What new will bring us Java 9?

Dominik Przybysz

@alien11689

http://przybyszd.blogspot.com

https://github.com/alien11689

Short history of Java part 1

1995 - JDK Aplha and Beta

1996/01 - JDK 1.0 - *Oak*

1997/02 - JDK 1.1 (RMI, JDBC, JIT)

1998/12 - J2SE 1.2

1999/12 - J2EE 1.2

2000/05 - J2SE 1.3

2001/09 - J2EE 1.3

Short history of Java part 2

2002/02 - J2SE 1.4

2003/11 - J2EE 1.4

2004/09 - J2SE 1.5

2006/05 - Java EE 5

2006/12 - Java SE 6

2009/04 - Oracle buys Sun

2009/12 - Java EE 6

Short history of Java part 3

2011/07 - Java SE 7

2013/06 - Java EE 7

2014/03 - Java SE 8

2016/09 - Java SE 9 / Java EE 8

2018 - Java 10

Java 9

2015/12/10 - Feature Complete

2016/02/04 - All Tests Run

2016/02/25 - Rampdown Start

2016/04/21 - Zero Bug Bounce

2016/06/16 - Rampdown Phase 2

2016/07/21 - Final Release Candidate

2016/09/22 - General Availability

Modularization

Modularization

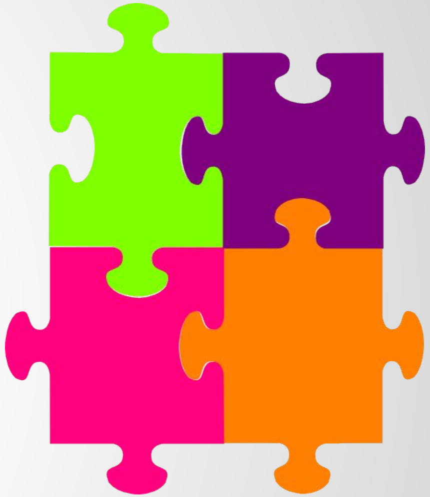
- improvements to performance
- improvements to security
- reduce system complexity
- fix classpath/jar hell

Compact profiles (Java 8)

- compiled for embedded systems
- compact1 - 14 MB
- compact2 - 18 MB
- compact3 - 21 MB
- Full - 45 MB

Project Jigsaw

- JEP 200: The Modular JDK
- JEP 201: Modular Source Code
- JEP 220: Modular Run-Time Images
- JSR 376: Java Platform Module System



What will change?

- no sun.* and *.internal.* packages
- no rt.jar
- improve SecurityManager.checkPackageAccess
- new linker - jlink
- no endorsed lib
- no extension mechanism
- no jre - building images

No sun.misc.Unsafe!

(maybe...)

sun.misc.Unsafe

Do we need to get rid of Unsafe, Yes! Do we need to keep Unsafe? Yes. How do we reconcile these seemingly conflicting positions? Well, we need to plan a migration path to move the stuff that we now know how it should be implemented out of Unsafe and into the JDK proper.

https://jaxenter.com/java-9-without-sun-misc-unsafe-119026.html

JEP 260: Encapsulate Most Internal APIs

Critical internal APIs for which replacements are introduced in JDK 9 will be deprecated in JDK 9 and either encapsulated or removed in JDK 10.

http://openjdk.java.net/jeps/260

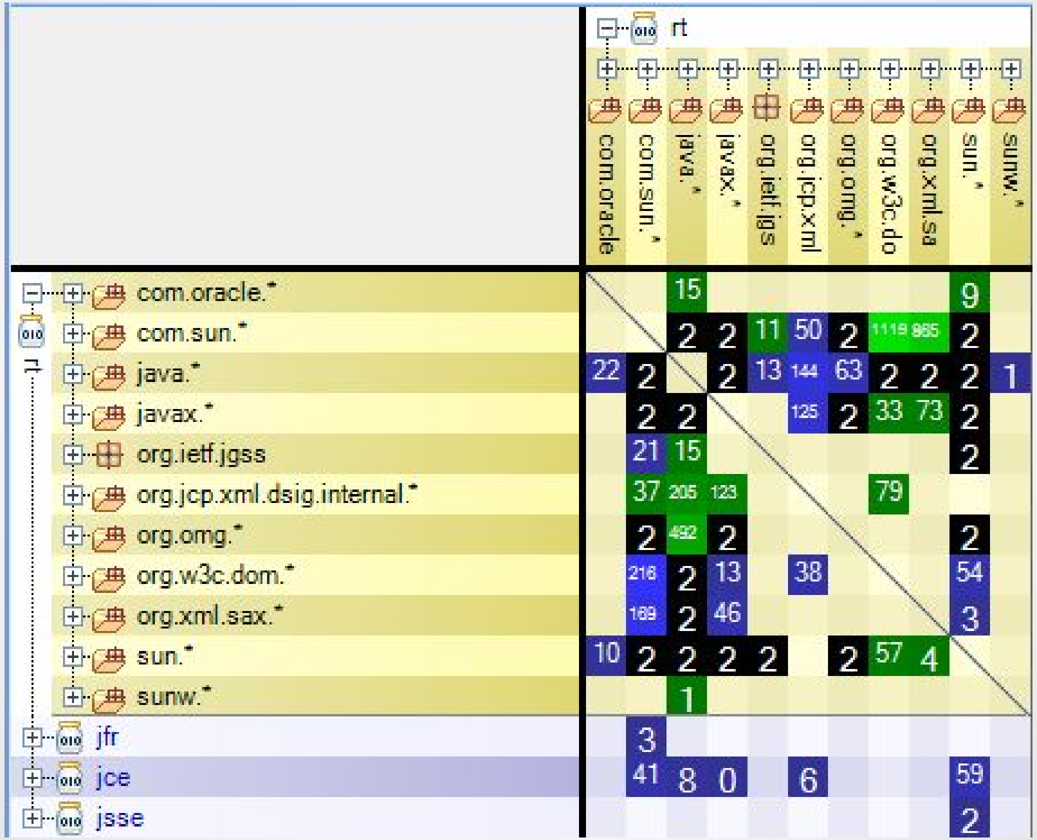
New URL schema

ClassLoader.getSystemResource(String name)

old url→ *jar:file:<path-to-jar>!<path-to-file-in-jar>*

new url → *jrt:/<module-name>/<path-to-file-in-module>*

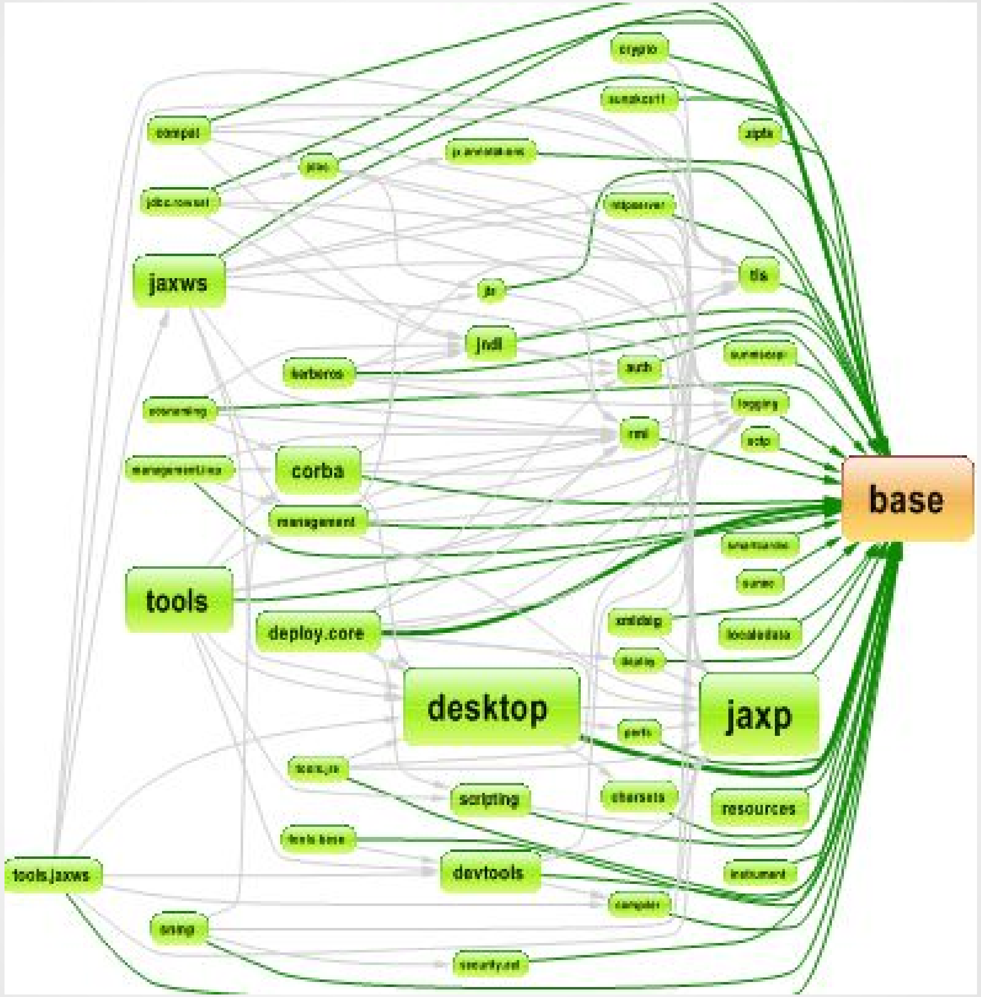
Problems?



rt

	com.oracle	com.sun.*	java.*	javax.*	org.ietf.jgs	org.jcp.xml	org.omg.*	org.w3c.do	org.xml.sa	sun.*	sunw.*
com.oracle.*			15							9	
com.sun.*			2	2	11	50	2	1119	985	2	
java.*	22	2		2	13	144	63	2	2	2	1
javax.*		2	2			125	2	33	73	2	
org.ietf.jgss		21	15							2	
org.jcp.xml.dsig.internal.*		37	205	123				79			
org.omg.*		2	492	2						2	
org.w3c.dom.*		216	2	13		38				54	
org.xml.sax.*		169	2	46						3	
sun.*	10	2	2	2	2		2	57	4		
sunw.*			1								
jfr		3									
jce		41	8	0		6				59	
jsse										2	

Modules


crypto
sunpkcs11
compat
zipfs
jdbc
jx.annotations
jdbc.rowset
httpserver
jaxws
tls
jfr
jndi
sunmscapi
kerberos
auth
cosnaming
logging
rmi
sctp
management.iiop
corba
base
management
smartcardio
sunec
tools
xmldsig
deploy.core
localedata
deploy
desktop
jaxp
prefs
tools.jre
scripting
charsets
resources
tools.base
devtools
instrument
tools.jaxws
compiler
snmp
security.acl

modules.xml

```
<modules>
  <module>
    <name>java.activation</name>
    <depend>java.base</depend>
    <depend re-exports="true">java.datatransfer</depend>
    <depend>java.desktop</depend>
    <depend>java.logging</depend>
    <export>
      <name>javax.activation</name>
    </export>
  </module>
...
```

jdeps (java 8) - part 1

```
$ jdeps -cp … \
pl/touk/mockserver/server/HttpMockServer.class
HttpMockServer.class -> .../groovy-all-2.4.3.jar
HttpMockServer.class -> java.base
HttpMockServer.class -> jdk.httpserver
HttpMockServer.class -> .../slf4j-api-1.7.9.jar
...
```

jdeps (java 8) - part 2

```
pl.touk.mockserver.server (HttpMockServer.class)
      -> com.sun.net.httpserver
      -> groovy.lang groovy-all-2.4.3.jar
      -> groovy.util.slurpersupport groovy-all-2.4.3.jar
      -> groovy.xml groovy-all-2.4.3.jar
      -> java.io
      -> java.lang
      -> java.lang.ref
      -> java.util
      -> java.util.concurrent
...
```

Example: Tree

```
src/dpr.api/com/dpr/api/IHello.java
src/dpr.api/module-info.java

src/dpr.impl/com/dpr/impl/Hello.java
src/dpr.impl/module-info.java

src/dpr.runner/com/dpr/runner/Main.java
src/dpr.runner/module-info.java
```

dpr.api

```
module dpr.api {
    requires java.base;
    exports com.dpr.api;
}
```

dpr.impl

```
module dpr.impl {
   requires dpr.api;
   provides com.dpr.api.IHello
      with com.dpr.impl.Hello;
}
```

dpr.runner

```
module dpr.runner {
    requires dpr.api;
    uses com.dpr.api.IHello;
}
```

Compilation

```
$ javac \
    -d modules
    -modulesourcepath src
    `find src -name '*.java'`
```

jmod

- like **jar** command
- creating modules
- printing modules descriptors
- listing modules content

jmod example

```
$ jmod -c --module-version=1.0.0 \
   --class-path modules/dpr.api/  \
   modules/dpr.api.jmod
```

jdeps (java 9) - part 1

```
$ ./bin/jdeps -genmoduleinfo . \
   ~/.
m2/repository/pl/touk/mockserver/mockserver/
2.1.1/mockserver-2.1.1.jar
```

jdeps (java 9) - part 2

```
$ cat  mockserver/module-info.java

module mockserver {

    requires java.xml;

    requires java.xml.bind;

    requires public jdk.httpserver;

    requires public NOT_FOUND;

    exports pl.touk.mockserver.server;

}
```

OSGI?

- execution environment
- module
- life-cycle
- service registry

com.dpr.api bundle

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: DPR API
Bundle-SymbolicName: com.dpr.api
Bundle-Version: 1.0.0
Exports: com.dpr.api;version=1.0.0
```

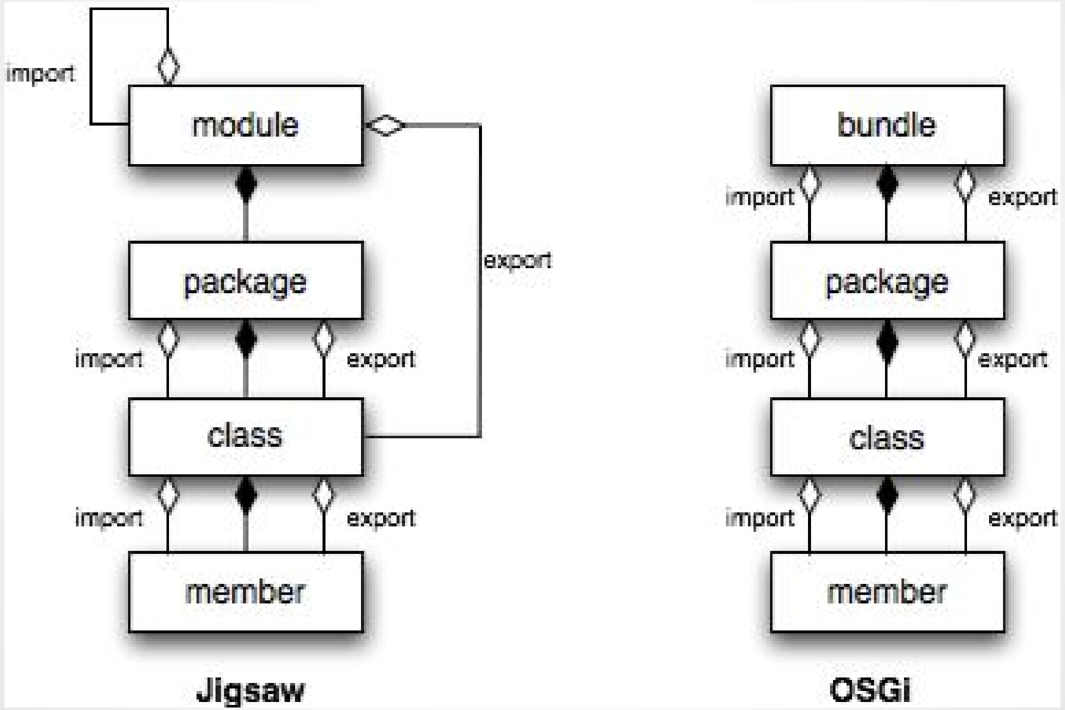
com.dpr.impl bundle

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: DPR IMPL
Bundle-SymbolicName: com.dpr.impl
Bundle-Version: 1.0.1
Imports: com.dpr.api;version=[1.0,2)
Bundle-RequiredExecutionEnvironment: JavaSE-
1.8
```

com.dpr.runner bundle

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: DPR RUNNER
Bundle-SymbolicName: com.dpr.runner
Bundle-Version: 1.0.0
Bundle-Activator: com.dpr.runner.Main
Imports: com.dpr.api;version=1.0.0
```

OSGI vs Jigsaw



We are still waiting...

“A shorter release cycle would be great.

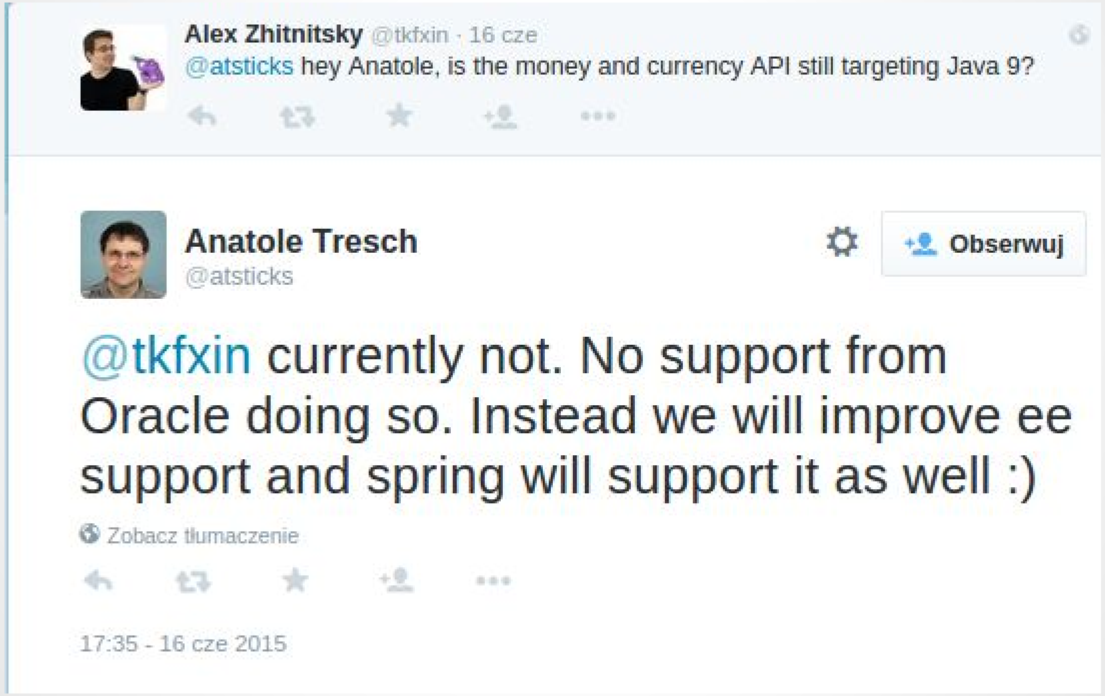
And for Jigsaw, just drop the idea, you are really loosing your time. OSGI is good enough, it's already working, and it's working well.”

http://mreinhold.org/blog/late-for-the-train

Money and Currency API

1.2626	EURO	1.0395
1.6888	USA	1.3947
12.7315	SOUTH AFRICA	9.9772
13.2175	HONG KONG	10.7869
162.29	JAPAN	130.37
1.8646	AUSTRALIA	1.5058
1.7278	CANADA	1.3924

Money and Currency API



Alex Zhitnitsky @tkfxin · 16 cze

@atsticks hey Anatole, is the money and currency API still targeting Java 9?

Anatole Tresch
@atsticks

Obserwuj

@tkfxin currently not. No support from Oracle doing so. Instead we will improve ee support and spring will support it as well :)

Zobacz tłumaczenie

17:35 - 16 cze 2015

Money and Currency API

Advice from Effective Java:

- do not use:
    - float
    - double
- use:
    - BigDecimal
    - Integer (int)
    - Long (long)

Money today

```
public class Money {
    private String currency;
    private BigDecimal amount;
    // ...
}
```

What for will it be?

- currency for each country
- exchange rate
- find rates
- standard ISO 4217

ISO 4217

```
<CcyNtry>
 <CtryNm>POLAND</CtryNm>
 <CcyNm>Zloty</CcyNm>
 <Ccy>PLN</Ccy>
 <CcyNbr>985</CcyNbr>
 <CcyMnrUnts>2</CcyMnrUnts>
</CcyNtry>
```

JSR 354 & Reference implementation

```
<dependency>
  <groupId>org.javamoney</groupId>
  <artifactId>moneta</artifactId>
  <version>1.0</version>
</dependency>
```

http://javamoney.github.io/ri.html

CurrencyUnit

```
CurrencyUnit pln =
   Monetary.getCurrency("PLN");

CurrencyUnit usd =
   Monetary.getCurrency(Locale.US);
```

Creating Money

```
MonetaryAmount normal = Money.of(new
   BigDecimal("1.5"), usd);
MonetaryAmount fast = FastMoney.of(new
   BigDecimal("1.5"), usd);

assertEquals("USD 1.5", normal.toString());
assertEquals("USD 1.50000",
   fast.toString());
```

Formatting Money

```
MonetaryAmountFormat format =
    MonetaryFormats.getAmountFormat(new
        Locale("pl_PL"));

assertEquals("USD 1.50",
    format.format(money));
```

Money operations

```
MonetaryAmount m1 = Money.of(1, usd);
MonetaryAmount m2 = Money.of(2, usd);
MonetaryAmount m3 = Money.of(3, usd);


assertEquals(m3, m1.add(m2));
assertEquals(m2, m1.multiply(2));
assertTrue(m1.isLessThan(m2));
```

Money functions

```
Arrays.asList(m1, m2, m3)
        .stream()
        .filter(MonetaryFunctions
            .isCurrency(usd))
        .reduce(MonetaryFunctions.sum()));
```

ExchangeRate

```
ExchangeRateProvider ecbRateProvider =
    MonetaryConversions.getExchangeRateProvider(
        "ECB");
ExchangeRate exchangeRate =
    ecbRateProvider.getExchangeRate(
        "PLN", "USD");
NumberValue factor = exchangeRate.getFactor();
// 0.27369381044951642996
```

CurrencyConversion

```
CurrencyConversion currencyConversion =
   ecbRateProvider.getCurrencyConversion(usd);
Money m = Money.of(5, pln)
   .with(currencyConversion);

// Money.of(new
      BigDecimal("1.3684690522475821498"), usd)
```

Rounding

```
MonetaryRounding rounding = Monetary
        .getRounding(RoundingQueryBuilder.of()
               .setCurrency(usd).build());

assertEquals(Money.of(
  new BigDecimal("1.37"), usd),
  rounding.apply(Money.of(
    new BigDecimal("1.3684690522475821498"),
  usd)));
```

JSON API

JSON API

- Parsing and generation of JSON without external libraries like Jackson.
- Navigation over JSON tree like in DOM or SAX for XML.
- Builder API for JSON.
- JSON tree transformation.

Road to JSON support

- JSR 353: Java API for JSON Processing - Java EE 7
- Nashorn in Java 8 - `JSON.parse()` and `JSON.stringify()`
- JEP 198: Light-Weight JSON API - Java 9

Nashorn

```
ScriptEngineManager m =
    new ScriptEngineManager();
ScriptEngine e =
    m.getEngineByName("nashorn");
try {
     e.eval("print('Hello World!');");
} catch(ScriptException se) {
     // ...
}
```

Nashorn

```
jjs> var me = JSON.parse('{"name":"John",
   "details":{"dob":"1976-05-21", "gender":"
   Male"}}');
jjs> print(me.name);
John
jjs> print(JSON.stringify(me));
   {"name":"John",
      "details":{"dob":"1976-05-21",
         "gender":"Male"}}
```

Nashorn

```
ScriptObjectMirror o =
    (ScriptObjectMirror) e.eval(
      "JSON.parse('{\"name\":\"John\"}');"
    );

assertEquals("John", o.getMember("name"));
```

Producing JSON

```
@GET
@Path("/{name}")
@Produces(MediaType.APPLICATION_JSON)
public Project get(@PathParam("name") String
name) {
    return projects.get(name);
}
```

Consuming JSON

```
@POST
@Consumes(MediaType.APPLICATION_JSON)
public void create(Project p) {
    projects.put(p.getName(), p);
}
```

HTTP2 client

Old HTTP client

- URLConnection class does not implement the AutoCloseable interface
- Blocking API
- Only support HTTP 1.0

HTTP2 client

JEP 110 will define and implement a new HTTP client for Java that will replace HttpURLConnection and also implement HTTP 2.0 and websockets.

GET

```
HttpResponse response = HttpRequest
        .create(new URI("http://www.foo.com"))
        .headers("Foo", "foovalue", "Bar", "barvalue")
        .GET()
        .response();
int statusCode = response.statusCode();
String responseBody = response.body(asString());
```

POST

```
HttpResponse response = HttpRequest
          .create(new URI("http://www.foo.com"))
          .body(fromString("param1=foo,param2=bar"))
          .POST()
          .response();
```

Collections factories

JEP 269: Convenience Factory Methods for Collections

*“Provide **static factory methods on the collection interfaces** that will **create compact**, **unmodifiable collection instances**. Provide static factory methods for creating the **most common concrete collections** classes.”*

http://openjdk.java.net/jeps/269

Factories for interfaces

```
List.of(1, 2, 3);

Set.of(“a”, “b”);

Map.of(5, “test”, 6, “test2”)

Map.fromEntries(

    entry(5, “test”),

    entry(6, “test2”));
```

Factories for concrete classes

```
ArrayList.of(1, 2, 3);

HashSet.of(“a”, “b”);

HashMap.of(5, “test”, 6, “test2”)

HashMap.fromEntries(

    entry(5, “test”),

    entry(6, “test2”));
```

Smart Java compilation

Smart Java compilation - phase 1

JEP 139: Enhance javac to Improve Build Speed in Java 8

by:

- javac use all cores
- reuse javac in a server process
- javac build incrementally
- internal wrapper of javac - sjavac

Smart Java compilation - phase 2

JEP 199: Smart Java Compilation, Phase Two in Java 9:

- expose sjavac in the JDK tool chain
- fix stability
- fix portability
- improve the quality of sjavac

Improved contended locking

Improved contended locking

JEP 143: Improve Contended Locking

Improve the overall performance of contended Java object monitors as measured by many tests.

For example volano29:

- a chat server with huge thread counts and client connections
- trying to access the same resources
- simulate a heavy duty real world application

What will be improved

- Field reordering and cache line alignment
- Speed up PlatformEvent::unpark()
- Fast Java monitor enter operations
- Fast Java monitor exit operations
- Fast Java monitor notify/notifyAll operations

Segmented code cache

Segmented Code Cache

JEP 197: Segmented Code Cache

- divide the code cache into distinct segments
- each segment contains compiled code of a particular type
- to improve performance and enable future extensions.

Example: the method sweeper will work faster because it could skip non-method code

Segmented Code Cache

3 segments:

- code that will stay in the cache forever (JVM internal / non-method code)
- short lifetime (Profiled code, specific to a certain set of conditions)
- potentially long lifetime (Non-profiled code)

Process API Improvements

Process API Improvements

JEP 102: Process API Updates

- get PID of current JVM process or processes created by it
- get and set name of JVM process or procesess created by it
- enumerate JVMs processes and processes on system
- working with process tree, e. g. destroy process tree
- working with hundreds of subprocess, e. g. one output stream and error stream for many processes, not one thread per subprocess

REPL

Java REPL - project KULLA

- JEP 222: jshell: The Java Shell (Read-Eval-Print Loop)
- evaluate declarations, statements, and expressions
- `jshell` command line tool

Unified JVM Logging

Unified JVM Logging

- JEP 158: Unified JVM Logging
- only infrastructure for logging
- multiple tags per log e. g. *gc, compiler, etc.*
- standard error levels
- redirecting log to console or file
- dynamic configuration via MBeans or jcmd
- decorators on demand - uptime, tags, level
- multiline print

Compiler Control

Compiler Control

- JEP 165: Compiler Control
- fine-grained and method-context dependent control of the JVM compilers
- change the JVM compiler control options in run time
- no performance degradation
- directives for compilation in file

New Version-String Scheme

JEP 223: New Version-String Scheme

- easier to parse and read versions
- compatible with Semantic Versioning (MAJOR.MINOR.PATCH)
- each version part encode only one information
- simple API for version-string parsing, validation, and comparison

Old versioning

Release type	Existing long	Existing short
Early Access	1.9.0-ea-b19	9-ea
Major	1.9.0-b100	9
Security #1	1.9.0_5-b20	9u5
Security #2	1.9.0_11-b12	9u11
Minor #1	1.9.0_20-b62	9u20
Security #3	1.9.0_25-b15	9u25
Minor #2	1.9.0_40-b45	9u40

Old versioning

Release type	Existing long	Existing short
Early Access	9.0.0-ea+19	9-ea
Major	9.0.0+100	9
Security #1	9.0.1+20	9.0.1
Security #2	9.0.2+12	9.0.2
Minor #1	9.1.2+62	9.1.2
Security #3	9.1.3+15	9.1.3
Minor #2	9.2.4+45	9.2.4

Version API

```
public class Version implements
Comparable<Version>, Serializable {
    public static Version parse(String);
    public static Version current();
    public int major();
    public int minor();
    public int security();
   // ...
```

Version API

```
   // ...
    public List<Integer> version();
    public Optional<String> pre();
    public Optional<Integer> build();
    public Optional<String> optional();
    public int compareTo(Version o);
}
```

Remove GC Combinations Deprecated in JDK 8

JEP 214: Remove GC Combinations Deprecated in JDK 8

- flags were deprecated with JEP 173: Retire Some Rarely-Used GC Combinations
- these options add very little value to the users
- these options add extra complexity to the GC code base
- in Java 8 there was a warning
- in Java 9 JVM will not start with such combinations

JEP 214: Remove GC Combinations Deprecated in JDK 8

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

JEP 214: Remove GC Combinations Deprecated in JDK 8

DefNew + iCMS	: -XX:+CMSIncrementalMode
	-XX:+UseConcMarkSweepGC
	-XX:-UseParNewGC
CMS foreground	: -XX:+UseCMSCompactAtFullCollection
CMS foreground	: -XX:+CMSFullGCsBeforeCompaction
CMS foreground	: -XX:+UseCMSCollectionPassing

Default GC

G1 will be default GC

- JEP 248: Make G1 the Default Garbage Collector
- current default GC is Parallel GC
- low-pause
- limiting GC pause

Milling Project Coin

JEP 213: Milling Project Coin

- JSR 334: Small Enhancements to the Java Programming Language in Java 7
- underscore will be not allowed as identifier
- interfaces with private methods
- allow @SafeVargs on private instance methods - nowadays it is allowed only on static and final methods
- final variable could be used in try-with-resources

Try-with-resources in Java 7/8

```
final BufferedReader reader =
    new BufferedReader(
        new InputStreamReader(System.in));
try (BufferedReader r = reader) {
    // ...
} catch (IOException ex) {
    // ...
}
```

Try-with-resources in Java 9

```
final BufferedReader reader =
    new BufferedReader(
        new InputStreamReader(System.in));
try (reader) {
    // ...
} catch (IOException ex) {
    // ...
}
```

Now and future

What we could try now?

- early access Oracle Java 9
- docker compiling the latest OpenJDK 9
- Money API and RI on github
- working jigsaw examples

Java 10

- in 2018
- change typing system
- Project Valhalla
  - value types
  - generics in generated classes names:
    - ArrayList${T=String}.class
  - List<int>

Summary

Adoption of Java 9 will slow to a crawl due to this, which would effect the entire Java ecosystem. The situation will be analogous to Python 2 and 3.

https://jaxenter.com/java-9-without-sun-misc-unsafe-119026.html

Q & A

References

- http://blog.codefx.org/java/dev/how-java-9-and-project-jigsaw-may-break-your-code/
- http://blog.takipi.com/java-9-the-ultimate-feature-list/
- http://comments.gmane.org/gmane.comp.java.openjdk.macosx-port.devel/6900
- http://docs.oracle.com/javase/8/embedded/develop-apps-platforms/jrecreate.htm#JEMAG270
- http://javadepend.files.wordpress.com/2012/10/jisgaw1.png?w=595
- http://javamoney.github.io/ri.html
- http://jaxenter.com/behind-scenes-java-9-new-features-come-113124.html
- http://jaxenter.com/java-9-release-date-announced-116945.html
- http://jaxenter.com/json-api-dropped-java-9-113028.html
- http://jaxenter.com/new-java-9-features-announced-112654.html
- http://mreinhold.org/blog/late-for-the-train
- http://openjdk.java.net/jeps/102
- http://openjdk.java.net/jeps/110
- http://openjdk.java.net/jeps/143
- http://openjdk.java.net/jeps/158
- http://openjdk.java.net/jeps/165
- http://openjdk.java.net/jeps/197

References

- http://openjdk.java.net/jeps/199
- http://openjdk.java.net/jeps/214
- http://openjdk.java.net/jeps/220
- http://openjdk.java.net/jeps/222
- http://openjdk.java.net/jeps/223
- http://openjdk.java.net/jeps/231
- http://openjdk.java.net/jeps/260
- http://openjdk.java.net/jeps/269
- http://openjdk.java.net/projects/jdk9/
- http://openjdk.java.net/projects/jigsaw/
- http://openjdk.java.net/projects/jigsaw/doc/lang-vm.html#jigsaw-1
- http://openjdk.java.net/projects/jigsaw/doc/ModulesAndJavac.pdf
- http://openjdk.java.net/projects/jigsaw/doc/quickstart.html
- http://radar.oreilly.com/2014/09/what-every-java-developer-needs-to-know-about-java-9.html
- https://blogs.oracle.com/jtc/entry/a_first_look_at_compact
- https://docs.oracle.com/javase/8/docs/technotes/guides/compactprofiles/compactprofiles.html
- https://github.com/JavaMoney/javamoney-examples

References

- https://github.com/neomatrix369/BuildHelpers
- https://jdk9.java.net/jigsaw/
- https://github.com/AdoptOpenJDK/jdk9-jigsaw
- https://msdn.microsoft.com/en-us/library/ms179882.aspx
- http://stackoverflow.com/questions/26424759/what-is-sjavac-who-is-it-for-and-how-do-i-use-it
- http://stackoverflow.com/questions/29366265/how-to-use-jigsaw-with-java-9
- https://weblogs.java.net/blog/otaviojava/archive/2014/08/25/java-9-coming-money-api
- https://www.eclipsecon.org/na2015/sites/default/files/slides/reinhold-eclipsecon-2015.pdf
- https://www.voxxed.com/blog/2015/01/new-try-resources-improvement-jdk-9/
- https://www.voxxed.com/blog/presentation/presentation-java-9-make-way-for-modules/
- http://www.baeldung.com/java-9
- http://www.codergears.com/Blog/?p=310
- http://www.codergears.com/Blog/wp-content/uploads/jigsaw1.png
- http://www.dobreprogramy.pl/Java-9-bedzie-lekka-oszczedna-i-modulowa-jak-LEGO,News,62108.html
- http://www.dzone.com/links/r/java_9_could_mess_with_your_code_new_early_access.html
- http://www.dzone.com/links/r/why_not_build_openjdk_9_using_docker_.html

References

- http://www.jarchitect.com/img/DocMatrix/img3.png
- http://www.javaworld.com/article/2878952/java-platform/modularity-in-java-9.html
- http://www.postgresql.org/docs/9.1/static/datatype-money.html
- http://www.slideshare.net/mfrancis/java-8-modules-jigsaw-and-osgi-neil-bartlett
- http://openjdk.java.net/projects/jigsaw/spec/reqs/02
- http://openjdk.java.net/projects/jigsaw/j1/
- https://github.com/sandermak/jigsaw-firstlook/blob/master/src/module1/com/test/TestClassModule1.java
- http://www.slideshare.net/delabassee/delabassee-http2
- https://neomatrix369.wordpress.com/2015/06/04/why-not-build-openjdk-9-using-docker/
- http://www.slideshare.net/SimoneBordet/http2-and-java-current-status
- http://cr.openjdk.java.net/~michaelm/8087112/
- http://hg.openjdk.java.net/jdk9
- https://vimeo.com/138736736
- http://www.infoq.com/articles/Java9-New-HTTP-2-and-REPL
- http://mail.openjdk.java.net/pipermail/core-libs-dev/2015-October/035743.html
- http://njbartlett.name/2015/11/13/osgi-jigsaw.html

Images

- http://www.clipartlord.com/wp-content/uploads/2014/11/puzzle4.png
- http://i537.photobucket.com/albums/ff332/Kento231/jigsaw.jpg
- http://www.codergears.com/Blog/wp-content/uploads/jigsaw1.png
- http://javadepend.files.wordpress.com/2012/10/jisgaw1.png?w=595
- http://www.osgi.org/wiki/uploads/Main/logo1.jpg
- http://i.telegraph.co.uk/multimedia/archive/02576/currency_2576865b.jpg
- http://media.techtarget.com/TheServerSideCOM/images/ModularDependants5.png

Thank you