Core Java

JDK 9 Overview

Angelika Langer & Klaus Kreft
Training/Consulting

http://www.AngelikaLanger.com/

a quick glance at Java 9

- Java 9
  - available since September, 21 2017
- many new features (> 90 JEPs)
  - "Collection Literals"
  - "Compact Strings" and related JEPs
  - Java Module System
  - "Replace sun.misc.*" and related JEPs
  - Enhanced Volatiles
  - JShell aka Java REPL
  - and many more ...

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

speaker's relationship to topic

- independent trainer / consultant / author
  - teaching C++ and Java for ~20 years
  - curriculum of some challenging seminars
  - JCP observer and Java champion since 2005
  - co-author of "Effective Java" column
  - author of Java Generics FAQ
  - author of Lambda Tutorial & Reference

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

agenda

- **"Collection Literals"**
- "Compact Strings" and related JEPs
- Java Module System
- "Replace sun.misc.*" and related JEPs
- Enhanced Volatiles
- JShell aka Java REPL
- and many more ...

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

collection literals

- proposed for "Project Coin" (Java 7)
  - convenient syntax for initialization of small immutable collections
- e.g.

```
List<Integer> list = #[ 1, 2, 3 ];
```

- no language extension in Java 9
  - value types (in Java 10/11) would change it anyway
- instead a library-based proposal (JEP 269)

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

of() factory methods

- static factory methods for List, Set, and Map interfaces
  - for creating unmodifiable instances of those collections
- e.g.

```
List<String> set = List.of("a", "b", "c");
```

- similar to existing factory methods in class Collections

```
List<String> noElement    = Collections.emptyList();
List<String> oneElement   = Collections.singletonList("a");
List<String> someElements = List.of("a", "b", "c");
```

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

minor language changes

- private interface methods
- diamond with anonymous classes
- effectively-final variables in try-with-resources
- @SafeVargs on private instance methods
- "_" is an illegal identifier

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

agenda

- "Collection Literals"
- **"Compact Strings" and related JEPs**
- Java Module System
- "Replace sun.misc.*" and related JEPs
- Enhanced Volatiles
- JShell aka Java REPL
- and many more ...

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

String related optimizations

- G1 deduplicates strings
  - since 8_u20

- several string related improvements in Java 9
  - JEP 254: Compact Strings
  - JEP 280: Indify String Concatenation
  - JEP 250: Store Interned Strings in CDS Archives

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

string deduplication

- observation:
  - roughly 25% of heap is consumed by strings
- goal:
  - reduce memory footprint by reducing number and size of strings

- deduplication
  - G1 GC looks for deduplication candidates
    - i.e., strings with a char sequence that already exists
  - deduplicates
    - i.e., re-assigns backing char array to a cached array with equal content

- available since JDK 8_u20

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

compact strings

- observation:
  - strings store characters in a char[]
    - using two bytes for each character
  - most string contain only Latin-1 characters
    - can be stored in one byte => half the space is going unused

- compact strings
  - change the internal representation
    from a UTF-16 char array to a byte array plus an encoding-flag
    - based on string content set an encoding flag
    - store character as
      - either ISO-8859-1/Latin-1 (one byte per character)
      - or UTF-16 (two bytes per character)

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

indify string concatenation

- javac translates string concatenation into StringBuilder::append chains
  - sometimes not optimal
  - sometimes need to presize the StringBuilder appropriately
- JIT compiler recognizes StringBuilder append chains
  - upon -XX:+OptimizeStringConcat option
  - applies aggressive optimizations => fragile and difficult to extend / maintain
- "indification"
  - change bytecode sequence generated by javac
  - use invokedynamic (INDY) calls to JDK library functions
  - enables future optimizations without changing javac

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

interned strings in CDS archives

- CDS = class data sharing
  - reduces startup time and footprint
  - JRE installer puts classes from rt. jar into a "shared archive" file
  - at runtime the shared archive is memory-mapped
    - saves cost of loading those classes and
    - allows sharing of class metadata among JVMs

- interned strings in CDS archives
  - do the same for strings in the classes' constant pools
    - dump into a shared archive and memory-map at runtime
  - only supported for G1 GC
    - shared strings require a pinned region, and only G1 supports pinning
  - only supported for 64-bit platforms

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.

agenda

- "Collection Literals"
- "Compact Strings" and related JEPs
- **Java Module System**
- "Replace sun.misc.*" and related JEPs
- Enhanced Volatiles
- JShell aka Java REPL
- and many more ...

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

Java Module System …

… aka Project Jigsaw or only Jigsaw

- JEP 200: The Modular JDK
- JEP 201: Modular Source Code
- JEP 220: Modular Run-Time Images
- JEP 260: Encapsulate Most Internal APIs
- JEP 261: Module System

- JSR 376: Java Platform Module System

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

what does it mean to you ?

- modularized JDK
  - new deployment form of the JDK
    - no rt.jar anymore
- ways and means how you and third party library vendors might (re-)organize your/their code into modules
  - optional, but recommended
  - starts with Java 9, transitional phase

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.

modules / module system in Java

- module contains classes, native code, properties, …

- module describes …
  - which modules it uses (readability)
  - which packages it allows to be accessed (accessibility)
    - both modules must agree upon their relationship
      - different from public, package, private protection scheme

- module system enforces that only these relationships can be used

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

agenda

- **Java Module System**
  - **motivation**
  - modular JDK
  - more about modules
  - module development
  - migration
  - critique

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

why modules ? - insufficient protection scheme

- existing public, package, protected, private scheme insufficient
- until now: package protected used to hide types
  - want to use an API from one package by two other packages
    - put all three in one package ? – end up with a bad package design
    - make the API public ? – others will use it, too
- happened in the JDK
  - public types in sun.*
    - by naming convention JDK internal
    - but accessible from all other Java code
    - and used, e.g. sun.misc.Unsafe

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.

why modules ? - jar hell

- don't know what version of a third party library your program needs ?
  - trial and error with different versions
    - until it starts, runs, passes all test suites ?
- know that different components of your program need different versions of a third party library ?
  - hope that the higher version library is downward compatible, and use this version
  - but that might be too optimistic
- program size
  - i.e. deploy only the modules needed
- …

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

agenda

- **Java Module System**
  - motivation
  - **modular JDK**
  - more about modules
  - module development
  - migration
  - critique

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

JEP 200: The Modular JDK

- took around 5 years

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.

what does it mean to you ?

- maybe nothing (good)
- maybe you have to adjust your program (not so good)
- possible issues
  - use of JDK internal API (sun.*)
  - direct access to the binary structure of the JDK or JRE (rt.jar)
  - parsing the Java version string
  - use of split packages
  - …

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.

use of JDK internal API

- jdeps
  - dependency analyzer (since Java 8)
  - helps to detect JDK internal usage + suggests alternative

```
        ......
        -> sun.security.x509.X500Name JDK internal API (java.base)
Warning: JDK internal APIs are unsupported and private to JDK implementation that are
subject to be removed or changed incompatibly and could break your application.
Please modify your code to eliminate dependency on any JDK internal APIs.
For the most recent update on JDK internal API replacements, please check:
https://wiki.openjdk.java.net/display/JDK8/Java+Dependency+Analysis+Tool

JDK Internal API              Suggested Replacement
----------------              ---------------------
sun.security.x509.X500Name    Use javax.security.auth.x500.X500Principal @since 1.4
```

  - change your code accordingly

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

use of JDK internal API (cont.)

- what if the dependency is not in your code, but in a third party library that you use ?
  - i.e. you cannot change the code
- new option (compile and start)

```
...... --add-exports java.base/sun.security.x509=ALL-UNNAMED,...
```

- breaks module encapsulation
  - allowed because you have control over the command line
  - support the transition to modular development
  - might not work forever
    - internal API, i.e. might change

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

agenda

- **Java Module System**
  - introduction
  - motivation
  - modular JDK
  - **more about modules**
  - module development
  - migration
  - critique

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

source code structure

```
src\de.mycompany.mymodule\
                          module-info.java
                          de\mycompany\mypackage1\ClassA.java
                          ...
                          de\mycompany\mypackage2\ClassB.java
                          ...
```

- additional directory on top of package layering
  - which matches the module name
- additional file
  - module-info.java

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

module-info.java

- contains

```
module de.mycompany.mymodule {
   ...
}
```

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

- specifies
  - what packages from the module are publicly accessible:

```
module de.mycompany.mymodule {
   exports de.mycompany.mypackage1;
   ...
}
```

package

  - qualified export:

```
module de.mycompany.mymodule {
   exports de.mycompany.mypackage1 to de.mycompany.myapp;
}
```

another module

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

module-info.java / **readability**

- specifies
  - what other modules are used:

```
module de.mycompany.myapp {
   ...
   requires de.mycompany.mymodule;
   ...
}
```

← module

  - note this is "de.mycompany.myapp's" module-info.java
  - can be confusing
    - ⍰ exports related to packages
    - ⍰ requires related to modules

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

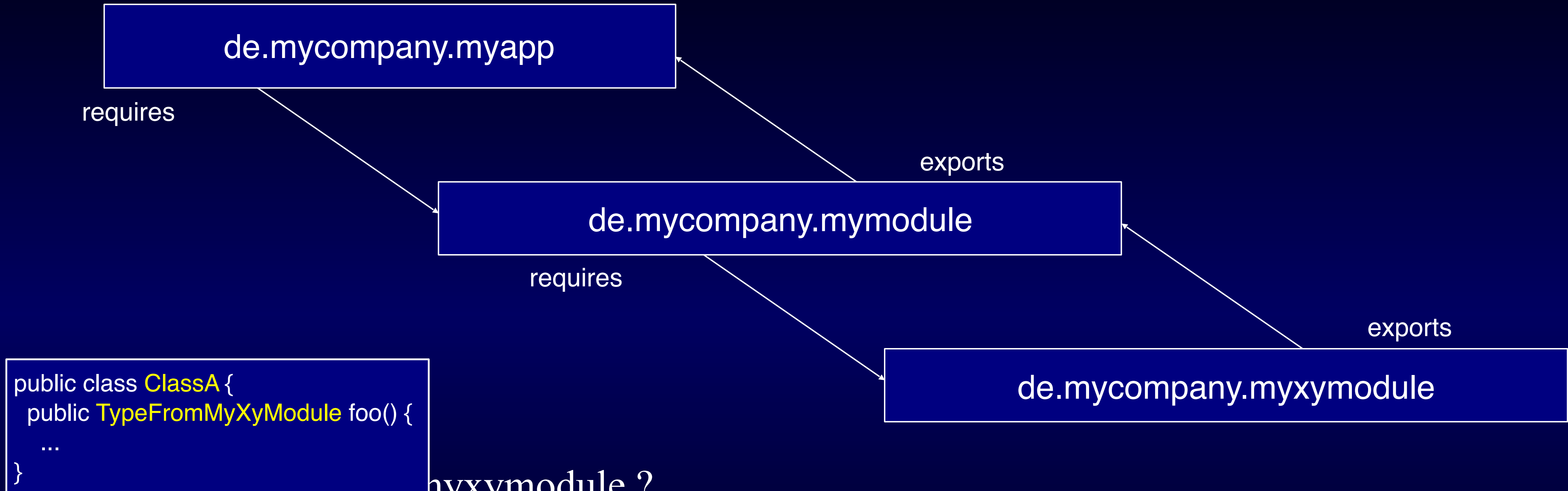
readability + accessibility …

… required

- compilation (checked by: javac)
- runtime (checked by: jvm, Core Reflection)

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

what if ...



myapp requires myxymodule ?

- if it uses foo() from ClassA from module mymodule

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.

solution: requires transitive ...

… transitive readability

```
module de.mycompany.myapp {
   requires de.mycompany.mymodule;
}
```

```
module de.mycompany.mymodule {
   exports de.mycompany.mypackage1;

   requires transitive de.mycompany.myxymodule;
}
```

```
module de.mycompany.myxymodule {
   exports de.mycompany.myxypackage;
}
```

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

agenda

- **Java Module System**
  - introduction
  - motivation
  - modular JDK
  - more about modules
  - **module development**
  - migration
  - critique

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

modular development

- module system changes
  - compilation
  - starting / running
- module path
  - --module-path, or –p
  - similar to class path
- new java option: --show-module-resolution
  - print the resolving of the required modules
  - similar to -verbose:class

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

modular jar

- package a module into a jar-file
  - module-info.class + everything else

- i.e. 1:1-relationship between module and jar-file

- new options for the jar-tool
  - --main-class, specify the main class
  - --list, print content of module
  - --describe-module, prints module description
  - …

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.

modular runtime image

- new tool: jlink
  - link to a modular runtime image
- result: not one executable file, but a directory structure
  - similar to the way the JRE / JDK is deployed in Java 9
- solves: program size
  - i.e. deploy only the modules needed

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

agenda

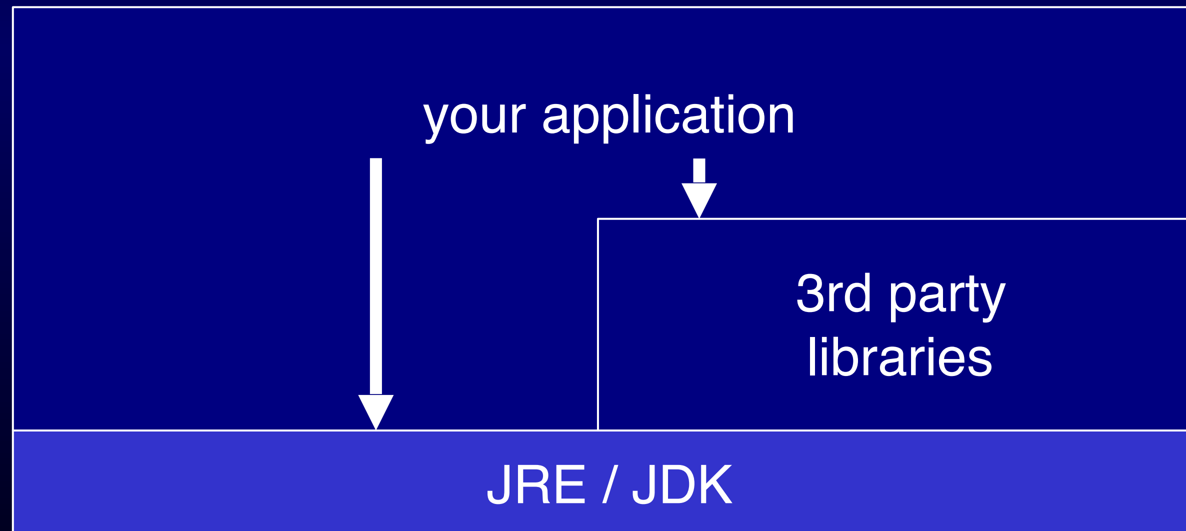
- **Java Module System**
  - introduction
  - motivation
  - modular JDK
  - more about modules
  - module development
  - **migration**
  - critique

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

migration

- assume no hiccups with modular JRE / JDK
  - discussed previously

- modularize your application



© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

modularize your application

- get an understanding of the dependencies
  - use jdeps
- apply a modular design to your application
  - change source directory structure
  - write module-info.java file(s)

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

problem

- 3rd party libraries may not be modules

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

class path in Java 9

- all jars from the class path form the *unnamed* module

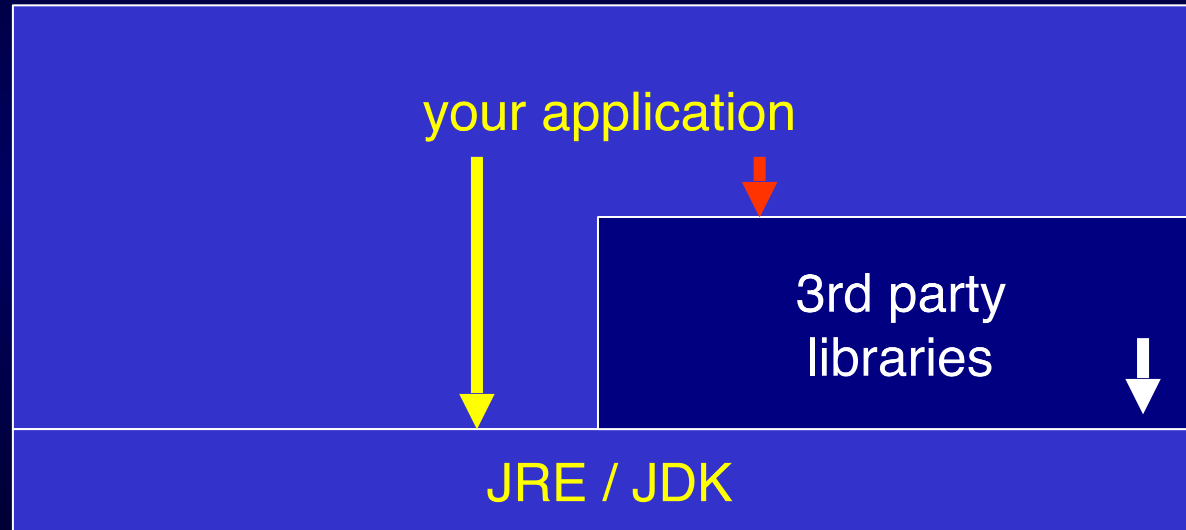
- unnamed module
  - exports all its packages ☺️
  - requires all other modules ☺️

- but
  - named modules cannot require the unnamed module ☹️

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

class path in Java 9 (cont.)

- but
  - named modules cannot require the unnamed module ☹



© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

automatic module …

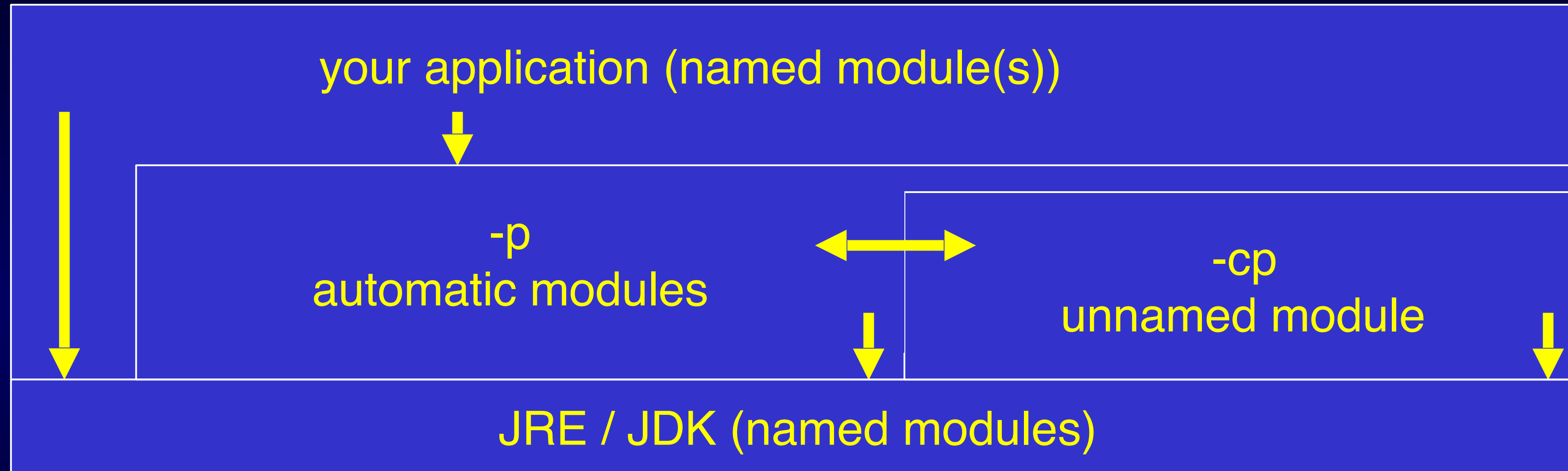
… existing (old) jar file put on the module path

- *automatic* module
  - module name is derived from the jar file name
  - exports all its packages
  - requires all other modules (including the unnamed module)
  - real module, i.e. can be required from a named module
  - but: might not be the module(s) that the 3rd party library provides after modularization

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.

module architecture

- migration



- idea for the future
  - automatic modules and jars from the unnamed module will become named modules

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.

agenda

- **Java Module System**
  - introduction
  - motivation
  - modular JDK
  - more about modules
  - module development
  - migration
  - **critique**

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

critique (1)

- access control might become confusing
  - old private, package, protected, public type-based scheme

  plus
  - module-info export for a package-based module scheme

  - have to look at both schemes to know what's going on
    - public types are not necessarily public anymore
    - type does not tell if its module-internal or not

  - details: mail.openjdk.java.net/pipermail/jpms-spec-observers/2015-November/000225.html

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

critique (2)

- adapting existing code to modules might be
  - a lot of work

    http://mail.openjdk.java.net/pipermail/jigsaw-dev/2015-December/005462.html
    http://mail.openjdk.java.net/pipermail/jigsaw-dev/2015-December/005454.html

  - or even nearly infeasible (split package)

    http://mail.openjdk.java.net/pipermail/jigsaw-dev/2015-October/005059.html

- depends significantly on how much the existing code pushed the limits

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.

critique (3)

- why don't modules have a version?
  - no solution to jar hell ☹
- early prototypes did have module versions
  - there is not just one way of resolving versions
    - each build tool approaches version resolution differently
    - Java would have added yet another approach
  - leave it to the build tools
    - major simplification + speeds up start-up time

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

further reading

The State of the Module System by Mark Reinhold, Oracle

openjdk.java.net/projects/jigsaw/spec/sotms/

Java SE Tools Reference - Section 2: Main Tools to Create and Build Applications

docs.oracle.com/javase/9/tools/main-tools-create-and-build-applications.htm

Java 9 Modularity by Paul Bakker & Sander Mak

O'Reilly, September 2017

The Java Module System by Nicolai Parlog

Manning, May 2018 (estimated)

see related blog: blog.codefx.org

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

agenda

- "Collection Literals"
- "Compact Strings" and related JEPs
- Java Module System
- **"Replace sun.misc.*" and related JEPs**
- Enhanced Volatiles
- JShell aka Java REPL
- and many more ...

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

JDK "internal" packages

- JDK has "internal" package (e.g. sun.misc)
  - never intended for public use, only for implementation of JDK classes
- sole encapsulation mechanism (until Java 9): package-protected
  - does not suffice - can't stuff entire JDK into one package

  => "internal" classes are public (and NOT internal)
- remedy for missing encapsulation mechanism: privilege check
  - for which there are work-arounds
- "internal" features are used in many libraries / frameworks
  - e.g. Akka, Grails, Guava, Hadoop, Hibernate, JRuby, LMAX Disruptor, Netty, Scala, Spring, etc.

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

sun.misc in the modular JDK (Java 9)

- Java 9 uses module system
  - limits access to sun.* features
  - encapsulates them into jdk.internal.* modules

  => sun.* operations will disappear (eventually)

- numerous internal sun.* APIs affected
  - see replacement table at wiki.openjdk.java.net/display/JDK8/Java+Dependency+Analysis+Tool

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

am I affected?

- use jdeps tool (available since Java 8)
  - with -jdkinternals option
  - to find dependencies to any JDK internal APIs

- if you use sun.misc features in your code
  - get rid of them & use replacements
- else
  - watch out for third-party software using it
  - libraries / frameworks might no longer work under Java 9

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.

replacements for critical sun.* internals

- some replacements available in Java 9
  - sun.reflect.Reflection.getCallerClass
    - replaced by a new "Stack-Walking API" (JEP 259)
  - sun.misc.Unsafe
    - major effort: several JEPs, some of them for Java 9

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.

uses of sun.reflect.Reflection.getCallerClass()

- find the immediate caller's class
  - caller-sensitive APIs in the JDK do this
  - e.g. Class.forName(), Class.getMethod(), ...
- filter out stack frames to find a specific class
  - java.util.logging API, Log4j, and Groovy runtime do this
  - skip intermediate implementation-specific stack frames to find the caller's class
- walk the entire stack
  - needed for stack trace of a Throwable and Thread.dumpStack()

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

replacement in Java 9

- Stack-Walking API (JEP 259)
  - an efficient standard API for stack walking
  - that allows easy filtering of stack traces, and
  - lazy access to stack traces

- class StackWalker

  `<T> T walk(Function<Stream<StackFrame>, T> function)`
  - opens a sequential stream of current thread's StackFrames
  - applies the function with the StackFrame stream

  `Class<?> getCallerClass()`
  - find the caller's frame
  - replaces sun.reflect.Reflection.getCallerClass()

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

stack walker example

- snapshot of current thread's stack trace

```
List<StackFrame> stack
= new StackWalker().walk(s->s.collect(Collectors.toList()));
```

- find first caller in stack with a particular trait

```
Class<?> find(Predicate<Class<?>> classTrait) {
  return new StackWalker().walk(
    s -> s.map(StackFrame::getDeclaringClass)
          .filter(classTrait)
          .findFirst()
   ).orElse(null);
}
```

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

use cases of sun.misc.Unsafe

- *common* use cases of sun.misc.Unsafe
  - enhanced atomic access
  - serialization
  - efficient memory management, layout, and access
  - interoperate across JVM boundary

- *uncommon* use cases
  - throw checked exception as unchecked
  - cast to an unrelated type, e.g. cast String to Integer
  - calculate shallow object size
  - corrupt memory
  - ...

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

enhanced atomic access

- uses
  - e.g. Unsafe.compareAndSwap or Unsafe.get/setVolatile
- reason
  - e.g. to avoid overhead of atomics or
  - to access array items with volatile semantic
- replacement
  - "Variable Handles" (JEP 193)
- when
  - Java 9

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.

serialization

- uses
  - e.g. Unsafe.allocateInstance
- reason
  - e.g. to speed up serialization or for mock object or proxies
- replacement
  - "Serialization 2.0" (JEP 187)
- when
  - some time in the future (Java 10, 11, ...)

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

efficient memory management

- uses
  - e.g. Unsafe.allocate/freeMemory or Unsafe.get/put (and JNI)
- reason
  - e.g. Guava library compares byte arrays using sun.misc.Unsafe
    to view byte[] as long[] to compare 8 bytes at a time
  - e.g. "big arrays" with 64-bit indices
- replacement
  - some smaller enhancements in Java 9
    - intrinsics for array bounds checks and array comparison => address the Guava use case
  - major efforts; some time in the future (Java 10, 11, ...)
    - Project Panama ("Interconnecting JVM and native code")
    - Project Valhalla ("Value Types / Generics Over Primitivies")
    - Arrays 2.0 (64-bit index, "frozen" arrays)

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

interoperate across JVM boundary

- uses
  - e.g. Unsafe.get/put (and JNI)
- reason
  - e.g. calling kernel methods or using native memory
- replacement
  - Project Panama ("Interconnecting JVM and native code")
  - "Foreign Function Interface" (JEP 191)
    - bind native functions to Java methods, and
    - directly manage blocks of native memory
- when
  - some time in the future (Java 10, 11, ...)

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

agenda

- "Collection Literals"
- "Compact Strings" and related JEPs
- Java Module System
- "Replace sun.misc.*" and related JEPs
- Enhanced Volatiles
- **JShell aka Java REPL**
- and many more ...

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

JShell

- JEP 222: JShell aka Java REPL
  - read-evaluate-print loop

- read-evaluate-print loop vs. edit-compile-execute cycle
  - type in Java code snippet, immediately see the result

- you don't need
  - class, main method
  - import (important ones included, add yours via start script)
  - semicolon (optional)

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

usage examples

- learn Java
  - immediate feedback without time consuming set-up
- explore new abstractions/libraries
  - create a new instance
  - get familiar with the API
  - see how it behaves
- get complex code right
  - e.g. Java 8 stream `collect()` with downstream collectors

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

usage details

- command line editing based on jline2 library

- auto-completion with tab-key

- command line in two modes
  - *code*: type in code snippet
  - *command*: type a JShell command
    - starts with '/', e.g.: /help

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

agenda

- "Collection Literals"
- "Compact Strings" and related JEPs
- Java Module System
- "Replace sun.misc.*" and related JEPs
- Enhanced Volatiles
- JShell aka Java REPL
- **and many more ...**

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.

more Java 9 features

- a new HTTP client API (JEP 110)
  - implements HTTP/2 and WebSocket
  - replaces the legacy HttpURLConnection API
- improved Process API (JEP 102)
  - for controlling and managing OS processes
- various JVM improvements
  - make G1 the default garbage collector (JEP 248)
  - unified JVM and GC logging (JEP 158 / 271)
  - additional diagnostic commands for Hotspot / JDK (JEP 228)
  - improved control of JVM (C1 / C2) compilers (JEP 165)
  - javadoc has a search box
  - ...

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

wrap-up

- Java 9 has a lot to offer
  - new features
    - collection literals, diamond for anon classes, ...
    - Java Module System
    - JShell aka Java REPL
    - unified JVM logging, lots of new flags, ...
  - optimizations
    - string compaction, deduplication, ...
    - VarHandles aka enhanced volatiles
  - gets rid of some legacy
    - replace sun.misc.*
    - carefully study the migration guide (docs.oracle.com/javase/9/migrate/toc.htm)

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

books on JPMS & Java 9 in general

Java 9 Modularity
Paul Bakker, Sander Mak
O'Reilly, September 2017

Java 9 Modularity Revealed
Alexandru Jecan
APress, September 2017

The Java Module System
Nicolai Parlog
Manning, Spring 2018 (estimated)

see related blog: blog.codefx.org

Java 9 Revealed
Kishori Sharan
APress, April 2017

55 New Features in JDK 9
Simon Ritter

https://www.youtube.com/watch?v=CMMzG8I23lY

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

Oracle documentation

The State of the Module System

Mark Reinhold, Oracle

openjdk.java.net/projects/jigsaw/spec/sotms/

Java SE Tools Reference

Section 2: Main Tools to Create and Build Applications

docs.oracle.com/javase/9/tools/main-tools-create-and-build-applications.htm

Oracle JDK 9 Migration Guide

docs.oracle.com/javase/9/migrate/toc.htm

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54

Q & A

Angelika Langer

http://www.AngelikaLanger.com
twitter: @AngelikaLanger

© Copyright 1995-2017 by Angelika Langer & Klaus Kreft. All Rights Reserved.
http://www.AngelikaLanger.com/
last update: 11/20/17,11:54