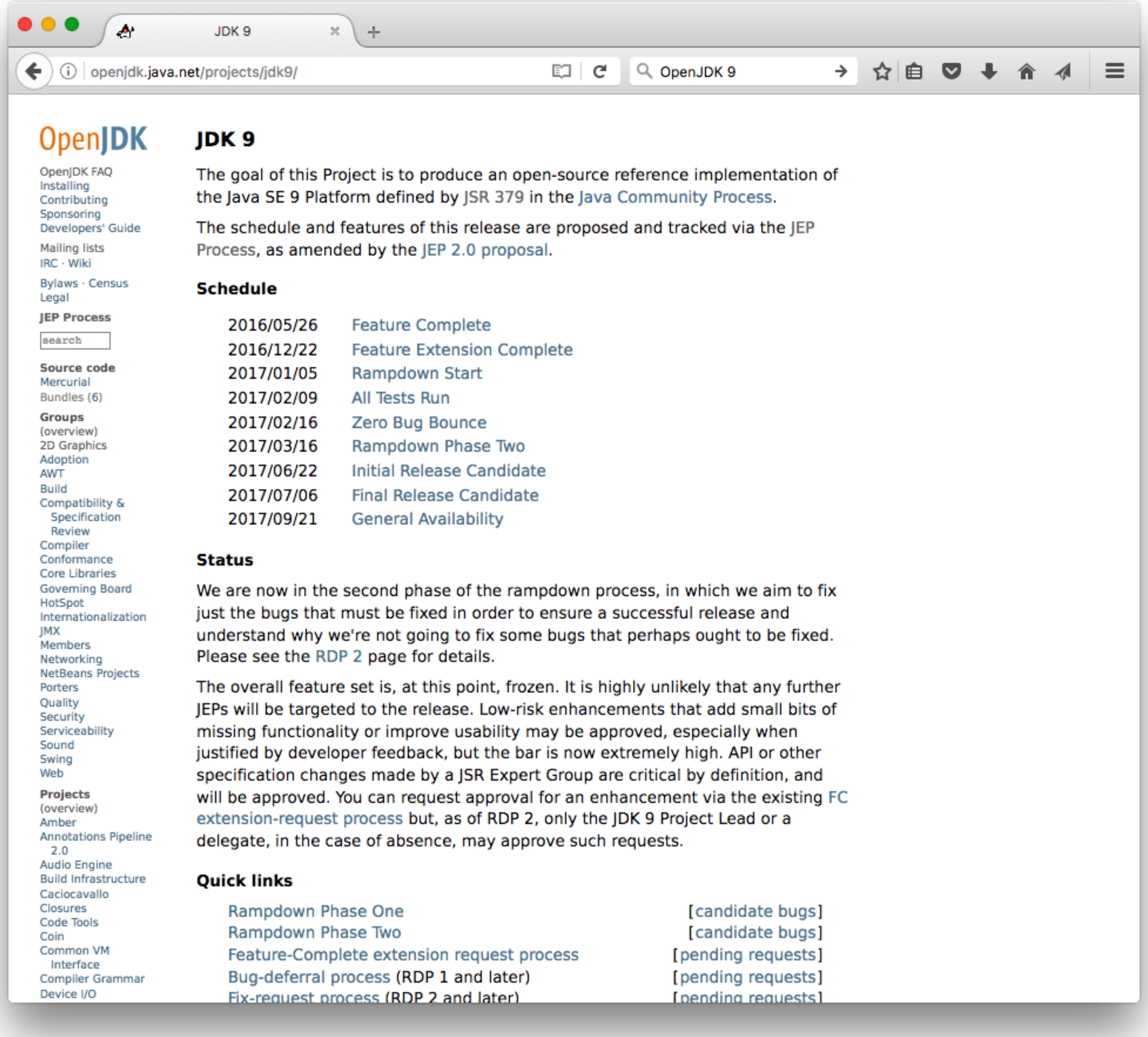
- Mission Control
- Flight Recorder
- Usage Tracker
- Advanced Management Console
- MSI Enterprise JRE Installer

JDK 9新特性

Java 9 议程

搜索:
OpenJDK 9

1. 新的特性和功能
2. 支持的新标准
3. 对用户透明的改进
4. 承上启下



OpenJDK

OpenJDK FAQ
Installing
Contributing
Sponsoring
Developers' Guide

Mailing lists
IRC · Wiki

Bylaws · Census
Legal

JEP Process

search

Source code
Mercurial
Bundles (6)

Groups
(overview)
2D Graphics
Adoption
AWT
Build
Compatibility & Specification Review
Compiler
Conformance
Core Libraries
Governing Board
HotSpot
Internationalization
JMX
Members
Networking
NetBeans Projects
Porters
Quality
Security
Serviceability
Sound
Swing
Web

Projects
(overview)
Amber
Annotations Pipeline 2.0
Audio Engine
Build Infrastructure
Caciocavallo
Closures
Code Tools
Coin
Common VM Interface
Compiler Grammar
Device I/O

JDK 9

The goal of this Project is to produce an open-source reference implementation of the Java SE 9 Platform defined by JSR 379 in the Java Community Process.

The schedule and features of this release are proposed and tracked via the JEP Process, as amended by the JEP 2.0 proposal.

Schedule

2016/05/26	Feature Complete
2016/12/22	Feature Extension Complete
2017/01/05	Rampdown Start
2017/02/09	All Tests Run
2017/02/16	Zero Bug Bounce
2017/03/16	Rampdown Phase Two
2017/06/22	Initial Release Candidate
2017/07/06	Final Release Candidate
2017/09/21	General Availability

Status

We are now in the second phase of the rampdown process, in which we aim to fix just the bugs that must be fixed in order to ensure a successful release and understand why we're not going to fix some bugs that perhaps ought to be fixed. Please see the RDP 2 page for details.

The overall feature set is, at this point, frozen. It is highly unlikely that any further JEPs will be targeted to the release. Low-risk enhancements that add small bits of missing functionality or improve usability may be approved, especially when justified by developer feedback, but the bar is now extremely high. API or other specification changes made by a JSR Expert Group are critical by definition, and will be approved. You can request approval for an enhancement via the existing FC extension-request process but, as of RDP 2, only the JDK 9 Project Lead or a delegate, in the case of absence, may approve such requests.

Quick links

Rampdown Phase One	[candidate bugs]
Rampdown Phase Two	[candidate bugs]
Feature-Complete extension request process	[pending requests]
Bug-deferral process (RDP 1 and later)	[pending requests]
Fix-request process (RDP 2 and later)	[pending requests]

Java 9 议程

1 新的特性和功能

2 支持的新标准

3 对用户透明的改进

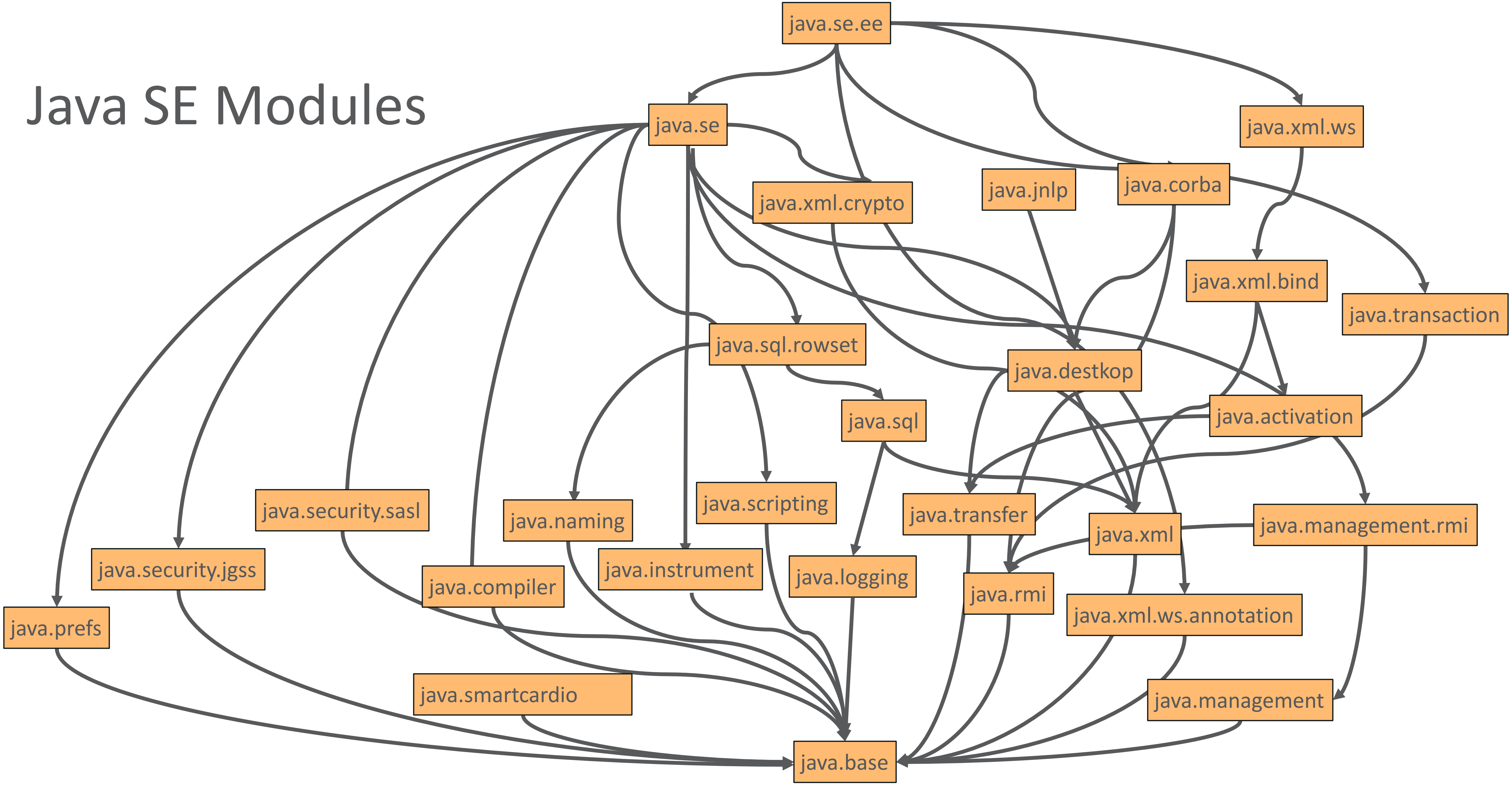
4 承上启下

Project Jigsaw

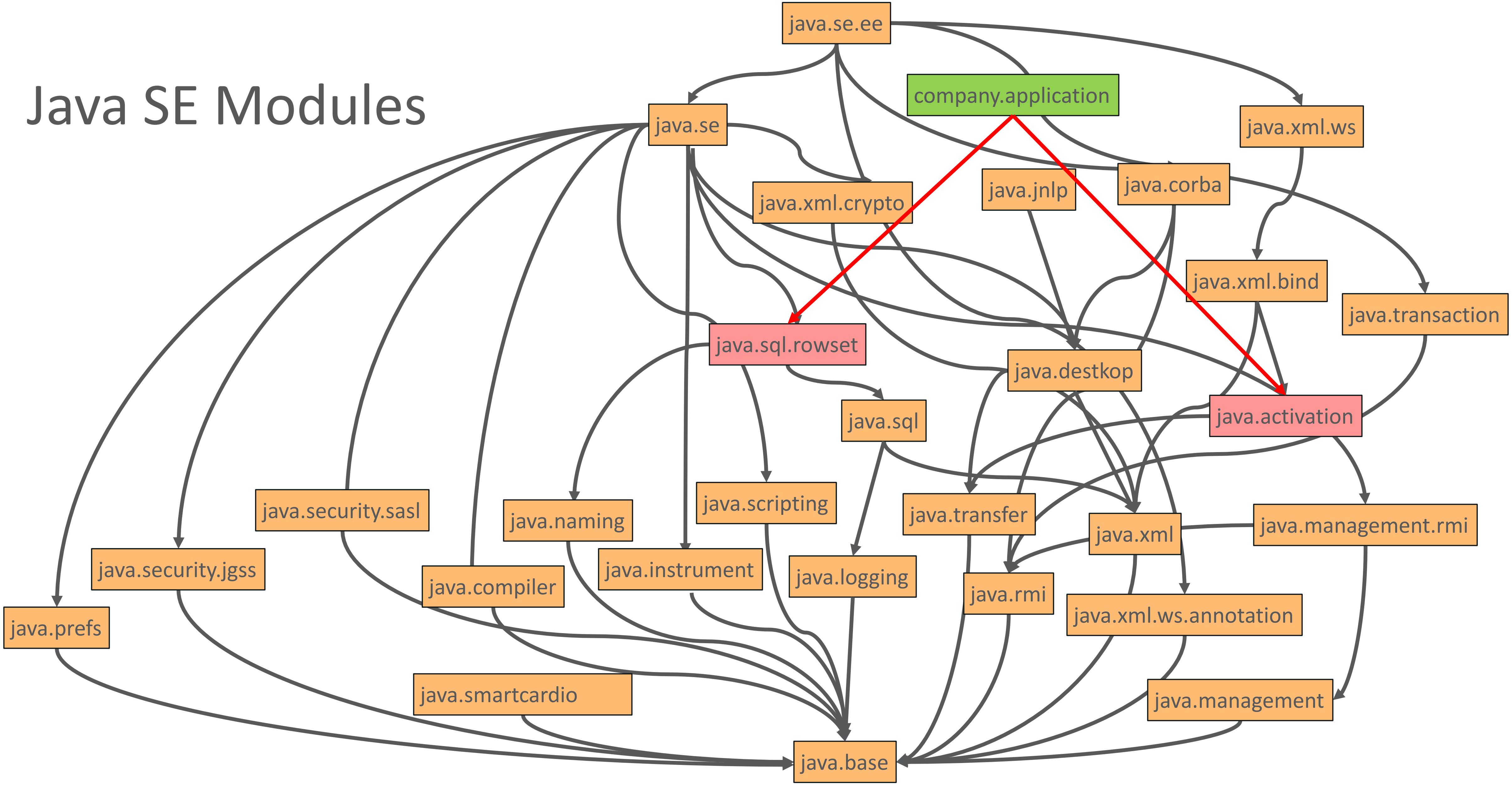
模块化Java平台

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

Java SE Modules

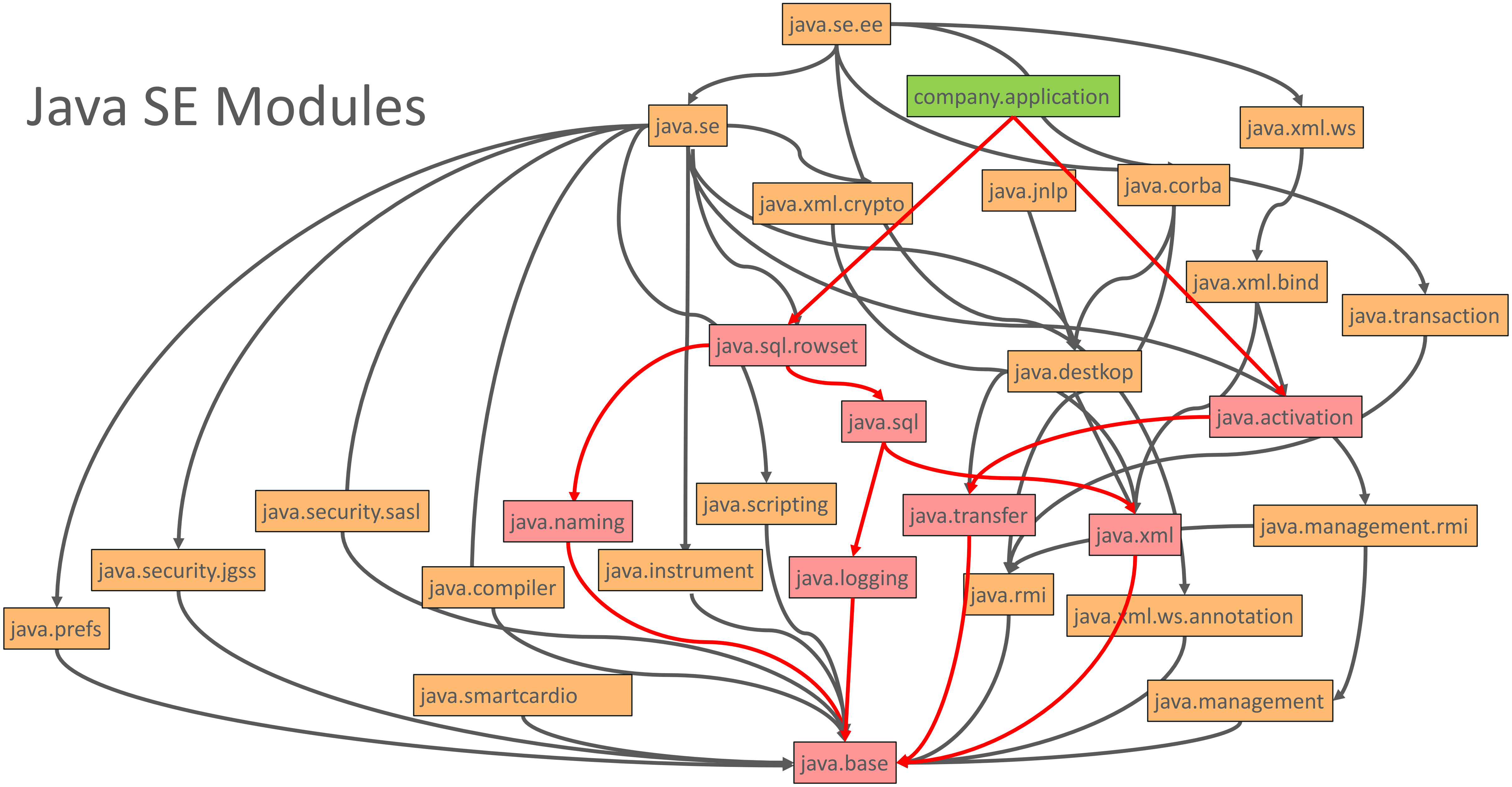


Java SE Modules



ORACLE

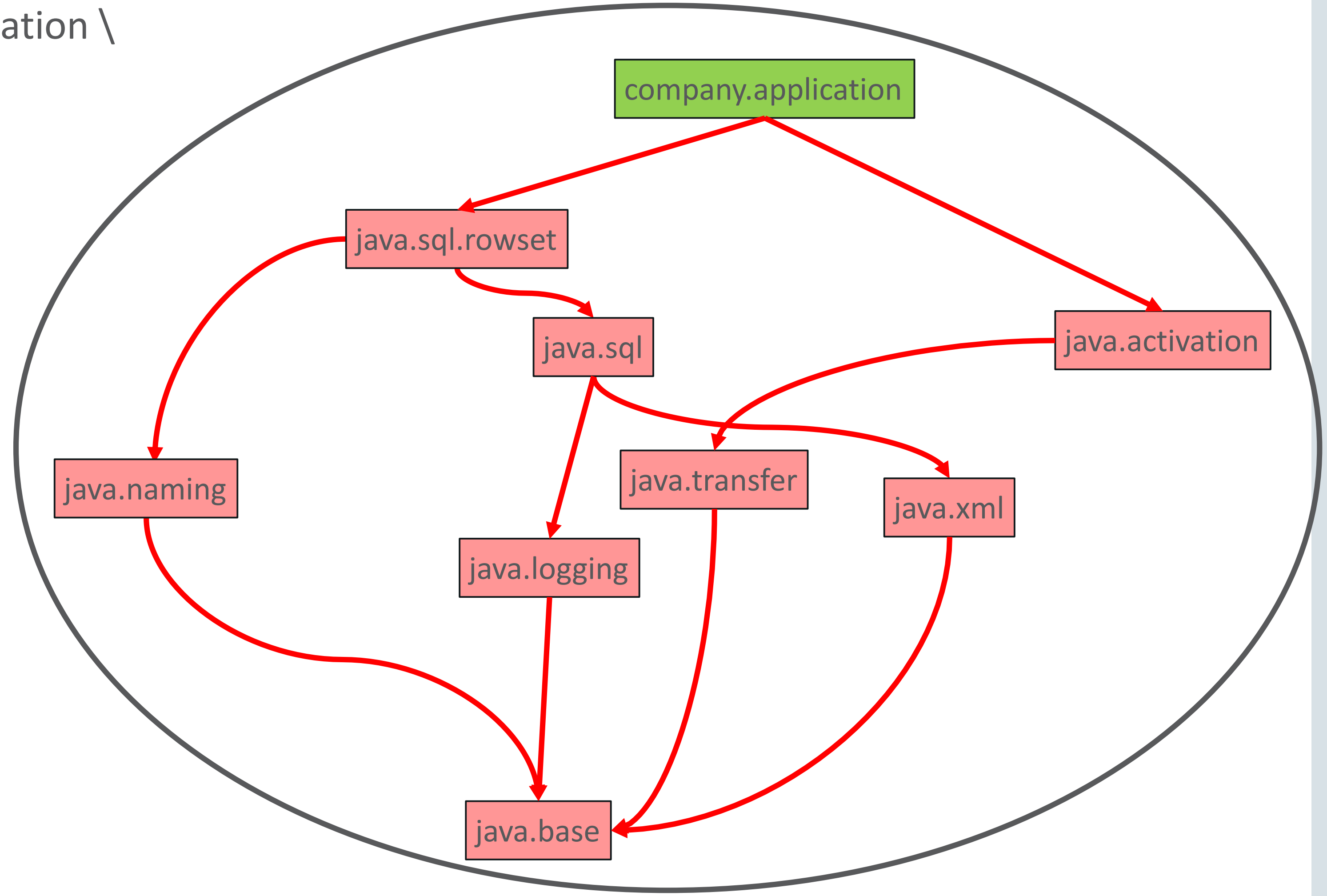
Java SE Modules



定制Java运行环境

```
$ jlink --module-path jmods/ \
        --add-modules java.sql.rowset,java.activation \
        --output myimage
$ myimage/bin/java --list-modules
java.activation@9
java.base@9
java.datatransfer@9
java.logging@9
java.naming@9
java.security.sasl@9
java.sql@9
java.sql.rowset@9
java.xml@9
$ myimage/bin/java –m company.application
```

~ 40 Mb



JEP 269: Convenience Factory Methods for Collections

core-libs / java.util:collections

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);
```

// 一行代码就搞定!

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

ORACLE

JEP 102: Process API Updates

core-libs / java.lang

- 改进管理进程的API

```
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()));
```

JEP 259: Stack-Walking API

core-libs

- 提供高效的标准API去遍历stack
  - 允许 filtering
  - Lazy access stack traces
  - 当前API会要求VM对整个stack进行 snapshot
- 使用场景：获取当前操作的调用者（caller class）最高效的方式？

```
// 使用Stack-Walking API
new StackWalker().walk(
      (s) ->s.map(StackFrame::declaringClass).skip(2).findFirst());
```

JEP 193: Variable Handles

core-libs / java.lang

- 提供标准的API操作object fields, array elements
  - 目前需要使用java.util.concurrent.atomic和sun.misc.Unsafe
- 提供标准的fence operations以影响内存排序
  - 目前需要调用sun.misc.Unsafe
- 提供标准的reachability fence operation

JEP 266: Java并发(Concurrency)API 更新

core-libs / java.util.concurrent

- 提供一个最小集合的API支持Reactive Stream
  - Flow API (Publisher, Subscriber, Processor)
  - 异步的方式处理数据流（Stream）
  - 避免操作线程、同步等，进而避免很多并发问题
  - Memory-efficient
- 改进CompletableFuture API等
  - 增强time-based支持
  - 增强扩展性，比如实现一个子类替换executor

hotspot/compiler新特性

- JEP 165: Compiler Control
  - 改进JVM compilers控制方式
  - 细粒度的控制JVM compilers (C1 and C2)
- JEP 197: Segmented Code Cache
  - 将code cache划分成段（segments），提高性能并便于进一步优化
- JEP 243: Java-Level JVM Compiler Interface
  - 提供Java based JVM compiler interface (JVMCI)，进而支持JVM使用Java编写动态编译器
  - 请注意这并不意味着集成一个动态编译器（比如Graal）

ORACLE

JEP 222: jshell: The Java Shell (Read-Eval-Print Loop -REPL)

tools / jshell

- 简单易用的交互式执行Java代码的工具
- 打开 command-line，然后:

  jdk-9\bin\jshell

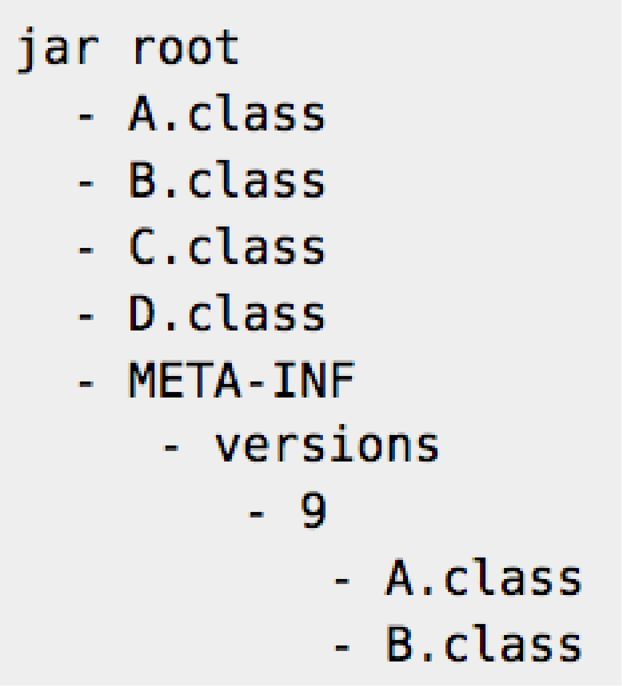
```
Jshell>/help
jshell> ProcessHandle ph = ProcessHandle.current();
jshell> ph.getPid();
jshell> ph.info().command();
```

ORACLE

JEP 238: Multi-Release JAR Files

tools / jar

- 扩展JAR文件格式，允许不同版本class文件共存
- 支持为不同JDK版本提供不同的代码实现



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

Java 9 议程

1 新的特性和功能

2 支持的新标准

3 对用户透明的改进

4 承上启下

Java安全类库支持的新标准或协议

- JEP 219: 支持Datagram Transport Layer Security (DTLS)
- JEP 229: 默认keystore格式从JKS替换为PKCS12
- JEP 244: TLS Application-Layer Protocol Negotiation(ALPN) Extension
  - 以全面支持HTTP/2协议
- JEP 249: 支持OCSP Stapling for TLS
- JEP 273: 实现基于DRBG的SecureRandom
  - 对于Deterministic Random Bit Generator (DRBG) 机制可参考NIST 800-90Ar1
- JEP 287: SHA-3 Hash Algorithms

JEP 110: HTTP/2 Client

core-libs / java.net

- 全新的HTTP client API
  - 支持HTTP/2和WebSocket
  - 用于替换老旧的HttpURLConnection
- 高性能但又简化、轻量级的API
  - 支持同步和异步操作
  - 支持Reactive style编程
- 目前还处于incubator阶段

```
HttpClient client = HttpClient.newBuilder()
          .sslContext(sslContext)
          .version(HTTP_2)
          .build();
HttpRequest req = HttpRequest.newBuilder(uri)
          .POST()
          .build();
client.sendAsync(req, abodyhandler)
         .thenApply(…);
```

其他新的标准

core-libs / java.lang

- Java SE 9支持
  - JEP 227: Unicode 7.0
  - JEP 267: Unicode 8.0
- JEP 226: UTF-8 Property Files
  - 以前的版本是基于ISO-8859-1，不支持的字符需要显示的替换为转义序列
  - 改进属性文件和ResourceBundle API以支持 UTF-8

Java 9 议程

1 新的特性和功能

2 支持的新标准

3 对用户透明的改进

4 承上启下

JEP 254: Compact Strings

core-libs / java.lang

- 优化存储字符串的空间
  - 常见应用中字符串（ String ）占用非常可观的内存
- 修改String实现，以 byte[]数组和一个编码标记替换char[] (16 bits)数组
- 对API的使用者完全透明

安全类库的一些有意思的改进

- **JEP 232: 提高安全应用性能**
  - 开启security manager通常会导致 10-15%的性能下降
  - Java 9的改进显著降低了开销
- **JEP 246: 利用CPU指令优化 GHASH and RSA**
  - 利用SPARC and Intel x64 CPU的部分新指令
  - 部分加密函数的性能非常显著，比如
    - 相比于JDK 8， AES 性能提高了8倍

JEP 250: Store Interned Strings in CDS Archives

hotspot / runtime

- 通过共享字符串和内部数组对象，降低多个JVM进程的内存消耗
  - 对CDS的进一步增强
  - 对于大规模的云计算部署有显著价值
- 仅支持G1
  - 共享字符串需要pinned region（只有G1支持pinning）
- 仅支持 64-bit 开启对象和类指针压缩的平台

JEP 143: 改进竞争锁（Contended Locking）

hotspot / runtime

- 改进高度竞争（high-contended）的Java Object Monitor性能
  - 高度竞争意味着多个（大量）线程同时试图获取一个锁
  - 实现更快的 Java monitor enter, exit, notify/notifyAll等
- 不影响 internal VM的monitors/mutex, 因为是不同的代码实现

Java 9 议程

1 新的特性和功能

2 支持的新标准

3 对用户透明的改进

4 承上启下

JEP 295: Ahead-of-Time Compilation

hotspot / compiler

- 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
    ```

ORACLE

JEP 248: 将G1作为默认垃圾收集器

hotspot / gc

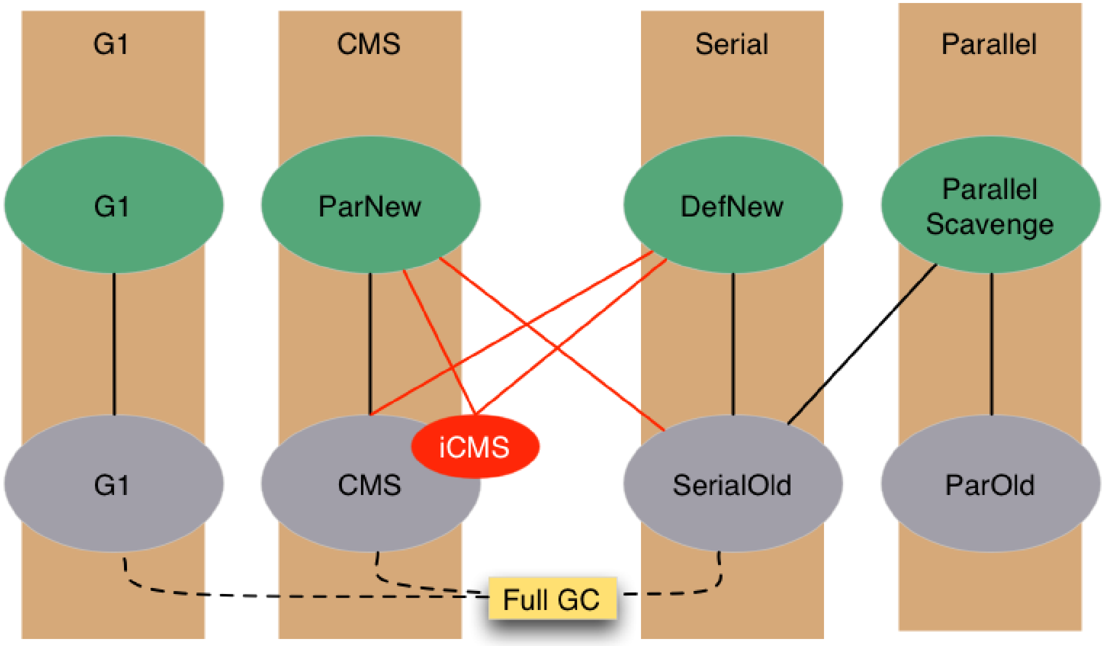
- 将G1作为server模式的默认选项
- 目前默认是Parallel GC（吞吐量优先）
- G1是一个非常健壮并经过充分测试的收集器:
  - 通用场景中，延迟比吞吐量更能提高用户体验
  - 直接设定延迟目标，能够达到延迟 SLAs
  - 最坏场景的延迟表现优于CMS (设计原理导致碎片化问题）

ORACLE

其他JVM变化

hotspot / gc

- JEP 214: 移除过时的GC组合，移除 Incremental CMS (iCMS)
- JEP 291: Deprecate CMS
- JEP 158/ JEP 271: 统一日志（JVM/GC ）
  - 引入适用于JVM各个模块的通用日志机制
  - 比如，“-Xlog:gc”提供”-XX:PrintGC”相似的能力



JEP 280: Indify String Concatenation

tools / javac

- 目前Javac会将字符串拼接操作转换成StringBuilder调用.
  - 这种优化有时候未必是最优
  - StringBuilder预分配的大小必须合适
  - 脆弱并难以维护
- 改为利用invokedynamic调用JDK类库.
  - 新添加 `java.lang.invoke.StringConcatFactory`
  - 编译器利用新的类库, JVM会对此进行优化

JEP 213: Project Coin的后续工作

tools / javac

- Project Coin / JSR 334 (Java SE 7)的一些遗留问题

1. 允许 @SafeVargs 使用在private instance methods
2. 允许 effectively-final变量用于 try-with-resources语句
3. 允许”<>”用于匿名类，如果类型推断有效
4. “_” 不再是合法的identifier名称
5. 支持private interface methods

- 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

Behind the scenes

- **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
- Compiler Control
- Java-Level JVM Compiler Interface
- Segmented Code Cache

New functionality

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

New functionality

- HTTP 2 Client
- Unicode 8.0
- UTF-8 Property Files
- Implement Selected ECMAScript 6 Features in Nashorn
- 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

New standards

- 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
- Ahead-of-Time Compilation
- 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
- 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
- Filter Incoming Serialization Data

Housekeeping

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

Gone

ORACLE

ORACLE®