



News in JDK8

Developer Conference Brno Feb. 2012

Jiří Vaněk



News in JDK8

Developer Conference Brno Feb. 2012

Jiří Vaněk

- **FOSDEM 2012:**
 - M. Reinhold: “There is nothing sure right now”
- Q: JDK 7 ?
- Q: JDK < 6 ?

Index

- 1) History of java
 - + list of JDK7 changes
 - 2) Summary
 - 3) OpenJDK8
 - a) Project Jigsaw
 - b) Project Lambda
 - c) Project Coin
 - d) JavaFX
 - e) HotSpot convergence
 - f) Project Avatar
 - 4) OpenJDK9 and conclusion
-

Brief history of java ...or not all was there from beginning

[wikipedia](#)

JDK 1.0 - 1996

- Codename Oak. **Initial** release after **aprox. 4 years of development**, The first stable version was JDK 1.0.2. is called Java 1
- Netbeans started as project Xelfi in CZ
- Each of the releases promised to have some **mayor improvements** and some of them as **leading changes** and some of them will be of **non-technical** character



Brief history of java 3/9

JDK 1.2 - 1998

- strictfp keyword
- Swing integrated
- **JIT compiler for the first time**
- **Java Plug-in**
- Java IDL, an IDL implementation for CORBA interoperability
- **Collections framework**
- Differences of J2SE x J2ME x J2EE



Brief history of java 4/9

JDK 1.3 - 2000

- **HotSpot** JVM included (the HotSpot JVM was first released in April, 1999 for the J2SE 1.2 JVM) => performance!
- RMI was modified to support optional compatibility with CORBA
- **JavaSound**
- Java Naming and Directory Interface (**JNDI**) included in core libraries (previously available as an extension)
- Java Platform Debugger Architecture (**JPDA**)
- Synthetic proxy classes
- NetBeans as we know them now

Brief history of java 5/9

JDK 1.4 - 2002

- **assert** keyword
- **regular expressions** modeled after Perl regular expressions
- **exception chaining** allows an exception to encapsulate original lower-level exception
- Internet Protocol version 6 (**IPv6**) support
- non-blocking IO (named **NIO**) (New Input/Output)
- **logging** API
- **image I/O** API for reading and writing images in formats like JPEG and PNG
- integrated XML parser and XSLT processor (**JAXP**)
- integrated **security and cryptography extensions** (JCE, JSSE, JAAS)
- **Java Web Start** included
- Preferences API (java.util.prefs)
- **Conquering EE** world and rise of **GNU classpath** (GJ, GCJ, rt.jar)

Brief history of java 6/9

JDK 5.0 - 2004

- Changed versioning
- **Generics**
- Metadata (**annotations**)
- **Autoboxing**/unboxing:
- **Enumerations**
- **Varargs**: (public void (String...s){})
- Enhanced **for each** loop (for (Widget w: widgets){})
- Static imports
- Swing: New skinnable **look and feel, called synth.**
- The concurrency utilities and Scanner class
- Java 5 is the last release of Java to officially support the Microsoft Windows 9x ;)
- **Eclipse opensourced**

Brief history of java 7/9

JDK 6 - 2006

- Another versioning change, and [release of OpenJDK and in 2007 rise of project IcedTea](#)
- [Contributions?](#)
- **Scripting** Language Support (eg. Rhino for javascript)
- Dramatic **performance** improvements for the core platform and Swing.
- JAX-WS and JDBC 4.0
- Java Compiler API - an API allowing a Java program to select and invoke a Java Compiler programmatically.
- Upgrade of JAXB 2.0 and StAX parser.
- Support for pluggable annotations
- Many GUI improvements, such as integration of **SwingWorker** in the API, **table sorting and filtering**, and true Swing **double-buffering**
- JVM improvements include: synchronization and compiler performance optimizations, new algorithms and upgrades to existing garbage collection algorithms, and application start-up performance
- **Acquisition** 2009/2010 maintainer of JDK changed from **Sun** to **Oracle**



Brief history of java 8/9

JDK 7 - 2011

- First oracle release, although presented proudly, only minor updates at all, and first version **was buggy**.
- JVM support for **dynamic languages (invoke dynamic)**, following the prototyping work currently done on the Multi Language Virtual Machine
 - JRuby/Scala/... call directly to JVM (and no transformation to java at first)
 - Via custom code which JVM inline through
- Compressed 64-bit pointers
- Small language changes (grouped under a project named **Coin**):
 - **Strings in switch**
 - Automatic resource management in try-statement (eg **AutoCloseable**)
 - Improved type inference for generic instance creation (eg <>)
 - Simplified varargs method declaration
 - Binary integer literals and spaces/underscores in numeric literals
 - Catching **multiple exception** types and rethrowing exceptions with improved type checking

Brief history of java 9/9

JDK 7 - continue

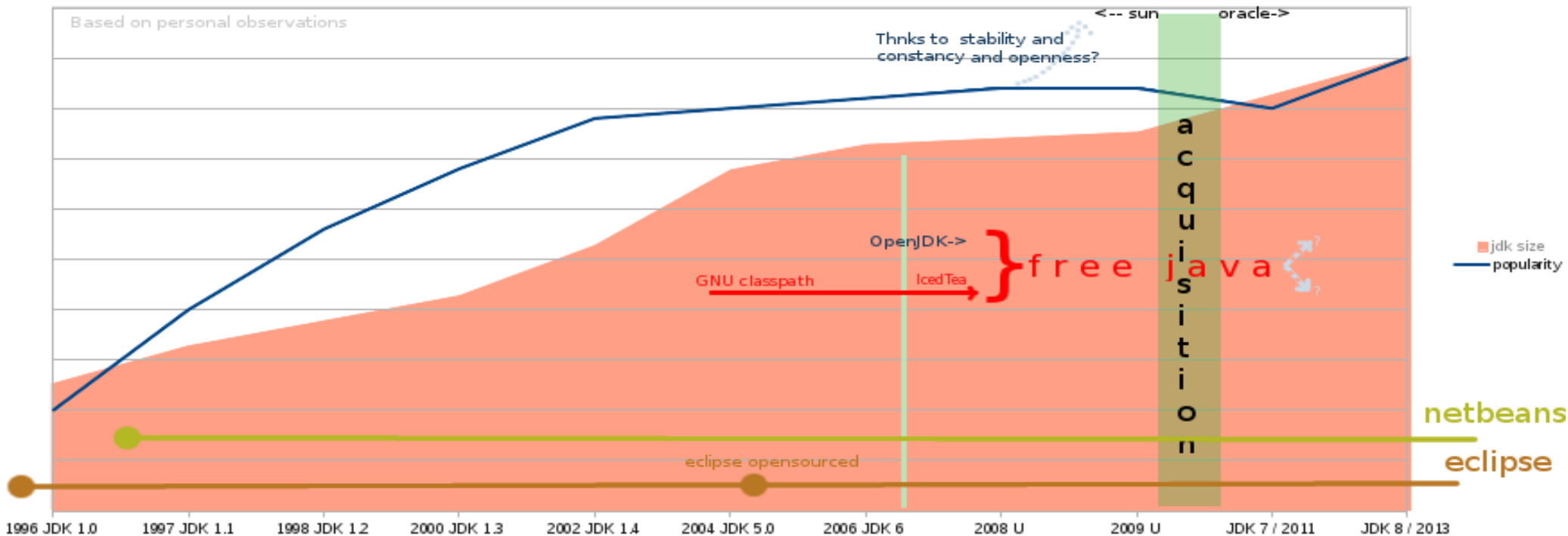
- Concurrency utilities (fork/join framework)
- New file I/O library to enhance platform independence and add support for metadata and symbolic links. The new packages are **java.nio.file** and `java.nio.file.attribute`
- Library-level support for Elliptic curve cryptography algorithms
- An **XRender pipeline** for Java 2D, which improves handling of features specific to modern GPUs
- New platform APIs for the graphics features originally planned for release in Java version 6u10
- Enhanced library-level support for **new network protocols**, including SCTP and Sockets Direct Protocol
- Upstream updates to XML and **Unicode**
- **OpenJDK** is build-able without additional projects
- Care is taken of **community**

JDK 8 - 2013

- Finish JDK7
- and much more!
- already now under more work then JDK7
- Even better care **of community**

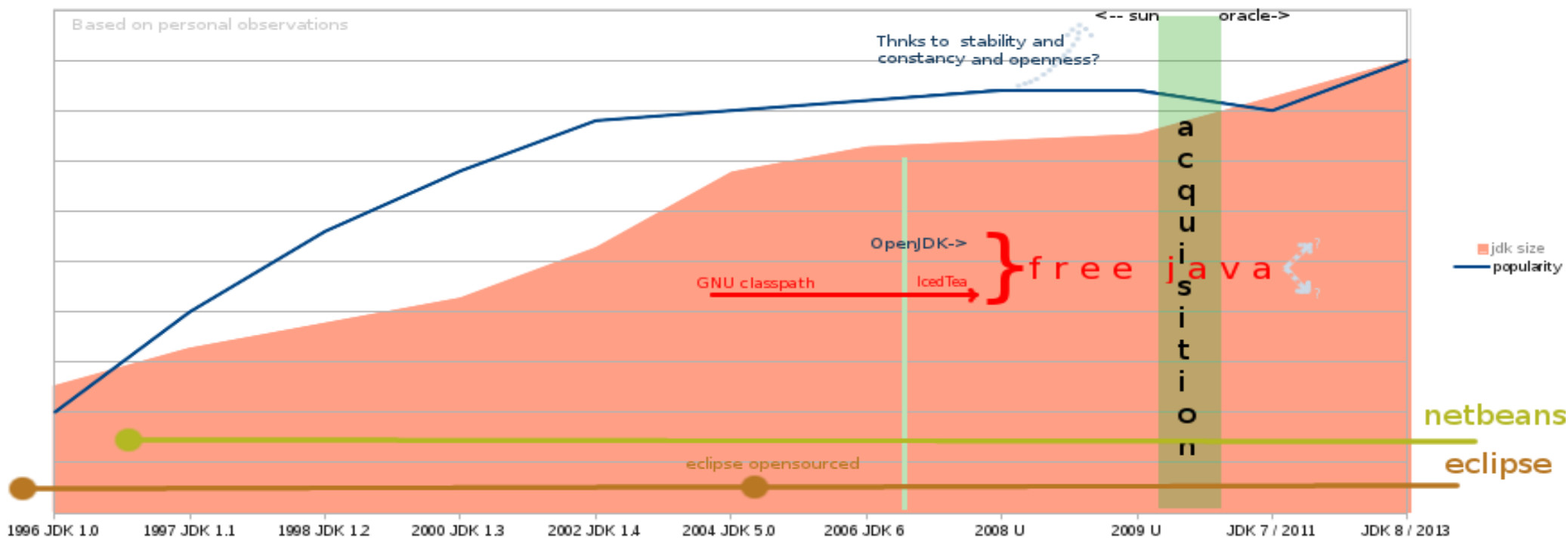


Summary 1/2



- Long way to OpenJDK 8! (**13+4 years**, and IBM is still supporting 1.4! (And oracle via very expensive support too))
- New release aprox. Every 2 years
 - Longest not-replaced release java 6 - 5 years
 - Caused by changing of maintainer from Sun to Oracle in 2009/2010?
- Each release have at least one mayor improvement
- Hugest release was (by surprise!) JDK 5.0
- **Oracle** looks to keep **2 years** period, is preparing **valuable changes**... This going **better** after long break!

Summary 2/2



- Small decrease of popularity when JDK7 was released was caused by unwillingness of developers to try changes (and by some mayor bugs O:)
- **Most (young) developers imagine under “java” release “JDK 6”, can not even imagine evolution!**
- Oracle is **maintaining community** (LCJ – community process to maintain community), is preparing valuable changes... This going **better** after long break!

OpenJDK7 -> OpenJDK8

or what did not affect (or Plan B)

[Future Of Java](#) and [Mark Reinholds's keynote \(2011\)](#)

Project Coin
Invoke Dynamic
Fork/Join Framework
(Penrose)

----->

Project Coin (finish)

----->

----->

+

+

?+?

?+?

Project Jigsaw

Project Lambda

Merge of HotSpot and JRocket

Java FX 3.0

Project Avatar(full HTML 5 support) ->9?

Graal

Strict Verification
Parallel Class Loaders
Phasers
Transfer Queues
More New I/O
Unicode 6.0
Enhanced Locales
SDP & SCTP
TLS 1.2
ECC
JDBX 4.1
Xrender Pipeline
Swing JLayer
Swing Nimbus
(EE) cache api (finally!)

----->

----->

----->

+

+

+

+

(EE?)Datagrid api (jsr 347)

Type Annotations

Bulk Data Operations

New Date/Time api

Multi-touch devices

Rewritten javascript engine (project

Nashorn)

+ Easier Java/Native integration ->9

OpenJDK 8

2013 (?)

- JDK7 was **very conservative** release
- JDK8 should be **big** update
- But developers see JDK7 as big set of changes. So?
- just finish 7? Hopes that not...
 - eg modularisation via project Jigsaw will be **the change**.
- Or will it achieve revolution instead of evolution?
- **FOSDEM** – no news:(
 - M. Reinhold: “There is nothing sure right now”
- JDK8 compared to JDK7 (compared to JDK6) have great improvements in **infrastructure** and **community** maintaining
 - LCJ – community process to maintain community
 - JEP – features proposal process
 - Community more involved into JSR process (eg.: approving or “adopt your JSR”)

OpenJDK8 - Project Coin 1/2

[J.Darcy's blog](#) and [draft](#)



- Gathering name for couple of small language changes
- Most already done in JDK7:
 - **Strings in switch**
 - Fastened by **hashcode** on byte code level (hashcodes compared first)
 - `String s; switch(s){ case "april":...}`
 - **Multi-catch** and more precise rethrow
 - `catch (MyException || TheirsException){ }`
 - Improved type inference for generic instance creation (**diamond**)
 - `List<String> a=new ArrayList<>`
 - try-with-resources statement (**AutoCloseable**)
 - `try (AutoCloseable q= new MyAutoCloseable())
{q.do()};`
//and ... q is closed, and exceptions handled
 - Simplified varargs method invocation
 - `@SafeVarargs`
 - **Binary** integral literals and **underscores** in numeric **literals**
 - Eg `int a=0b10010; int b= 1 000_000;`

OpenJDK8 - Project Coin 2/2

- To be done in JDK8:
 - Language support for **collections** (access **through []**)
 - Eg `List<String> a;...; String q=a[5];`
 - **Elvis** and other **Nullsafe operators** (in case that subject is null then nothing upon him will be invoked, and no exception thrown)
 - JDK9?
 - To be done at all?
 - Currently implemented as variations of “?”
 - `a?.foo();`
 - Or `a ? a : b`
 - Or `a ?: b`
 - Large **arrays** - declaration via **long** (currently maximally via int)
 - `Long l=15; String[] s=new String[l]`

OpenJDK8 - Project Jigsaw 1/4

[draft](#)

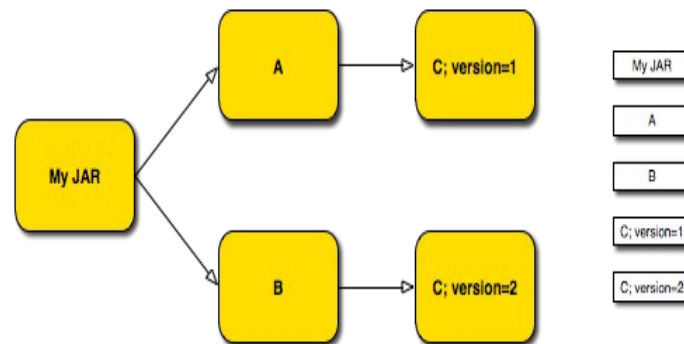


- Leading change in JDK8 - **modularisation of java platform**
- Continuous integration into JDK7 via project [penrose](#) (approved Jan/Feb 2012)
- Probably hugest change since JDK1
- Current JDK is **monolithic** and **huge** (more then 100M)
- Modules **will replace class path** (unix and maven like approach)
 - Eg. by Maven - Build-time, install-time, test-time and run-time
 - Eg from packages - shared versions and modules
 - Inspired and compatible with OSGI
- Modularization of native-binary parts of JDK will probably comes up to JDK9 :(

OpenJDK8 - Project Jigsaw 2/4

[solution](#)

- What it should solve:
 - JAR hell
 - Too many **transitive** references
 - Dependence on **multiple versions**



- Unmanaged Dependencies (only via **classloaders** hierarchy) => ServiceLoader API
- Use of **private code** - no longer possible?
- Stomping - **name clash** in jars

OpenJDK8 - Project Jigsaw 3/4

- Platform fragmentation
 - Will allow unification of SE x ME and EE
 - **No more rt.jar** (separate jars for separate technologies - swing, xml, language...)
 - **Startup** performance
 - (**pre**)loading only what needed (**pre-downloading**?)
 - Already JDK6 have lazy loading of parts of RT (but still whole RT must be available)
 - Integration with **native packaging** systems
 - Rpm/deb... inspiration <-> compatibility
 - Support for better cooperation with native modules probably moved to JDK 9
 - Package granularity
 - Libraries consisting from more and more jars?
 - Can lead to “new” “**modules hell**” ?
(lot of work done to not so)
 - What is module?

OpenJDK8 - Project Jigsaw 4/4

[jigsaw big picture](#) and [lang.support](#)

- Descriptors are **plain-text .java** files “inside” module/jar
- Module declaration:

```
module a.b @ 1.0 {  
  requires      c.d @ /* Use v2 or above */ >= 2.0 ;  
  requires service e.f;
```

```
  provides      g.h @ 4.0;  
  provides service i.j with k.l;  
  exports m.n;  
  permits o.p;  
  class cc.dd;
```

```
view a.b.c {  
  provides      q.r @ 1.0;  
  provides service s.t with u.v;  
  exports w.x;  
  permits y.z;  
  class aa.bb;  
}
```

```
---->jar  
      (classical,  
      classpath re-usable jar)  
---->jmod  
---->rpm  
---->deb  
---->war,ear (JDK 9?)
```

Maven --->
(pom compatibility)

OpenJDK8 - Project Lambda 1/2

[introduction](#), [draft](#)

- Known also as **Closures**, close connection to JDK7's invoke dynamic
- Anonymous functions:
 - Eg: `#()(42) is int getMeaningOfLive(){return 42}`
 - Eg: `#(int x,int y)(x*y) is int multiply(){return x*y}`
- **-> lambda operator**
 - `Squarer<Integer, Integer> s = (x) -> x * x; // 1-ary Lambda expression`
`System.out.println(s.square(5));`
 - Anonymous call: `(x,y)->x*y`
- **default keyword**
 - Defenders methods - Default implementation of method in interface
 - “breakng” interface pureness, but still forbid fields
 - When implementing more interfaces, must be declared directly
- What is it for? – **parallelism** and **bulk data operations**



OpenJDK8 - Project Lambda 2/2

1)

```
Class Student {
  String name
  Int gradYear;
  Double score
};
```

2) Naive implementation of searching

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

double max= Double.MIN_VALUE;

For (Student s: students){
  If (s.gradYear==2011)
    max=Math.max(max,s.score)
}Return max;
```

3) Task:

```
Double max
= students.filter
  s.gradYear == 2011
  .map
  s.score
  .reduce
  Mth.max
```

4)JDK6 - set of interfaces:

```
double max= students.filter(new
Predicate<Student>() {public boolean eval(Student s)
{
return s.gradYear==2011)
}).map(new Mapper<Student,Double>() {public
Double map(Student s){
return s.score)
})..reduce(new Reduce<Double,Double>() {public
Double reduce(Double max,Double score){
Return Math.max(max,score)
}
});
```

5)JDK7 - Job for fork/join framework

6) JDK8 - lambda

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

And parallelism will come from JDK.

7)result

```
double max= students
.filter(s -> s.gradYear==2011)
.map(s -> s.score)
.reduce(0.0,Math#max) //wrap
```

Parallelism possible, but...

Without parallelisms lambda implementation is currently slow!

OpenJDK8 - Hotspot convergence

jrocket mission control



HotSpot merge with JRocket

[announcement](#)

- Merging the best from two leading java virtual machines
- **HotSpot** (Oracle) – faster JIT and much better optimization
- **JRocket** (created by BEA, currently IBM) – more scalable and observable
- Will reach OpenJDK?
- JDK 9?

OpenJDK8 - Java FX (3.0?) [announcement](#)

- Already well know as Dead technology => What benefits can be bring in JDK8
- **Open-sourced and version 3**
- FX 3.0 - extended FXML (Reaction to flex?)
- Inclusion into Java Embedded
- Standardized as JSR
- Full support for **Linux** and Mac OS X

(Open?)JDK8 - Graal [announcement](#)

- Close connection with Invoke dynamic
- Portable, extendable multi-VM JIT written in java
- Personally, I do not believe in Graal.
- JDK9?



(Open?)JDK8 – Project Avatar

[announcement](#)

- Full HTML 5 support
- Second step to ME, SE and EE Unification
 - Serialization over JSON
 - Bi-directional “eternal” web-sockets
- To much shadows and questions about it.
- EE?
- Probably bigger change then jigsaw => I believe to come in JDK9, but some teasers can be available during JDK8, as JDK9 can be to late for first real introduction of HTML5 to JAVA

OpenJDK8 (EE?) – Datagrid API

[JSR-347](#) and [JSR-107](#)

- JVM-embedded non-relation-database
 - Same or different one virtual machine
- Current implementation is eg: jboss-comunity [infinispan](#)
- Based on oldest unimplemented JSR-107 – Caching API
 - Caching API comes with JDK7 EE

OpenJDK9 – 2015/16?

- **Self tuning JVM**
- **Improved Native Integration**
- Big Data
- Reification
- Tail Calls/Continuation
- Meta Object Protocol
- Multi Tenancy
- Resource Management
- Heterogenous Compute model
- **Finish avatar, Jigsaw....**
- Do you want something? Introduce it via **JEP** – enhancement proposal

- *Sci-fi?*

Conclusion

- There are **small** changes – rest of project Coin
- Strange changes – JavaFX
- Very **interesting** changes – Project Lambda, JVMs merging and Graal
- And **Huge** change under project Jigsaw
- And more huge, but still unclear changes in Project Avatar
- It will change the perception of java...
 - ... Or update like every else before?*
 - ... Or will become Revolution instead of Evolution?*

Questions?

- http://en.wikipedia.org/wiki/Java_version_history
- http://gotocon.com/dl/goto-prague-2011/slides/TerrenceBarr_FutureOfJava.pdf
- <http://www.oracle.com/javaone/live/index.html>
- http://blogs.oracle.com/darcy/entry/project_coin_final_five
- <http://openjdk.java.net/projects/lambda/>
- <http://openjdk.java.net/projects/jigsaw/doc/draft-java-module-system-requirements-12>
- <http://cr.openjdk.java.net/~mr/jigsaw/notes/jigsaw-big-picture-01>
- <http://openjdk.java.net/projects/jigsaw/doc/lang-vm.html>
- <http://cr.openjdk.java.net/~mr/lambda/straw-man/>
- <http://openjdk.java.net/projects/lambda/>
- <http://www.oracle.com/us/corporate/press/512956>
- <http://www.theserverside.com/feature/Project-Avatar-One-HTML5-Strategy-to-Rule-Them-All>
- <http://mail.openjdk.java.net/pipermail/discuss/2012-January/002320.html>
- <http://www.wiki.jvmlangsummit.com/images/8/8e/GraalJVMSummit2011.pdf>
- <http://www.infoq.com/news/2011/10/java-data-grid>
- <http://www.jboss.org/infinispan> and <http://jcp.org/en/jsr/detail?id=107>

Thank you for your attention!