ORDINA

Java9 Launch Party

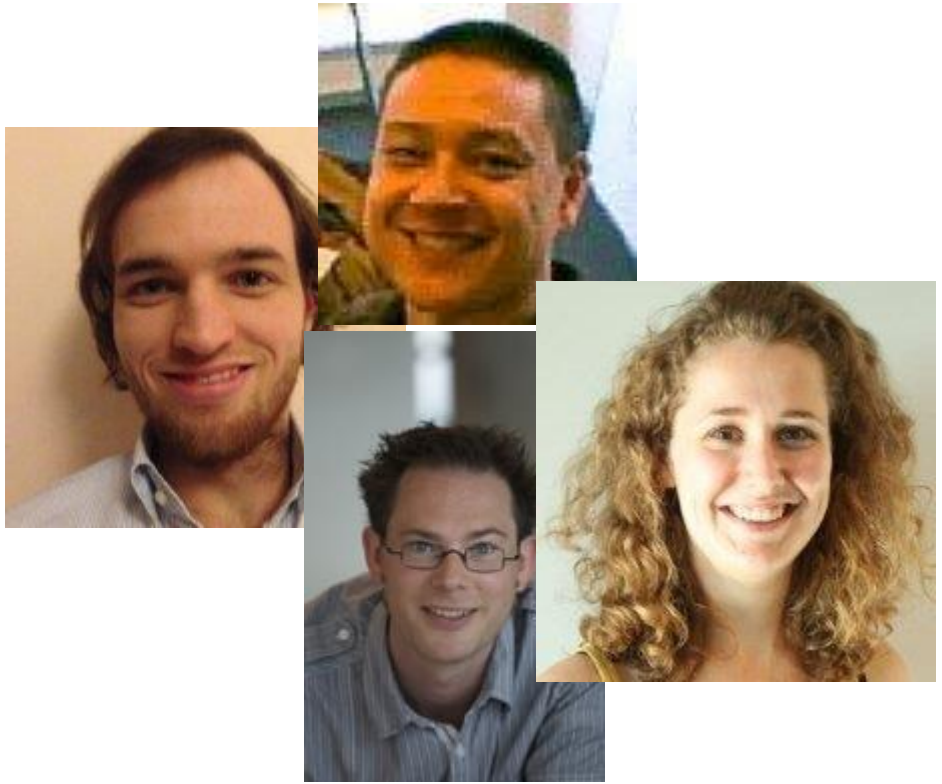
Donderdag

21 September 2017

Even voorstellen

Uw JTECH Java9 team:

- Bas: jigsaw
- Rosanne : jigsaw
- Philippe: jshell
- Hedzer: rest



Programma

- 17:00 – 18:00 Eten 
- 18:00 – 19:30 Presentatie & demo’s
- 19:30 – 21:00 Workshop / lab
- 21:00 – 22:00 Borrel / Party!

Wat vooraf ging..

- Java 7 Release … July 28, 2011
- Java 8 Release … March 18, 2014
- Java 9
  - First Features Announced August 11, 2014
  - Early Access Releases sinds April 2014
  - General Availability ~~July 27, 2017~~ Sep 21, 2017

The State of the Module System

Version 9 UNRELEASED

Start date not set Release date not set Release Notes

20039 Issues in version	20032 Issues done	6 Issues in progress	1 Issues to do

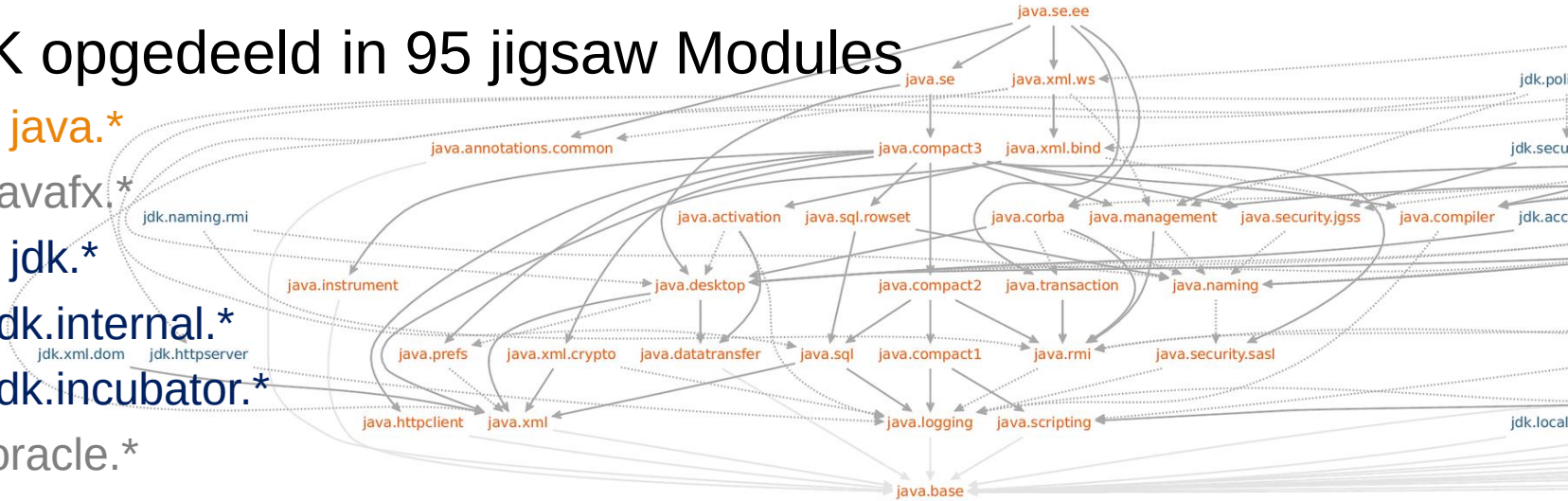
1–6 of 6

View in Issue Navigator

P	T	Key	Summary	Assignee	Status
1	+	JDK-8051618	JEP 200: The Modular JDK	Mark Reinhold	INTEGRATED
1	+	JDK-8051619	JEP 201: Modular Source Code	Alan Bateman	COMPLETED
1	+	JDK-8061971	JEP 220: Modular Run-Time Images	Alan Bateman	INTEGRATED
1	+	JDK-8061972	JEP 261: Module System	Mark Reinhold	INTEGRATED
1	+	JDK-8132928	JEP 260: Encapsulate Most Internal APIs	Chris Hegarty	COMPLETED
2	☑	JDK-8161312	MOS Upload set-up through Hudson for JDK 9	Saravana Vijayasek...	IN PROGRESS

Java 9 in getallen

- 181 early access builds
- 91 JEPs
- 2 JSRs
- JDK opgedeeld in 95 jigsaw Modules
  - 28 java.*
  - 8 javafx.*
  - 51 jdk.*
  - 5 jdk.internal.*
  - 1 jdk.incubator.*
  - 2 oracle.*



Politics!

- JSR-376 JPMS vote 1 8-mei: 10 voor, 13 tegen

Azul Systems, Inc. ✓	Credit Suisse ✗	Eclipse Foundation, Inc ✗	Fujitsu Limited ✓
Gemalto M2M GmbH ✓	Goldman Sachs & Co. ✓	Grimstad, Ivar ✗	Hazelcast ✗
Hewlett Packard Enterprise ✗	IBM ✗	Intel Corp. ✓	Keil, Werner ✗
London Java Community ✗	MicroDoc ✓	NXP Semiconductors ✓	Oracle ✓
Red Hat ✗	SAP SE ✗	Software AG ✗	SouJava ✓
Tomitribe ✗	Twitter, Inc. ✗	V2COM ✓	

- open brief Mark Reinhold 5-mei
  - Oracle vs. Red Hat & IBM – en meelopers
- Voorstel Reinhold 30-mei: 6 weken uitstel
  - Tekstuele aanpassingen, enkele regels code
- Vote 2 26-jun: 1 Abstain: Red Hat, 24 Yes
  - Alle 25 EC members, incl. ARM & JetBrains!

- *Project Jigsaw*: Modules
- *Project Kulla*: JShell
- Milling Project Coin
- Concurrency, o.a. Reactive Streams
- Diverse JDK API uitbreidingen
- Security
- Performance

Verder nog – komt vanavond aan bod

- Factory methods voor Collections
- `@Deprecated(since,forRemoval)`
- Process API improvements
- *Project Verona*: New Version-String Scheme
- Multi-Release JARs

Verder nog – teveel om te behandelen!

- Elide Deprecation Warnings on Import Statements
- GC (Garbage Collector) Improvements
- Diverse Compiler & JVM Improvements
- Stack-Walking API
- Platform Logging API and Service & Unified JVM Logging
- Javascript: Parser API for Nashorn & enkele ES6 features
- Javadoc Search & HTML5 Javadoc
- Unicode 7&8, UTF-8 property files
- XML Catalogs
- Div. GUI-gerelateerde updates (JavaFx)
- hprof & jhat verwijderd
- Linux AArch64, s390x & arm32/arm64 ports
- …

Helaas (net) niet in Java9 terechtgekomen

- HTTP/2.0 Client — incubated, JEP 110, 2014

- Standardized lightweight JSON API — JEP 198, 2014
- Money and Currency API — JSR 354, 2012
- Local-Variable Type Inference — JEP 286, 2016
- Value types — JEP 169, 2012
  - *“Codes like a class, works like an int!”*
  - Denk: C structs in Java – memory & performance


KEEP
CALM
AND
lets get
Started

Project Jigsaw!

Expected since Java 7 and finally here

Wat is Jigsaw?

- Module System
- New component in Java 9
  - Class
  - Interface
  - Package
  - Module

welk probleem lost het op?

- Classpath ‘breekbaar’
  - Reliable configuration
- *Alle* public types *altijd* zichtbaar
  - Strong encapsulation
- Altijd hele JVM geladen
  - Modules & jlink

What is a module?

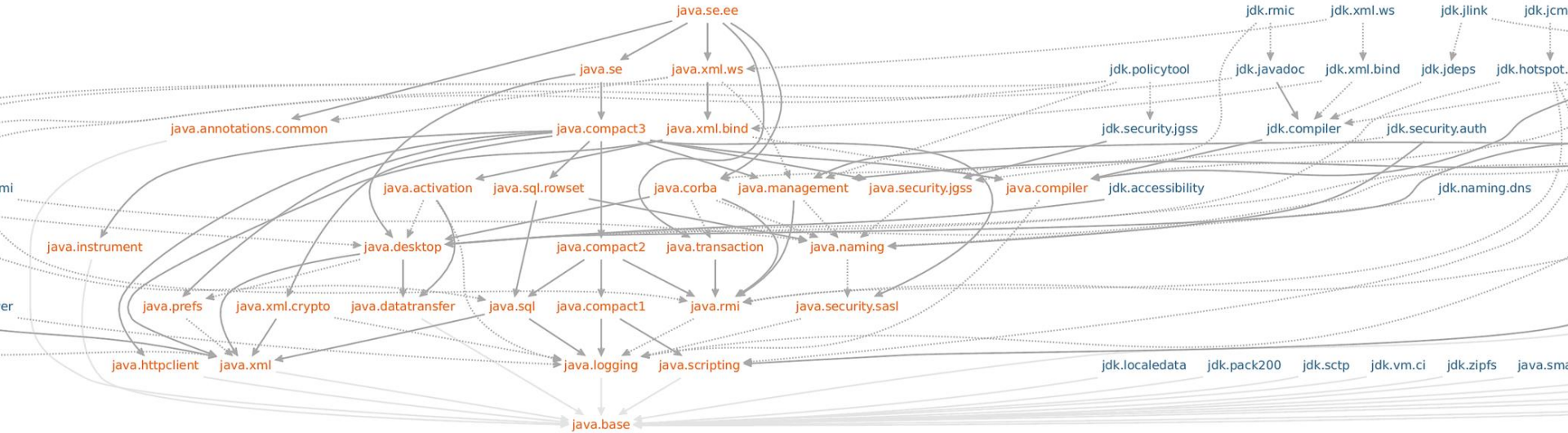
- Container of packages
- Defines its required modules
- Exports specific packages

Modulepath vs. classpath

- Classpath blijft bestaan voor backwards compatibility
- Alles op modulepath moet voldoen aan JPMS eisen

De modulaire JDK

- overzicht van java packages per module



Internal API's

- sun.*
  - Bijv. sun.misc.BASE64En/Decoder
  - Bijv. Sun.misc.Unsafe
- *.internal.*
  - Bijv. com.sun.istack.internal.Nullable
- Not meant for usage

module-info.java

```
module com.foo.bar {
    requires com.foo.baz;
    exports com.foo.bar.alpha;
    exports com.foo.bar.beta
      to some.module,other.module;
    provides com.foo.baz.Service
    with com.foo.baz.impl.ServiceImpl,
         com.foo.baz.impl.ServiceImpl2
}
```

module-info.java

```
module com.foo.bar {
   requires com.foo.baz;      //module
   exports com.foo.bar.alpha;
   exports com.foo.bar.beta
      to some.module,other.module;
   provides com.foo.baz.Service
   with com.foo.baz.impl.ServiceImpl,
        com.foo.baz.impl.ServiceImpl2
}
```

module-info.java

```
module com.foo.bar {
   requires com.foo.baz;
   exports com.foo.bar.alpha; //package
   exports com.foo.bar.beta
      to some.module,other.module;
   provides com.foo.baz.Service
   with com.foo.baz.impl.ServiceImpl,
        com.foo.baz.impl.ServiceImpl2
}
```

module-info.java

```
module com.foo.bar {
    requires com.foo.baz;
    exports com.foo.bar.alpha;
    exports com.foo.bar.beta    //package
        to some.module,other.module; // module
    provides com.foo.baz.Service
    with com.foo.baz.impl.ServiceImpl,
         com.foo.baz.impl.ServiceImpl2

}
```

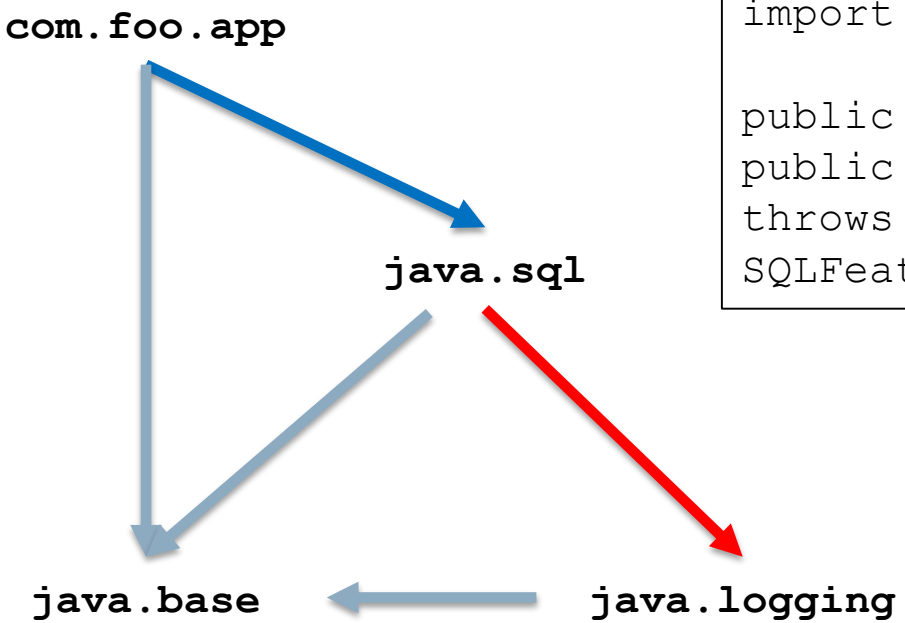
module-info.java

```
module com.foo.bar {
   requires com.foo.baz;
   exports com.foo.bar.alpha;
   exports com.foo.bar.beta
      to some.module,other.module;
   provides com.foo.baz.Service  //interface
   with com.foo.baz.impl.ServiceImpl, // class
        com.foo.baz.impl.ServiceImpl2
}
```

module-info.java

```
module com.foo.alp {
   requires com.foo.baz;
   uses com.foo.baz.Service;      // interface
}
```

Dependency graph




```
package java.sql

import java.util.logging.Logger;

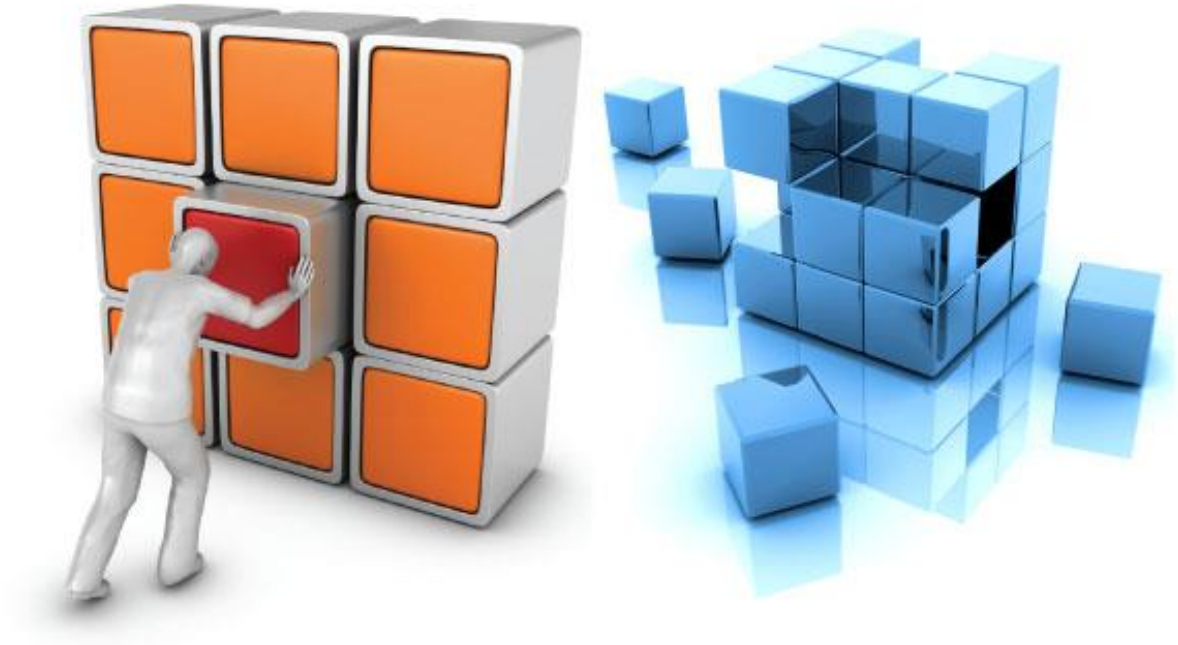
public class Driver {
public Logger getParentLogger()
throws
SQLFeatureNotSupportedException;
```

module-info.java

```
module java.sql {
   requires java.xml;
   requires transitive java.logging;
}
```

Re-exporteert `java.logging` naar alle modules die aangeven `java.sql` te willen gebruiken

Tada! Modules




>_


DEMO

JShell – Quick start

#> jshell

Shifting

8 << 2
8 >> 2
-8 << 2
-8 >> 2
-8 >>> 2

Maar wat is dat nu?

Code completion

```
Integer.(tab)
Integer.to(tab)
Integer.toB(tab)
Integer.toB(tab)( 8<< 2)
```

Define methods

```
String intToString(int number, int groupSize) {
    StringBuilder result = new StringBuilder();
    for(int i = 31; i >= 0 ; i--) {
        int mask = 1 << i;
        result.append((number & mask) != 0 ? "1" : "0");
        if (i % groupSize == 0) {
            result.append(" ");
        }
    }
    result.replace(result.length() - 1, result.length(), "");
    return result.toString();
}
```

Call Methods

```
intToString(8, 3)
intToString(8 >> 2, 3)
intToString(-8, 3)
intToString(-8 >> 2, 3)
intToString(-8 >>> 2, 3)
```

Import statements

```
import java.util.stream.*
```

Lambda’s

```
IntStream.range(0, 31)
IntStream.range(0, 31).mapToObj(i -> (8 & (1 << i)) != 0 ? "1" : "0" )
IntStream.range(0, 31).mapToObj(i -> (8 & (1 << i)) != 0 ? "1" : "0" ).reduce((a, b) -> a + b);
IntStream.range(0, 31).mapToObj(i -> (-8 & (1 << i)) != 0 ? "1" : "0" ).reduce((a, b) -> a + b);
IntStream.range(-31, 0).mapToObj(i -> (-8 & (1 << i*-1)) != 0 ? "1" : "0" ).reduce((a, b) -> a + b);
IntStream.range(-31, 0).mapToObj(i -> (8 & (1 << i*-1)) != 0 ? "1" : "0" ).reduce((a, b) -> a + b);
IntStream.range(-31, 0).mapToObj(i -> (8 >> 2 & (1 << i*-1)) != 0 ? "1" : "0" ).reduce((a, b) -> a + b);
IntStream.range(-31, 0).mapToObj(i -> (-8 >> 2 & (1 << i*-1)) != 0 ? "1" : "0" ).reduce((a, b) -> a + b);
IntStream.range(-31, 0).mapToObj(i -> (-8 >>> 2 & (1 << i*-1)) != 0 ? "1" : "0" ).reduce((a, b) -> a + b);
IntStream.range(0, 31).mapToObj(i -> (-8 & (1 << i)) != 0 ? "1" : "0" ).sorted((a, b) ->
  b.compareTo(a)).reduce((a, b) -> a + b);
```

“Five small amendments”:

- Diamond operator for anonymous inner classes
- Try-with-resources enhancement
- Underscore ('_') character is a keyword
- @SafeVarargs on private methods
- Private methods in interfaces

Milling Project Coin 1/5: Diamond operator for anonymous inner classes

- JAVA 8

```
<T> Package<T> createPackage(T packageContent) {
        // type vereist: 'Non-denotable type'
        return new Package<T>(packageContent) { ... };
}
```

- JAVA 9

```
<T> Package<T> createPackage(T packageContent) {
        // denotable type wordt afgeleid!
        // Java8: "cannot infer type arguments for Package<E>"
        return new Package<>(packageContent) { ... };
}

Package<?> createPackage(Object content) {
        List<?> innerList = Arrays.asList(content);
        // helaas geen diamond - List<?> is 'non-denotable':
        return new Package<List<?>>(innerList) { };
}
```

Milling Project Coin 2/5: Try-with-resources enhancement

- ```
  BufferedReader reader1 = ...
  try (BufferedReader reader2 = reader1) { .. }
  ```

  wordt:

  ```
  BufferedReader reader1 = ...
  try (reader1) { .. }
  ```

- als `reader1` *na* try re-assigned wordt:
- "variable `reader1` used as a try-with-resources resource neither final nor effectively final“
  ```

Milling Project Coin 4/5: @SafeVarargs on private methods

- Signaleert de compiler dat combinatie varargs & generics ‘veilig’ is
- Java 7: alleen op niet-overridable methods
  - static en final methods
- Java 9: ook private methods

```
@SafeVarargs
private void safeVarargsOnPrivateInstanceMethods(List<String>... stringLists) {
    for (final List<String> stringList : stringLists) {
        System.out.println("list: " + stringList);
    }
}
```

Varargs: Java Puzzler…

- ```
  @SafeVarargs
  <T> T[] asArray(T... args) {
        return args;
  }
  ```
- ```
  <T> T[] arrayOfTwo(T a, T b) {
        return asArray(a, b);
  }
  ```

- Faalt *at runtime*:

  ```
  String[] arrayOfTwo = arrayOfTwo("a", "b");
  ```
- Waarom – en met welke Exceptie?

- ```
  java.lang.ClassCastException: java.base/[Ljava.lang.Object; cannot be cast
  to java.base/[Ljava.lang.String;
  ```
  ```

Milling Project Coin 5/5: Private methods in interfaces

- private & private static methodes
  - Uitbreiding van Java 8 default methods
  - Nut: betere encapsulatie & factorisatie van default interface methods

```
interface NameAndCountry {
    static String getName() { return "Jan Modaal"; }
    default String getCountry() { return defaultCountry(); }

    private String defaultCountry() { return "Nederland"; }
}

System.out.println(NameAndCountry.getName() + " woont in " + someNac.getCountry());
```

Collections

- Factory Methods voor Immutable List, Set, Map & Map.Entry:

```
List<String> emptyList = List.of();
List<String> list = List.of("foo", "bar");
Map<String, String> map = Map.of(
  "key1", "value1",
  "key2", "value2");
Map<String, String> mapOfEntries =
   Map.ofEntries( Map.entry("key1", "value1"),
                  Map.entry("key2", "value2") );
```

Concurrency

- Uitbreidingen op:
  - `Optional`
  - `Collectors`
  - `Stream`
  - `CompletableFuture`
- Reactive Streams: Flow

- 3 nieuwe methods:

  - `Optional.`**`stream`**`()`
    - `Stream` met 0 of 1 elementen

  - `Optional.`**`ifPresentOrElse`**`(`

```
    i -> System.out.println("number is " + i),
   () -> System.out.println("empty"))
```

  - *`someOptional`*`.`**`or`**`("fallback value")`

- 2 nieuwe methods:

  - `Collectors.filtering(..)`
    - Variant van `filter()` om samen met `groupingBy()` te gebruiken

  - `Collectors.flatMapping()`
    - efficiënte combinatie van `mapping()` en `flatMap()`

- 4 nieuwe methods:
  - `Stream.takeWhile(..)`
  - `Stream.dropWhile(..)`
  - `Stream.iterate(1, i -> i < 20, i -> i + 2)`
    - Hee, een Stream versie van de `for`-loop!
    - Er bestond al een *infinite* versie: `iterate(1, i->i+2)`
  - `Stream.ofNullable(..)`

- Delays & timeouts
  - `completeAsync(..)`
  - `orTimeout(..)`
  - `delayedExecutor(..)`

- Ondersteuning voor subclassen van `CompletableFuture`

- 4 subinterfaces in java.util.concurrent.Flow:
  - `Publisher`: publishes messages
  - `Subscriber`: receives messages - 4 events
    - PROTOCOL: onSubscribe onNext* (onError | onComplete)?
  - `Processor<T,R> extends Subscriber<T>, Publisher<R>`
  - `Subscription`: connection between publisher & subscriber
    - back pressure / cancel

- 1 implementatie: SubmissionPublisher

@Deprecated

- ```
  @Deprecated(since = "2.0" [, forRemoval = false])
  public void someAncientMethod(){
      System.out.println("Ancient stuff...");
  }
  ```

- @SuppressWarnings(“deprecation”)
  - Java 9: onderdrukt alleen `forRemoval=false`
  - Niet-gewijzigde code kan dus opeens extra warnings krijgen
  - Bijv. `Thread.stop(Throwable) / destroy()`

- @SuppressWarnings(“removal”)
- @SuppressWarnings({“deprecation”, “removal”})
  ```

Verder deprecated / verwijderd

- **Applet** plugin & API
- CORBA
- Explicit constructors for primitive wrappers: `new Integer(1)` i.p.v. `valueOf()` & `parse()`
- `java.base/java.util.Observer & Observable: @Deprecated(since="9")`
- `Object.finalize()`
- **SHA-1 certificates**
- **'_' character**
- De meeste JDK internal APIs zijn inaccessible by default
- **rt.jar & tools.jar** verwijderd – zijn verplaatst naar modules (.jmod)
  - Nu nog: jar/zip met andere extensie en 4B header. Kan veranderen!
- `java.xml.bind, java.se.ee` modules & 4 andere
- **Java DB** wordt Apache Derby
- Enkele obscure GC opties
- **VisualVM**
  - wordt een los Open Source project, wellicht bij Apache
  - NB heel **NetBeans** gaat naar Apache!
- AppleScript
- HTTP Proxying from RMI (was al deprecated)

New Version-String Scheme

“The format of the new version-string is as follows: $MAJOR(.$MINOR.$SECURITY)?(+$PATCH)?”

Voorheen vele waarden, bijv.

- 1.8
- 1.8.0
- 1.8.0_25
- 1.8.0_25-b18
- 8u25

Lastig parsable voor tools zoals maven

System property	Existing	Proposed
java.version	1.9.0	9
java.runtime.version	1.9.0-b100	9+100
java.vm.version	1.9.0-b100	9+100
java.specification.version	1.9	9
java.vm.specification.version	1.9	9

- Datagram Transport Layer Security (DTLS)
- PKCS12 Keystores by default
  - .jks wordt .p12
- DRBG SecureRandom
- SHA-3 Hash
- Weiger SHA-1 Certificates
- TLS ALPN (voor HTTP/2)

- Backwards & forwards compatibility
  - M.n. voor libraries & frameworks

- Maak gebruik van nieuwe taalfeatures waar mogelijk
  - Voorheen:
    - Reflection, bijv. `Class.forInstance("java.time.LocalDate")`
    - Aparte builds:
      - org.apache.tika:tika**-java7**:1.12 vs. org.apache.tika:tika:1.12
      - com.google.oauth-client:google-oauth-client**-java7**:1.16.0-rc vs. com.google.oauth-client:google-oauth-client**-java6**:1.22.0
    - Support ‘laagste’ JDK & VM:
      - Guava 1.0-11.0: Java5, 12.0-20.0: Java6, 21.0: Java8
      - Alleen in documentatie terug te vinden

JAR file layout

```
my-jar.jar/
  A.class                  # wordt gebruikt door Java 8 VM en ouder
  B.class
  C.class
  D.class
  META-INF
  versions/
    9/                      # verwijst naar java.version
       A.class             # vervangt 'base' A.class in Java 9 VM
    10/
       B.class             # vervangt 'base' B.class in Java 10 VM
```

- MANIFEST.MF

  - `Multi-Release: true`

- Problemen:
  - Testing nightmare! Security, anyone?
  - Geen IDE support
  - Maven heeft nog geen nette oplossing

- Crux: welke source file structuur, en hoe te combineren met project java version config?

Migration

Maven

- Maven3.0+
- Cross-compilen:
  - toolchain-plugin (mvn3.3.x), animal-sniffer-plugin of maven.compiler.release property
  - Java <= 5: fork
- Upgrade o.a. compiler & war plugins
  - M.n. voor ‘new version-string scheme’
- Nog onduidelijk hoe multi-release JARs ‘landen’ in Maven
  - alternatieven: /src/main/java{|9|10}, multi-module met assembly of nieuw packaging type
- Best practice: geen module-info.java in src/test/java; alleen voor src/main/java
- Tips:
  - `mvn dependency:list` helpt bij bepalen module names voor module-info.java
  - `mvn jdeps:jdkinternals` toont afhankelijkheden op JDK internals
  - Voeg enforcer rule `banDuplicateClasses` toe

- Zie https://cwiki.apache.org/confluence/display/MAVEN/Java+9+-+Jigsaw

What's in it for me?

- JShell: live coding?
- Modularity in je applicatie ??
  - Eigen distro incl. afgeslankte JRE9; jlink & packaging JEPs
- Performance & security
- Bijblijven met latest-and-greatest
  - Java7 end-of-life April 2015, Java8 per Sep 2018?
- API features, o.a. Collections Factory methods
  - Milling Project Coin overtuigt niet echt..
- Concurrency & Reactive apps
- JavaFx apps, JavaScript

Migratiepad

- Update bouwstraat (Maven *plugins*, Jenkins)
- Update appservers of docker container

Migratiepad

- Update JDK & cfg toolchain
- Update IDE: IntelliJ 2017.1.3+, Eclipse Oxygen??, NetBeans nightly
- Compileer tegen Java9
  - Run jdeps (toont ‘illegaal’ internal API gebruik)
  - Fix fouten, zoals ‘_’, deprecated API usage en unit tests
  - Update zo nodig dependencies & tools
    - Zoals mocking frameworks, aspect weavers en code generators
  - Optioneel: modulariseer je applicatie
    - Maven recipe om op zowel JVM 8 als 9 te kunnen draaien

Migratiepad

- Zie verder OpenJDK 9 Outreach


Workshop!

Workshop

- WiFi: Public / GuestAccess2012

- Zie https://github.com/Ordina-JTech/java9-launch-party
  - Git clone
  - Lees readme.md in een Markdown viewer
  - Volg de instructies

- We zijn er om je te helpen met al je vragen & problemen!

Referenties

- Downloaden: http://jdk.java.net/9
- OpenJDK 9
- JDK9 Javadoc
- What's new (Oracle)
- https://github.com/yannickdeturck/java9workshop
- Let op bij blogs – er is al veel over Java9 geschreven dat er uiteindelijk niet, of heel anders, in is gekomen

Contact