Under the Roof of OpenJDK8

DevConf, Feb 2013

Jiří Vaněk

Under the Roof of OpenJDK8

DevConf, Feb 2013

- Jigsaw
- Lambda
- Small features

Under the Roof of OpenJDK8

DevConf, Feb 2013

- Lambda
- Small features
- Jigsaw

Index

One page of history

devconf 2012 - http://rvokal.fedorapeople.org/devconf2012/jvanek-jdk8.pdf

1992 - started at Sun labs as project Oak

1996 - Version 1 publicly released

Cca **1998** - plugin, JIT, GNU classpath

Cca **2001** - HotSpot, javaws

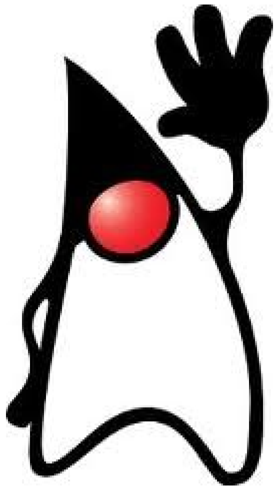
Cca **2006** - JDK6, OpenJDK, IcedTea,

Cca **2009** - acquisition by Oracle, IcedTea-Web

Cca **2011** - OpenJDK7,

- merging Of IcedTea to OpenJDK, jigsaw to JDK9

Middle of **2013** – OpenJDK8....



Well.. two of them

- OpenJDK7 – released in time, July 2011
  - JVM support for **dynamic languages (invoke dynamic)**,
    - JRuby/Scala/... call directly to JVM (and no transformation to Java at first) – really great performance boosts
    - Via custom code which JVM inline through
  - Small language changes (grouped under a project named **Coin**):
    - **Strings in switch, AutoCloseable, numeric literals**
    - Catching **multiple exception** types and rethrowing exceptions with improved type checking
  - 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
  - Care is taken of **community**
  - It **was** conservative change at the end!

OpenJDK8

- Jigsaw ---> JDK9 :(, decided October 2012
- Lambda
- Small features
    - Finish Coin
- JavaFX 3.0 opensource
- Grail
- ....

OpenJDK8

- http://openjdk.java.net/projects/jdk8/
- Cloned from JDK7 at 2012/04/26
    - ...
    - **2013/01/31 Feature Complete**
        - Not true – 14.1 several features dropped, many rescheduled to next milestones
    - 2013/02/21 Developer Preview
    - 2013/07/05 Final Release Candidate
    - 2013/09/09 General Availability

OpenJDK8 – sources and build

- Bundles available at http://jdk8.java.net/download.html, and are going to be in Fedora 19 (of course from source)
- Or you can get bleeding edge by mercurial:
  - *hg clone* http://hg.openjdk.java.net/jdk8/jdk8
    - + *sh ./get_source.sh*
  - or
    - *hg clone* http://hg.openjdk.java.net/jdk8/jdk8/jdk
    - *hg clone* http://hg.openjdk.java.net/jdk8/jdk8/corba
    - *hg clone* http://hg.openjdk.java.net/jdk8/jdk8/hotspot
    - *hg clone* http://hg.openjdk.java.net/jdk8/jdk8/jaxp
    - *hg clone* http://hg.openjdk.java.net/jdk8/jdk8/jaxws
    - *hg clone* http://hg.openjdk.java.net/jdk8/jdk8/langtools
  - *./configure; make* (jdk7 needed to compile it)
  - ^new:)^
  - http://hg.openjdk.java.net/jdk8/build/raw-file/tip/README-builds.html

    But well, maintained just sporadically

- But be aware, no jigsaw, no lambda.... (for Fedora we will try.. :o)

Lambda – where to get it

- http://openjdk.java.net/projects/lambda/
  - *hg clone http://hg.openjdk.java.net/lambda/defender-prototype*
  - *ant*
  - *hg clone* http://hg.openjdk.java.net/lambda/lambda
  - *sh ./get_source.sh*
    - Will add additions to corba, jaxp, jaxws, langtools, hotspot, jdk (rest of jdk is needed too)
    - ./configure
    - make
- JDK8 bundles: http://jdk8.java.net/lambda/
- Supported by Eclipse HEAD and Netbeans 8, IDEA 12
  - http://git.eclipse.org/c/jdt/eclipse.jdt.core.git/log/?h=BETA_JAVA8

Lambda – what it is?

- What is lambda?
- An anonymous function
    - Parametrize behavior
    - Treat behavior as data
    - Provides closure mechanism
- Provides:
    - More effective code
    - Parallelism

λ

Lambda – what it is?

To request some kind of functionality, java is using so called ***functional interface****. Eg* ***Runnable***

```
public void spamMatchingPersons(Predicate<Person> predicate) {
  List<Person> persons = gatherPersons();
  for (Person p : persons) {
    if (predicate.test(p)) {
      EmailAddress emailAddress = p.getEmailAddress();
      sendEmail(emailAddress);
    }
  }
}
```

Lambda – what it is?

- Declaration of predicates is to verbose

```
public void spamPossibleAlcoholics() {
  SpamMatchingPersons( new Predicate<Person>() {
    public boolean test(Person p){
        return p.getAge() >= 18;
    }
  });
}
```

- The only stuff we wanted to say **if getAge() >= 18;**

Lambda – why to change well known pattern?

- Generalization is even more verbose
    - By generalized predicates
        - spamPersonsOlderThan(n)
        - spamPersonsBetween(m,n)
        - Some really complicated Predicate
    - Soon you are in generics, substituted methods, own interfaces or whatever...

Lambda

```
public void spamPossibleAlcoholics() {
  SpamMatchingPersons( new Predicate<Person>() {
    public boolean test(Person p){
        return p.getAge() >= 18;
    }
  });
}
```

- Our first lambda expression (Person p) -> p.getAge() => 18

```
public void spamPossibleAlcoholics() {
  SpamMatchingPersons((Person p) -> p.getAge() => 18);
}
```

Lambda

- (int x, int y) -> x+y
- () -> 42
- (String s) -> {System.out.println(s)}
- Lambda will enhance many parts of JDK itself
  - (String s) -> s.toLowerCase()
- Is still object
  - Is translated to the functional interface
    - Guessing types during compile time
    - But invoked via **invoke dynamic** (no instance during runtime)
- Should be stateless because of possible parallelism
- Can receive **effective final** outer local variables
- Jdk8 is going beyond user forEach(()->{})
  - Collections framework enhancements
    - Bulk data operations
    - Iteration delegated into inside of collection
  - Parallelism
  - Defender methods
  - Incorporated into JDK

Lambda – Defender methods

- Default (dummy) implementations of interface methods
  - Disturbing pureness of interface just a bit
    - No global variables
    - Stateless
  - Can be removed lower in hierarchy (**none** keyword)
  - Support **supper**

```
public interface IterateV5{
   public void forEach(Block<T> b)
       default Collections.<T>setForEach;
    public void doMagic() default{
       System.out.println("Default from interface");
     }
   }
```

Where

```
class Collections {
   public static<T> void setForEach(Set<T> set,Block<T> block) {
     ...
   }
}
```

Lambda – under the roof

Defender methods – inheritance

- Mess by inheriting methods from multiple ancestors become somehow possible
- Inheritance algorithms is pretty complicated
  - Closest, best type-matched implementation
  - One can declare the supper default directly

```
interface A { void m() default X.a; }
interface B extends A { void m() default X.b; }
interface C extends A { }
abstract class D implements B, C { }
```

Straight forward to X.b

```
interface A { void m() default X.a; }
interface B extends A { void m() default X.b; }
interface C extends A { void m() default X.b; }
abstract class D implements B, C { }
```

Compile time error unless D implements m()

Lambda – under the roof

Enhancing collections

- The only reason defender methods were **added** was to be able to add methods to collection's interfaces and to not to destroy backward **compatibility** and so **evolve** collections framework
  - If defender methods are + or – will be proven in time
  - Now all the **Collections** (and iterator and some more) have **stream()** method (and some more)

```
public interface Stream<T> {
    Stream<T> filter(Predicate<? super T> predicate);
    <R> Stream<R> map(Mapper<? extends R, ? super T> mapper);
    <R> Stream<R> flatMap(FlatMapper<? extends R, ? super T> mapper);
    Stream<T> uniqueElements();
    Stream<T> sorted(Comparator<? super T> comparator);
    Stream<T> cumulate(BinaryOperator<T> operator);
    void forEach(Block<? super T> block);
    Stream<T> tee(Block<? super T> block);
    Stream<T> limit(int n);
    Stream<T> skip(int n);
    <A extends Destination<? super T>> A into(A target);
    Object[] toArray();
    <U> Map<U, Collection<T>> groupBy(Mapper<? extends U, ? super T> classifier);
    <U, W> Map<U, W> reduceBy(Mapper<? extends U, ? super T> classifier,
                          Factory<W> baseFactory,
                          Combiner<W, W, T> reducer);
    T reduce(T base, BinaryOperator<T> op);
    Optional<T> reduce(BinaryOperator<T> op);
    <U> U fold(Factory<U> baseFactory,
              Combiner<U, U, T> reducer,
              BinaryOperator<U> combiner);
    boolean anyMatch(Predicate<? super T> predicate);
    boolean allMatch(Predicate<? super T> predicate);
    boolean noneMatch(Predicate<? super T> predicate);
    Optional<T> findFirst();
    Optional<T> findAny();
    Stream<T> sequential();
   Stream<T> unordered();
}
```

```
someCollection.stream()
    .filter(...)
    .map(...)
    .forEach(...)
```

```
public void spamHomePossibleAlcoholics() {
    gatherPersons().stream()
        .filter(p -> p.getAge() >= 18)
        .map(p -> p.getHomeEmailAddress())
        .forEach(emailAddress ->
sendEmail(emailAddress));
}
```

Lambda – under the roof

Parallelism

- Lambda methods can be easily parallelized Collections' **Stream<T> parallelStream()** method and **ParallelIterable<T> parallel()**
- Interfaces *java.util.*
    - Spliterator<T>
    - Splittable<T, S extends Splittable<T, S>>
    - ParallelIterable<T> extends Splittable<T, ParallelIterable<T>>

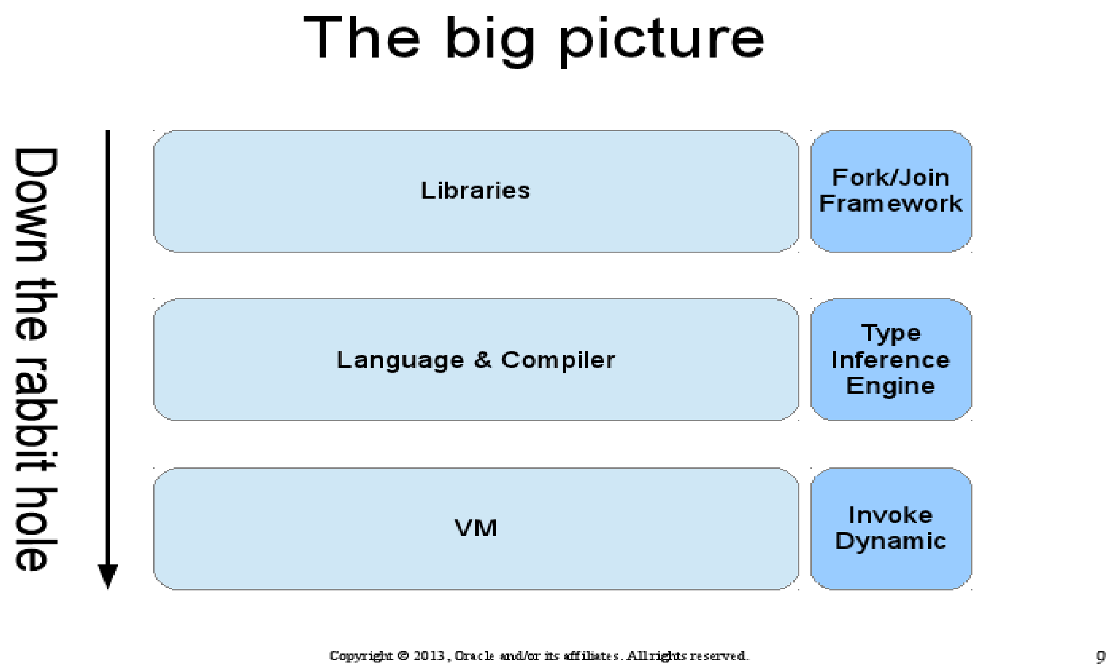
```
public void spamHomePossibleAlcoholics() {
    gatherPersons().parallelStream()
        .filter(p -> p.getAge() >= 18)
        .map(p -> p.getHomeEmailAddress())
        .forEach(emailAddress -> sendEmail(emailAddress));
}
```

- can be controlled, but default implementation is good enough
- can be dangerous and slow if done wrong

Lambda – under the roof

Parallelism

- Under the roof parallelism
  - is nothing more then **Fork and Join framework**
  - **spliterator** nothing more then better way how abstract and specify the dividing of work
  - .paralel() is just simplified and generalized access to it
  - Anonymous classes into which the Lambda is compiled are never instantiated by **invoked dynamically**



Small changes

1) Miscellaneous
2) Javadoc
3) Infrastructure
4) Lang
    1) JVM
    2) Lambda connected features
5) Networking
6) Cryptography

Small Features of OpenJDK8 – Miscellaneous

Locale Matching

- New API to with full implementation of BCP 47 (Internet best current practice for tags for identifying languages)
- Will provide sorted list with best matching locales for user

Date and Time API

- New (immutable) date,time,calendar API
- Based on JodaTime
- Basic set of extensible calendars
- Performance boost

Charset Implementation Improvements

- Improve maintainability and performance and decrease size
- A lot of work done in 7 (eg loading of system fonts instead of plain-text mapping)
  - In 7 some of it just do not work
  - New String(byte[]) and string.getBytes()

Small Features of OpenJDK8 – Miscellaneous

Adopt unicode CLDR Data for i18n

- Set of new tools
  - to convert between individual formats
  - To pack them
- Another “java's own format” replaced by standard one

Unicode 6.2

- Adapt to latest (September 2012) unicode

Base64 Encoding/Decoding

- Unify all 5(!) internal implementations to one improved java.util public api

Small Features of OpenJDK8 - JavaDoc

DocTree API

- Enable access to syntactic elements of javadoc
- Prepare path for javadoc tools evolution (finally!)

Javadoc to javax.tools

- Starting the new javadoc evolution
- Allowing execution of javadoc via api
  - Instead of new process “javadoc”

DocLint

- Detect errors in javadoc in compile time
  - Bad syntax
  - Bad html
  - Bad annotations
  - Bad targets
  - ....

Small Features of OpenJDK8 - infrastructure

Compact profiles

- Specify profiles, so java aps will not need to load whole JDK
  - Eg no-gui app will no longer load swing from rt.jar
- Jigsaw?
  - Compact1 java.{io,lang,math,nio,security,text,util,crypto,net}
  - Compact2 java.{rmi,sql,transaction,xml}, org.w3c.{sax,dom}
  - Compact3 java.{lang,management,naming,security,sql,util,xml,tools}, org.ieft.jgss
    - The same packages in profiles have mostly empty intersection of subpackages

Prepare for modularization

- Provide substitute API for some commonly used private stuff
- Deprecate APIs which will become unavailable after modularization is done

Small Features of OpenJDK8 - infrastructure

Autoconf-Based built system

- Introduce autoconf (./configure-style) build setup, refactor the Makefiles to remove recursion
  - Increase build speed radically
  - Simplify build-system source code (Makefiles, etc.)
  - Simplify work for developers
  - Get exact and reproducible build output
  - Update the Makefile structure
  - Add parallel Java compilation support
  - Make Java builds incremental
- Result of M4 compilation (generated ./configure script) **will be checked** to repository

Launch JavaFX applications

- Enable commandline java command to launch also JavaFX applications directly

Small VM

- Support the creation of a small VM that is no larger than 3MB.
  - Make necessary modifications so that we can optionally build a small VM no larger than 3MB. (now client and server VMs are around 6 and 9MB)
  - Allowing some features to be excluded at build time, and by optimizing the C++ compiled code for space when possible.
  - A performance degradation with the small VM of up to 5% is acceptable.
  - There is no plan to retain full capabilities
  - There is no plan to make functionality optional at runtime.

Small Features of OpenJDK8 – Lang

Annotation on Java Types

- Now annotations are allowed for
  - Classes
  - methods
- @Interval(min=10,max 20) int sizeOfSquirel;
  - Easy pluggable data checkers on top of it

Generalised Target-Type Infrence

- Remove burden of redundant type declaration
- Eg from
  - String s = List.<String>nill().head();
- To
  - String s = List.nill().head();

Access to Parameter Names at Runtime

- Get rid of custom *@ParameterName*
- Java have access to all names except parameters' ones
  - Smallest change with impact to byte code

Small Features of OpenJDK8 – Lang

Repeating annotations

- Now annotations are allowed for language member just oncetime per annotation
- This should allow multiple same-name annotations

javax.lang.model backed by reflection

- Move the responsibility from javac to public api
  - access and process reflective information about loaded classes by this API

Jdbc 4.2

- Just minor changes

Reduce Core-Library Memory Usage

- Reducing heap size occupied by core libraries without lost of performance
- Candidates:
  - Reduce base Object size
  - Disable reflection compiler
  - Direct memory reductions found by heap analyze

Small Features of OpenJDK8 – Lang - JVM

Remove permanent generation

- Part of jrockit and hotspot conversion
  - Jrockit customers do not need to tune permanent generation
  - There should be no need for it in JVM too

G1 GC: Reduce need for full GCs

- Enhance G1 so that it does not r4ely on full GCs to perform class unloading or any other critical operation
  - Shorter pauses during GC
  - There should be no need for it in JVM too
- Dropped 14.1.2013

Compiler control

- Unify all wide-spread compiler flags and settings to one well documented common way
  - Affect both C1 and C2
  - Possibility to change this settings in runtime

Fence Intrinsic

- Adding memory-ordering intrinsic to *sun.misc.Unsafe* as known from C11/C++11 on JVM level
  - void loadFence()
  - void storeFence()
  - void fullFence()
- Maybe public in java.util.concurent later

Small Features of OpenJDK8 – Lang - JVM

Limited doPrivllдged

- Enable asserted code to run without full access-control stack walk to check for permissions
- Possible security impact?

Concurrency Updates

- Scalable updatable variables, cache-oriented enhancements to the
  - ConcurrentHashMap API
  - ForkJoinPool improvements,
  - additional Lock and Future classes and better support for software transactional memory (STM) frameworks(?)

Reduce Class Metadata Footprint

- Reduce HotSpot’s class metadata memory footprint in order to improve performance on small devices
- Many manual actions
  - Reducing offsets and pointers
  - Squeeze what can be squeezed (eg 33b to 32b instead of 64 :)
  - Put away rarely used fields
- Better usage of automated optimization

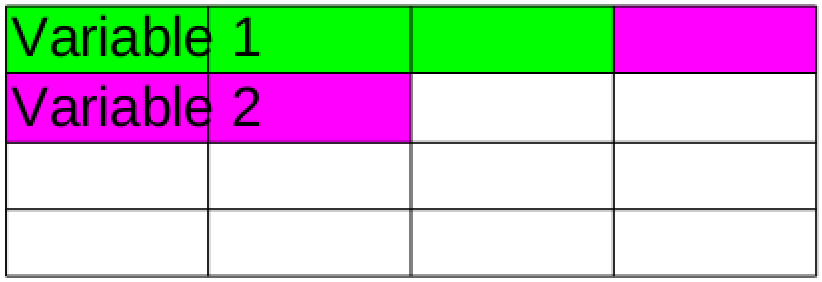
Small Features of OpenJDK8 – Lang - JVM

Enhanced Verification Errors

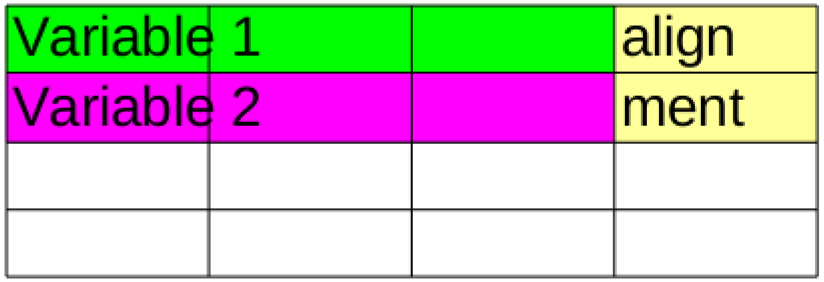
- Bytecode is verified in JVM, but in case of failure exceptions are to vague or misleading
- Although rare, those will be enhanced

Reduce Cache Contention on Specified Fields

- Find way how to specify fields which can spread across multiple cores or share lines in caches
  - ..and avoid it
- By aligning the fields
  - By Adding padding before and/or after
- Performance
- Parallelisation



cache without alignment



cache with alignment

Small Features of OpenJDK8 – Lang - Lambda connected features

Bulk data operations (filter, map, reduce)

Lambda expressions themselves

Integrate Lambda into Core Libraries of JDK

- where useful (and possible)

Collections Enhancements from Third-Party Libraries

- Goal si not to eliminate v3rd parties, but to learn form them and use what can be used
- Dropped 14.1.2013

Parallel Array sorting

- New methods to Arrays class like
  - *public static parallelSort*
- Dependence and similarity on Lambda with Fork and Join framework
  - Implementation based on ParallelArray framework

Lambda-Form representation of Method Handles

- Improve performance, quality and portability of method handles and invoke dynamic
  - Reduce assembly code in jvm
  - Reduce native calls

Small Features of OpenJDK8 – Networking

New HTTP Client

- Current URLConnection done with Legacy (ftp, gopher) protocols in mind
- Goal si to made new extensible Api
- Based on NIO
- Keep EE in mind
- Dropped 14.1.2013

TLS Server Name (SNI) Extension

- TLS is is extension fore more flexible and secure virtual-machine/server/hosting infrastructures based on SSL *rt*
- Already supported by most major
  - Browsers
  - Servers

Network Interference Aliases Events and Defaults

- Make java SE to work on devices with multiple network (or hierarchical) interfaces
  - Listen to changes **of** the devices configuration
  - Select device

Small Features of OpenJDK8 – Cryptography

Configurable Secure Random-Number generation

- Currently
  - Mix of blocking/not-blocking system calls
  - No configureability and wrong documentation
  - On linux reading of */dev/random* and is blocking until enough entrophy
- */dev/urandom* can provide good randomnes without blockin
- Except */dev/** also custom algorithms
- Both runtime and deplyment configuration
  - sr = New SecureRandom(TYPE_OF_QUALITY)
  - java.security policy

MS-SFU Kerberos 5 Extension

- Specified since 2003
- Client-server and server-client delegation

Stronger Algorithms for Password-Based Encryption

- Lot of current algorithms i in JDK are legacy
  - DES
  - RC2
  - ...
- Need to add new ones
  - SHA-2
  - PBE

Small Features of OpenJDK8 – Cryptography

Enhance the Certificate Revocation-Checking API

- *java.security.cert* API to be enhanced
  - Current api is pass/fail only
  - New API should be independently checkin each step and have callbacks and fallbacks

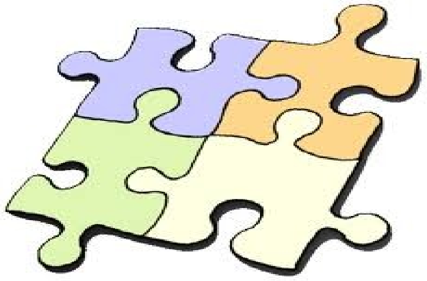
Overhaul JKS-JCEKS-PKCS12 Keystore

- Migrate current keystore format to standard PKCS#12
- Update java tools
- Update API
- Another “java's own format” replaced by standard one

Various suites

- **AEAD Cipher suite**
- **NSA Suite B**
- **SHA-224**
- **PKCS#11 crypto provider for Windwos64b (finally!)**

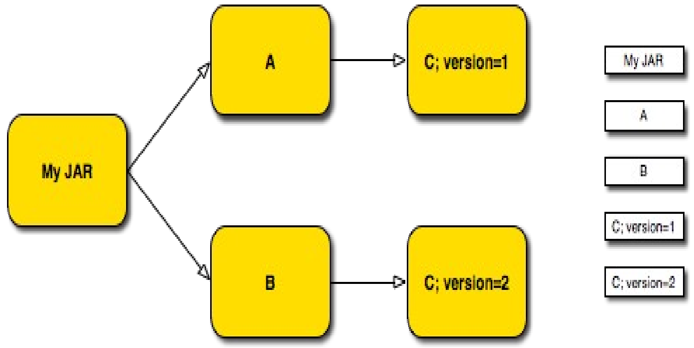
Project Jigsaw



- Postponed to JDK9– **modularisation of Java platform**
- Continuous integration into JDK7/8 via project penrose (approved Jan/Feb 2012)
- Current JDK is **monolithic** and **huge** (more then 100MB)
  - “Hello world” in Maven can have up to 4MB
- 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 is not sure even for JDK9 :(

Project Jigsaw

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



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

Project Jigsaw

- Platform fragmentation
    - Will allow unification of SE x ME (and EE?)
        - There are complicated license issues for 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
        - Windows will get shorten? O:)
    - Support for better cooperation with native modules also in JDK9?
- 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?

Project Jigsaw

- 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;
 }
}
```

Maven ---> (pom compatibility)

---->jar (classical, classpath re-usable jar)

---->jmod

---->rpm

---->deb

---->war,ear (JDK 9?)

Project Jigsaw - build

- hg clone http://hg.openjdk.java.net/jigsaw/jigsaw
- cd jigsaw
- bash get_sources.sh
- ./configure
- make all

Project Jigsaw - build

- Result
    - build/linux-{i586 amd64}/jdk-module-image”.
- In bin are new tools
    - jmod
    - jpkg
- The ”lib/modules”
    - folder contains a myriad of modules.
    - The JDK is no longer this huge ”rt.jar” with a gravity of JARs around
    - it is now a set of modules.
    - Each module contains (except classes)
        - index
        - metadata

Project Jigsaw – first module

- **mkdir** -p *sources/fact/fact*
- **mkdir** *modules*

- *fact/Factorial.java*

```
package fact;

public class Factorial {
  public static int factorial(int n) {
    if (n <= 0) { return 1; }
    else        { return n * factorial(n - 1); }
  }
}
```

- *fact/module-info.java*

```
module fact @1.0 {
   exports fact}
```

- javac -d modules -modulepath modules -sourcepath sources \

```
    sources/fact/module-info.java \
    sources/fact/fact/Factorial.java
```

Project Jigsaw – first dependent module

- *hello/Main.java*

```
package hello;

import static fact.Factorial.factorial;

      public class Main {
        public static void main(String... args) {
          System.out.println(factorial(10));
        }
      }
```

- *hello/module-info.java*

```
module hello @1.0 {
  requires fact @1.0;
  class hello.Main;
}
```

- javac -d modules -modulepath modules -sourcepath sources \
      sources/fact/* sources/hello/*
- java -m hello

Project Jigsaw – deploy and run

- **jmod** create -L repo
- **jmod** install modules hello fact -L repo
    - **find** repo/
        - repo/
        - repo/fact
        - repo/fact/1.0
        - repo/fact/1.0/index
        - repo/fact/1.0/info
        - repo/fact/1.0/classes
        - repo/%jigsaw-library
        - repo/hello
        - repo/hello/1.0
        - repo/hello/1.0/config
        - repo/hello/1.0/index
        - repo/hello/1.0/info
        - repo/hello/1.0/classes
- **java** -L repo -m hello
    - 3628800
- Modules can be used also directly from **modules** dir where were built

Project Jigsaw – deploy and run

- Make jmod packages
  - **jpkg** -m modules/fact jmod fact
  - **jpkg** -m modules/hello jmod hello
    - fact@1.0.jmod hello@1.0.jmod
- Make linux packages
  - jpkg -m modules/ deb hello
  - jpkg -m modules/ rpm fact
    - fact_1.0_x86_64.deb hello-1.0.x86_64.rpm
- Install module back from a jmod package
  - jmod install -L repo hello@1.0.jmod
  - java -L repo -m hello
    - 3628800

Project Jigsaw – little bit under the hood

- Declaration

```
module foo{}
module foo @1.0 {}
```

  - Version is optional
  - Name is qualified java identifier
  - No annotations

- Exports

```
module foo{
    exports foo;
}
```

  - Exports all public types in foo, but not in subpackages
  - Name convention

```
module foo{
    exports foo;
    exports foo.bar;
    exports foo.baz;
}
```

  - No private members export ever!

Project Jigsaw – little bit under the hood

- Requires

```
module bar{
        requires foo;
}
```

  - **foo** and **bar** will have **different classloaders**
  - Do **not** export foo's classes
  - Optional version constraints

```
module bar{
        requires foopa @ >=1.0;
        requires foot  @ <2.3a;
}
```

- Re-exports

```
module bar{
        requires public foo;
}
```

  - Reexports foo's classes (otherwise same)

Project Jigsaw – little bit under the hood

- Services

```
module bar{
    provides service servers.Server with myServers.MyServerImpl;
}
```

  - Provides implementation of service

```
module bar{
    requires service servers.Server
}
```

  - Is requiring implementation(s) of service
  - will got myServers.MyServerImpl in this case
- Enhanced ServiceLoader API with possibility of select the impl
- Services creation:

```
Class<Foo> serviceInterface = ...;
ClassLoader serviceConsumer = ...;
// Lazy,  No service instances are instantiated
Iterable<Foo> services = ServiceLoader.load(serviceInterface, serviceConsumer);
// Instantiation occurs on each call to Iterator.next()
for (Foo service : services) { if (service.isCapableOf(...)) {
    return service;}}
return new DefaultFoo();
```

Project Jigsaw – little bit under the hood

- Permits

  module foopa{
  permits bar;
  }

  - **foopa** can be required **only** by bar
  - Otherwise same

- Local dependence

  module bar{
  requires local foopa;
  }

  - **foopa** must explicitly permits bar
  - **foo** and **bar** will have **same classloaders**
    - **The only case of shared classlaoder**
    - Multi-module packages

- Optional dependence

  module foopa2{
  requires optional bar2;
  }

  - Must be ready to work without it

Project Jigsaw – little bit under the hood

- Entry point
  ```
  module foo{
      class foo.Main;
  }
  ```
  - Alternative to manifest entry with main method
  - Java -m foo

- Base module

  Jdk itself - *java.base*

  If module is not requiring exact version, then platform default is added

Project Jigsaw – little bit under the hood

- Aliases

```
module foo{
        provides bar;
}
```

  - Renaming of bar?
  - Necessary for renaming of known packages to new modules

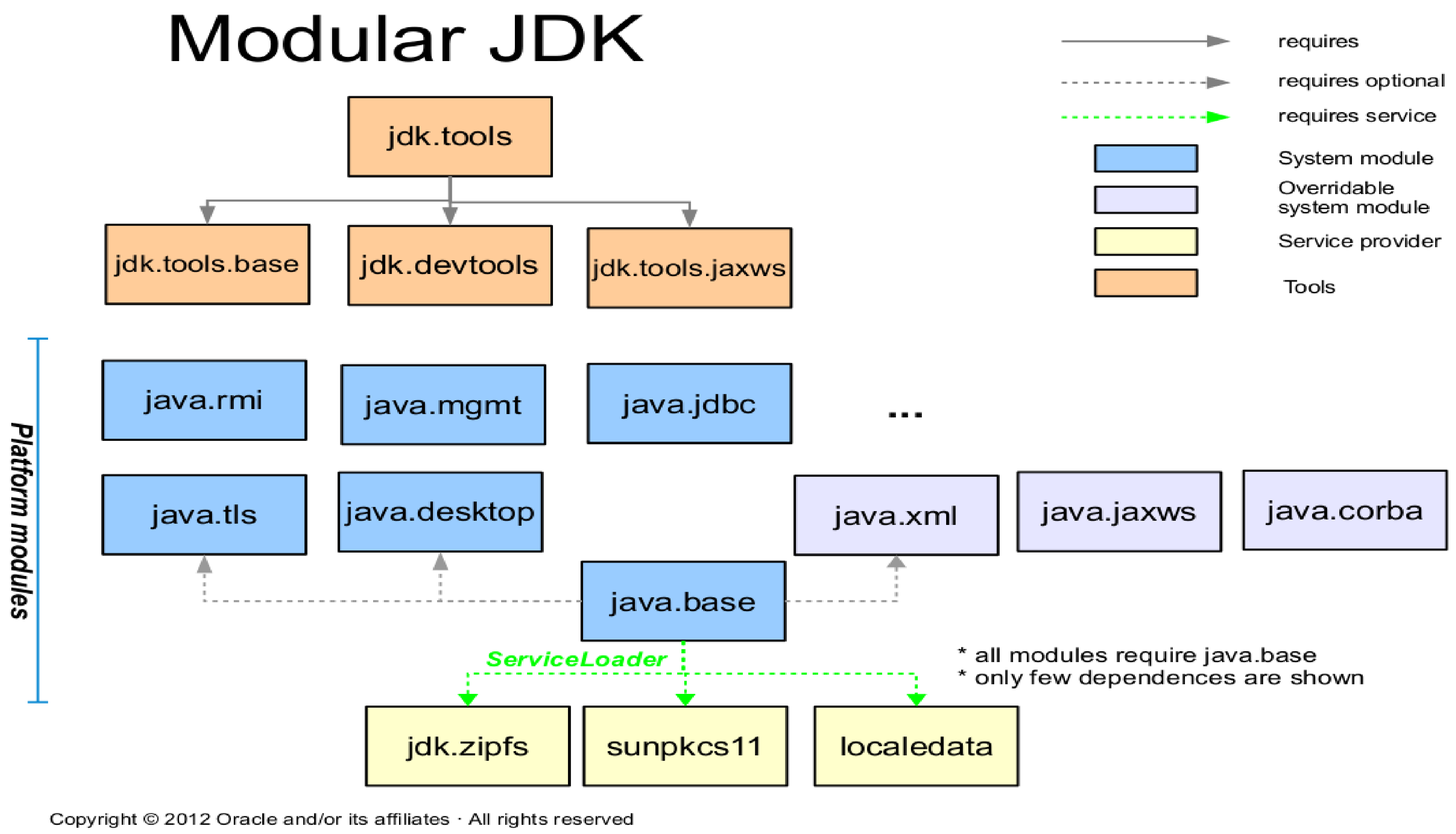
- view

```
module bar{
        requires foo;
        exports bar
        view bar.internal {
                permits baz;
                exports bar.private;
        } view cat {
                class org.foo.Cat
        } view ls{
                class org.foo.List
        }
}
```

- java -m cat   x   java -m ls

Project Jigsaw – modular jdk

- Aliased by *java.base*



Project Jigsaw – classlaoders

- Class.getClassLoader() will never be null
  - There will be classlaoder(s) for *java.base*
  - Replacement for bootclassloader
- Each module will have its own classlaoder
  - Except multi-module package
- No possibility to access private classes of other modules
  - Some hackish way to get its classlaoder and then access via some new reflection tricks??

Project Jigsaw – byte code

- The module-info.java is compiled into module-info.class
- New ClassFile.access_flag ACC_MODULE (0x80000) added on byte code level
- Major/minor version limitation (>= 53.0, jdk 9)
- No implicit reexports – just expanded
- Also views are expanded
- Dependencies, exports and services are tables with indexes to constant pool

Conclusion

- Oracle have fulfill some of his promises
    - Community is taken care about
    - Lambda is going in
    - Most of the small changes are going in
- Dropping of jigsaw in October is sad but probably worthy
    - Modularisation of binary parts?
    - Convergence of java ME?
- Dropping of some of some in January 2013 ..
    - Well smells like problems
- At least it is still evolution and not revolution

Questions?

- http://openjdk.java.net/projects/jdk8/
- http://wiki.eclipse.org/JDT_Core/Java8
- http://dharrigan.com/2011/11/20/building-jdk8-with-lambda-support/
- http://openjdk.java.net/projects/jigsaw/doc/module-class-loading.pdf
- http://cr.openjdk.java.net/~briangoetz/lambda/Defender%20Methods%20v4.pdf
- https://docs.google.com/file/d/0BxQTeZmiQCClcEVtOXdqZ25Zem8/edit
- https://wiki.engr.illinois.edu/download/attachments/202146190/L7_ParallelArrary.pdf
- http://openjdk.java.net/projects/jdk8/features
- http://openjdk.java.net/jeps/0 (all the JEPs of Small features)
- http://julien.ponge.info/notes/building-openjdk8-with-jigsaw/
- http://openjdk.java.net/projects/jigsaw/
- Http://openjdk.java.net/projects/jigsaw/doc/openjdk-jigsaw-modular-services.pdf

Thank you for your attention!