



Java™ NEW FEATURES

A close-up photograph of a hand holding a blue pen, poised to write on a piece of paper. The hand is wearing a grey, textured sweater sleeve. The background is blurred, showing more of the paper and the pen.

2

HELLO!

Technical Workshop Team

@Phong Pham

@Ky Huynh

@Thien Tai

@Dieu Pham



JAVA ROAD MAP

Jul Oct Jan 2018 Apr Jul Oct Jan 2019 Apr Jul Oct Jan 2020

9 - 9.0.1 — 9.0.4 — 10 — 10.0.1 — 10.0.2 — 11 - 11.0.1 — 11.0.2 — 12 - 12.0.1 — 12.0.2 — 13 - 13.0.1 — 13.0.2

Oracle JDK Releases

18.9 LTS

11 - 11.0.1 — 11.0.2 — 11.0.3 — 11.0.4 — 11.0.5 — 11.0.6

18.3

10 — 10.0.1 — 10.0.2

9 - 9.0.1 — 9.0.4

8u141 — 8u151 — 8u161 — 8u171 — 8u181 — 8u191 — 8u201 — 8u211 — 8u221 — 8u231 — 8u241
8u152 — 8u162 — 8u172 — 8u182 — 8u192 — 8u202 — 8u212 — 8u222 — 8u232 — 8u242

7u151 — 7u161 — 7u171 — 7u181 — 7u191 — 7u201 — 7u211 — 7u221 — 7u231 — 7u241 — 7u251

6u161 — 6u171 — 6u181 — 6u191 — 6u201 — 6u211

Last update
for JDK 6

OpenJDK: GPL Releases - Public

Java SE Advanced - Customers

BCL - Public

BCL - Public for non-commercial
Java SE Advanced for commercial use

Last reviewed on 2018/02

All future release dates subject to change

4



Martin Thompson

**“What goes into
Java is more
important than
how often we
get releases”**

5

AGENDA

- ▶ Module System
- ▶ Reactive Streams
- ▶ Stream, Collections API Improvements
- ▶ Miscellaneous Features
- ▶ Challenge: Migrate a Java 8 application

6

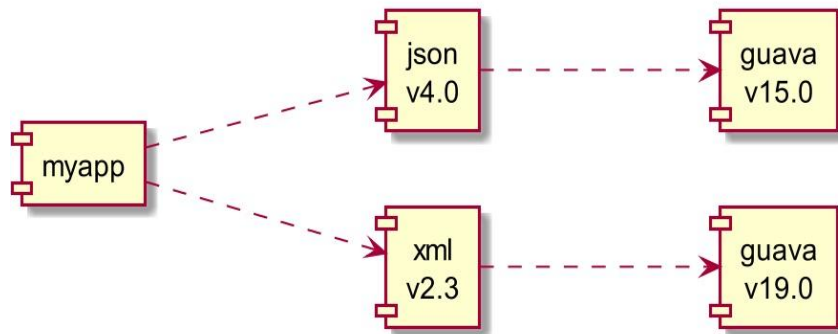
MODULE SYSTEM

7

ClassNotFoundException? NoClassDefFoundError?

Module System

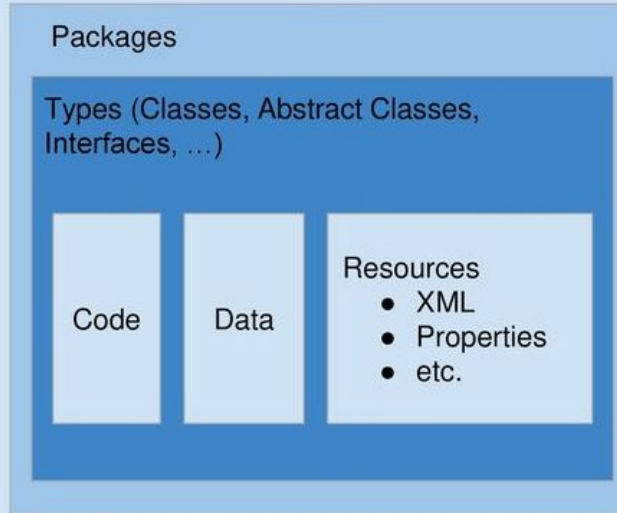
JAR HELL



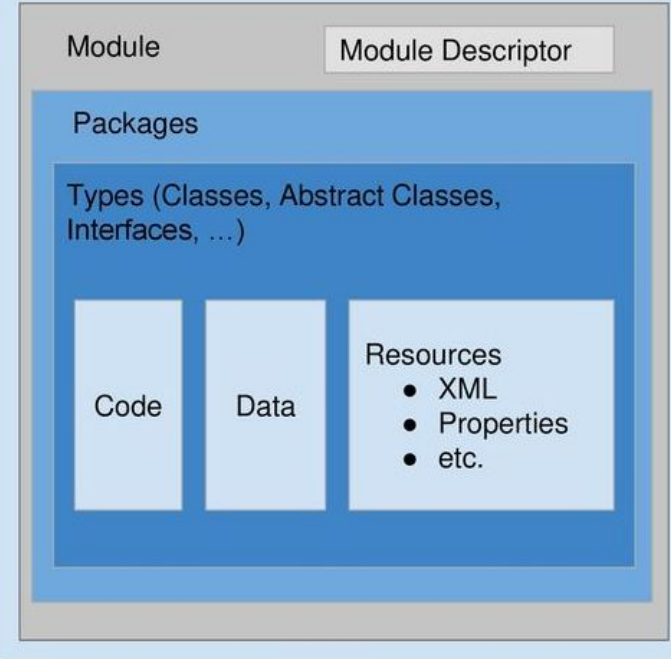
8

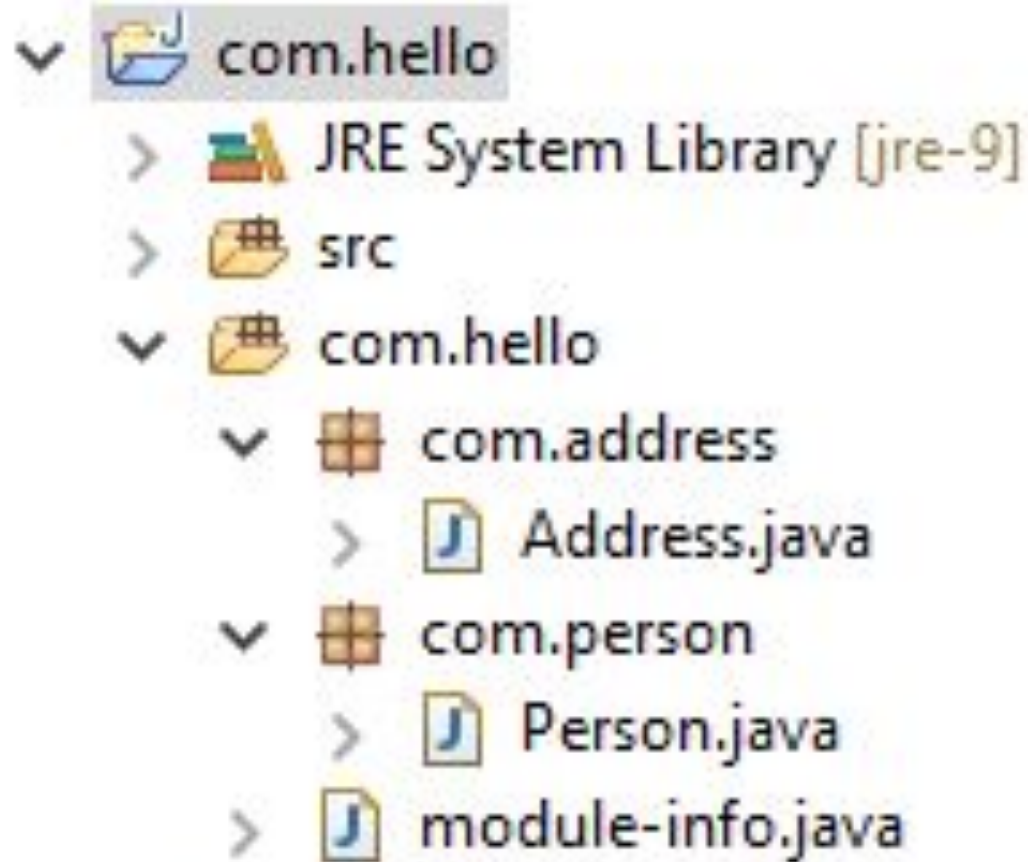
Module System

Java 8 Application



Java 9 Application





10

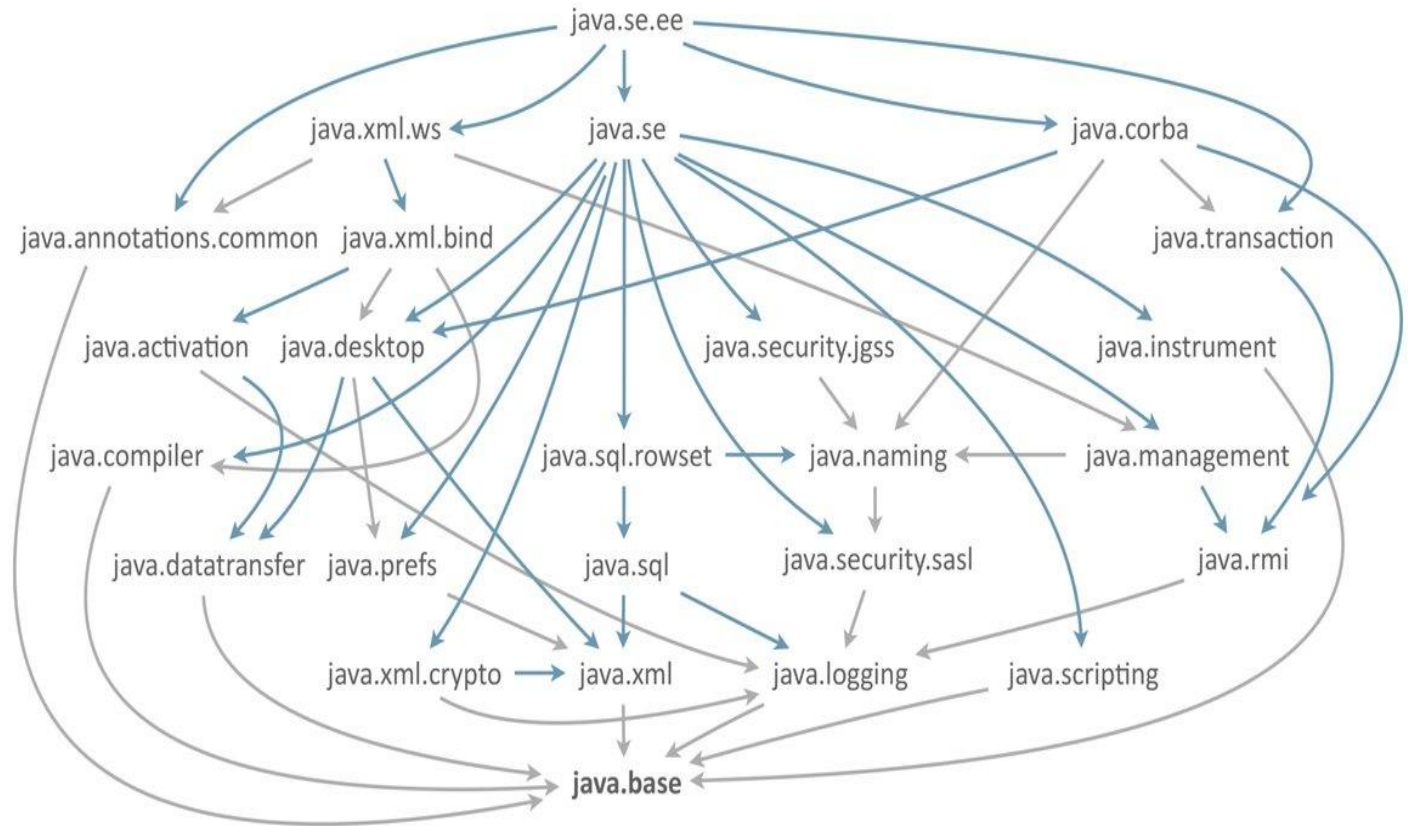
Module System

```
// module-info.java
```

```
module com.hello {  
    exports com.person;  
    exports com.address to com.hello.client;  
}
```

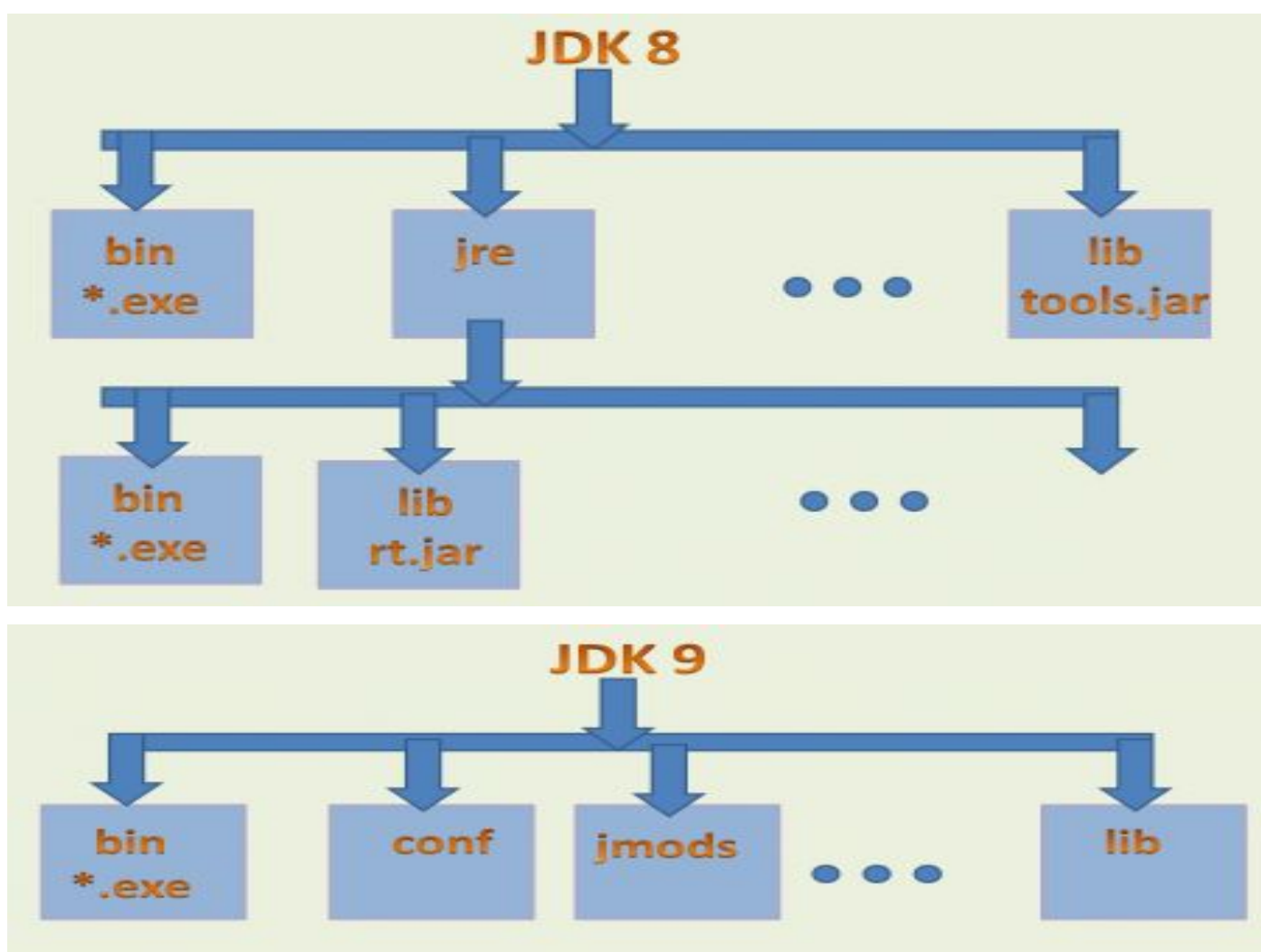
```
module com.hello.client {  
    requires com.hello;  
}
```

Reliable Configuration



12

Module System



13

Module System

Monolithic JDK



14

Module System



15

Module
System





DEMO

17

REACTIVE STREAMS

18

Reactive
Streams

Stream

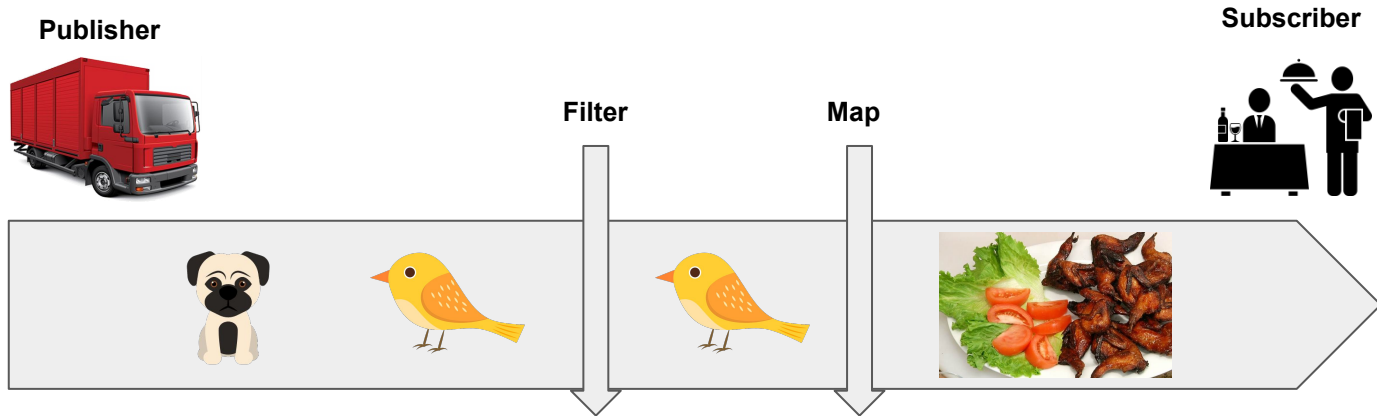
Asynchronous

Observer

Backpressure

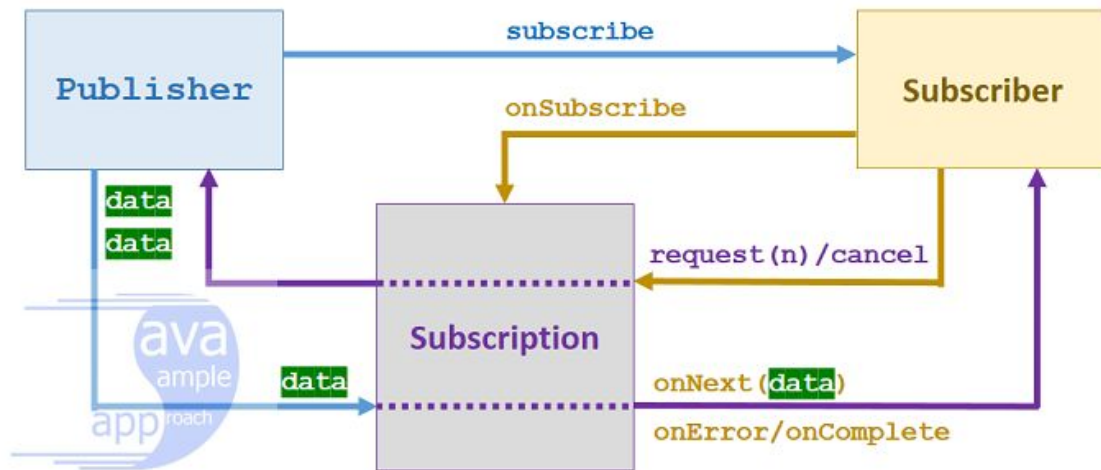
19

Reactive Streams



20

Reactive Streams



Flow API:
`java.util.concurrent.Flow.*`



DEMO

22

STREAM API IMPROVEMENTS

23

Collections

// Java 8

```
List<String> list = new ArrayList<>();  
list.add("Workshop");  
list.add("Java New Features");
```

```
List<String> immutableList = Collections.unmodifiableList(list);
```

// Java 9

```
List<String> immutableList = List.of("Workshop", "Java New  
Features");
```

// List.of() doesn't allow null values => NullPointerException

```
Set.of();
```

```
Set.of("Hello", "World");
```

```
Map.of();
```

```
Map.of(1, "Workshop", 2, "Java New Features");
```

```
// don't allow null keys or values, duplicate element.
```


takeWhile()

```
Stream.of(  
    "Workshop ",  
    "Java New Feature ",  
    "Axon Active",  
    "",  
    "Can Tho",  
    "Branch"  
).takeWhile(s -> !s.isEmpty())  
  .forEach(System.out::printf);
```

Workshop Java New Feature Axon Active Can Tho Branch

Workshop Java New Feature Axon Active

26

Streams

dropWhile()

```
Stream.of(  
    "Workshop",  
    "Java New Feature",  
    "Axon Active",  
    "",  
    "Can Tho ",  
    "Branch"  
).dropWhile(s -> !s.isEmpty())  
  .forEach(System.out::printf);
```

Workshop Java New Feature Axon Active
Can Tho Branch

```
List<Integer> ages = List.of(20, 25, 40,  
                             13, 30, 8,  
                             16, 15, 4, 9);  
  
ages.stream()  
    .takeWhile(age -> age > 8)  
    .dropWhile(age -> age % 5 == 0)  
    .forEach(System.out::println);
```



28

Streams

```
List<Integer> ages = List.of(20, 25, 40,  
                             13, 30, 8,  
                             16, 15, 4, 9);
```

```
ages.stream()  
    .takeWhile(age -> age > 8)  
    .dropWhile(age -> age % 5 == 0)  
    .forEach(System.out::println);
```

```
// takeWhile: 20, 25, 40, 13, 30, 8, 16, 15, 4, 9
```

```
// dropWhile: 20, 25, 40, 13, 30
```

```
// 13, 30
```

```
// Java 8
Optional<Integer> result = Optional.empty();

if(result.isPresent()) {
    System.out.println("Result " + result.get());
} else {
    System.out.println("Empty " + result.orElse(222));
}
```

```
// Java 9
Optional<Integer> result = Optional.empty();

result.ifPresentOrElse(
    x -> System.out.println("Result " + x),
    () -> System.out.println("Empty")
);
```

30

Optional

```
public static void main(String[] args) {  
    System.out.println(getOptionalEmpty()  
        .or(() -> getOptionalHasEmptyValue())  
        .or(() -> getOptionalHasAnotherValue()));  
}  
  
private static Optional<String> getOptionalEmpty() {  
    return Optional.empty();  
}  
  
private static Optional<String> getOptionalHasEmptyValue() {  
    return Optional.empty();  
}  
  
private static Optional<String> getOptionalHasAnotherValue(){  
    return Optional.empty();  
}  
}
```

31

MISCELLANEOUS FEATURES

JEP 222: jshell: The Java Shell

```
[pankaj:~ pankaj$ jshell  
| Welcome to JShell -- Version 9  
| For an introduction type: /help intro
```

```
[jshell> 10+5  
$1 ==> 15
```

```
[jshell> 10/5  
$2 ==> 2
```

```
[jshell> 10/3  
$3 ==> 3
```

```
[jshell> 10.0/3  
$4 ==> 3.3333333333333335
```

```
[jshell> 10*5  
$5 ==> 50
```


JEP 286: Local-Variable Type Inference

```
// infers ArrayList<String> and Stream<String>  
List<String> list = new ArrayList<>();  
Stream stream = list.stream();
```

```
// equivalent to  
var stream = list.stream();  
var list = new ArrayList<String>();
```

JEP 323: Local-Variable Syntax for Lambda Parameters

```
(x, y) -> x.process(y)
```

```
// equivalent to
```

```
(var x, var y) -> x.process(y)
```

```
// uniformity
```

```
@NonNull var x = new Foo();
```

```
(@NonNull var x, @Nullable var y) -> x.process(y)
```

JEP 323: Local-Variable Syntax for Lambda Parameters

// Cannot mix 'var' and 'no var'

```
(var x, y) -> x.process(y)
```

// Cannot mix 'var' and manifest types

```
(var x, int y) -> x.process(y)
```

JEP 320: Remove the Java EE and CORBA Modules

```
java.xml.ws           // JAX-WS
java.xml.bind          // JAXB
java.activation        // JAF
java.xml.ws.annotation // Common Annotations
java.corba             // CORBA
java.transaction       // JTA
java.se.ee             // Aggregator module for the 6 version
jdk.xml.ws            // Tools for JAX-WS
jdk.xml.bind          // Tools for JAXB
```

JEP 330: Launch Single-File Source-Code Programs

```
// run a program supplied as a single file of  
java source code
```

```
java -classpath /home/foo/java Hello.java Bonjour
```

```
// equivalent to
```

```
javac -classpath /home/foo/java Hello.java
```

```
javac -classpath /home/foo/java Hello Bonjour
```

JEP 230: Microbenchmark Suite

JEP 325: Switch Expressions

JEP 326: Raw String Literals (dropped from JDK 12 release)

JEP 334: JVM Constants API

JEP 340: One AArch64 Port, Not Two

39

THANKS!

Any questions?





40

CHALLENGE: MIGRATE A JAVA 8 APPLICATION

41



Banana

- ▶ **Center** as a server application to manage employee, including time management.
- ▶ **Timer** as a desktop app allows check-in and check out.

As **manager**,

Java has released a new version, and we want to **migrate the center server to version 12** so that we can the **benefits of upgrading the application to Java 12**.

- ▶ Migration can be done incrementally: **run, compile, modularize.**
- ▶ Full source code and it should be clean.
- ▶ Reactive streams.
- ▶ Enhancements.



45

Information

Time: 9:00 - 11:45

Rule: The Technical Workshop Team will make final decision.

Result: Full source code

Date of Result: 31/07/2019

46

How-to

1. Enter **bit.ly/banana-company**
2. Prepare workspace.
3. Preliminary investigation.
4. Write down your tasks and who will implement it.
5. Enjoy!

1. Exploring Java 9, Fu Cheng

Build Modularized Applications in Java

2. Java 9 Modularity, Paul Bakker, Sander Mak

Patterns and Practices for Developing Maintainable Applications

3. Reactive Programming With Java 9, Tejaswini Mandar Jog

Develop concurrent and asynchronized application with Java 9