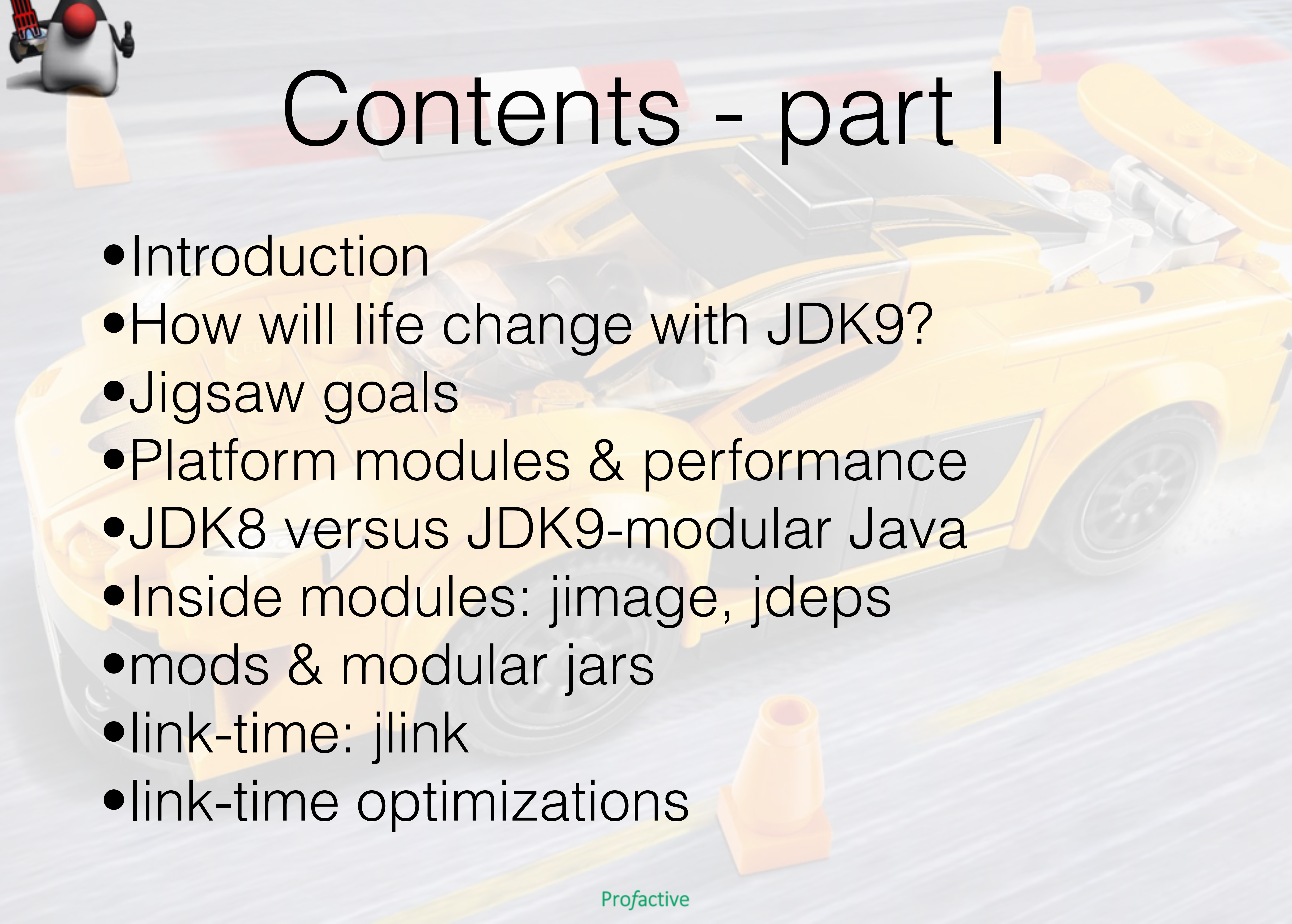
Java 9 Performance

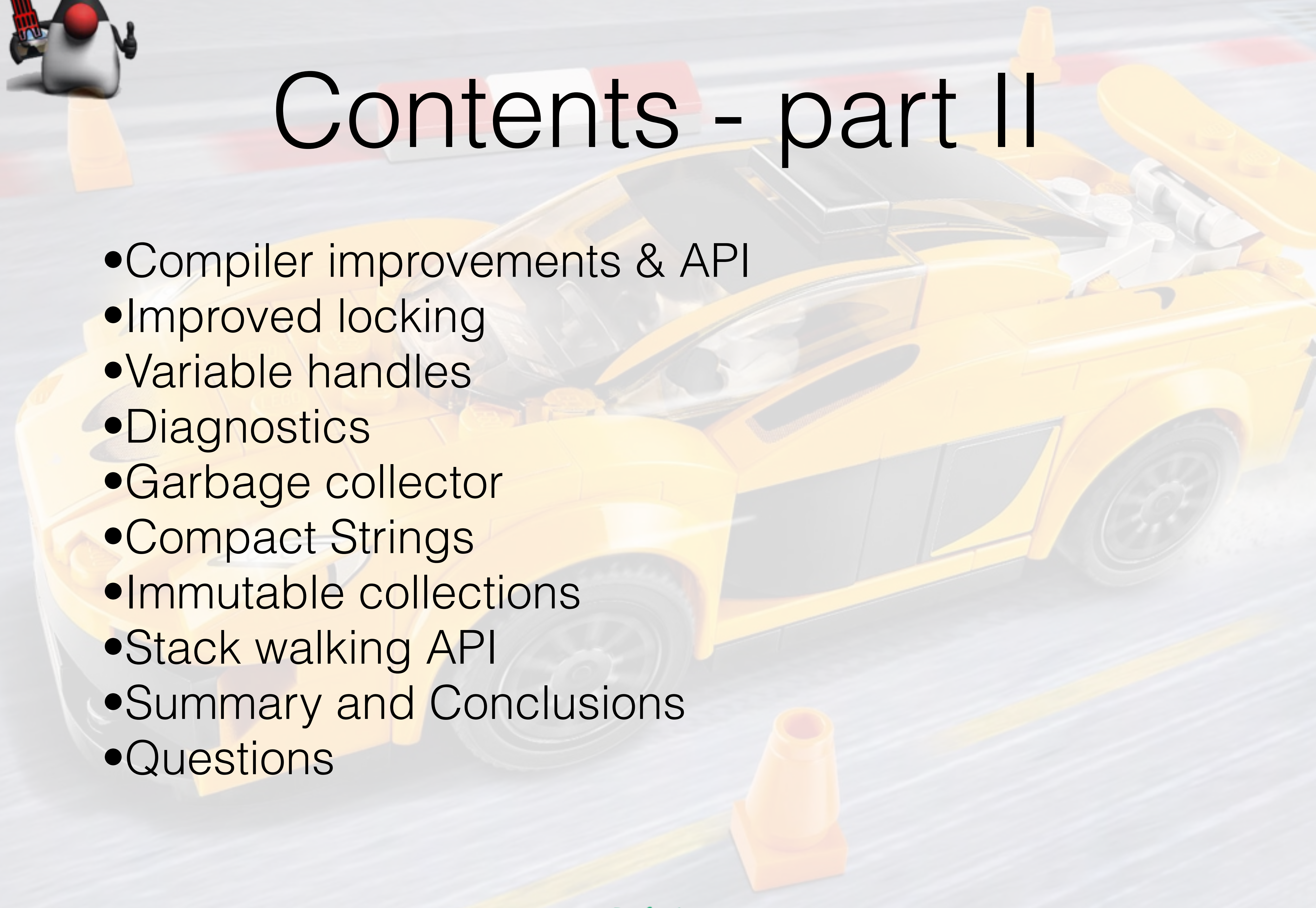
By Jeroen Borgers

jPinpoint

Profactive



Contents - part I

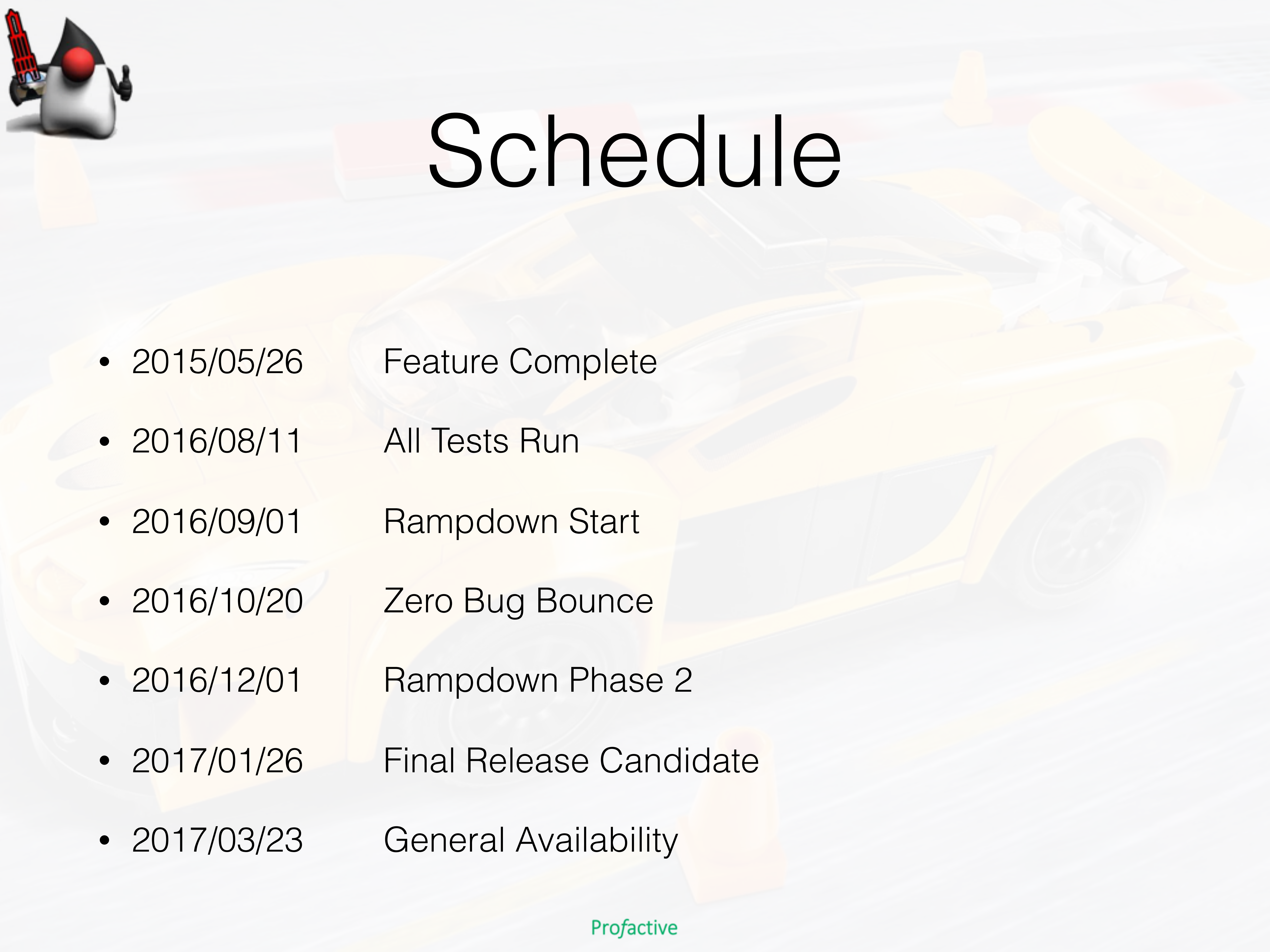


Contents - part II

- Compiler improvements & API
- Improved locking
- Variable handles
- Diagnostics
- Garbage collector
- Compact Strings
- Immutable collections
- Stack walking API
- Summary and Conclusions
- Questions

Introduction

- Java 8 introduced lambda’s and (parallel) streams
- Java 9 introduces Jigsaw
- Will life change with Java 9?
- What about performance?



Schedule

• 2015/05/26	Feature Complete
• 2016/08/11	All Tests Run
• 2016/09/01	Rampdown Start
• 2016/10/20	Zero Bug Bounce
• 2016/12/01	Rampdown Phase 2
• 2017/01/26	Final Release Candidate
• 2017/03/23	General Availability

Schedule

2015/05/26	**Feature Complete**
2016/08/11	All Tests Run
2016/09/01	Rampdown Start
2016/10/20	Zero Bug Bounce
2016/12/01	Rampdown Phase 2
2017/01/26	Final Release Candidate
2017/03/23	**General Availability**

How will life change?

- No more rt.jar, tools.jar in Java runtime
  - Tools like IntelliJ and Eclipse currently rely on it and don't run
    - Beta versions available for jdk-9
  - Modules instead: added logical layer
  - Accessible at runtime via URL:
    - jrt:/**java.base**/java/lang/String.class
- Unrecognized VM options
  - Deprecated in JDK 8, removed now: -XX:MaxPermSize

How will life change? -2

- Several Java API's not accessible anymore
  - internal, unsupported and not portable: sun.*, com.sun.*, java.awt.peer
  - jdeps from Java 8/9 helps to find static dependencies
- G1 is default collector
- '_' no longer allowed as identifier by itself
- private interface methods (instance and static)
- No more support for java -source and -target < 1.6

How will life change? -3

- Javadoc search
- Factory methods for small immutable collections
- Stack walking API
- Concurrency updates for reactive programming
- Unified GC logging
- Optimize String concatenation

Project Jigsaw goals

Project Jigsaw goals

- Make platform&JDK more easily scalable down to small computing devices;
- Improve security and maintainability
- Enable **improved application performance**; and
- Make it easier for developers to construct and maintain libraries and large applications.

Platform Module System, JSR 376 - Improved performance

- Platform, library, and application components are put in one runtime and dependencies are known
- Ahead-Of-Time and Whole-Program optimizations are more effective

Modules enable optimizations

- Known where code will be used, optimizations more feasible;
- JVM-specific memory images that load faster than class files;
    - Fast lookup of both JDK and application classes;
- early bytecode verification;
- ahead-of-time (AOT) compilation of method bodies to native code;
- the removal of unused fields, methods, and classes; and
- aggressive inlining of, e.g., lambda expressions.

Side step: inlining

```
String wrap(String a, String b) { return b + a + b; }

String getPersonText(String wrapper) {

    StringBuilder persons = new StringBuilder();

    persons.append(wrap("Brian", wrapper));

    persons.append(wrap("John", wrapper));

    return persons.toString();

}
```

- Dependent on size of method
    - default <= 35 bytes of byte code
    - 325 bytes for hot methods

Side step: inlining

```
String wrap(String a, String b) { return b + a + b; }

String getPersonText(String wrapper) {

    StringBuilder persons = new StringBuilder();

    persons.append(wrap(“Brian”, wrapper));

    persons.append(wrap(“John”, wrapper));

    return persons.toString();

}
```

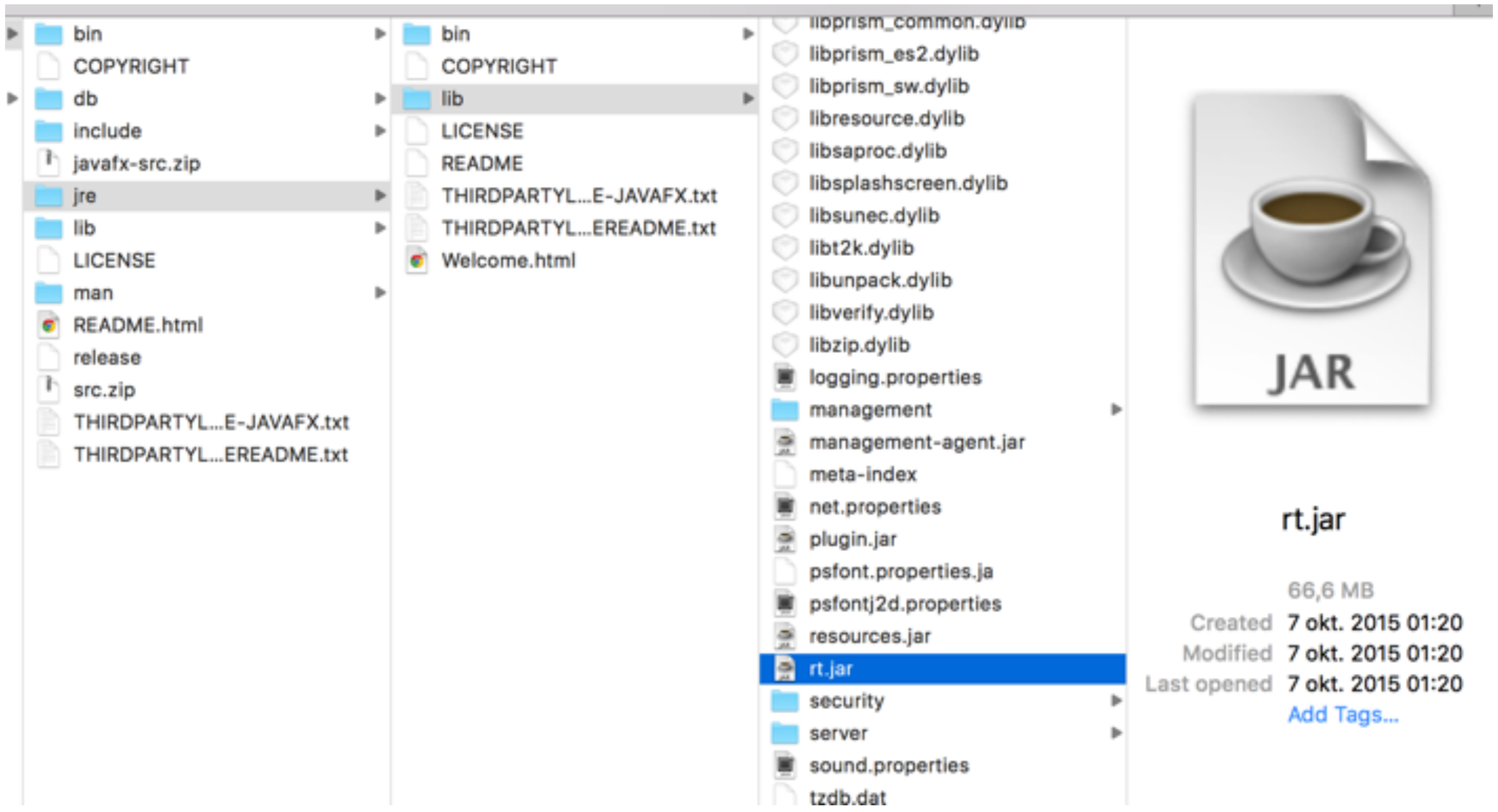
After inlining:

```
String getPersonText(String wrapper) {

    StringBuilder persons = new StringBuilder();

    persons.append(wrapper).append(“Brian”).append(wrapper));

    persons.append(wrapper).append(“John”).append(wrapper));

    return persons.toString();

}
```

Startup Performance

- Current JVM startup:
  - class loading slow: executes a linear scan of all JARs on classpath
  - Annotation detection requires to read all classes in package(s)
  - Spring: <context:component-scan base-package="your.package.name" />
  - Modules will provide a fast class-lookup, including by annotation, without reading all class files
  - Indexes created when the module is compiled

JDK 8



bin
COPYRIGHT
db
include
javafx-src.zip
jre
lib
LICENSE
man
README.html
release
src.zip
THIRDPARTYL...E-JAVAFX.txt
THIRDPARTYL...EREADME.txt

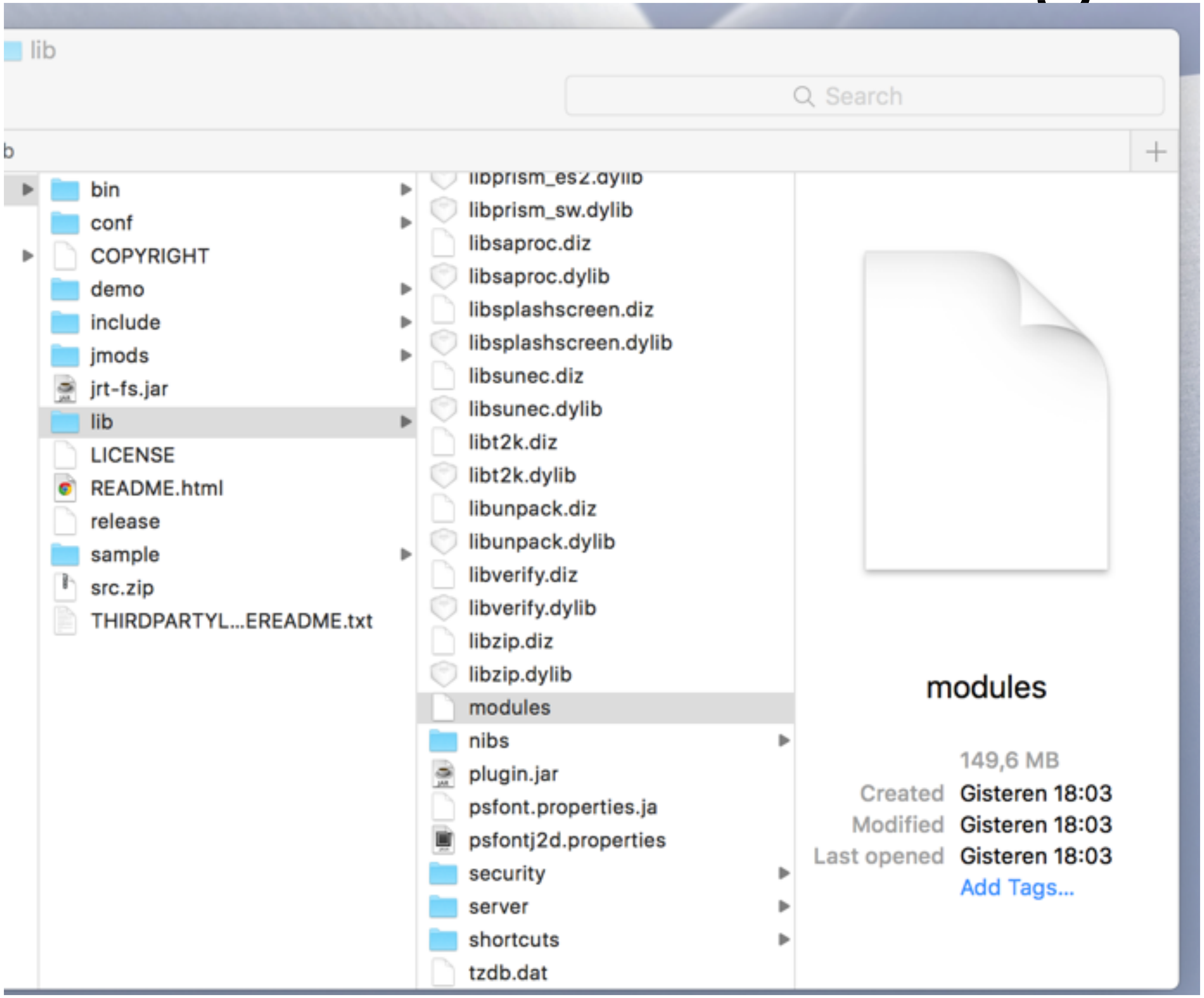
bin
COPYRIGHT
lib
LICENSE
README
THIRDPARTYL...E-JAVAFX.txt
THIRDPARTYL...EREADME.txt
Welcome.html

libprism_common.dylib
libprism_es2.dylib
libprism_sw.dylib
libresource.dylib
libsaproc.dylib
libsplashscreen.dylib
libsunec.dylib
libt2k.dylib
libunpack.dylib
libverify.dylib
libzip.dylib
logging.properties
management
management-agent.jar
meta-index
net.properties
plugin.jar
psfont.properties.ja
psfontj2d.properties
resources.jar
rt.jar
security
server
sound.properties
tzdb.dat

rt.jar

66,6 MB
Created 7 okt. 2015 01:20
Modified 7 okt. 2015 01:20
Last opened 7 okt. 2015 01:20
Add Tags...

Modular Java - JEP 220: Modular Run-Time Images



lib

Search

bin
conf
COPYRIGHT
demo
include
jmods
jrt-fs.jar
lib
LICENSE
README.html
release
sample
src.zip
THIRDPARTYL...EREADME.txt

libprism_es2.dylib
libprism_sw.dylib
libsaproc.diz
libsaproc.dylib
libsplashscreen.diz
libsplashscreen.dylib
libsunec.diz
libsunec.dylib
libt2k.diz
libt2k.dylib
libunpack.diz
libunpack.dylib
libverify.diz
libverify.dylib
libzip.diz
libzip.dylib
modules
nibs
plugin.jar
psfont.properties.ja
psfontj2d.properties
security
server
shortcuts
tzdb.dat

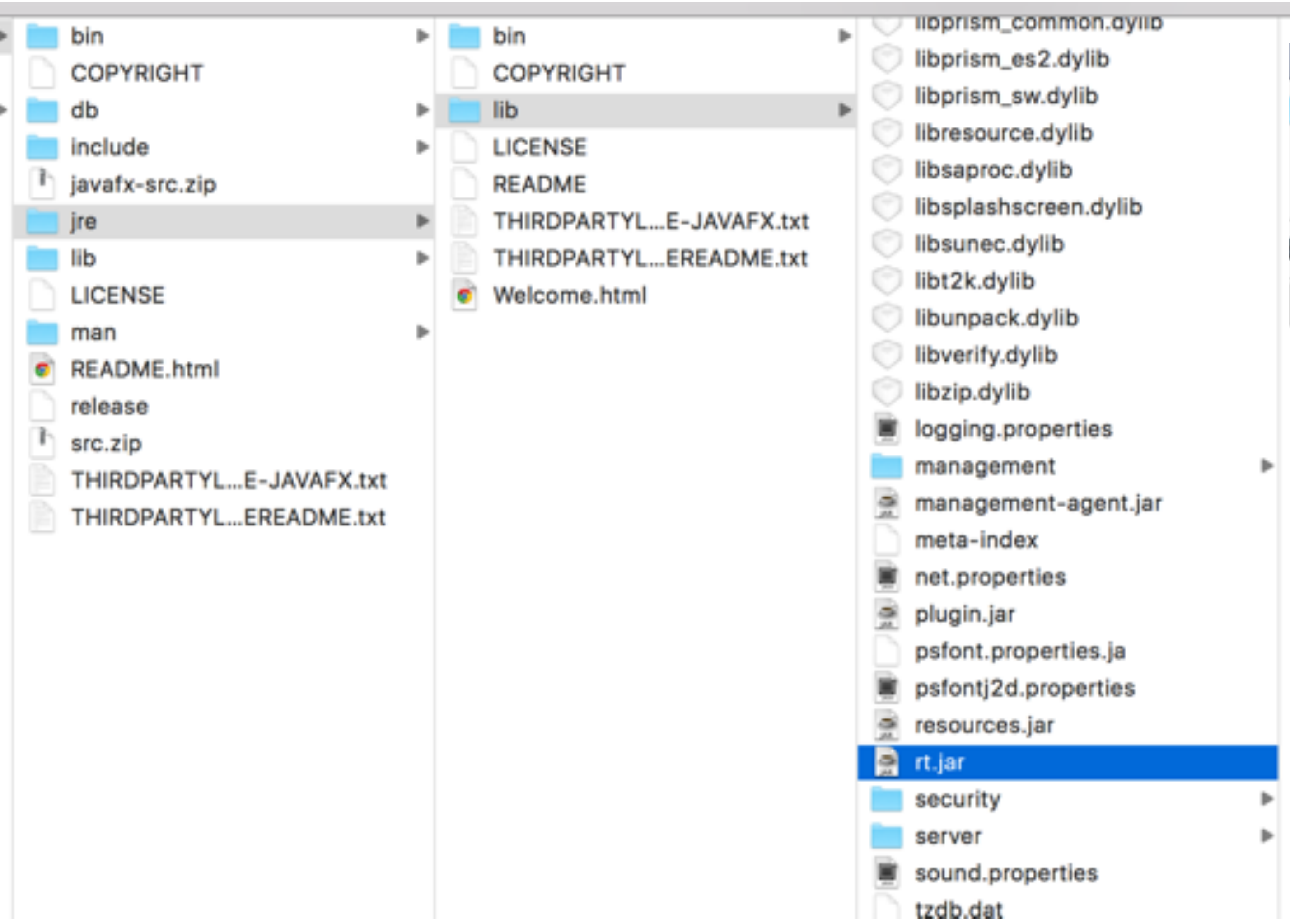
modules

149,6 MB

Created	Gisteren 18:03
Modified	Gisteren 18:03
Last opened	Gisteren 18:03

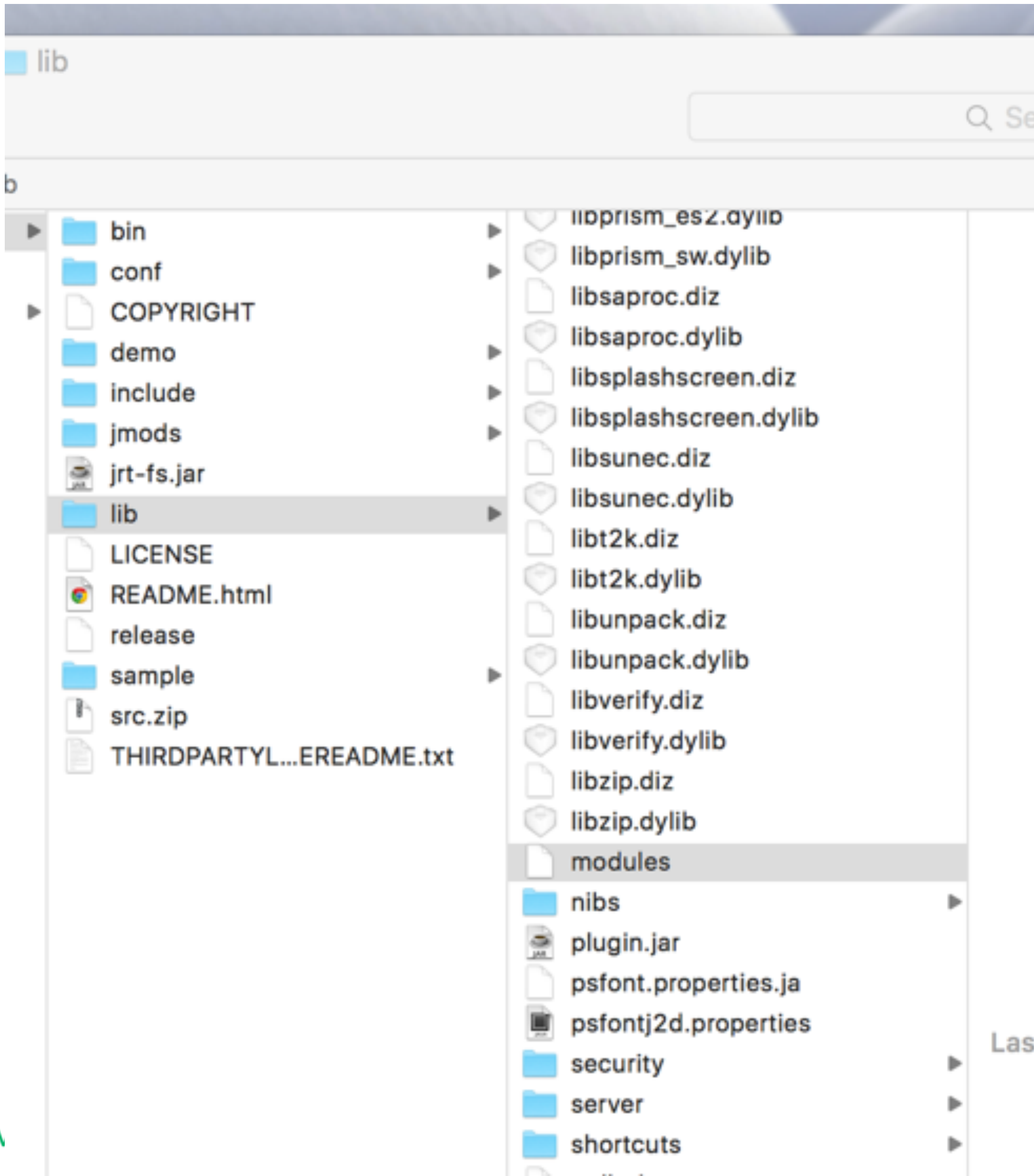
Add Tags...

Modular Java - JEP 220: Modular Run-Time Images



```
bin                      bin                          libprism_common.dylib
COPYRIGHT                COPYRIGHT                    libprism_es2.dylib
db                       lib                          libprism_sw.dylib
include                  LICENSE                      libresource.dylib
javafx-src.zip           README                       libsaproc.dylib
jre                      THIRDPARTYL...E-JAVAFX.txt   libsplashscreen.dylib
lib                      THIRDPARTYL...EREADME.txt    libsunec.dylib
LICENSE                  Welcome.html                 libt2k.dylib
man                                                   libunpack.dylib
README.html                                           libverify.dylib
release                                               libzip.dylib
src.zip                                               logging.properties
THIRDPARTYL...E-JAVAFX.txt                            management
THIRDPARTYL...EREADME.txt                             management-agent.jar
                                                      meta-index
                                                      net.properties
                                                      plugin.jar
                                                      psfont.properties.ja
                                                      psfontj2d.properties
                                                      resources.jar
                                                      rt.jar
                                                      security
                                                      server
                                                      sound.properties
                                                      tzdb.dat
```

Profacti

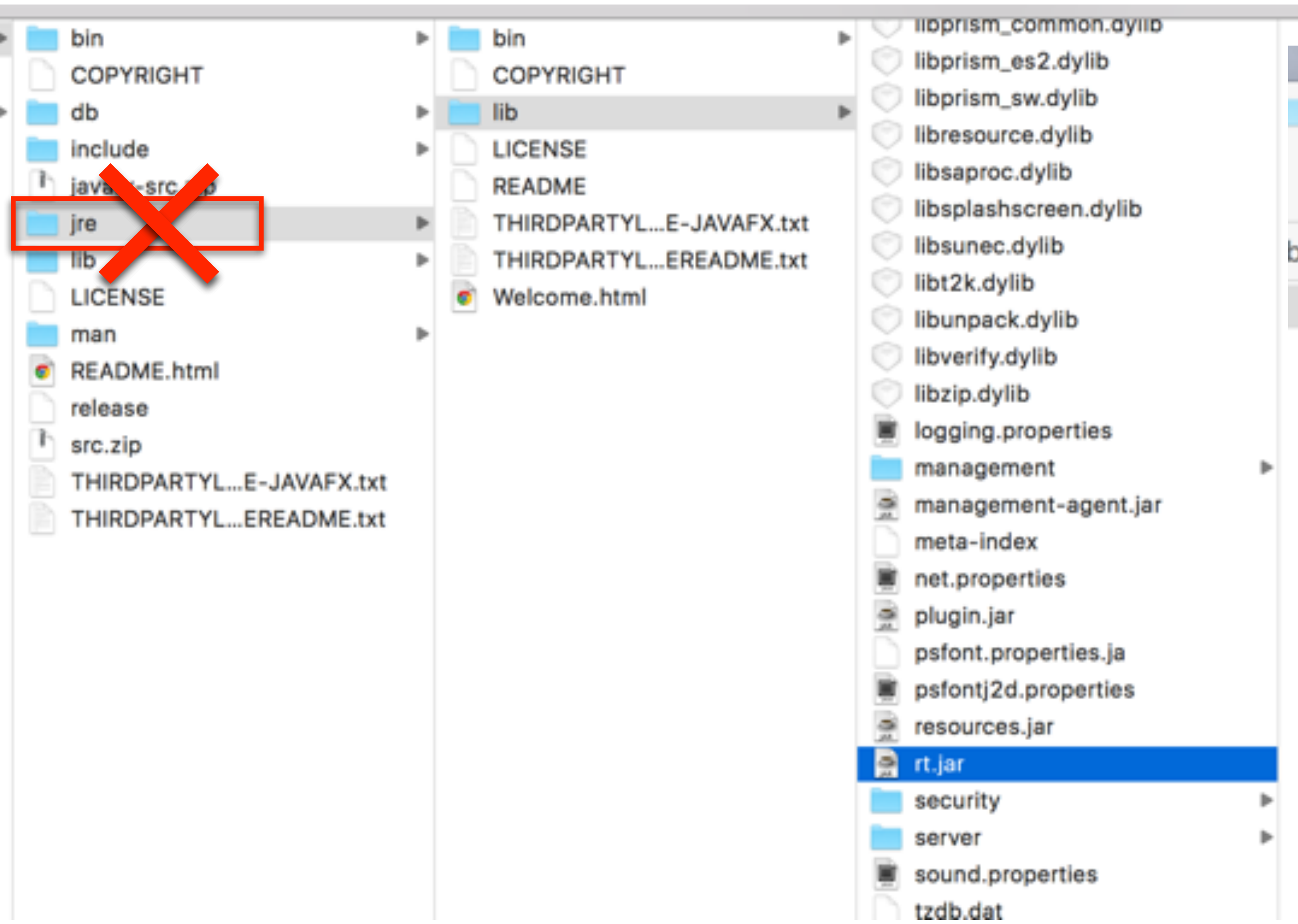


lib

```
bin                          libprism_es2.dylib
conf                         libprism_sw.dylib
COPYRIGHT                    libsaproc.diz
demo                         libsaproc.dylib
include                      libsplashscreen.diz
jmods                        libsplashscreen.dylib
jrt-fs.jar                   libsunec.diz
lib                          libsunec.dylib
LICENSE                      libt2k.diz
README.html                  libt2k.dylib
release                      libunpack.diz
sample                       libunpack.dylib
src.zip                      libverify.diz
THIRDPARTYL...EREADME.txt    libverify.dylib
                             libzip.diz
                             libzip.dylib
                             modules
                             nibs
                             plugin.jar
                             psfont.properties.ja
                             psfontj2d.properties
                             security
                             server
                             shortcuts
```

Las

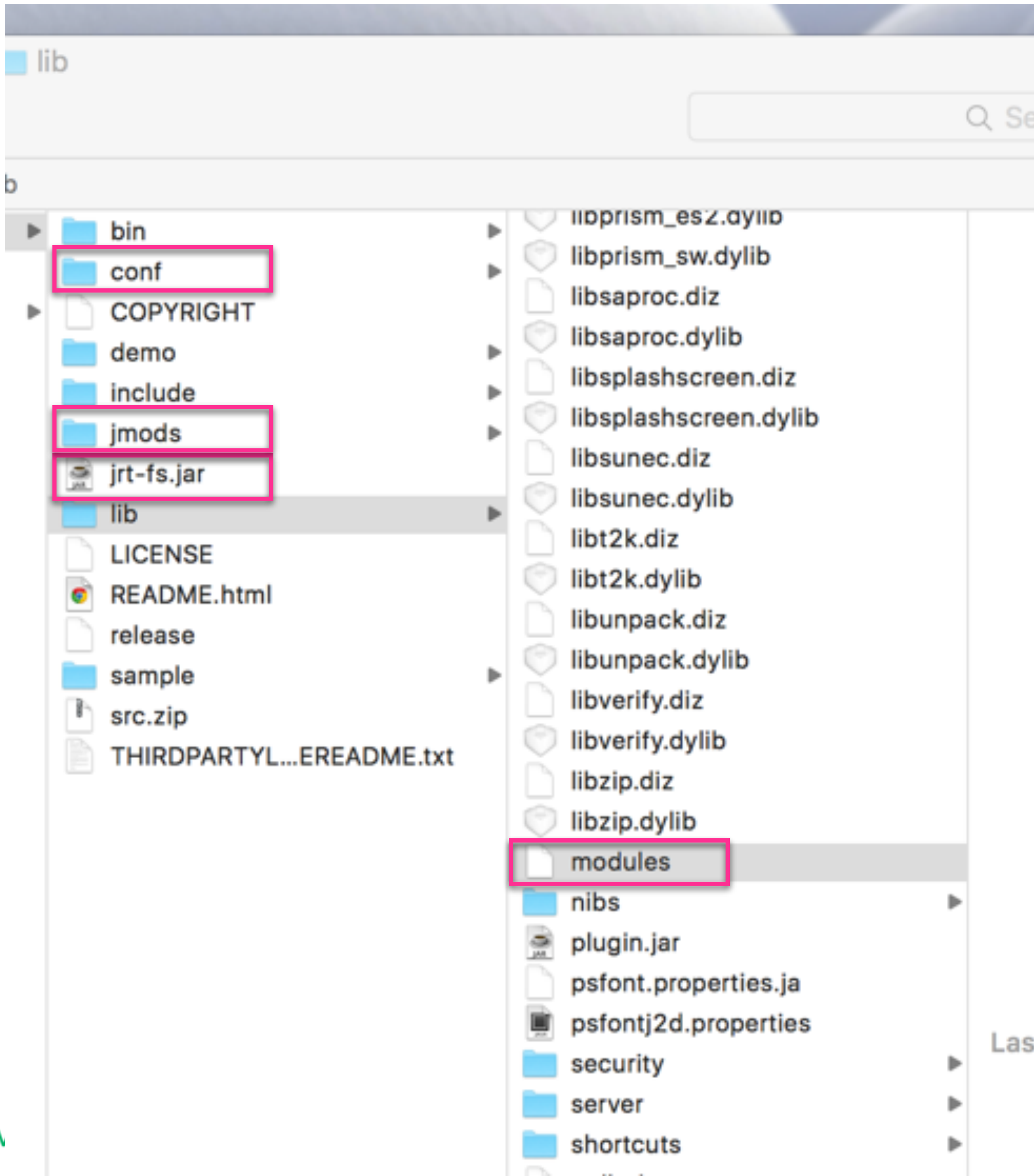
Modular Java - JEP 220: Modular Run-Time Images



bin
COPYRIGHT
db
include
javafx-src.zip
jre
lib
LICENSE
man
README.html
release
src.zip
THIRDPARTYL...E-JAVAFX.txt
THIRDPARTYL...EREADME.txt

bin
COPYRIGHT
lib
LICENSE
README
THIRDPARTYL...E-JAVAFX.txt
THIRDPARTYL...EREADME.txt
Welcome.html

libprism_common.dylib
libprism_es2.dylib
libprism_sw.dylib
libresource.dylib
libsaproc.dylib
libsplashscreen.dylib
libsunec.dylib
libt2k.dylib
libunpack.dylib
libverify.dylib
libzip.dylib
logging.properties
management
management-agent.jar
meta-index
net.properties
plugin.jar
psfont.properties.ja
psfontj2d.properties
resources.jar
rt.jar
security
server
sound.properties
tzdb.dat

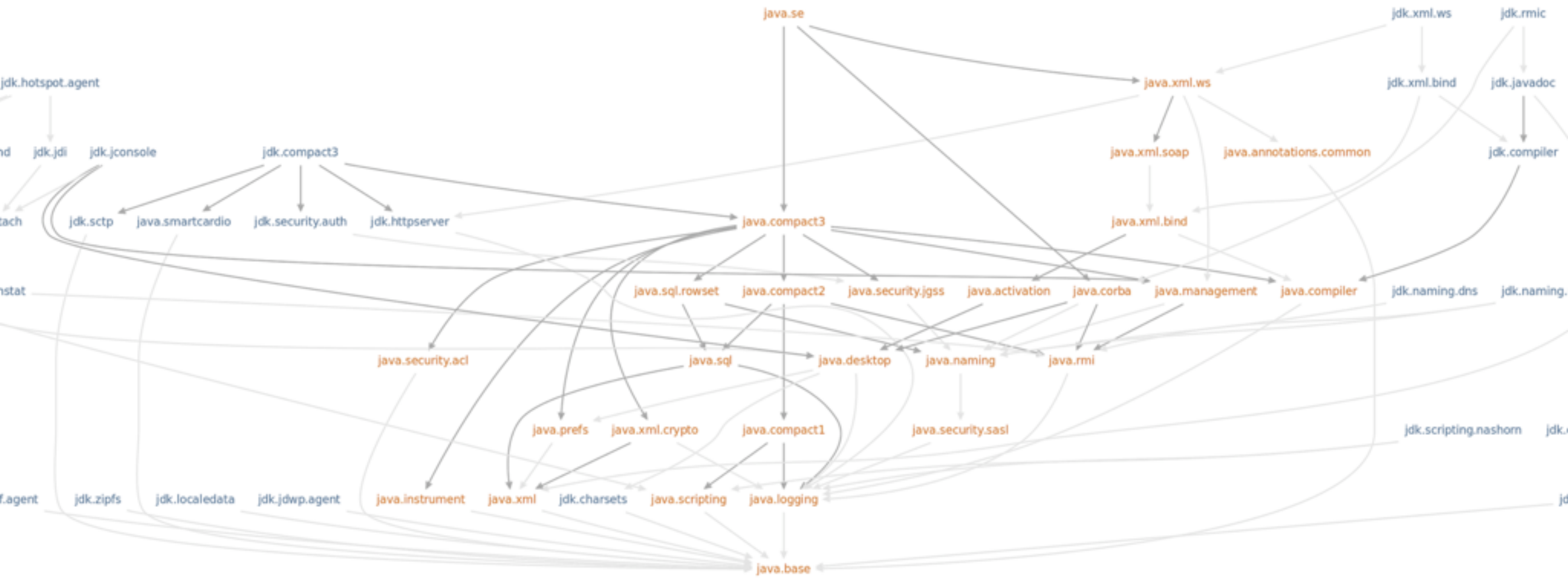


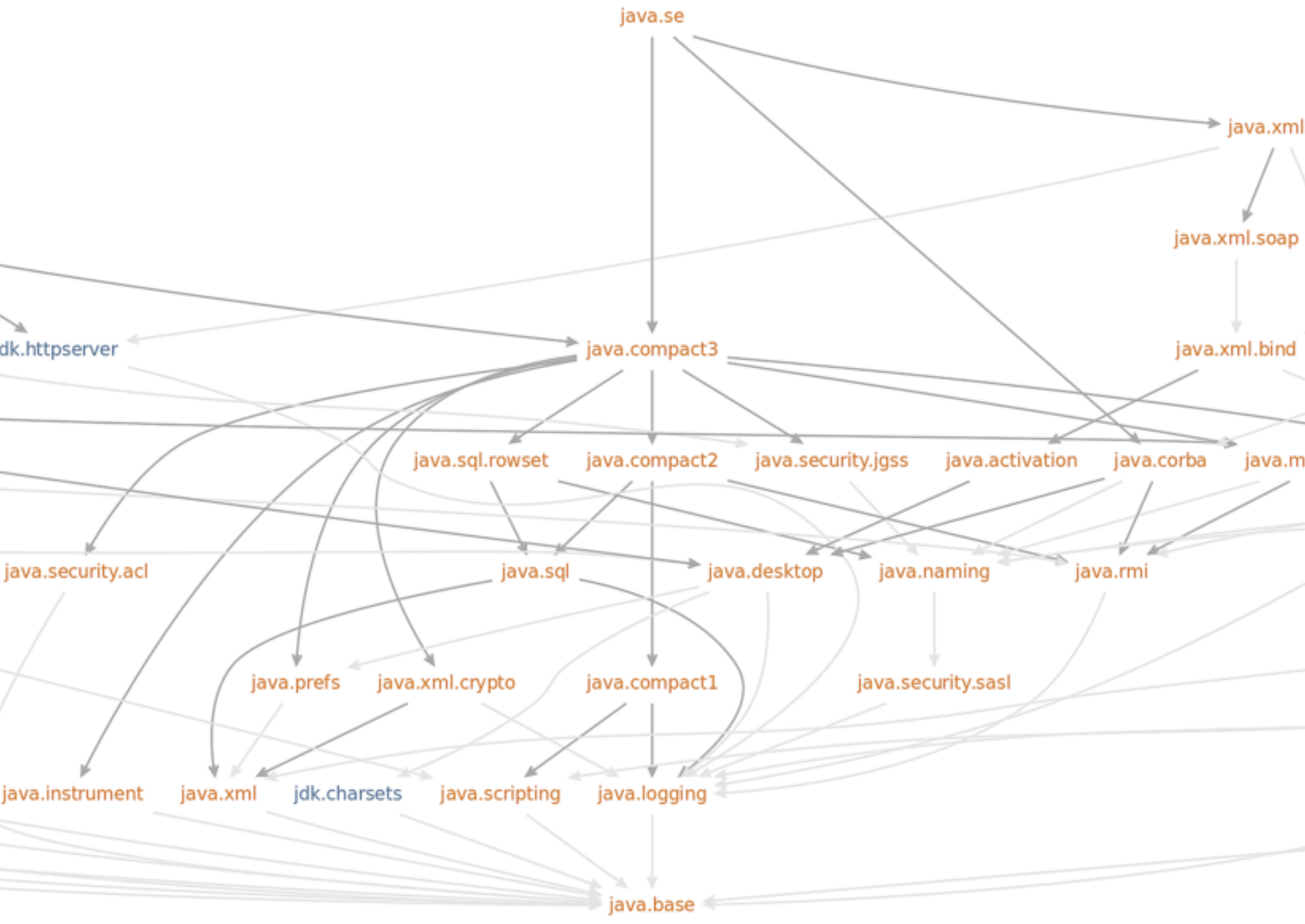
bin
conf
COPYRIGHT
demo
include
jmods
jrt-fs.jar
lib
LICENSE
README.html
release
sample
src.zip
THIRDPARTYL...EREADME.txt

libprism_es2.dylib
libprism_sw.dylib
libsaproc.diz
libsaproc.dylib
libsplashscreen.diz
libsplashscreen.dylib
libsunec.diz
libsunec.dylib
libt2k.diz
libt2k.dylib
libunpack.diz
libunpack.dylib
libverify.diz
libverify.dylib
libzip.diz
libzip.dylib
modules
nibs
plugin.jar
psfont.properties.ja
psfontj2d.properties
security
server
shortcuts

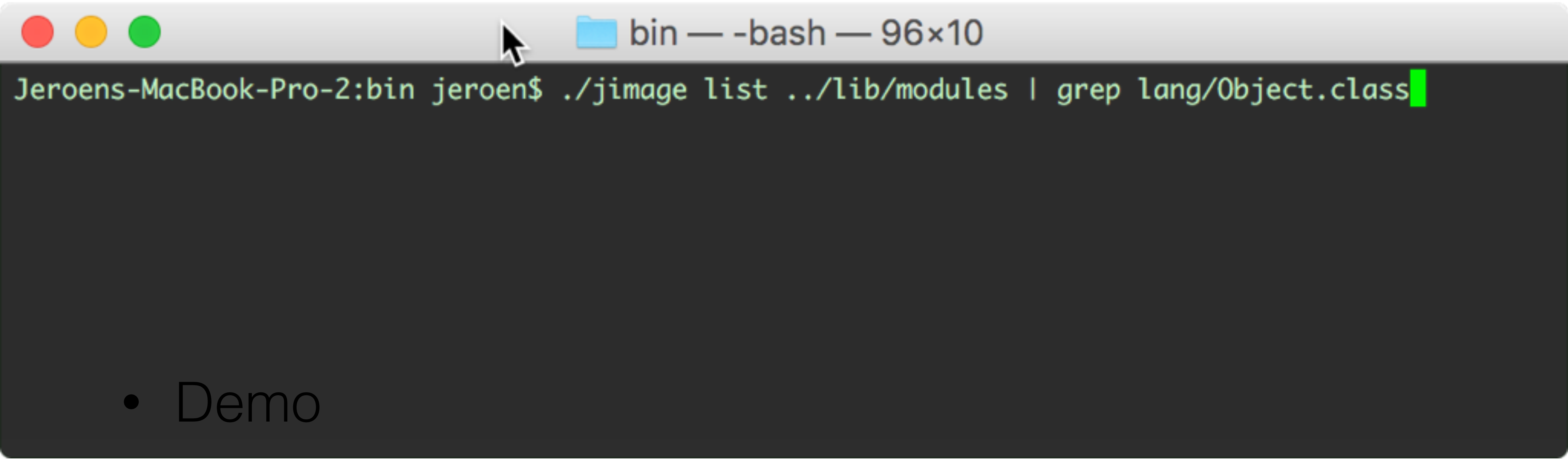
Profactiv

JDK Module Graph




java.se
java.xml
java.xml.soap
dk.httpserver
java.compact3
java.xml.bind
java.sql.rowset
java.compact2
java.security.jgss
java.activation
java.corba
java.m
java.security.acl
java.sql
java.desktop
java.naming
java.rmi
java.prefs
java.xml.crypto
java.compact1
java.security.sasl
java.instrument
java.xml
jdk.charsets
java.scripting
java.logging
java.base

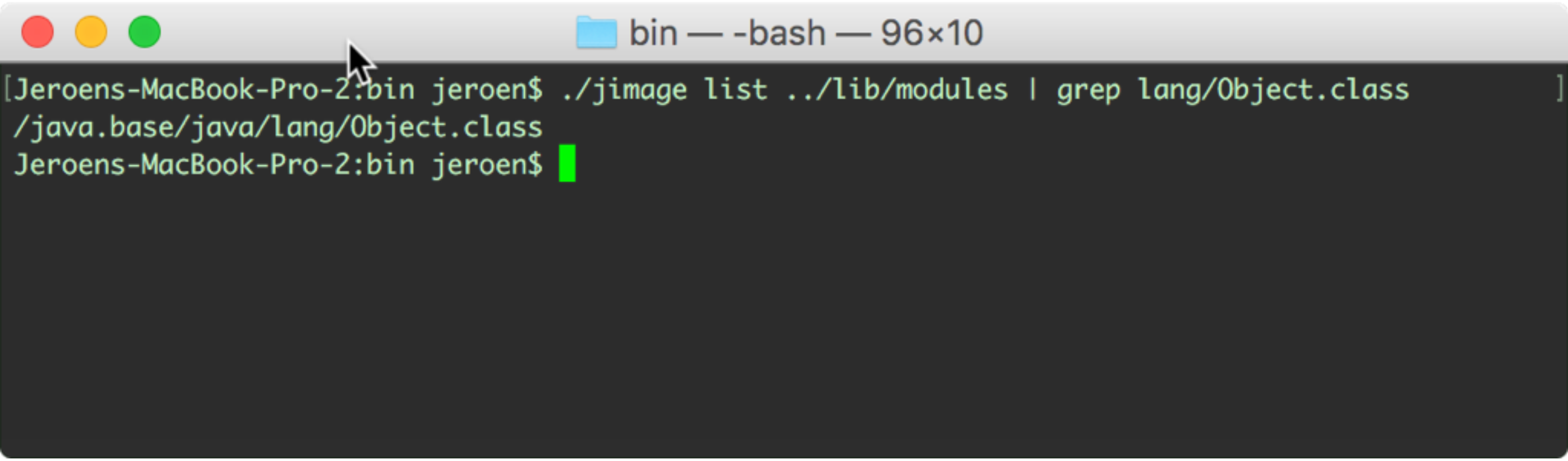
inside the modules file - jimage tool



```
Jeroens-MacBook-Pro-2:bin jeroen$ ./jimage list ../lib/modules | grep lang/Object.class
```

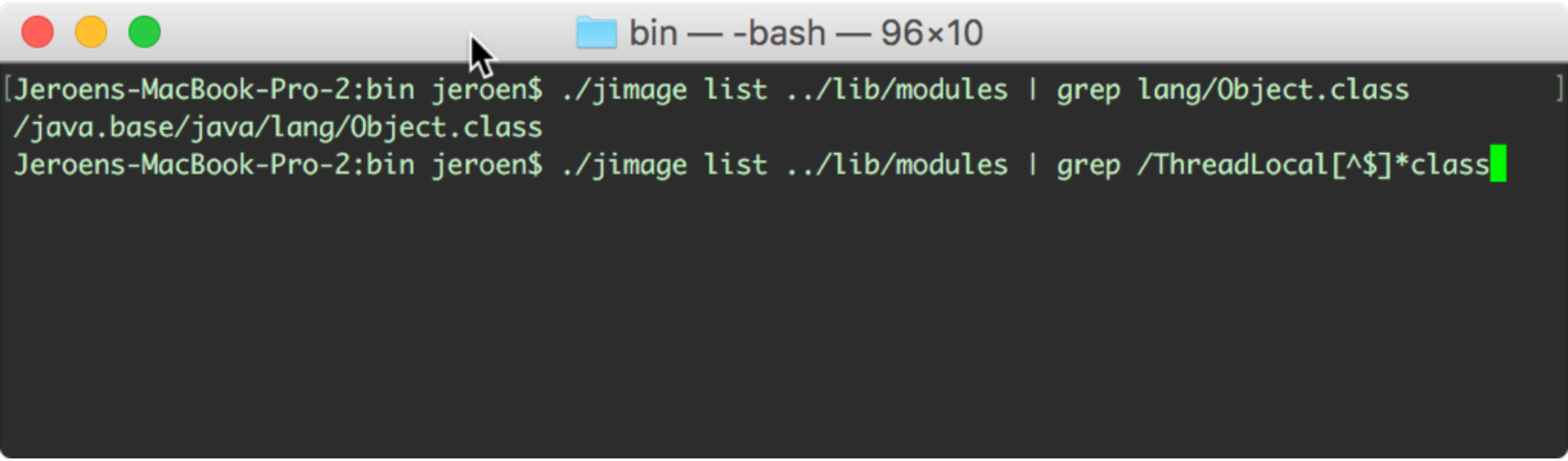
- Demo

jimage tool



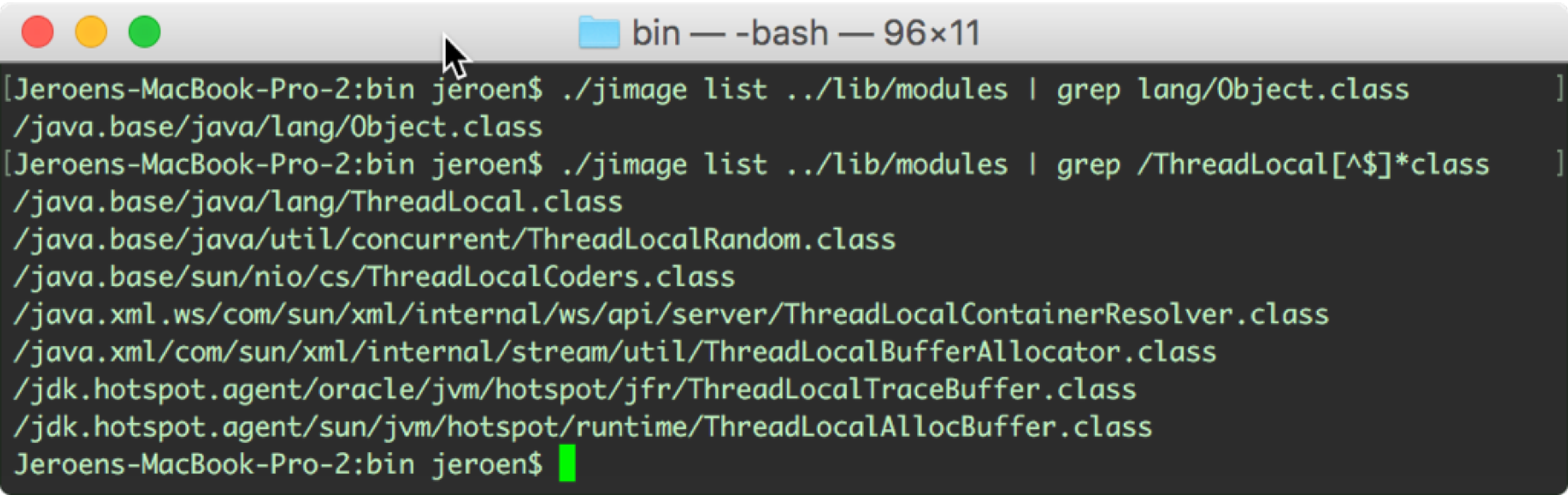
```
[Jeroens-MacBook-Pro-2:bin jeroen$ ./jimage list ../lib/modules | grep lang/Object.class        ]
/java.base/java/lang/Object.class
Jeroens-MacBook-Pro-2:bin jeroen$ 
```

jimage tool



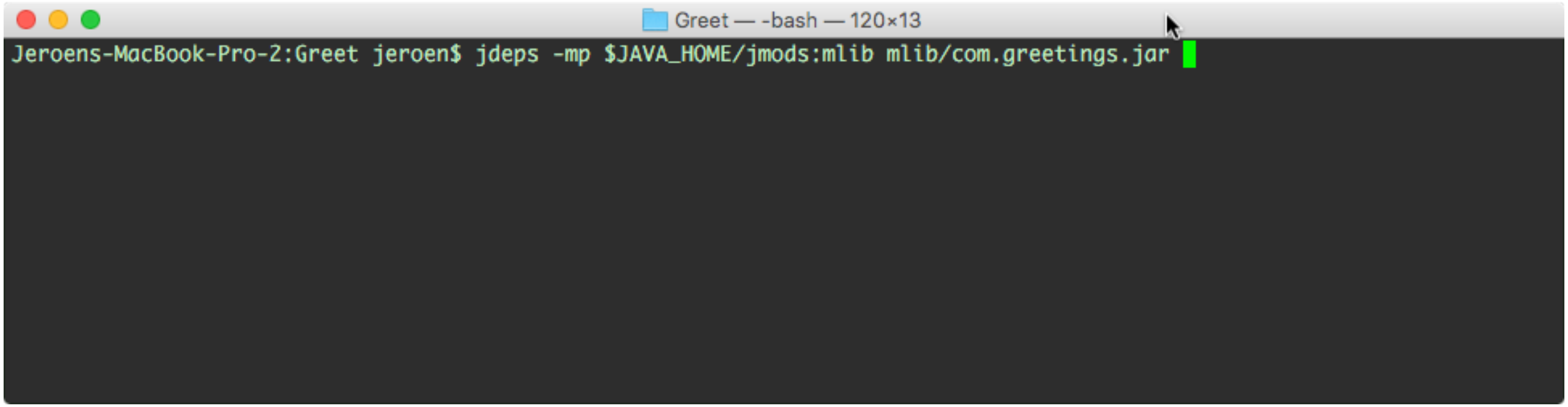
```
[Jeroens-MacBook-Pro-2:bin jeroen$ ./jimage list ../lib/modules | grep lang/Object.class      ]
/java.base/java/lang/Object.class
Jeroens-MacBook-Pro-2:bin jeroen$ ./jimage list ../lib/modules | grep /ThreadLocal[^$]*class
```

jimage tool



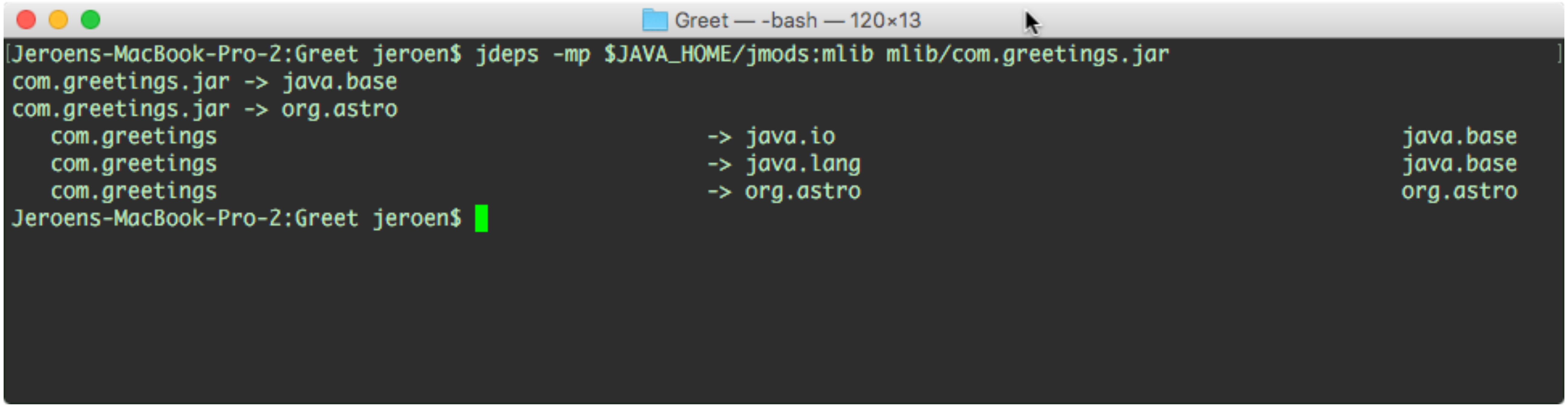
```
Jeroens-MacBook-Pro-2:bin jeroen$ ./jimage list ../lib/modules | grep lang/Object.class
/java.base/java/lang/Object.class
Jeroens-MacBook-Pro-2:bin jeroen$ ./jimage list ../lib/modules | grep /ThreadLocal[^$]*class
/java.base/java/lang/ThreadLocal.class
/java.base/java/util/concurrent/ThreadLocalRandom.class
/java.base/sun/nio/cs/ThreadLocalCoders.class
/java.xml.ws/com/sun/xml/internal/ws/api/server/ThreadLocalContainerResolver.class
/java.xml/com/sun/xml/internal/stream/util/ThreadLocalBufferAllocator.class
/jdk.hotspot.agent/oracle/jvm/hotspot/jfr/ThreadLocalTraceBuffer.class
/jdk.hotspot.agent/sun/jvm/hotspot/runtime/ThreadLocalAllocBuffer.class
Jeroens-MacBook-Pro-2:bin jeroen$ 
```

jdeps tool



```
Jeroens-MacBook-Pro-2:Greet jeroen$ jdeps -mp $JAVA_HOME/jmods:mlib mlib/com.greetings.jar
```

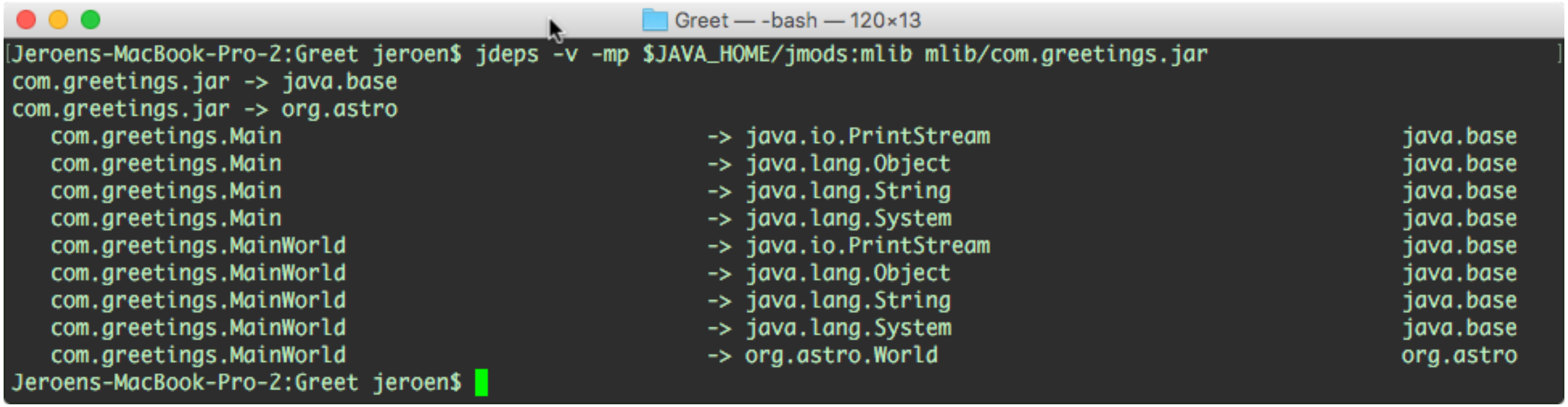
jdeps tool



Greet — -bash — 120×13

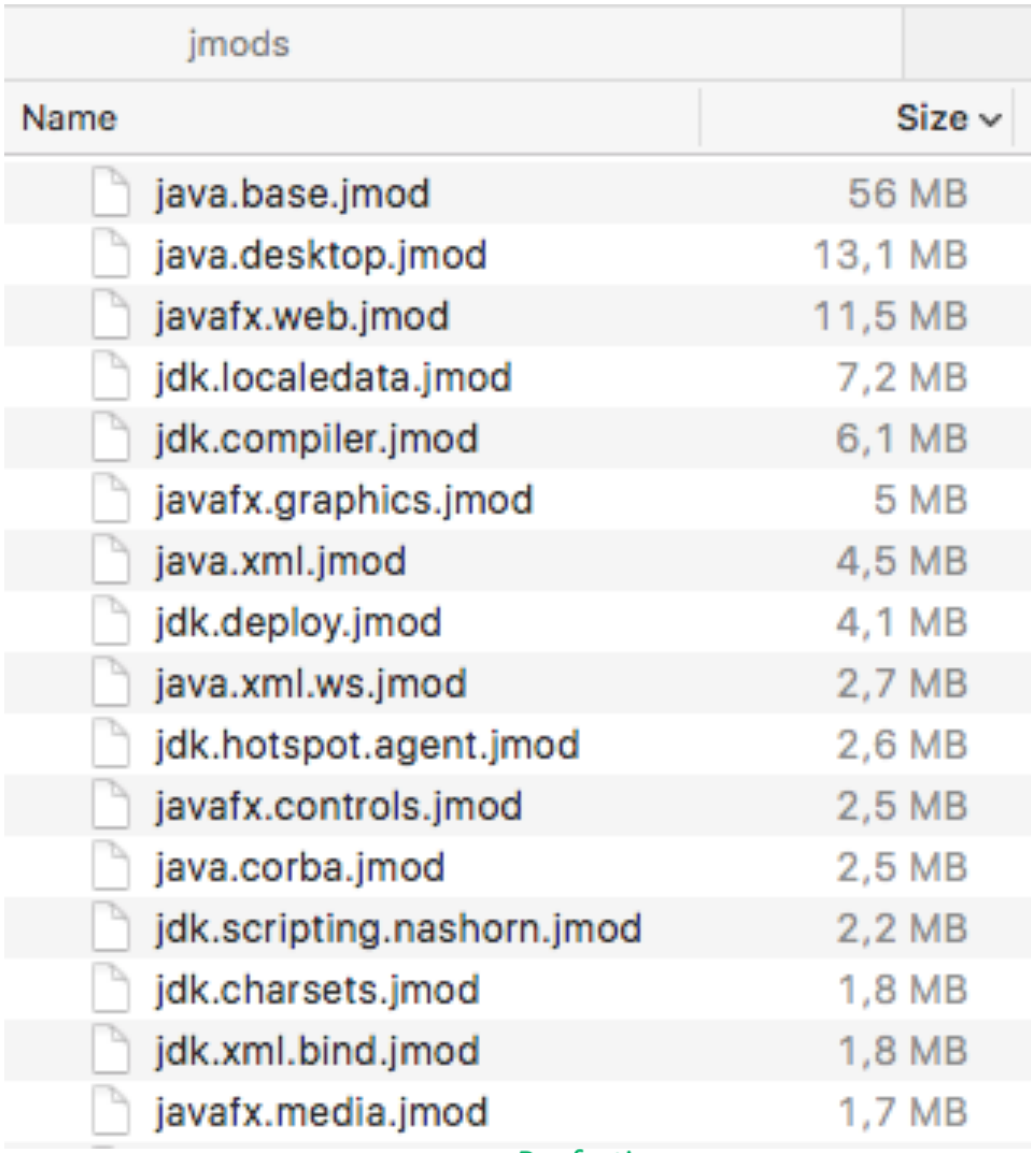
```
Jeroens-MacBook-Pro-2:Greet jeroen$ jdeps -mp $JAVA_HOME/jmods:mlib mlib/com.greetings.jar
com.greetings.jar -> java.base
com.greetings.jar -> org.astro
   com.greetings                                      -> java.io                                            java.base
   com.greetings                                      -> java.lang                                          java.base
   com.greetings                                      -> org.astro                                          org.astro
Jeroens-MacBook-Pro-2:Greet jeroen$
```

jdeps tool


Greet — -bash — 120×13

```
Jeroens-MacBook-Pro-2:Greet jeroen$ jdeps -v -mp $JAVA_HOME/jmods:mlib mlib/com.greetings.jar
com.greetings.jar -> java.base
com.greetings.jar -> org.astro
   com.greetings.Main                                 -> java.io.PrintStream                                java.base
   com.greetings.Main                                 -> java.lang.Object                                   java.base
   com.greetings.Main                                 -> java.lang.String                                   java.base
   com.greetings.Main                                 -> java.lang.System                                   java.base
   com.greetings.MainWorld                            -> java.io.PrintStream                                java.base
   com.greetings.MainWorld                            -> java.lang.Object                                   java.base
   com.greetings.MainWorld                            -> java.lang.String                                   java.base
   com.greetings.MainWorld                            -> java.lang.System                                   java.base
   com.greetings.MainWorld                            -> org.astro.World                                    org.astro
Jeroens-MacBook-Pro-2:Greet jeroen$
```

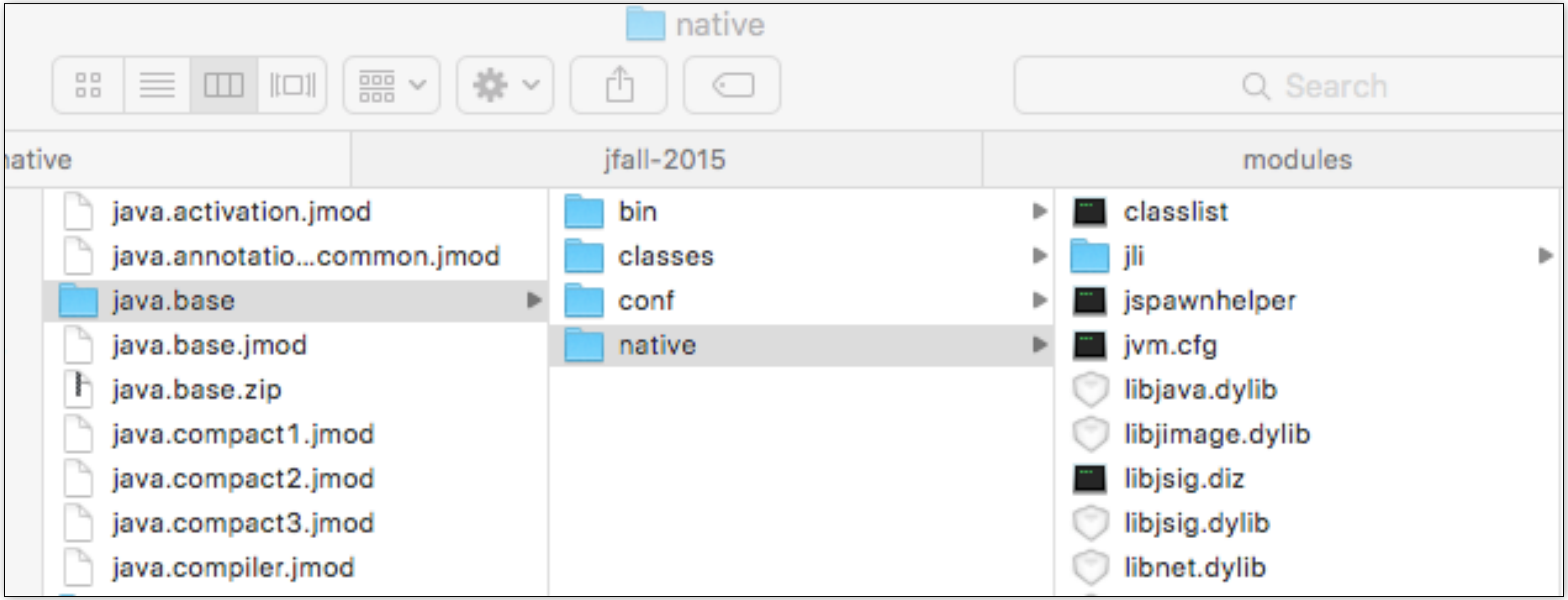
Packaging: JMOD files



jmods

Name	Size
java.base.jmod	56 MB
java.desktop.jmod	13,1 MB
javafx.web.jmod	11,5 MB
jdk.localedata.jmod	7,2 MB
jdk.compiler.jmod	6,1 MB
javafx.graphics.jmod	5 MB
java.xml.jmod	4,5 MB
jdk.deploy.jmod	4,1 MB
java.xml.ws.jmod	2,7 MB
jdk.hotspot.agent.jmod	2,6 MB
javafx.controls.jmod	2,5 MB
java.corba.jmod	2,5 MB
jdk.scripting.nashorn.jmod	2,2 MB
jdk.charsets.jmod	1,8 MB
jdk.xml.bind.jmod	1,8 MB
javafx.media.jmod	1,7 MB

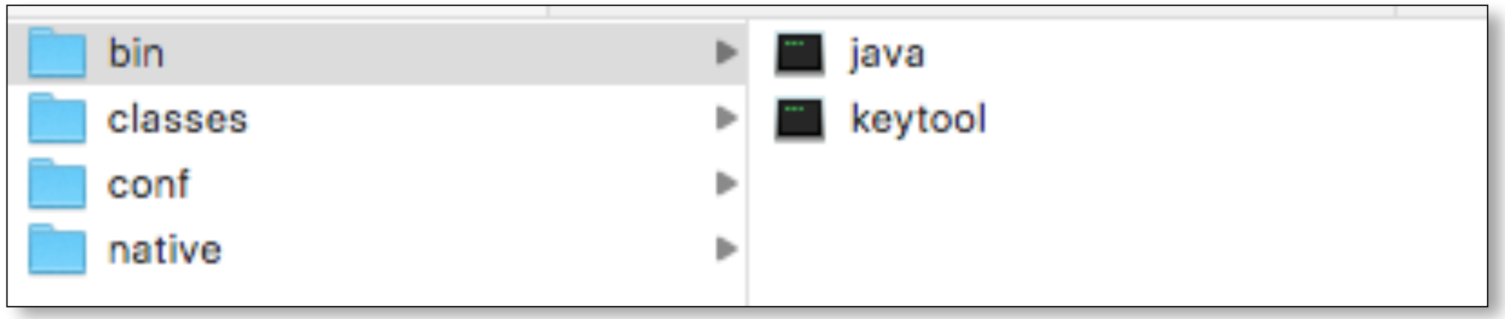
jmod = jar++ for compile and link time



java.activation.jmod
java.annotatio...common.jmod
java.base
java.base.jmod
java.base.zip
java.compact1.jmod
java.compact2.jmod
java.compact3.jmod
java.compiler.jmod

bin
classes
conf
native

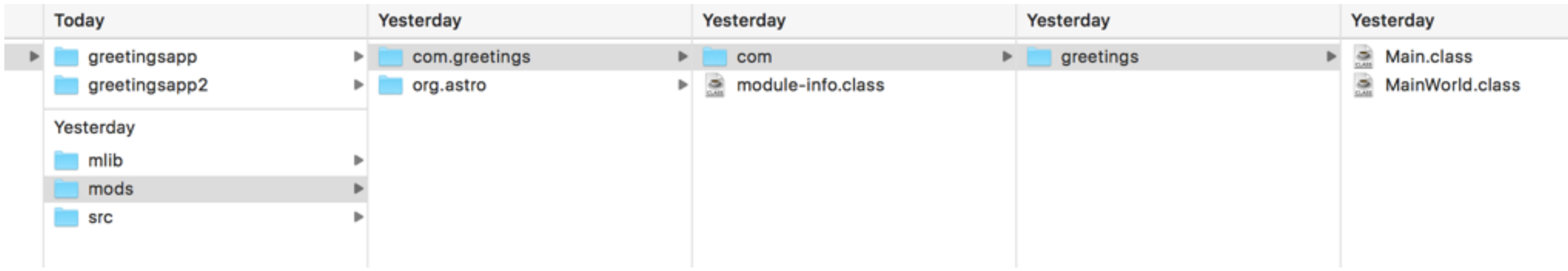
classlist
jli
jspawnhelper
jvm.cfg
libjava.dylib
libjimage.dylib
libjsig.diz
libjsig.dylib
libnet.dylib

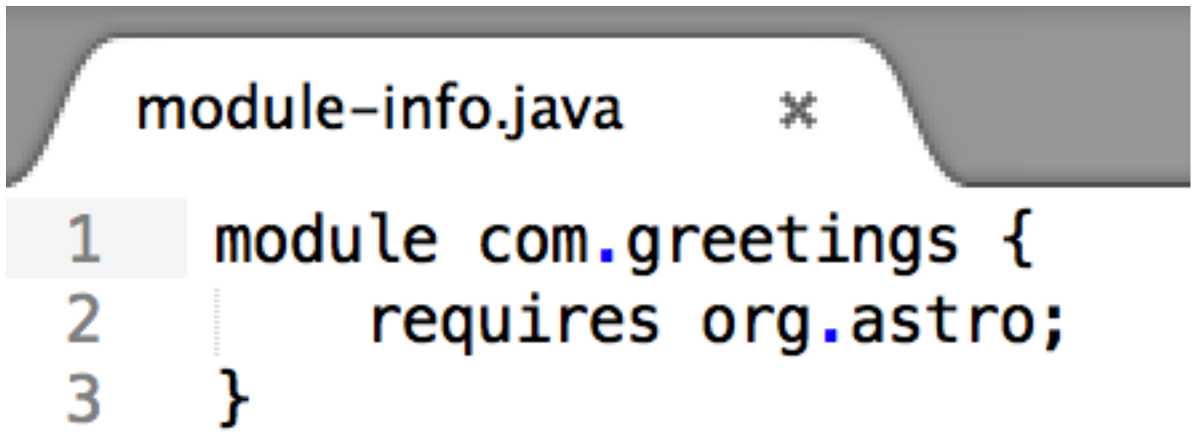


bin
classes
conf
native

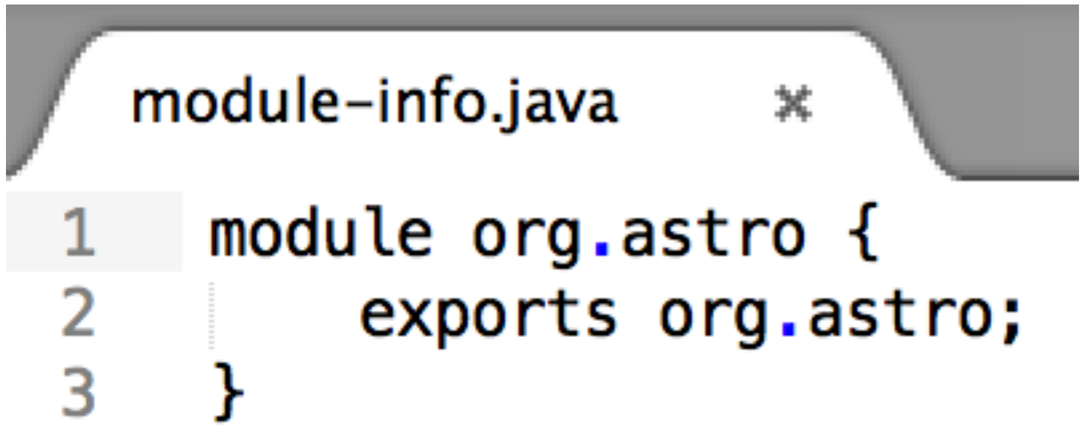
java
keytool

mods of project



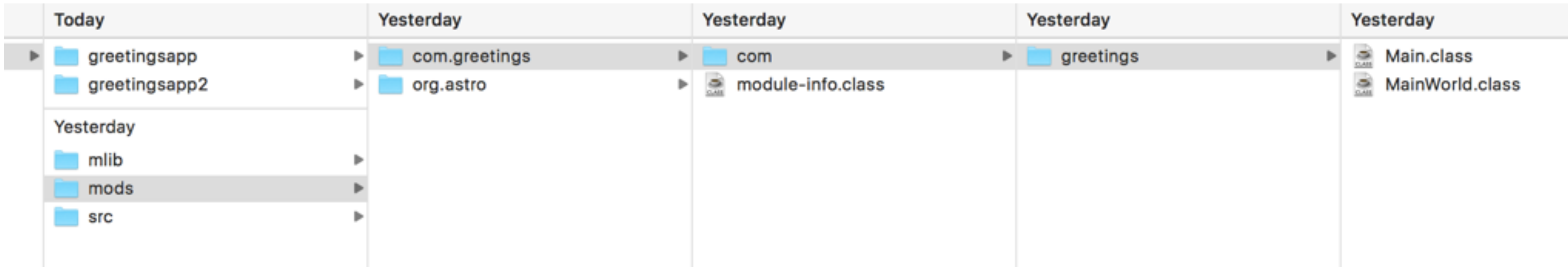


```
module com.greetings {
    requires org.astro;
}
```



```
module org.astro {
    exports org.astro;
}
```

mods of project

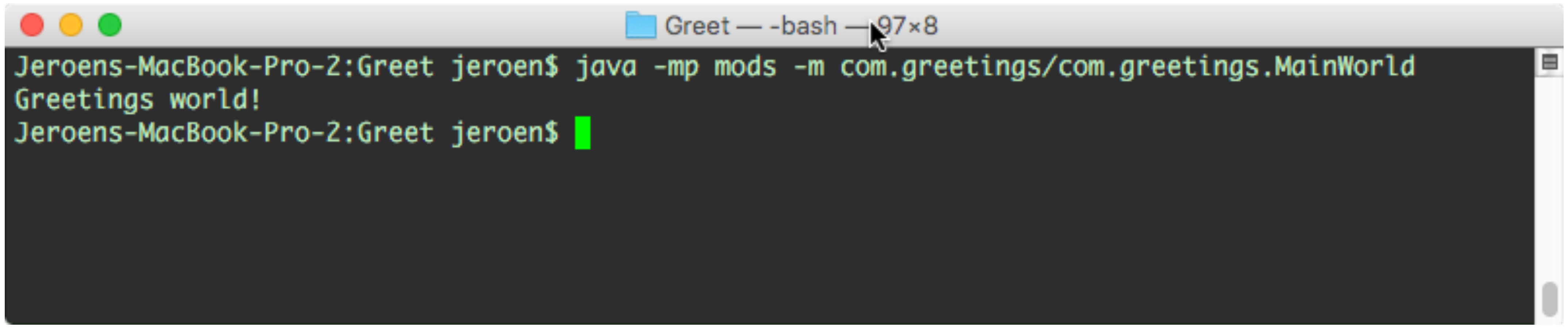

Today
greetingsapp
greetingsapp2
Yesterday
mlib
mods
src

Yesterday
com.greetings
org.astro

Yesterday
com
module-info.class

Yesterday
greetings

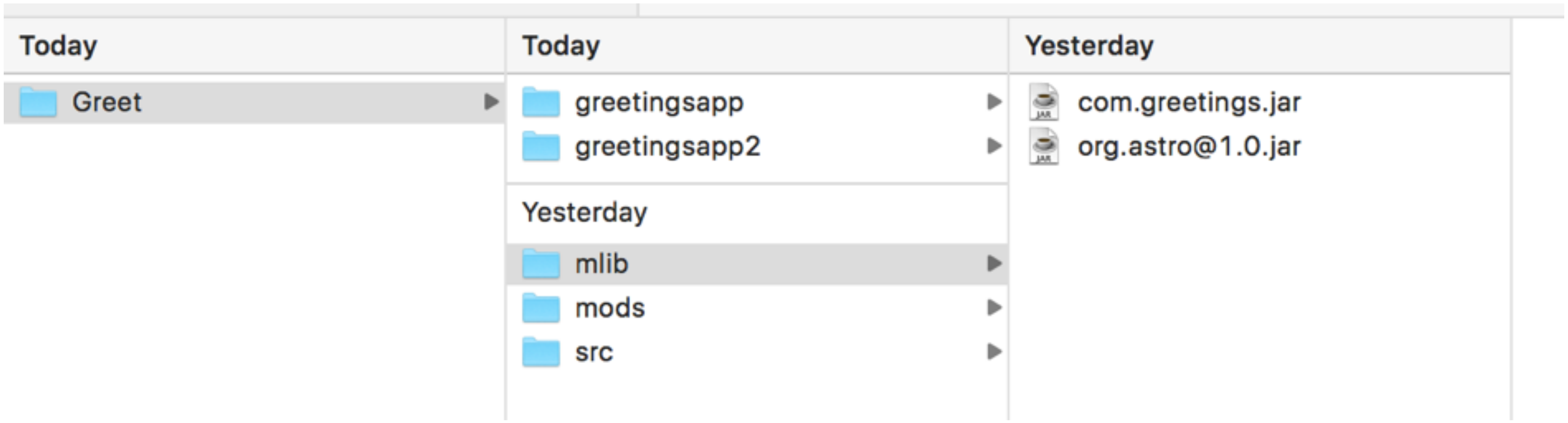
Yesterday
Main.class
MainWorld.class

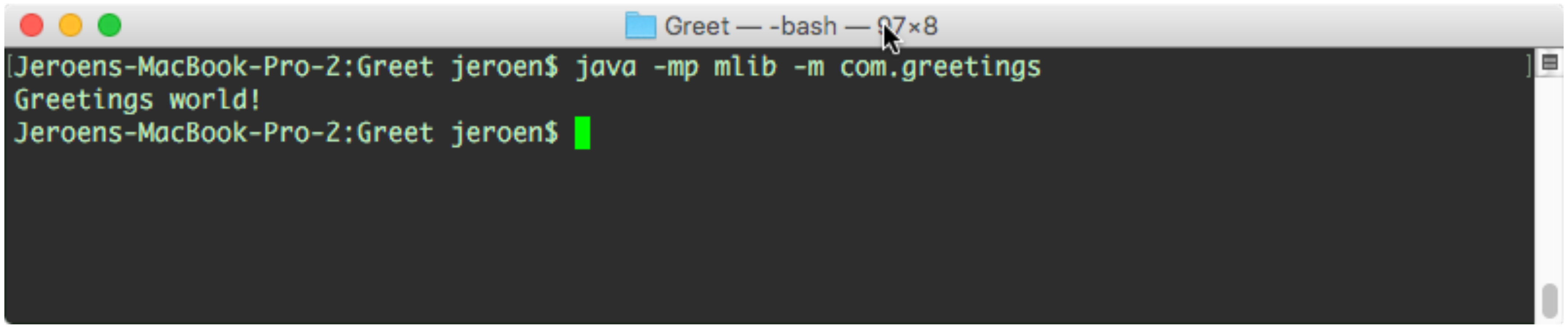

Greet — -bash — 97×8

```
Jeroens-MacBook-Pro-2:Greet jeroen$ java -mp mods -m com.greetings/com.greetings.MainWorld
Greetings world!
Jeroens-MacBook-Pro-2:Greet jeroen$ 
```

Profactive

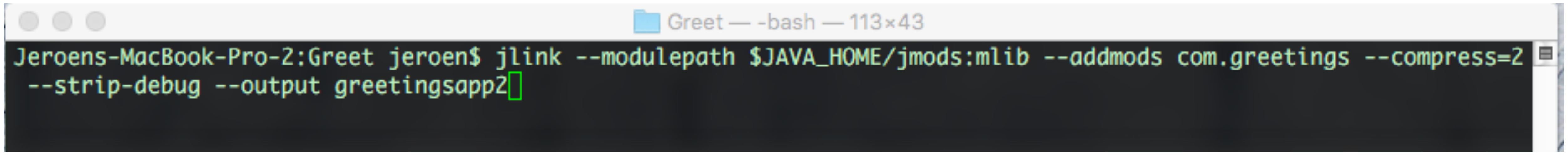
packaging: modular jars





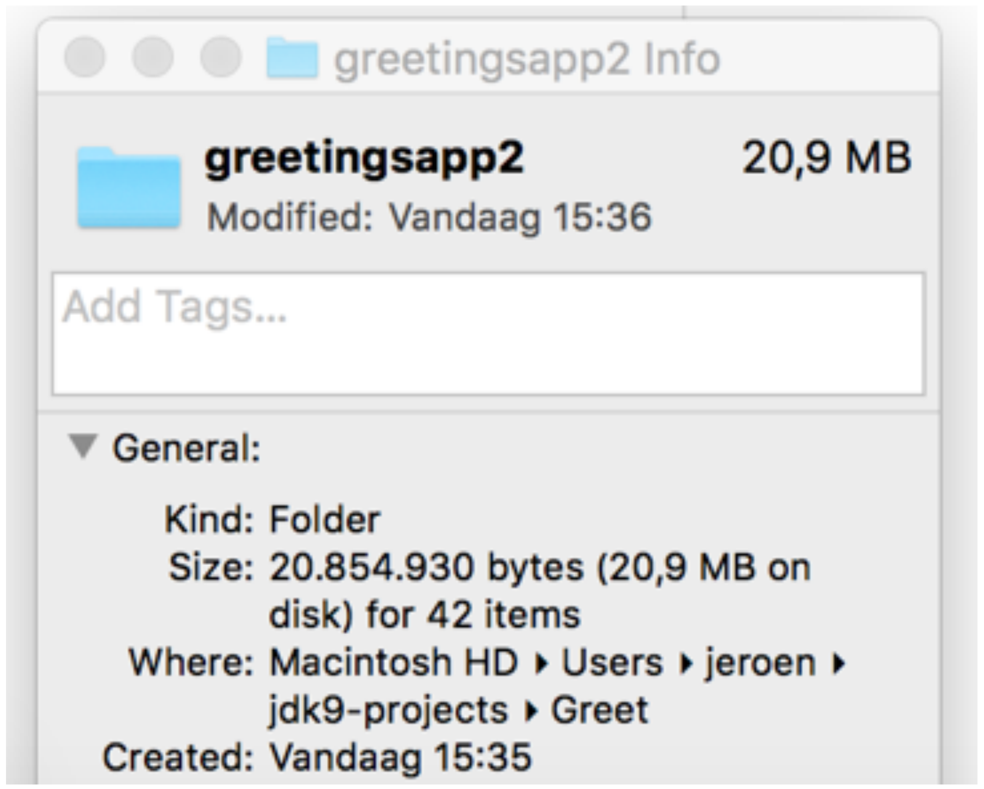
```
Jeroens-MacBook-Pro-2:Greet jeroen$ java -mp mlib -m com.greetings
Greetings world!
Jeroens-MacBook-Pro-2:Greet jeroen$
```

jlink



```
Jeroens-MacBook-Pro-2:Greet jeroen$ jlink --modulepath $JAVA_HOME/jmods:mlib --addmods com.greetings --compress=2
 --strip-debug --output greetingsapp2
```

- Small size output image:
  - 70 MB with native debug files
  - 21 MB without, can be 12 MB



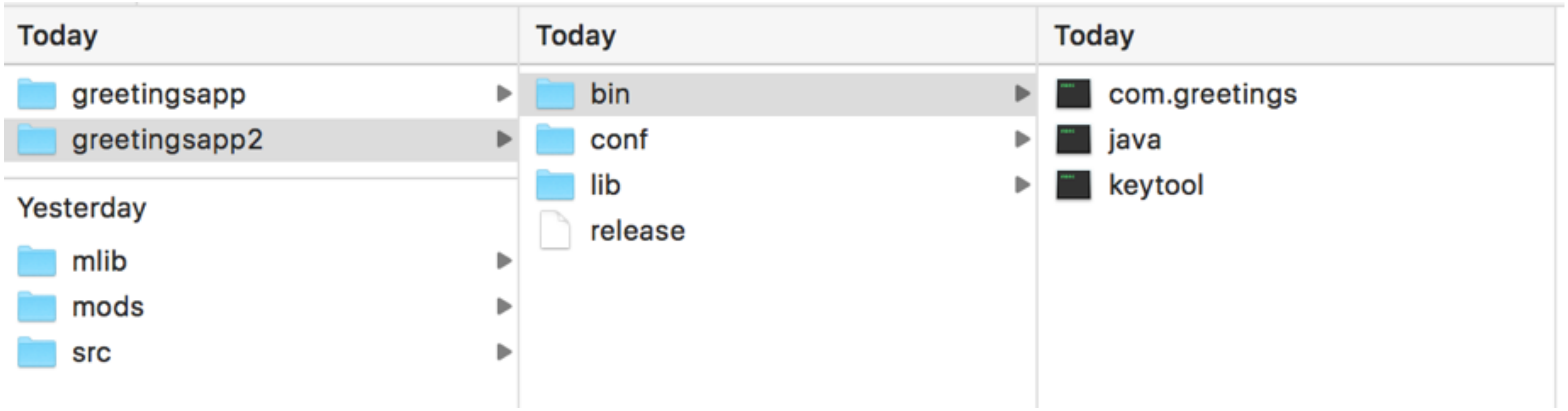
greetingsapp2 20,9 MB
Modified: Vandaag 15:36

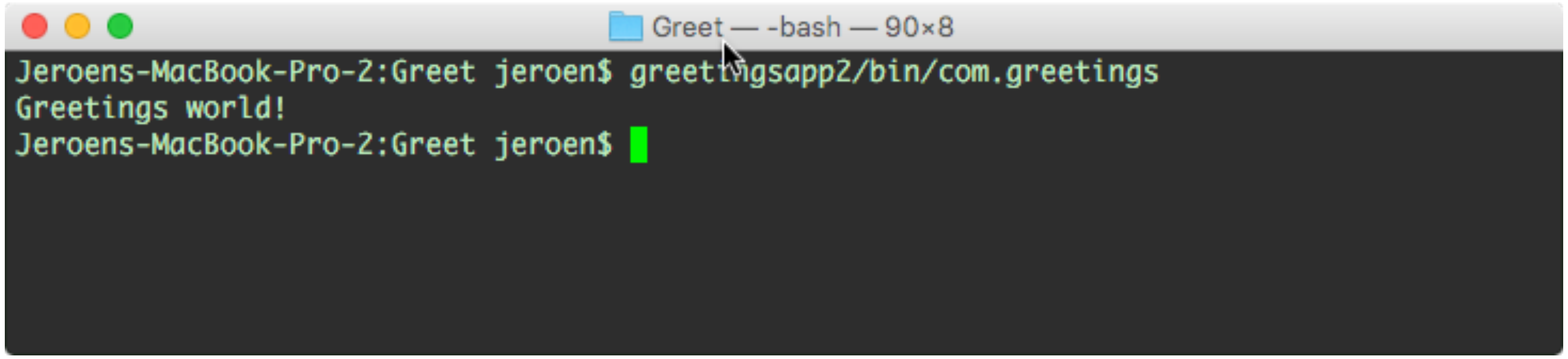
Kind: Folder
Size: 20.854.930 bytes (20,9 MB on disk) for 42 items
Where: Macintosh HD ▸ Users ▸ jeroen ▸ jdk9-projects ▸ Greet
Created: Vandaag 15:35

Profactive

jlink

- Produces a custom modular run-time image

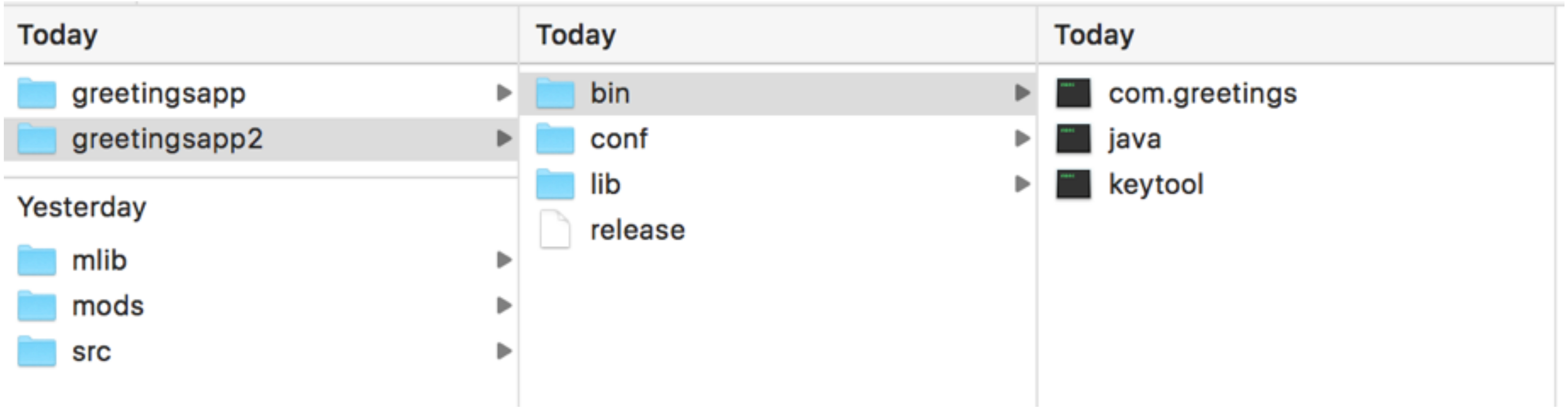


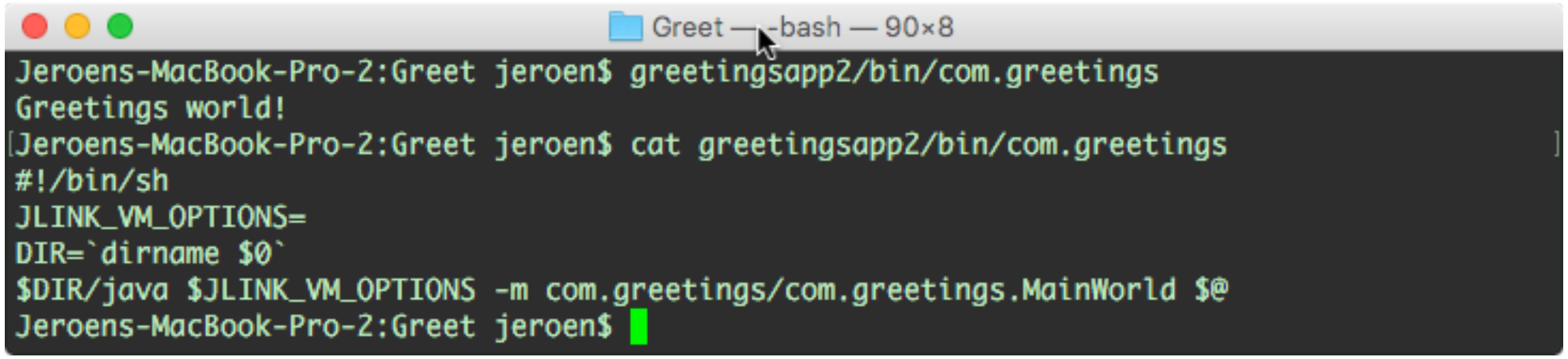


```
Jeroens-MacBook-Pro-2:Greet jeroen$ greetingsapp2/bin/com.greetings
Greetings world!
Jeroens-MacBook-Pro-2:Greet jeroen$ 
```

jlink

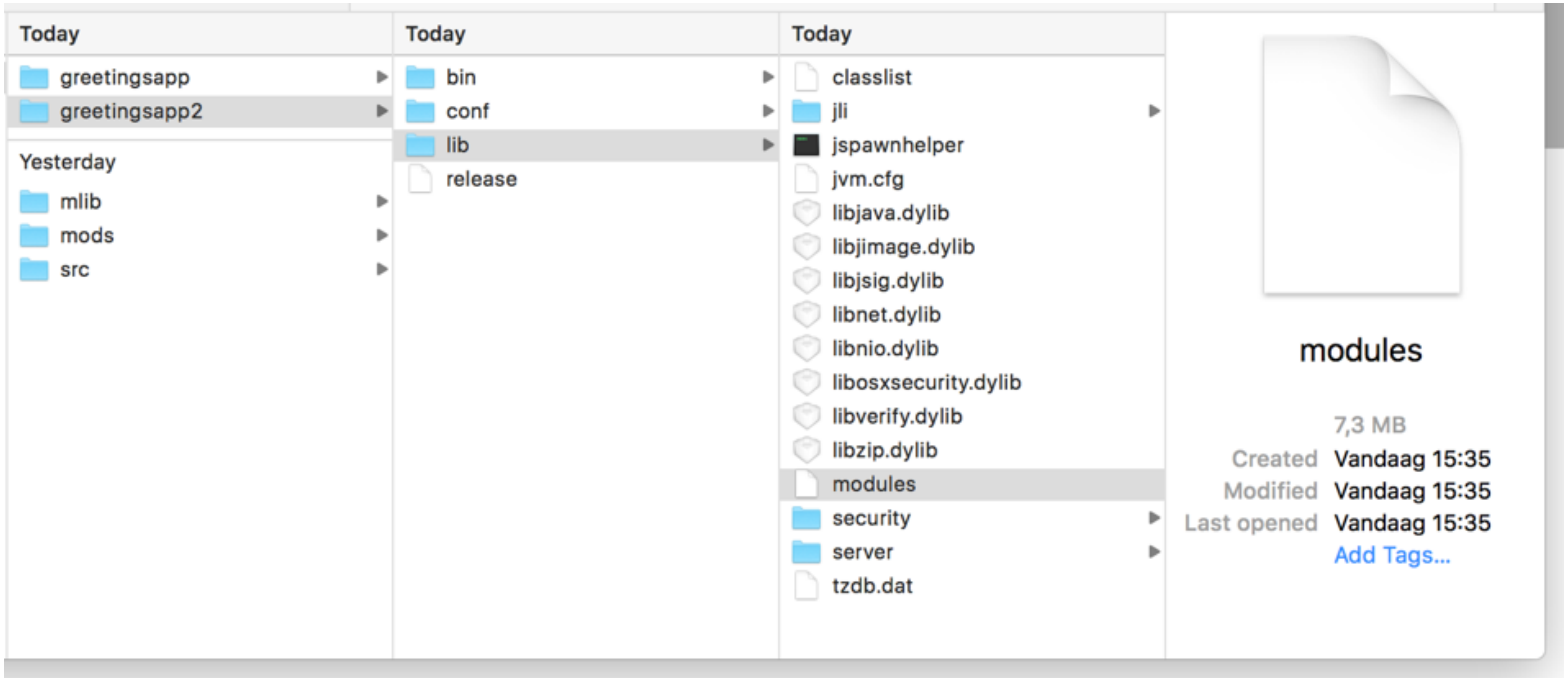
- Produces a custom modular run-time image





```
Jeroens-MacBook-Pro-2:Greet jeroen$ greetingsapp2/bin/com.greetings
Greetings world!
Jeroens-MacBook-Pro-2:Greet jeroen$ cat greetingsapp2/bin/com.greetings
#!/bin/sh
JLINK_VM_OPTIONS=
DIR=`dirname $0`
$DIR/java $JLINK_VM_OPTIONS -m com.greetings/com.greetings.MainWorld $@
Jeroens-MacBook-Pro-2:Greet jeroen$
```

inside the run-time image



Today
- greetingsapp
- greetingsapp2

Yesterday
- mlib
- mods
- src

Today
- bin
- conf
- lib
- release

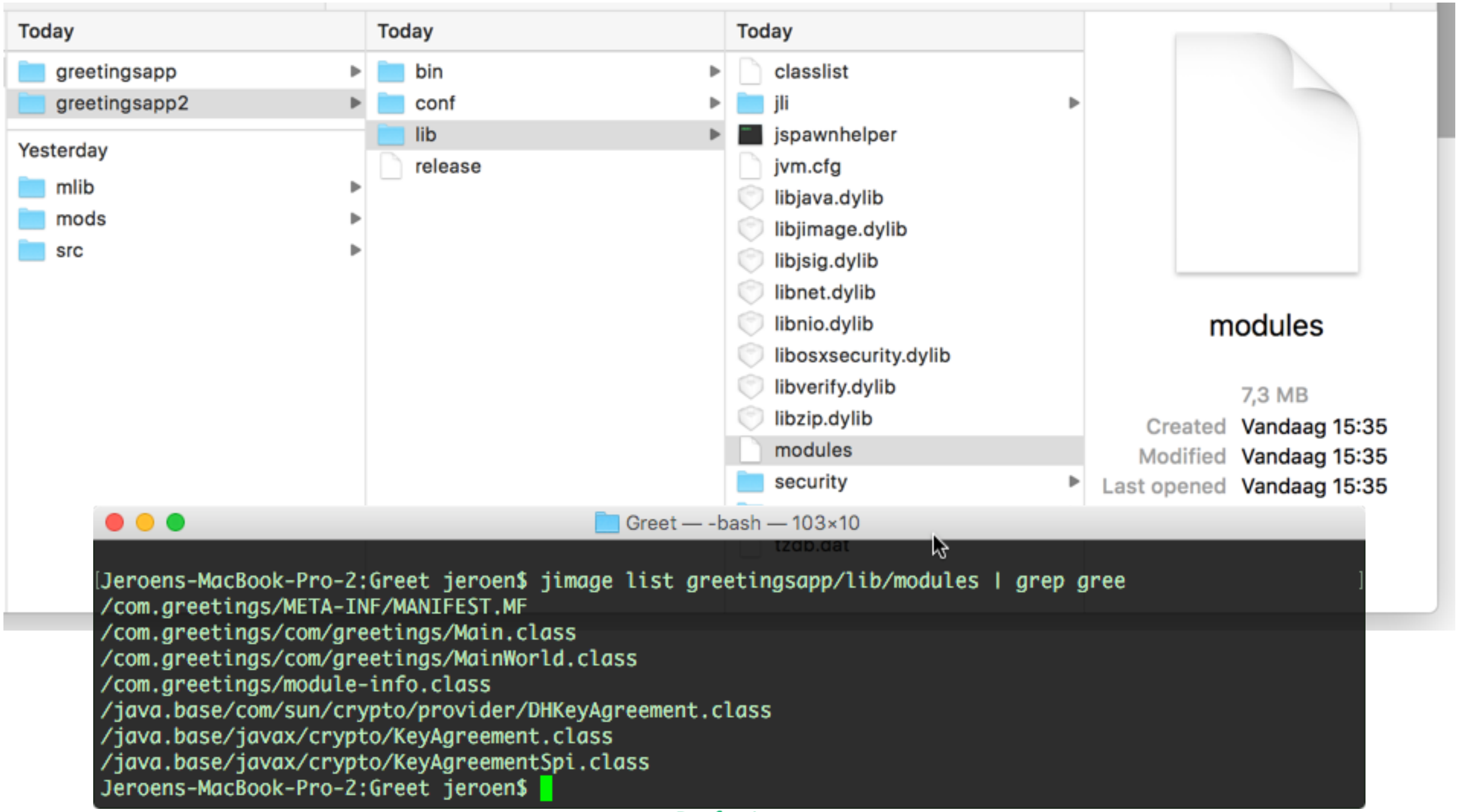
Today
- classlist
- jli
- jspawnhelper
- jvm.cfg
- libjava.dylib
- libjimage.dylib
- libjsig.dylib
- libnet.dylib
- libnio.dylib
- libosxsecurity.dylib
- libverify.dylib
- libzip.dylib
- modules
- security
- server
- tzdb.dat

modules

7,3 MB

Created	Vandaag 15:35
Modified	Vandaag 15:35
Last opened	Vandaag 15:35

inside the run-time image



```
Jeroens-MacBook-Pro-2:Greet jeroen$ jimage list greetingsapp/lib/modules | grep gree
/com.greetings/META-INF/MANIFEST.MF
/com.greetings/com/greetings/Main.class
/com.greetings/com/greetings/MainWorld.class
/com.greetings/module-info.class
/java.base/com/sun/crypto/provider/DHKeyAgreement.class
/java.base/javax/crypto/KeyAgreement.class
/java.base/javax/crypto/KeyAgreementSpi.class
Jeroens-MacBook-Pro-2:Greet jeroen$
```

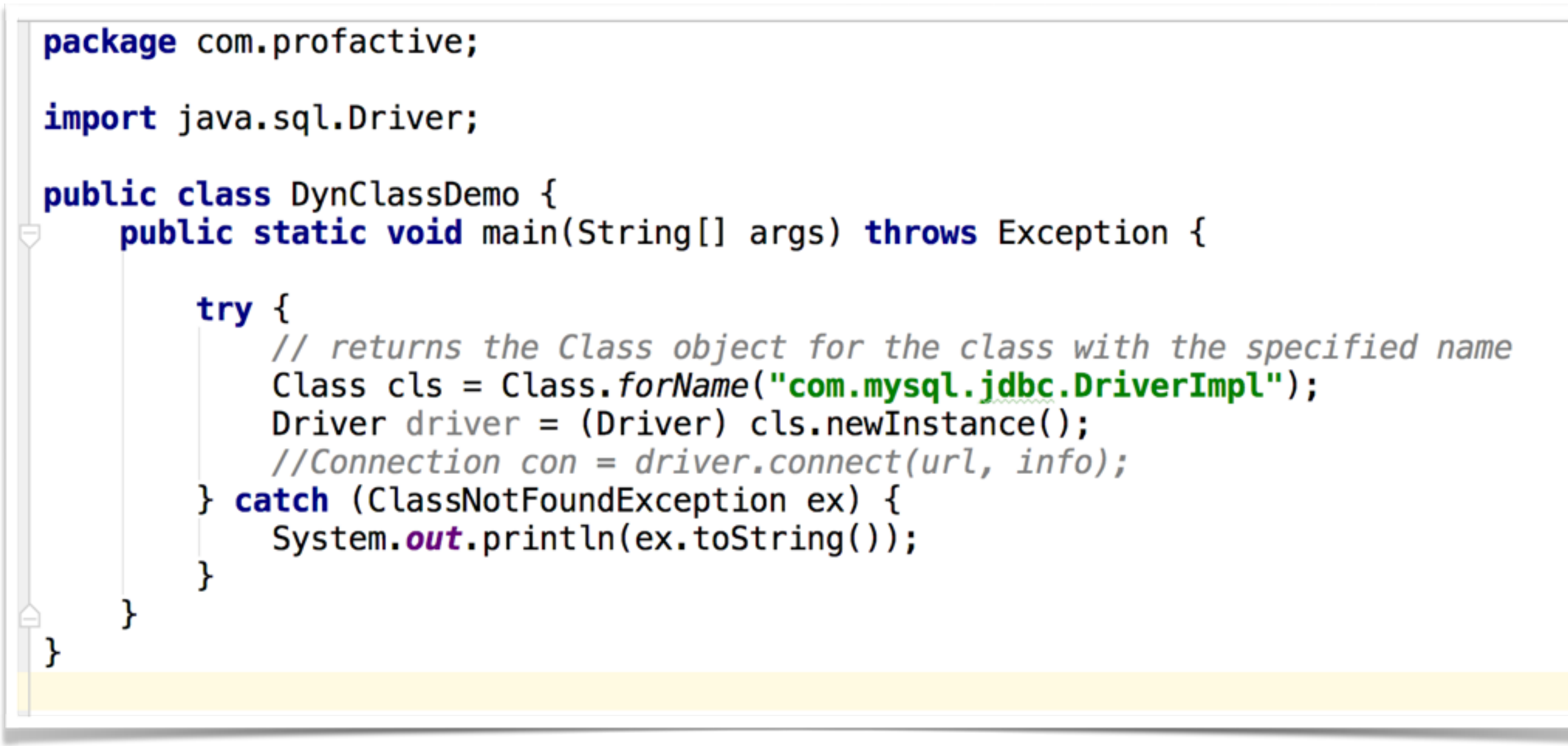
jlink - to assemble and optimize

- link-time: optional phase between `javac` and `java`
- assemble and optimize a set of modules
    - and their transitive dependencies
- creates a run-time image or executable
- apply *whole-world* optimizations
    - otherwise difficult at `javac` time or costly at `java` time

jlink plugins

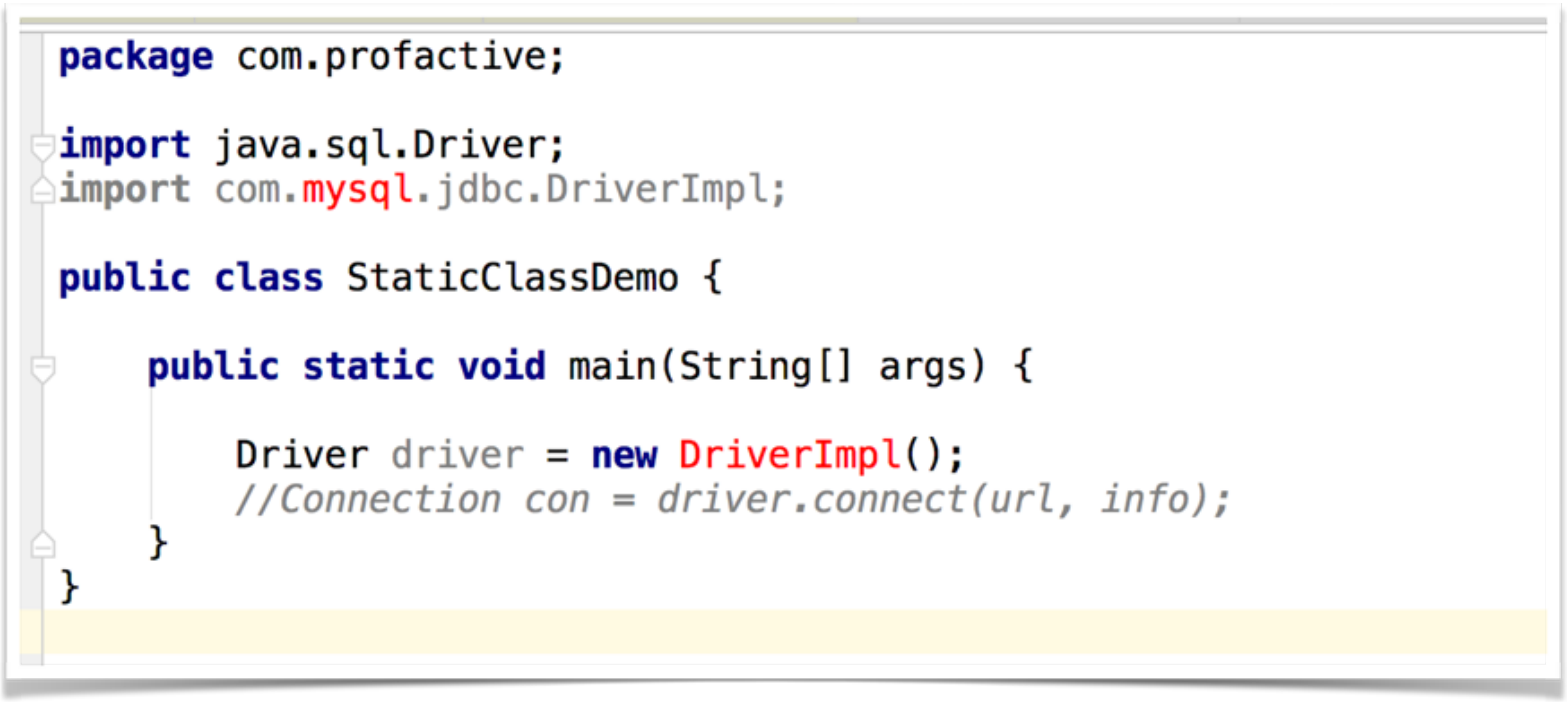
- compress
- strip-debug
- installed-modules
    - fast loading of module descriptors
- class-optim=<all|forName-folding>[:log=<log file>]
    - experimental

Dynamic class loading

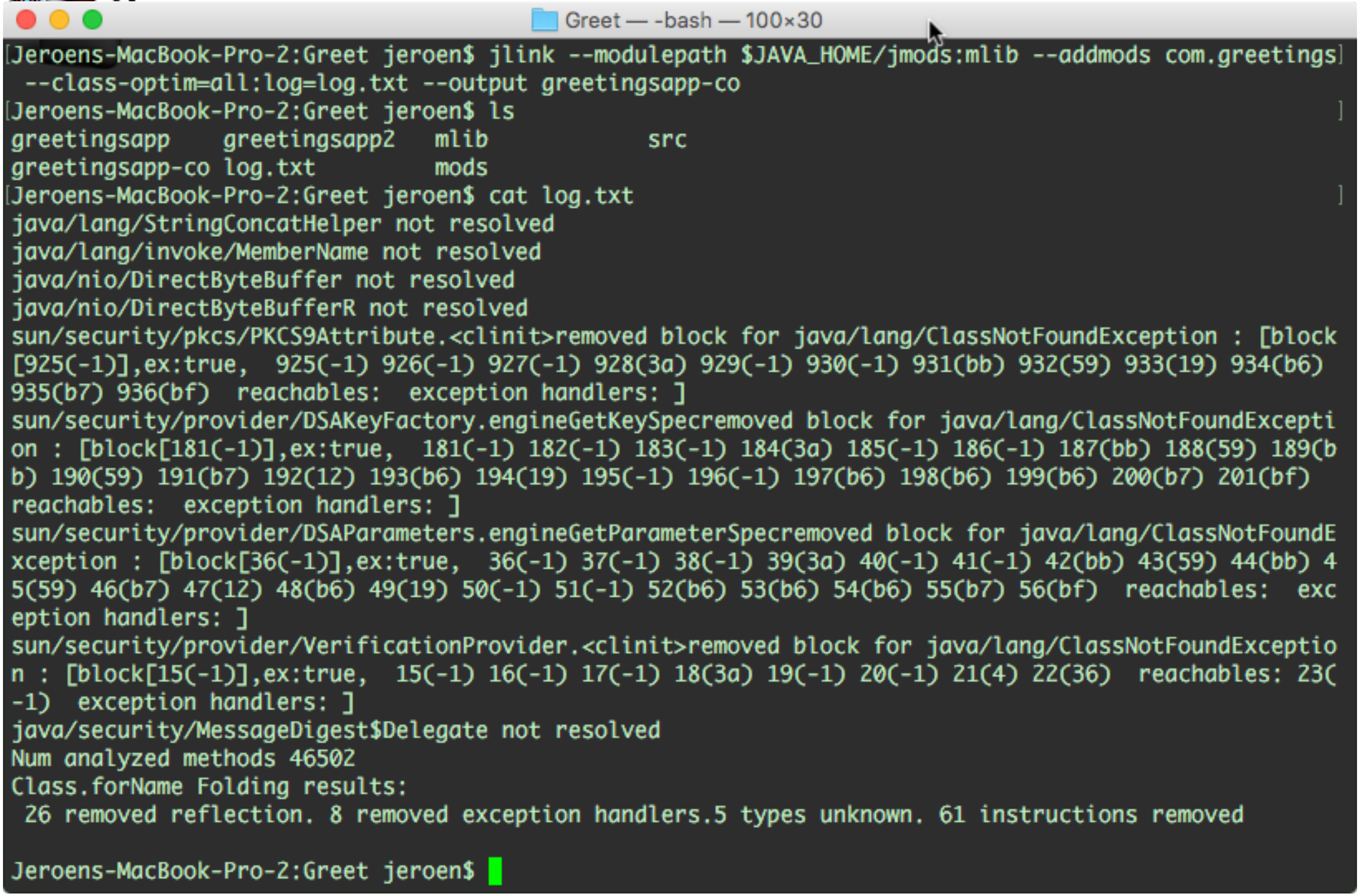


```java
package com.profactive;

import java.sql.Driver;

public class DynClassDemo {
    public static void main(String[] args) throws Exception {

        try {
            // returns the Class object for the class with the specified name
            Class cls = Class.forName("com.mysql.jdbc.DriverImpl");
            Driver driver = (Driver) cls.newInstance();
            //Connection con = driver.connect(url, info);
        } catch (ClassNotFoundException ex) {
            System.out.println(ex.toString());
        }
    }
}
```

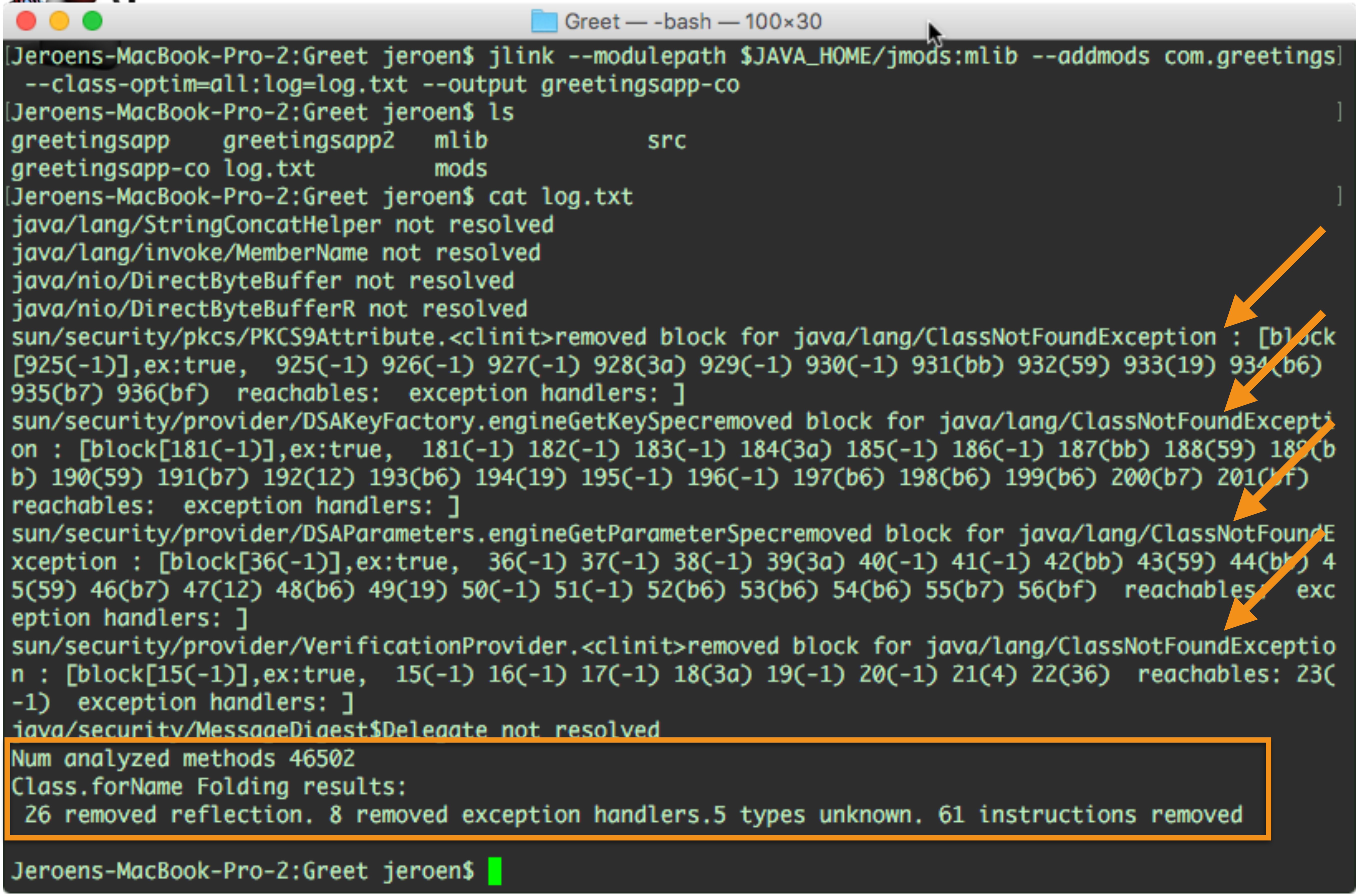
Static class loading



```java
package com.profactive;

import java.sql.Driver;
import com.mysql.jdbc.DriverImpl;

public class StaticClassDemo {

    public static void main(String[] args) {

        Driver driver = new DriverImpl();
        //Connection con = driver.connect(url, info);
    }
}
```


Greet — -bash — 100×30

```
Jeroens-MacBook-Pro-2:Greet jeroen$ jlink --modulepath $JAVA_HOME/jmods:mlib --addmods com.greetings
 --class-optim=all:log=log.txt --output greetingsapp-co
Jeroens-MacBook-Pro-2:Greet jeroen$ ls
greetingsapp    greetingsapp2   mlib            src
greetingsapp-co log.txt         mods
Jeroens-MacBook-Pro-2:Greet jeroen$ cat log.txt
java/lang/StringConcatHelper not resolved
java/lang/invoke/MemberName not resolved
java/nio/DirectByteBuffer not resolved
java/nio/DirectByteBufferR not resolved
sun/security/pkcs/PKCS9Attribute.<clinit>removed block for java/lang/ClassNotFoundException : [block
[925(-1)],ex:true,  925(-1) 926(-1) 927(-1) 928(3a) 929(-1) 930(-1) 931(bb) 932(59) 933(19) 934(b6)
935(b7) 936(bf)  reachables:  exception handlers: ]
sun/security/provider/DSAKeyFactory.engineGetKeySpecremoved block for java/lang/ClassNotFoundExcepti
on : [block[181(-1)],ex:true,  181(-1) 182(-1) 183(-1) 184(3a) 185(-1) 186(-1) 187(bb) 188(59) 189(b
b) 190(59) 191(b7) 192(12) 193(b6) 194(19) 195(-1) 196(-1) 197(b6) 198(b6) 199(b6) 200(b7) 201(bf)
reachables:  exception handlers: ]
sun/security/provider/DSAParameters.engineGetParameterSpecremoved block for java/lang/ClassNotFoundE
xception : [block[36(-1)],ex:true,  36(-1) 37(-1) 38(-1) 39(3a) 40(-1) 41(-1) 42(bb) 43(59) 44(bb) 4
5(59) 46(b7) 47(12) 48(b6) 49(19) 50(-1) 51(-1) 52(b6) 53(b6) 54(b6) 55(b7) 56(bf)  reachables:  exc
eption handlers: ]
sun/security/provider/VerificationProvider.<clinit>removed block for java/lang/ClassNotFoundExceptio
n : [block[15(-1)],ex:true,  15(-1) 16(-1) 17(-1) 18(3a) 19(-1) 20(-1) 21(4) 22(36)  reachables: 23(
-1)  exception handlers: ]
java/security/MessageDigest$Delegate not resolved
Num analyzed methods 46502
Class.forName Folding results:
 26 removed reflection. 8 removed exception handlers.5 types unknown. 61 instructions removed

Jeroens-MacBook-Pro-2:Greet jeroen$ 
```



```
Jeroens-MacBook-Pro-2:Greet jeroen$ jlink --modulepath $JAVA_HOME/jmods:mlib --addmods com.greetings
 --class-optim=all:log=log.txt --output greetingsapp-co
Jeroens-MacBook-Pro-2:Greet jeroen$ ls
greetingsapp    greetingsapp2   mlib            src
greetingsapp-co log.txt         mods
Jeroens-MacBook-Pro-2:Greet jeroen$ cat log.txt
java/lang/StringConcatHelper not resolved
java/lang/invoke/MemberName not resolved
java/nio/DirectByteBuffer not resolved
java/nio/DirectByteBufferR not resolved
sun/security/pkcs/PKCS9Attribute.<clinit>removed block for java/lang/ClassNotFoundException : [block
[925(-1)],ex:true,  925(-1) 926(-1) 927(-1) 928(3a) 929(-1) 930(-1) 931(bb) 932(59) 933(19) 934(b6)
935(b7) 936(bf)  reachables:  exception handlers: ]
sun/security/provider/DSAKeyFactory.engineGetKeySpecremoved block for java/lang/ClassNotFoundExcepti
on : [block[181(-1)],ex:true,  181(-1) 182(-1) 183(-1) 184(3a) 185(-1) 186(-1) 187(bb) 188(59) 189(b
b) 190(59) 191(b7) 192(12) 193(b6) 194(19) 195(-1) 196(-1) 197(b6) 198(b6) 199(b6) 200(b7) 201(bf)
reachables:  exception handlers: ]
sun/security/provider/DSAParameters.engineGetParameterSpecremoved block for java/lang/ClassNotFoundE
xception : [block[36(-1)],ex:true,  36(-1) 37(-1) 38(-1) 39(3a) 40(-1) 41(-1) 42(bb) 43(59) 44(bb) 4
5(59) 46(b7) 47(12) 48(b6) 49(19) 50(-1) 51(-1) 52(b6) 53(b6) 54(b6) 55(b7) 56(bf)  reachables:  exc
eption handlers: ]
sun/security/provider/VerificationProvider.<clinit>removed block for java/lang/ClassNotFoundExceptio
n : [block[15(-1)],ex:true,  15(-1) 16(-1) 17(-1) 18(3a) 19(-1) 20(-1) 21(4) 22(36)  reachables: 23(
-1)  exception handlers: ]
java/security/MessageDigest$Delegate not resolved
Num analyzed methods 46502
Class.forName Folding results:
 26 removed reflection. 8 removed exception handlers.5 types unknown. 61 instructions removed

Jeroens-MacBook-Pro-2:Greet jeroen$
```

Contents - part I - wrap up

- Introduction
- How will life change with JDK9?
- Jigsaw goals
- Platform modules & performance
- JDK8 versus JDK9-modular Java
- Inside modules: jimage, jdeps
- mods & modular jars
- link-time: jlink
- link-time optimizations

Contents - part II

- Compiler improvements & API
- Improved locking
- Variable handles
- Diagnostics
- Garbage collector
- Compact Strings
- Immutable collections
- Stack walking API
- Summary and Conclusions
- Questions

Compiler improvements

- JEP 165: Compiler Control
  - method specific flags, file: `inline:["+java.util.*", "-com.sun.*"]`
  - runtime manageable: `jcmd <pid> Compiler.add_directives <file>`
- JEP 199: Smart Java Compilation
  - sjavac: smart wrapper around javac
  - incremental compiles - recompile only what's necessary
  - parallel compilation - utilize cores during compilation
  - keep compiler in hot VM - reuse JIT'ed javac instance for consecutive invocations

Compiler API - JEP 243

- Allow Java code to observe, query, and affect JVM's compilation
- Pluggable JIT compiler architecture
  - Graal
- May persist code profile and reuse it AOT, avoid JVM warm-up
  - Like Azul's ReadyNow!

JEP 143: Improve contended locking

- 22 many-threads benchmarks
- Field reordering and cache line alignment
- Fast Java monitor enter and exit operations
- Fast Java monitor notify/notifyAll operations

JEP 193: Variable handles

- Typed reference to a variable
- Atomicity for object fields, array elements and ByteBuffers
  - like java.util.concurrent.atomic, sun.misc.Unsafe operations
  - java.lang.invoke.VarHandle, next to MethodHandle from Java7
  - java.util.concurrent will move from use of Unsafe to VarHandles
  - VH will use Unsafe internally
  - What is that Unsafe class? In thread stacks I see: Unsafe.park

Every time I see this:

```
java.lang.Thread.State: WAITING
    at sun.misc.Unsafe.park(Native Method)
```

I think on this:

By: @arturotena

Unsafe.park - 2

www.circlecity.co.uk

Profactive

Side step: sun.misc.Unsafe

- Better alternative to native C or assembly code via JNI
- Atomic compare-and-swap operations like in AtomicInteger, ConcurrentHashMap

```
public final native boolean compareAndSwapInt(Object o, long
offset, int expected, int x)
```

- Direct access to native, off-heap memory

```
public native long allocateMemory(long bytes); //quite unsafe!
```

- Creating objects without calling constructor like in Serialization
- High performance; special handling by JVM
    - methods are intrinsified: assembler instruction inlined to caller, no JNI-call overhead

Side step: sun.misc.Unsafe

- Access to Unsafe is restricted to JDK classes however
  - Can be worked around by reflection
- Java 9 puts Unsafe in jdk internal module
  - Safe and updated alternatives come available: VarHandles
- Libs currently using Unsafe: Netty, Hazelcast, Kryo, Cassandra, Spring, Akka, ..
- command line flag makes Unsafe readable for transition period

JEP 193: Variable handles

- Use case:

```
class Position {
 private volatile int x = 0;
 public void walkRight() {
  x++;
 }
}
```

  - Is it thread safe?

JEP 193: Variable handles

- Use case:

```
class Position {
  private volatile int x = 0;
  public void walkRight() {
    x++;
  }
}
```

  - Not thread-safe because x++ is in fact two operations:

```
int tmp = this.x;
this.x = tmp + 1;
```

  - Other thread may walkRight in between these two and have his result lost

JEP 193: Variable handles

- Solution:

```
class Position {
 private AtomicInteger x = new AtomicInteger();
 public void walkRight() {
  x.incrementAndGet();
 }
}
```

  - memory usage compared to previous?

JEP 193: Variable handles

```
class Pos {

 private int x = 0;

 public void walkRight() {

  x = VH_POS_X.addAndGet(this, 1);

 }

}
```

JEP 193: Variable handles

```
class Pos {
    private static final VarHandle VH_POS_X;
    private int x = 0;
     static {
         try {
             VH_POS_X = MethodHandles.lookup().
                 in(Pos.class).findFieldVarHandle(Pos.class, "x", int.class);
         } catch (Exception e) { throw new Error(e); }
     }
    public void walkRight() {
         VH_POS_X.addAndGet(this, 1);
    }
}
```

More diagnostic commands

```
Jeroens-MacBook-Pro-2:Home jeroen$ jcmd 31142 VM.class_hierarchy
31142:
java.lang.Object/null
|--java.lang.reflect.Proxy$ProxyBuilder$$Lambda$122/123322386/null
|--jdk.internal.jimage.ImageBufferCache/null
|--org.netbeans.core.windows.view.ModeAccessor/0x00007faf026d8730 (intf)
|-java.lang.invoke.LambdaForm$DMH/1841321848/null

Jeroens-MacBook-Pro-2:Home jeroen$ jcmd 31142 VM.stringtable
\31142:
StringTable statistics:
Number of buckets       :     60013 =    480104 bytes, avg   8.000
Number of entries       :     17882 =    429168 bytes, avg  24.000
Number of literals      :     17882 =   1604736 bytes, avg  89.740
Total footprint         :           =   2514008 bytes
Average bucket size     :     0.298
Variance of bucket size :     0.299
Std. dev. of bucket size:     0.547
Maximum bucket size     :         4
```

- Compiler.queue .codelist, .codecache
- VM.set_flag

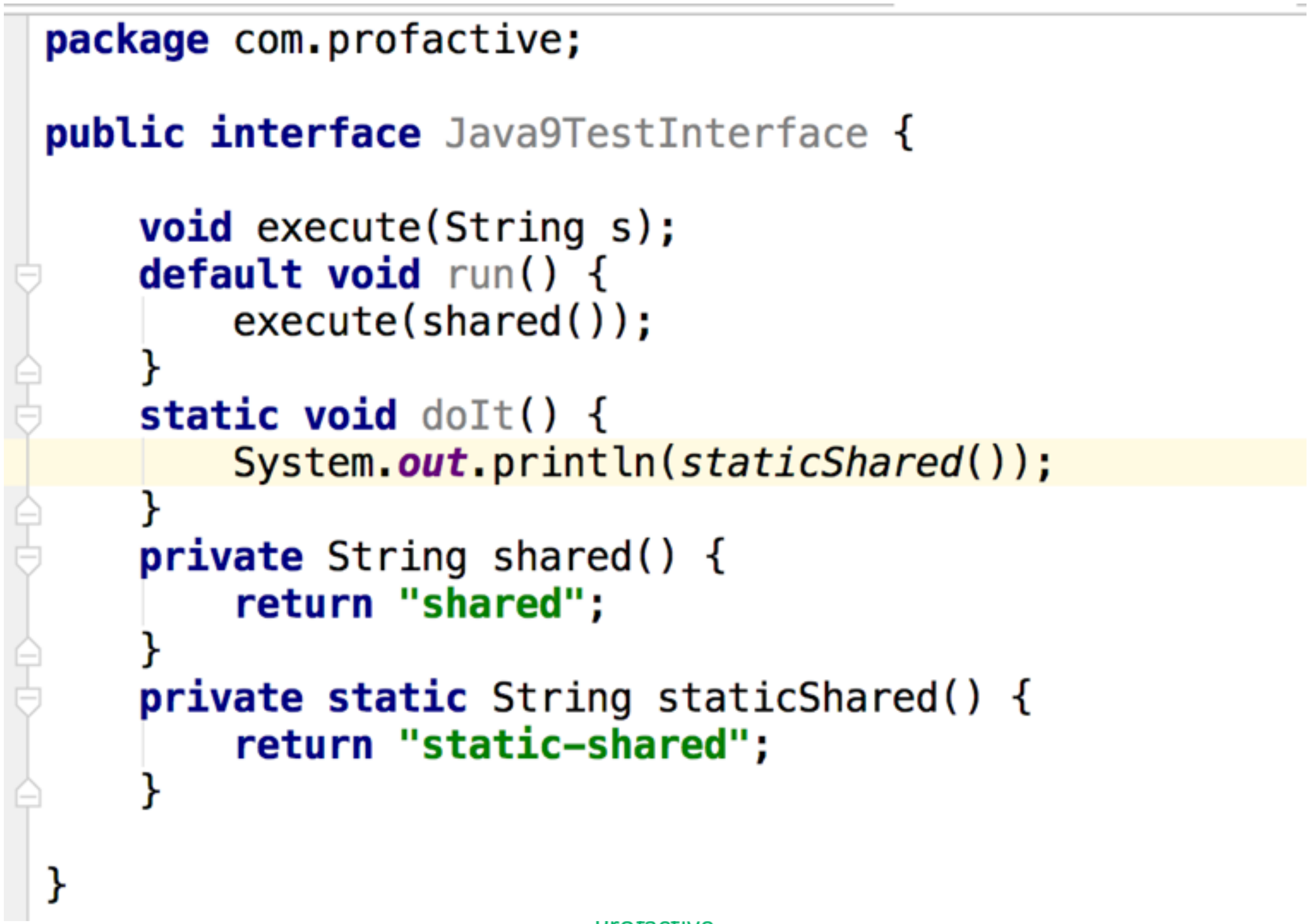
G1 as default collector

- G1 default on 32 and 64 bit server configs
- Replaces Parallel GC as default
  - Parallel GC shows long pauses for large heaps
- JDK8_u40 / JEP 156: G1 now supports class unloading instead of needing a full GC
- Optimizes for low pause time
  - Not for throughput nor CPU load!
- May need more tuning
  - -XX:MaxGCPauseMillis=n

Compact Strings

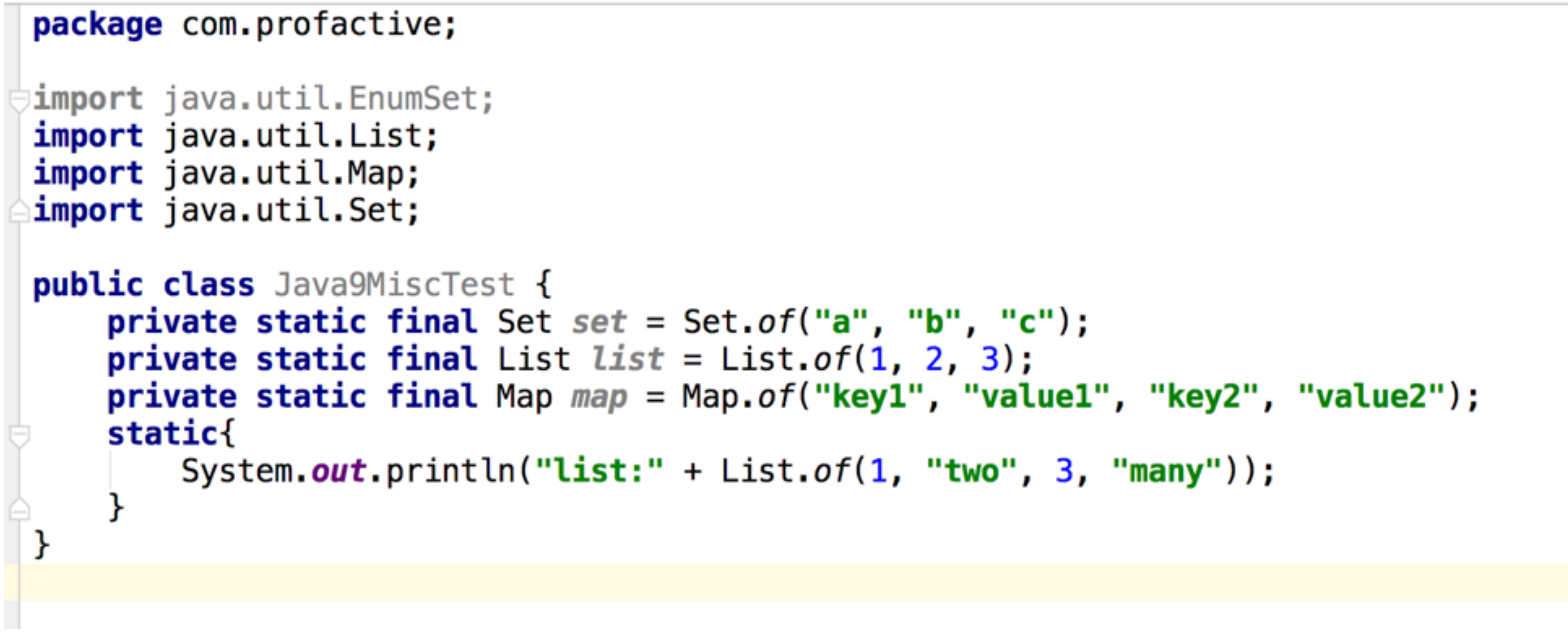
- Improve space efficiency of String, StringBuilder, etc.
- String is often biggest consumer of the heap
- Characters are UTF-16: 2 bytes, while most apps use only Latin-1: 1 byte
- New: byte[] or char[], + encoding flag field
- Less allocation, less GC, less data on bus: so also better time efficiency!
- SPECjbb2005 server app benchmark:
    - 21% less live data
    - GC: 21% less frequent, 17% less long
    - 10% better app throughput

private methods on interfaces



```java
package com.profactive;

public interface Java9TestInterface {

    void execute(String s);
    default void run() {
        execute(shared());
    }
    static void doIt() {
        System.out.println(staticShared());
    }
    private String shared() {
        return "shared";
    }
    private static String staticShared() {
        return "static-shared";
    }

}
```

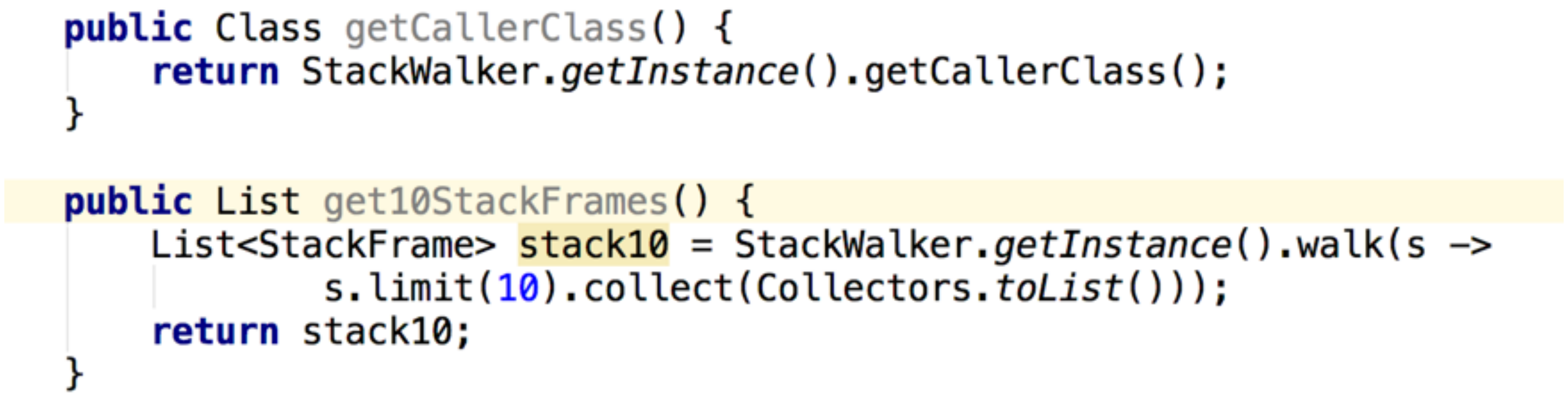
Factory methods for small immutable collections



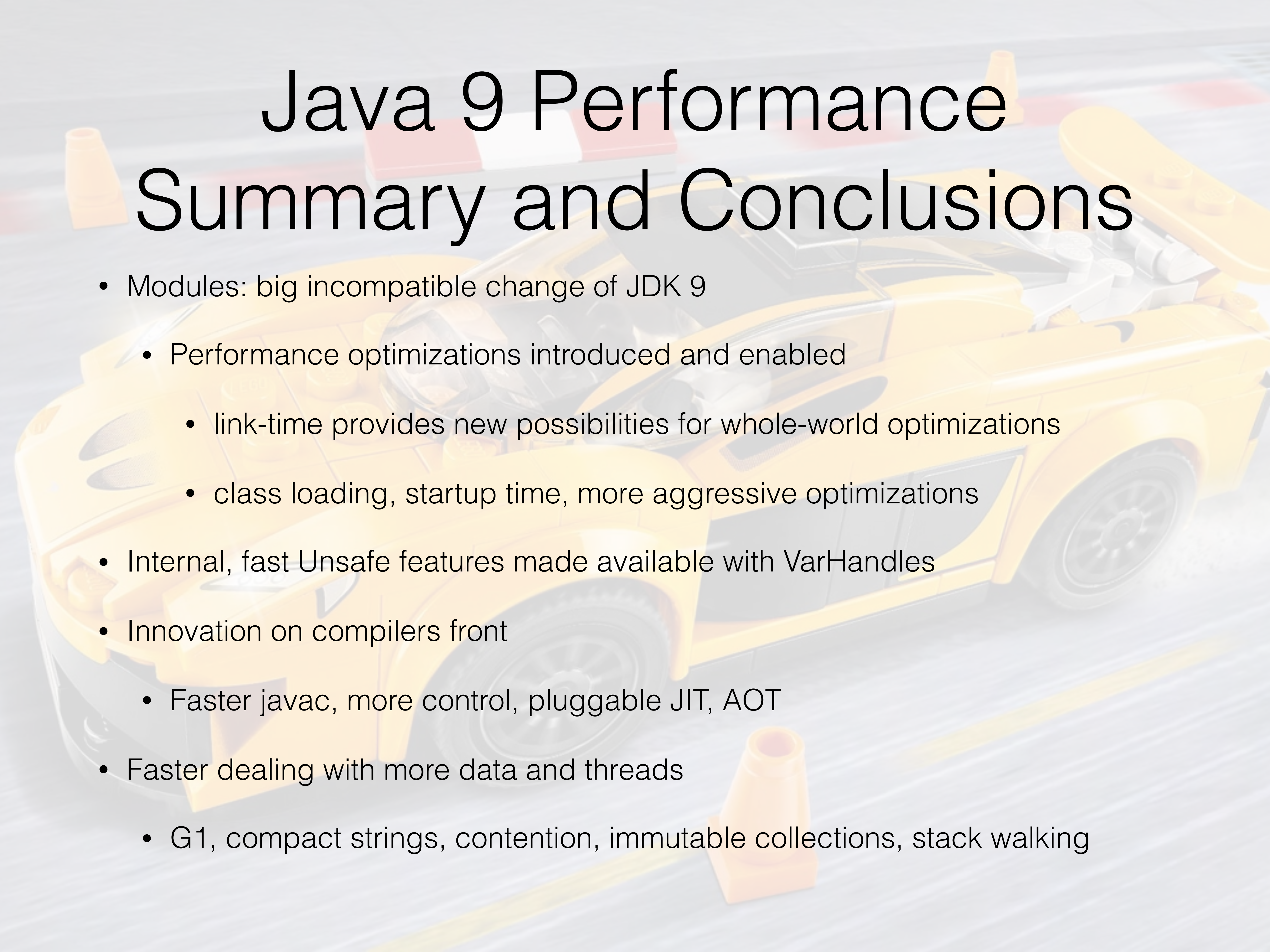
```
package com.profactive;

import java.util.EnumSet;
import java.util.List;
import java.util.Map;
import java.util.Set;

public class Java9MiscTest {
    private static final Set set = Set.of("a", "b", "c");
    private static final List list = List.of(1, 2, 3);
    private static final Map map = Map.of("key1", "value1", "key2", "value2");
    static{
        System.out.println("list:" + List.of(1, "two", 3, "many"));
    }
}
```

Stack-walking API

- Throwable::getStackTrace and Thread::getStackTrace return all StackTraceElement[] containing class name + method name
  - expensive
- sun.reflect.Reflection::getCallerClass is fast, only JDK-internal
  - replacement needed



```
public Class getCallerClass() {
    return StackWalker.getInstance().getCallerClass();
}

public List get10StackFrames() {
    List<StackFrame> stack10 = StackWalker.getInstance().walk(s ->
            s.limit(10).collect(Collectors.toList()));
    return stack10;
}
```



Java 9 Performance Summary and Conclusions

- Modules: big incompatible change of JDK 9
  - Performance optimizations introduced and enabled
    - link-time provides new possibilities for whole-world optimizations
    - class loading, startup time, more aggressive optimizations
- Internal, fast Unsafe features made available with VarHandles
- Innovation on compilers front
  - Faster javac, more control, pluggable JIT, AOT
- Faster dealing with more data and threads
  - G1, compact strings, contention, immutable collections, stack walking

Java 9 Performance Questions?

Want to learn more?

- www.jpinpoint.com / www.profactive.com
  - references, presentations
- Accelerating Java Applications
  - 3 days technical training. October 10-12, 2016
  - Info/subscribe: http://www.jpinpoint.com/training-accelerate.html
- Performance oriented Java coding (NEW)
  - 1 day developer workshop. November 2, 2016
  - Info/subscribe: jborgers@jpinpoint.com
- For both use code: ‘*UJM2016*’ for 10% discount.