
ORACLE

# JDK 8 & Beyond

Dalibor Topić (@robilad)
Principal Product Manager, Java Platform Group
January 17th, 2013 - JUG Ostfalen, Braunschweig

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.

# Evolving the Language

*From "Evolving the Java Language" - JavaOne 2005*

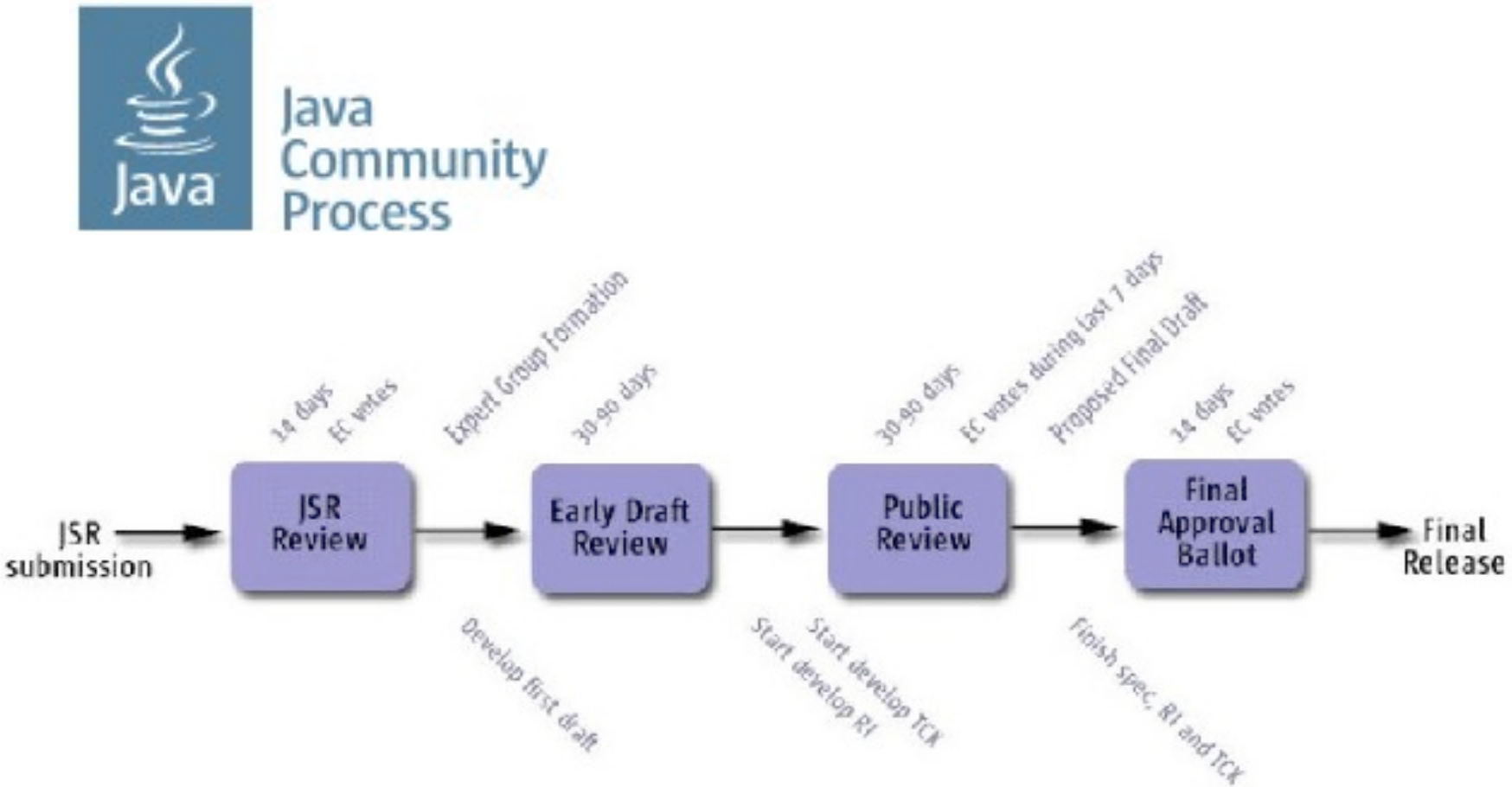
- Java language principles
  - Reading is more important than writing
  - Code should be a joy to read
  - The language should not hide what is happening
  - Code should do what it seems to do
  - Simplicity matters
  - Every "good" feature adds more "bad" weight
  - Sometimes it is best to leave things out
- One language: with the same meaning everywhere
  - No dialects
- We will evolve the Java language
  - But cautiously, with a long term view
  - "first do no harm"

*also "Growing a Language" - Guy Steele 1999*
*"The Feel of Java" - James Gosling 1997*

# Java SE 7 Release Contents

- Java Language
  - Project Coin (JSR-334)
- Class Libraries
  - NIO2 (JSR-203)
  - Fork-Join framework, ParallelArray (JSR-166y)
- Java Virtual Machine
  - The DaVinci Machine project (JSR-292)
  - InvokeDynamic bytecode
- Miscellaneous things
- JSR-336: Java SE 7 Release Contents

# How Java Evolves and Adapts




**JSR-348: JCP.next**





JavaOne ORACLE

OpenJDK

OpenJDK FAQ
Installing
Contributing
Sponsoring
Developers' Guide

Mailing lists
IRC · Wiki

Bylaws · Census
Legal

**Groups**
(overview)
2D Graphics
AWT
Build
Compiler
Conformance
Core Libraries
Governing Board
HotSpot
Internationalization
JMX
Members
Networking
NetBeans Projects
Porters
Quality
Security
Serviceability
Sound

# Java SE 8 Platform Umbrella JSR (337)

This is the primary web page for JSR 337, the Platform Umbrella JSR for Java SE 8.

The original JSR submission may be found on the official JCP page.

## Expert Group

- Kevin Bourrillion (Google)
- Andrew Haley (Red Hat)
- Steve Poole (IBM)
- Mark Reinhold (Oracle)

## Mailing lists

There are three mailing lists:

- java-se-8-spec-experts is the Expert Group (EG) list. Only EG members may subscribe and post to this list, but the archives are public.
- java-se-8-spec-observers is for those who wish to monitor and, perhaps, discuss the EG's progress. Messages sent to the primary EG list are automatically forwarded to this list. Anyone may subscribe to this list, and any subscriber may post. EG members are under no obligation to follow the traffic on this list.
- java-se-8-spec-comments is for sending comments, suggestions, and other feedback directly to the EG. Only EG members may subscribe to this list, but anyone may post, and the archives are public. The EG will read all messages sent to this list.

## Issue tracker

Comments on specific elements of draft specifications should be submitted into the issue tracker.





JavaOne ORACLE


## JDK 8

« home · features · milestones · builds »

The goal of this Project is to to produce an open-source reference implementation of the Java SE 8 Platform, to be defined by JSR 337 in the Java Community Process.

### Content

JDK 8 is the second part of Plan B. The proposed release-driver features are the Lambda and Jigsaw Projects (though note that a proposal has been made to defer Jigsaw to the next release). Additional features proposed via the JEP Process will be included, but they must fit into the overall schedule required for the release drivers. Detailed information on the features funded and targeted to the release, so far, can be found on the features page.






JavaOne ORACLE


The Source for Java Technology Collaboration

**JDK 8**

- Downloads
- Feedback Forum
- OpenJDK
- Planet JDK

# JDK 8 Project

Building the next generation of the Java SE platform

## Download JDK 8

- JDK 8 snapshot release
- **Source code** (instructions)

For details about JDK 8, please see the JDK 8 and Lambda project pages.

## JDK 8 Early Access Now Available!

Try it out today!






Java.net The Source for Java Technology Collaboration

Login | Register | Help

JDK 8
- Downloads
- Feedback Forum
- OpenJDK
- Planet JDK

## Java™ Platform, Standard Edition 8 Early Access Releases

*January 10, 2013*

# 8 Build b72

Summary of changes in JDK 8 build b72

Previous Early Access Releases
Feedback forum
Report Bugs

**Please note:**

This list offers files for different platforms - please be sure to select the proper file(s) for your platform.
Carefully review the files listed below to select the ones you want, then click the link(s) to download.

### License Agreement:

You must accept the Pre-Production Software Evaluation Agreement for Java SE to download this software.

◯ Accept License Agreement | ◯ Decline License Agreement

### Documentation

- JDK Docs (61.20 MB zip | HTML)
- JavaFX Docs (8.91 MB zip | HTML)

| Platforms | | JRE | JDK |
|---|---|---|---|
| Windows | 32-bit | exe ( md5) 27.76 MB | exe ( md5) 99.96 MB |
| | 64-bit | exe ( md5) 29.31 MB | exe ( md5) 102.45 MB |
| Mac OS X | 64-bit | dmg( md5 ) 53.54 MB | dmg ( md5) 150.00 MB |
| Linux | 32-bit | tar.gz ( md5) 50.90 MB | tar.gz ( md5) 98.04 MB |
| | 64-bit | tar.gz ( md5) 49.59 MB | tar.gz ( md5) 96.75 MB |
| Solaris SPARC | 32-bit | tar.gz ( md5) 57.14 MB | tar.gz ( md5) 99.91 MB |
| | 64-bit* | tar.gz ( md5) 17.55 MB | tar.gz ( md5) 17.73 MB |
| Solaris | 32-bit* | tar.gz ( md5) 96.20 MB | tar.gz ( md5) 96.20 MB |
| | 64-bit* | tar.gz ( md5) 14.99 MB | tar.gz ( md5) 15.15 MB |

*Note: Solaris 64-bit requires users to first install the 32-bit version.





JavaOne ORACLE

Java.net The Source for Java Technology Collaboration

Login | Register | Help

JDK 8
- Downloads
- Feedback Forum
- OpenJDK
- Planet JDK

## JDK 8 (with JavaFX) for ARM Early Access

This page contains a JDK 8 including JavaFX on Linux for ARM processors. The Early Access is provided to the community so that we can get feedback on the ongoing progress of the project. We wanted to get this release out to you as quickly as we can so you can start using this build of JDK 8 on an ARM device, such as the a Raspberry Pi.

## Please Help Us Test!

1. Follow the documentation instructions to setup JDK 8 (with JavaFX) on a Raspberry Pi device
2. Run the Demos and Samples below
3. You could also write your own application and run it
4. See below for reporting feedback and issues

You must accept the Pre-Production Software Evaluation Agreement for the JDK and JRE to download this software.

◯ Accept License Agreement | ◯ Decline License Agreement

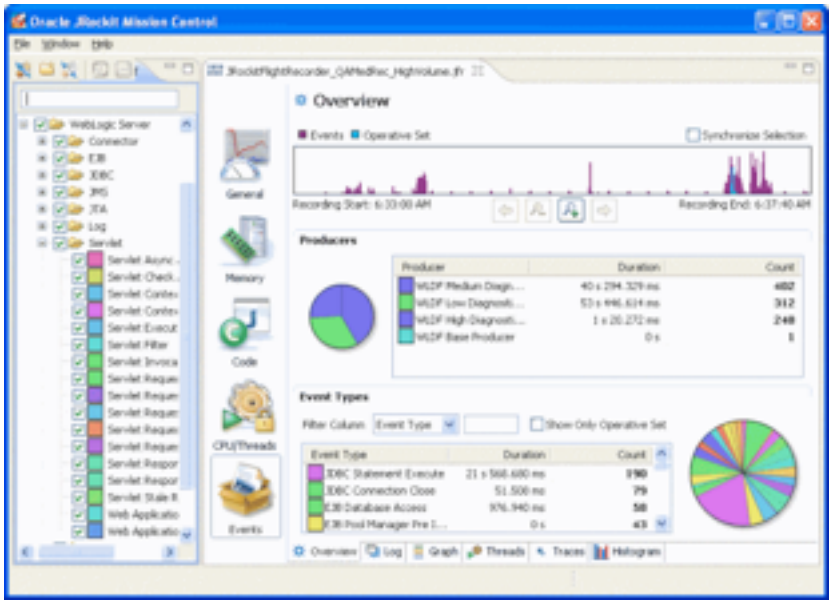
| Download | Size | Release |
|---|---|---|
| Oracle JDK 8 (with JavaFX) for ARM Early Access | 58 MB (md5) | Dec 18, 2012 |





JavaOne ORACLE

# JVM Convergence







JavaOne ORACLE

## JEP 122: Remove the Permanent Generation

| | |
|---|---|
| *Author* | Jon Masamitsu |
| *Organization* | Oracle |
| *Created* | 2010/8/15 |
| *Updated* | 2012/8/20 |
| *Type* | Feature |
| *State* | Funded |
| *Component* | vm/gc |
| *Scope* | Impl |
| *RFE* | 6964458 |
| *Internal-refs* | Oracle:A360:682265 |
| *Discussion* | hotspot dash dev at openjdk dot java dot net |
| *Start* | 2010/Q3 |
| *Effort* | XL |
| *Duration* | XL |
| *Reviewed-by* | Paul Hohensee |
| *Endorsed-by* | Paul Hohensee |
| *Funded-by* | Oracle |
| *Release* | 8 |

### Summary

Remove the permanent generation from the Hotspot JVM and thus the need to tune the size of the permanent generation.

### Non-Goals

Extending Class Data Sharing to application classes. Reducing the memory needed for class metadata. Enabling asynchronous collection of class metadata.

### Success Metrics

Class metadata, interned Strings and class static variables will be moved from the permanent generation to either the Java heap or native memory.

The code for the permanent generation in the Hotspot JVM will be removed.

Application startup and footprint will not regress more than 1% as measured by a yet-to-be-chosen set of benchmarks.

### Motivation

This is part of the JRockit and Hotspot convergence effort. JRockit customers do not need to configure the permanent generation (since JRockit does not have a permanent generation) and are accustomed to not configuring the permanent generation.

**hsx/hotspot-gc/hotspot / changeset**

summary | shortlog | changelog | tags | manifest | changeset | raw | bz2 | zip | gz

**6964458: Reimplement class meta-data storage to use native memory** default tip

| | |
|---|---|
| author | coleenp |
| | Sat Sep 01 13:25:18 2012 -0400 (45 hours ago) |
| changeset 3599 | da91efe96a93 |
| parent 3598 | 36d1d483d5d6 |

6964458: Reimplement class meta-data storage to use native memory
Summary: Remove PermGen, allocate meta-data in metaspace linked to class loaders, rewrite GC walking, rewrite and rename metadata to be C++ classes
Reviewed-by: jmasa, stefank, never, coleenp, kvn, brutisso, mgerdin, dholmes, jrose, twisti, roland
Contributed-by: jmasa <jon.masamitsu@oracle.com>, stefank <stefan.karlsson@oracle.com>, mgerdin <mikael.gerdin@oracle.com>, never <tom.rodriguez@oracle.com>

# Big Disclaimer

The syntax used in the following slides may change

Caveat emptor

```
class Student {
    String name;
    int gradYear;
    double score;
}

Collection<Student> students = ...;
```

```
Collection<Student> students = ...;

double max = Double.MIN_VALUE;

for (Student s : students) {
    if (s.gradYear == 2011)
        max = Math.max(max, s.score);
}
```

```
Collection<Student> students = ...;

double max = 0.0;

for (Student s : students) {
    if (s.gradYear == 2011)
        max = Math.max(max, s.score);
}
```

```
Collection<Student> students = ...;

max = students.filter(new Predicate<Student>() {
                  public boolean op(Student s) {
                     return s.gradYear == 2011;
                  }
               }).map(new Extractor<Student, Double>() {
                  public Double extract(Student s) {
                     return s.score;
                  }
               }).reduce(0.0, new Reducer<Double, Double>() {
                  public Double reduce(Double max, Double score) {
                    return Math.max(max, score);
                  }
               });
```

# Inner Classes Are Imperfect Closures

- Bulky syntax
- Unable to capture non-final local variables
- Transparency issues
  - Meaning of return, break, continue, this
- No non-local control flow operators

# Single Abstract Method (SAM) Types

- Lots of examples in the Java APIs
  - `Runnable, Callable, EventHandler, Comparator`

```
foo.doSomething(new CallbackHandler() {
  public void callback(Context c) {
    System.out.println(c.v());
  }
});
```

- Noise:Work ratio is 5:1
- Lambda expressions grow out of the idea of making callback objects easier

```
Collection<Student> students = ...;

max = students.filter((Student s) -> s.gradYear == 2011)
              .map((Student s) -> s.score)
              .reduce(0.0,
                (Double max, Double score) ->
                   Math.max(max, score));

max = students.filter(s -> s.gradYear == 2011)
              .map(s -> s.score)
              .reduce(0.0, Math::max);

max = students.parallel()
              .filter(s -> s.gradYear == 2011)
              .map(s -> s.score)
              .reduce(0.0, Math::max);
```

```
Collection<Student> students = ...;

double max =          // Lambda expressions
  students.filter(Students s -> s.gradYear == 2011})
          .map(Students s -> s.score })
          .reduce(0.0, Math::max);
```

```
interface Collection<T> {
   int add(T t);
   int size();
   void clear();
   ...
}
```

# How to extend an interface in Java SE 8

tells us this method extends the interface

```
public interface Set<T> extends Collection<T>
{

    public int size();

  ...    // The rest of the existing Set methods


    public extension T reduce(Reducer<T> r)
      default Collections.<T>setReducer;
}
```

Implementation to use if none exists for the implementing class

```
Collection<Student> students = ...;

double max =          // Lambda expressions
  students.filter(Students s -> s.gradYear == 2010)
          .map(Students s -> s.score )
          . reduce(0.0, Math#max);

interface Collection<T> {  // Default methods
   extension Collection<E> filter(Predicate<T> p)
        default Collections.<T>filter;

   extension <V> Collection<V> map(Extractor<T,V> e)
        default Collections.<T>map;

   extension <V> V reduce()
        default Collections.<V>reduce;
}
```

Java.net The Source for Java Technology Collaboration

**JDK 8**

- Downloads
- Feedback Forum
- OpenJDK
- Planet JDK

## Java™ Platform, Standard Edition 8 Early Access with Lambda Support

This page provides an Early Access of OpenJDK with Lambda (JSR 335) support. The Lambda project aims to support programming in a multicore environment by adding closures and related features to the Java language

For documentations and other details, please see the Lambda project page.

**Please note:**
The Lambda project has used source files that are not yet available in JDK8; therefore, these early access builds are created using the latest OpenJDK 7 source repository. This project will merge into OpenJDK 8 when the source files are available.

These bundles are meant to allow developers to try the Lambda features without making their own compilations. If you are looking for the latest JDK 8 builds, please download from here.

**License Agreement:**
You must accept the Pre-Production Software Evaluation Agreement for Java SE to download this software.

Accept License Agreement | Decline License Agreement

**Downloads (b50)**

| Platforms | | JDK |
|---|---|---|
| Windows | | zip ( md5) 82 MB |
| Windows x64 | | zip ( md5) 76 MB |
| Solaris SPARC | 32-bit | tar.gz ( md5) 333 MB |
| | 64-bit* | tar.gz ( md5) 492 MB |
| Solaris | x86 | tar.gz ( md5) 338 MB |
| | x64* | tar.gz ( md5) 495 MB |
| Linux | | tar.gz ( md5) 80 MB |
| Linux x64 | | tar.gz ( md5) 137 MB |
| Mac OS X | | tar.gz ( md5) 66 MB |

*Note: Solaris 64-bit requires users to first install the 32-bit version.





JavaOne ORACLE

Java.net The Source for Java Technology Collaboration

JDK 8
- Downloads
- Feedback Forum
- OpenJDK
- Planet JDK

## Java™ Platform, Standard Edition 8 Early Access with Type Annotation Support

The Type Annotations project (JSR 308) extends the Java language so that annotations may appear on essentially any use of a type. This page provides an Early Access of OpenJDK with Type Annotations (JSR 308) support.

For documentation and other details, please see the Type Annotations project page.

**Please note:**
These bundles are meant to allow developers to try the Type Annotations feature without building JDK8 themselves.
If you are looking for the very latest JDK8 builds (without Type Annotations), please download from here.

### License Agreement:

You must accept the Pre-Production Software Evaluation Agreement for Java SE to download this software.

◯ Accept License Agreement | ◯ Decline License Agreement

### Downloads (b69)

| Platforms | | JDK |
|---|---|---|
| Windows | | zip ( md5) 85 MB |
| Windows x64 | | zip ( md5) 80 MB |
| Solaris SPARC | 32-bit | tar.gz ( md5) 332 MB |
| | 64-bit* | tar.gz ( md5) 489 MB |
| Solaris | x86 | tar.gz ( md5) 332 MB |
| | x64* | tar.gz ( md5) 486 MB |
| Linux | | tar.gz ( md5) 84 MB |
| Linux x64 | | tar.gz ( md5) 129 MB |
| Mac OS X | | tar.gz ( md5) 69 MB |

*Note: Solaris 64-bit requires users to first install the 32-bit version.




JavaOne ORACLE

## *There's not a moment to lose!*

Mark Reinhold's Blog

### Project Jigsaw: Late for the train

2012/07/17 08:58:00 -07:00

The aim of Project Jigsaw is to design and implement a standard module system for the Java SE Platform, and to apply that system to the Platform itself and to the JDK.

Jigsaw is currently slated for Java 8. The proposed development schedule for Java 8 expects work on major features to be finished by May 2013, in preparation for a final release around September. Steady progress is being made, but some significant technical challenges remain. There is, more importantly, not enough time left for the broad evaluation, review, and feedback which such a profound change to the Platform demands.

I therefore propose to defer Project Jigsaw to the next release, Java 9. In order to increase the predictability of all future Java SE releases, I further propose to aim explicitly for a regular two-year release cycle going forward.

## *There's not a moment to lose!*

Mark Reinhold's Blog

### Project Jigsaw: Late for the train: The Q&A

2012/08/24 08:52:12 -07:00

I recently proposed, to the Java community in general and to the SE 8 (JSR 337) Expert Group in particular, to defer Project Jigsaw from Java 8 to Java 9. I also proposed to aim explicitly for a regular two-year release cycle going forward. Herewith a summary of the key questions I've seen in reaction to these proposals, along with answers.

### Making the decision

**Q** Has the Java SE 8 Expert Group decided whether to defer the addition of a module system and the modularization of the Platform to Java SE 9?

**A** No, it has not yet decided.

**Q** By when do you expect the EG to make this decision?

**A** In the next month or so.

**Q** How can I make sure my voice is heard?

**A** The EG will consider all relevant input from the wider community. If you have a prominent blog, column, or other communication channel then there's a good chance that we've already seen your opinion. If not, you're welcome to send it to the Java SE 8 Comments List, which is the EG's official feedback channel.

**Q** What's the overall tone of the feedback you've received?

**A** The feedback has been about evenly divided as to whether Java 8 should be delayed for Jigsaw, Jigsaw should be deferred to Java 9, or some other, usually less-realistic, option should be taken.

```
$ java org.planetjdk.aggregator.Main
```

```
$ java -cp $APPHOME/lib/jdom-1.0.jar:
$APPHOME/lib/jaxen-1.0.jar:
$APPHOME/lib/saxpath-1.0.jar:
$APPHOME/lib/rome.jar-1.0.jar:
$APPHOME/lib/rome-fetcher-1.0.jar:
$APPHOME/lib/joda-time-1.6.jar:
$APPHOME/lib/tagsoup-1.2.jar:
org.planetjdk.aggregator.Main
```

```
$ java -cp $APPHOME/lib/jdom-1.0.jar:
$APPHOME/lib/jaxen-1.0.jar:
$APPHOME/lib/saxpath-1.0.jar:
$APPHOME/lib/rome.jar-1.0.jar:
$APPHOME/lib/rome-fetcher-1.0.jar:
$APPHOME/lib/joda-time-1.6.jar:
$APPHOME/lib/tagsoup-1.2.jar:
org.planetjdk.aggregator.Main
```

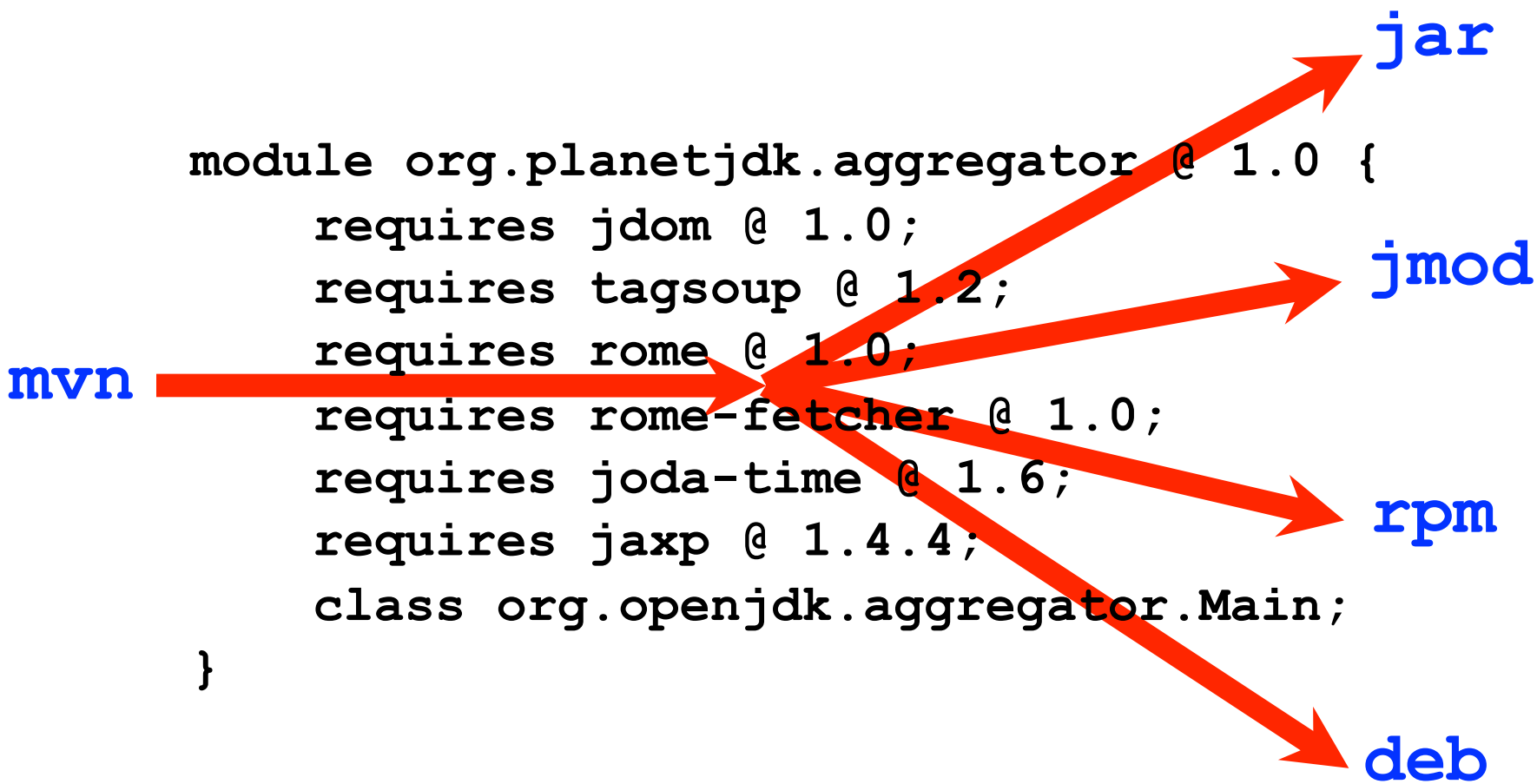
# module-info.java

```
module org.planetjdk.aggregator @ 1.0 {
    requires jdom @ 1.0;
    requires tagsoup @ 1.2;
    requires rome @ 1.0;
    requires rome-fetcher @ 1.0;
    requires joda-time @ 1.6;
    requires jaxp @ 1.4.4;
    class org.openjdk.aggregator.Main;
}
```






JavaOne ORACLE



Java.net The Source for Java Technology Collaboration

**JDK 8**

- Downloads
- Feedback Forum
- OpenJDK
- Planet JDK

## Java™ Platform, Standard Edition 8 Early Access with Project Jigsaw

This page provides an Early Access of OpenJDK with Project Jigsaw support. The goal of Project Jigsaw is to design and implement a standard module system for the Java SE Platform, and to apply that system to the Platform itself and to the JDK.

For documentations and other details, please see the Project Jigsaw page.

These bundles are meant to allow developers to try out Project Jigsaw without needing to build it from sources. If you are looking for the latest JDK 8 builds, please download from here.

**License Agreement:**

You must accept the Pre-Production Software Evaluation Agreement for Java SE to download this software.

Accept License Agreement | Decline License Agreement

**JDK modules image**: This download is equivalent to the normal JDK download except that all components are pre-installed as modules. Note that the runtime no longer contains a "jre" directory, and rt.jar and tools.jar no longer exist.

**JDK base image + jmod packages**: This download contains a minimal "base" runtime and a directory of jmod packages with the JDK modules. The jmod packages can be installed directly via the "jmod install" command, or added to a file or http based module repository and installed automatically when required.

**Downloads (b42):**

See Quick Start Guide to get started.

See Release Notes for known issues.

| Platforms | JDK modules image | JDK base image + jmod packages |
|---|---|---|
| Windows | zip ( md5) 52 MB | zip ( md5) 55 MB |
| Windows x64 | zip ( md5) 46 MB | zip ( md5) 44 MB |
| Solaris SPARC | tar.gz ( md5) 310 MB | tar.gz ( md5) 566 MB |
| Solaris SPARCv9 | tar.gz ( md5) 468 MB | tar.gz ( md5) 719 MB |
| Solaris x86 | tar.gz ( md5) 315 MB | tar.gz ( md5) 578 MB |
| Solaris x64 | tar.gz ( md5) 330 MB | tar.gz ( md5) 591 MB |
| Linux | tar.gz ( md5) 57 MB | tar.gz ( md5) 63 MB |
| Linux x64 | tar.gz ( md5) 115 MB | tar.gz ( md5) 178 MB |
| Mac OS X | tar.gz ( md5) 43 MB | tar.gz ( md5) 32 MB |





JavaOne ORACLE

# Additional Disclaimers

- Some ***ideas*** for the Java Platform are shown on the following slides
- Large R&D effort required
- Content and timing highly speculative
- Some things will turn out to be bad ideas
- New ideas will be added
- Java's future is bright (in our humble opinion)!

# Java SE 9 (and beyond…)

| | |
|---|---|
| **Interoperability** | • Multi-language JVM<br>• Improved Java/Native integration |
| **Cloud** | • Multi-tenancy support<br>• Resource management |
| **Ease of Use** | • Self-tuning JVM<br>• Language enhancements |
| **Advanced Optimizations** | • Unified type system<br>• Data structure optimizations |
| **Works Everywhere and with Everything** | • Scale down to embedded, up to massive servers<br>• Support for heterogenuous compute models |

# Vision: Interoperability

- Improved support for non-Java languages
  - Invokedynamic (done)
  - Java/JavaScript interop (in progress – JDK 8)
  - Meta-object protocol (JDK 9)
  - Long list of JVM optimizations (JDK 9+)
- Java/Native
  - Calls between Java and Native without JNI boilerplate (JDK 9)

# Vision: Cloud

- Multi-tenancy (JDK 8+)
  - Improved sharing between JVMs in same OS
  - Per-thread/threadgroup resource tracking/management
- Hypervisor aware JVM (JDK 9+)
  - Co-operative memory page sharing
  - Co-operative lifecycle, migration

# Vision: Language Features

- Large data support (JDK 9)
  - Large arrays (64 bit support)
- Unified type system (JDK 10+)
  - No more primitives, make everything objects
- Other type reification (JDK 10+)
  - True generics
  - Function types
- Data structure optimizations (JDK 10+)
  - Structs, multi-dimensional arrays, etc
  - Close last(?) performance gap to low-level languages

# Vision: Integration

- Modern device support (JDK 8+)
  - Multitouch (JDK 8)
  - Location (JDK 8)
  - Sensors – compass, accelerometer, temperature, pressure, ... (JDK 8+)
- Heterogenous compute models (JDK 9+)
  - Java language support for GPU, FPGA, offload engines, remote PL/SQL...

# The Path Forward

- Open development
  - Prototyping and R&D in OpenJDK
  - Cooperate with partners, academia, greater community
- Work on next JDK, future features in parallel
- 2-year cycle for Java SE releases

# Conclusions

- The Java platform will continue to evolve
- Java SE 8 will add some nice, big features
- Expect to see more in Java SE 9 and beyond

# Q&A