Java™ NEW FEATURES

AXON ACTIVE
www.axon.vn

HELLO!

Technical Workshop Team

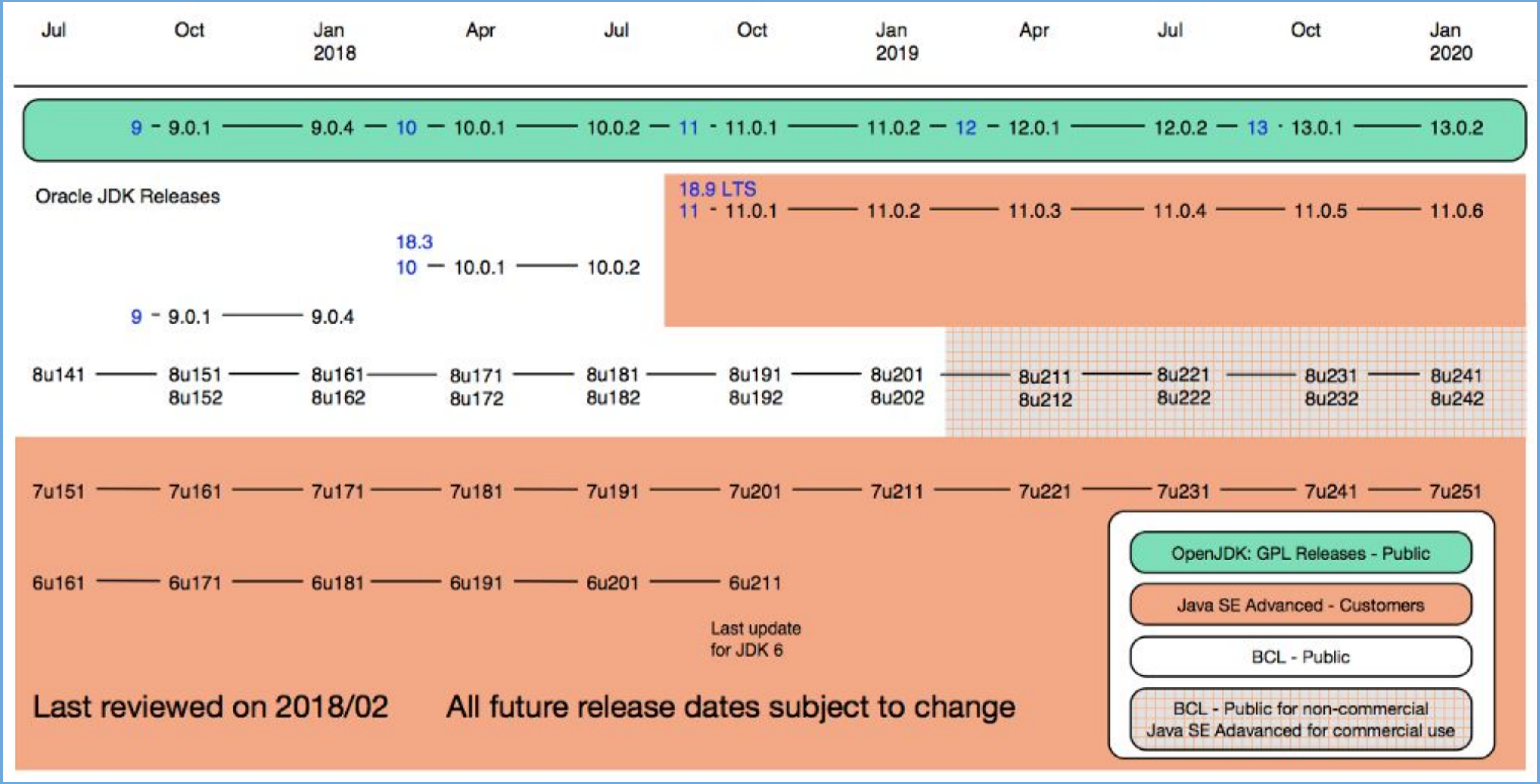
@Phong Pham

@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 8u152 — 8u161 8u162 — 8u171 8u172 — 8u181 8u182 — 8u191 8u192 — 8u201 8u202 — 8u211 8u212 — 8u221 8u222 — 8u231 8u232 — 8u241 8u242
7u151 — 7u161 — 7u171 — 7u181 — 7u191 — 7u201 — 7u211 — 7u221 — 7u231 — 7u241 — 7u251
6u161 — 6u171 — 6u181 — 6u191 — 6u201 — 6u211
Last update for JDK 6
Last reviewed on 2018/02
All future release dates subject to change
OpenJDK: GPL Releases - Public
Java SE Advanced - Customers
BCL - Public
BCL - Public for non-commercial
Java SE Adavanced for commercial use

Martin Thompson

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

AGENDA

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

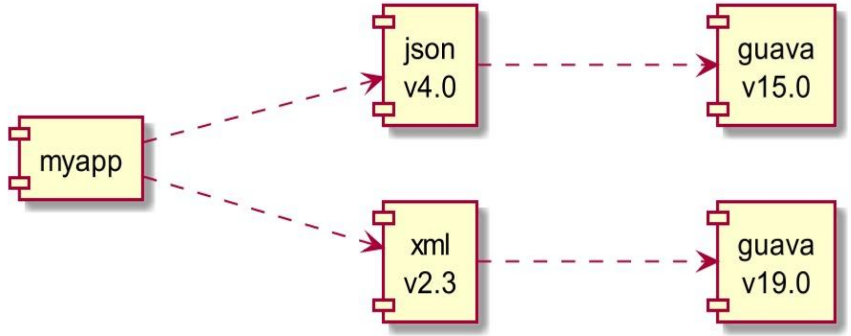
6

MODULE SYSTEM

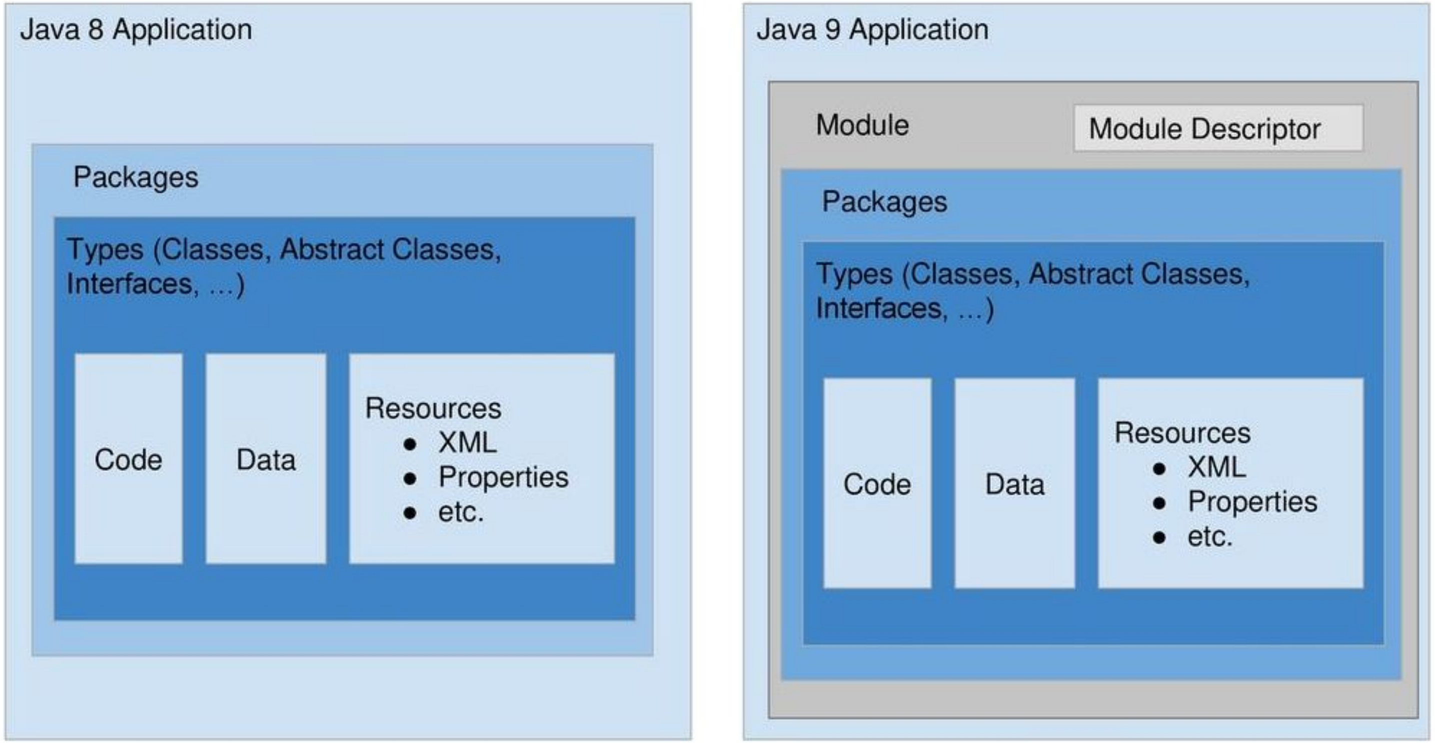
Module System

ClassNotFoundException?

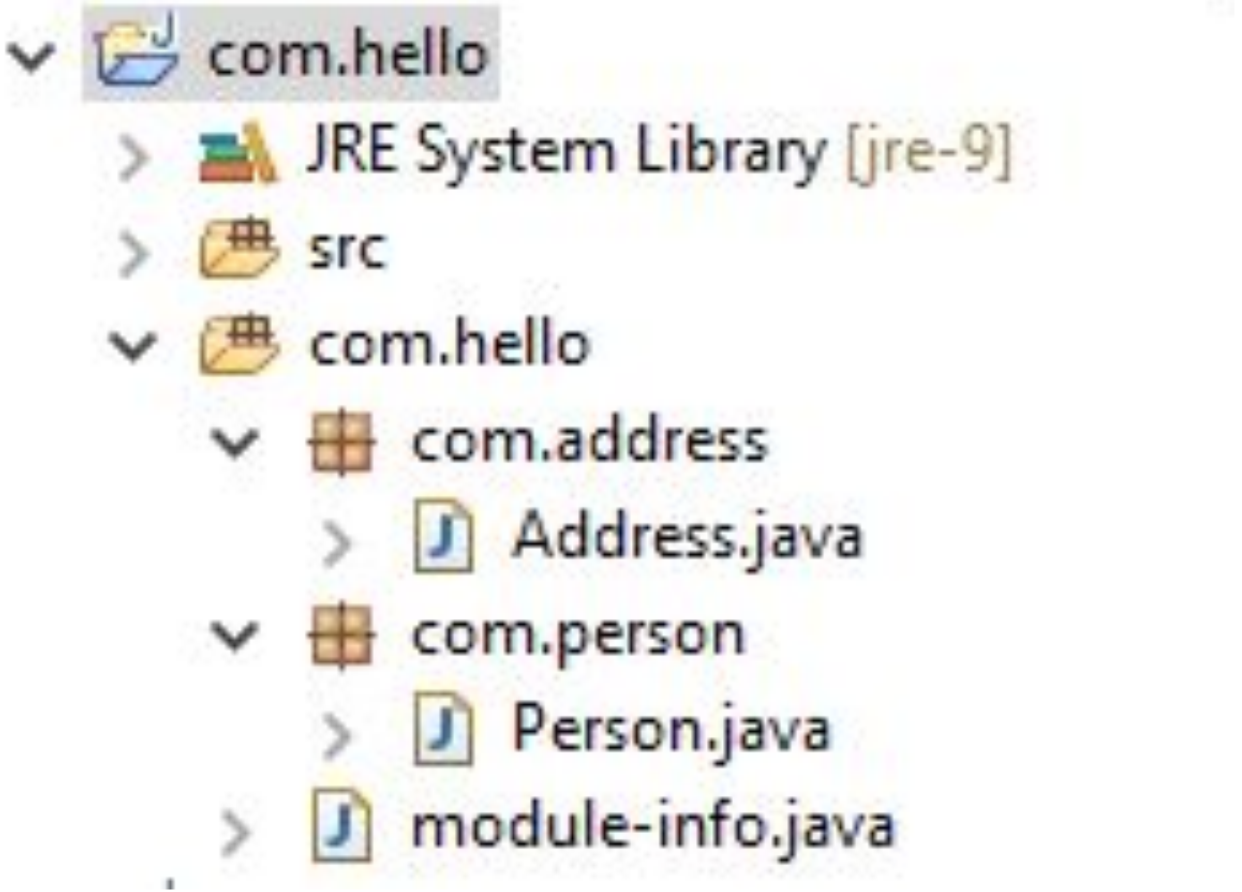
NoClassDefFoundError?



Module System



Module System



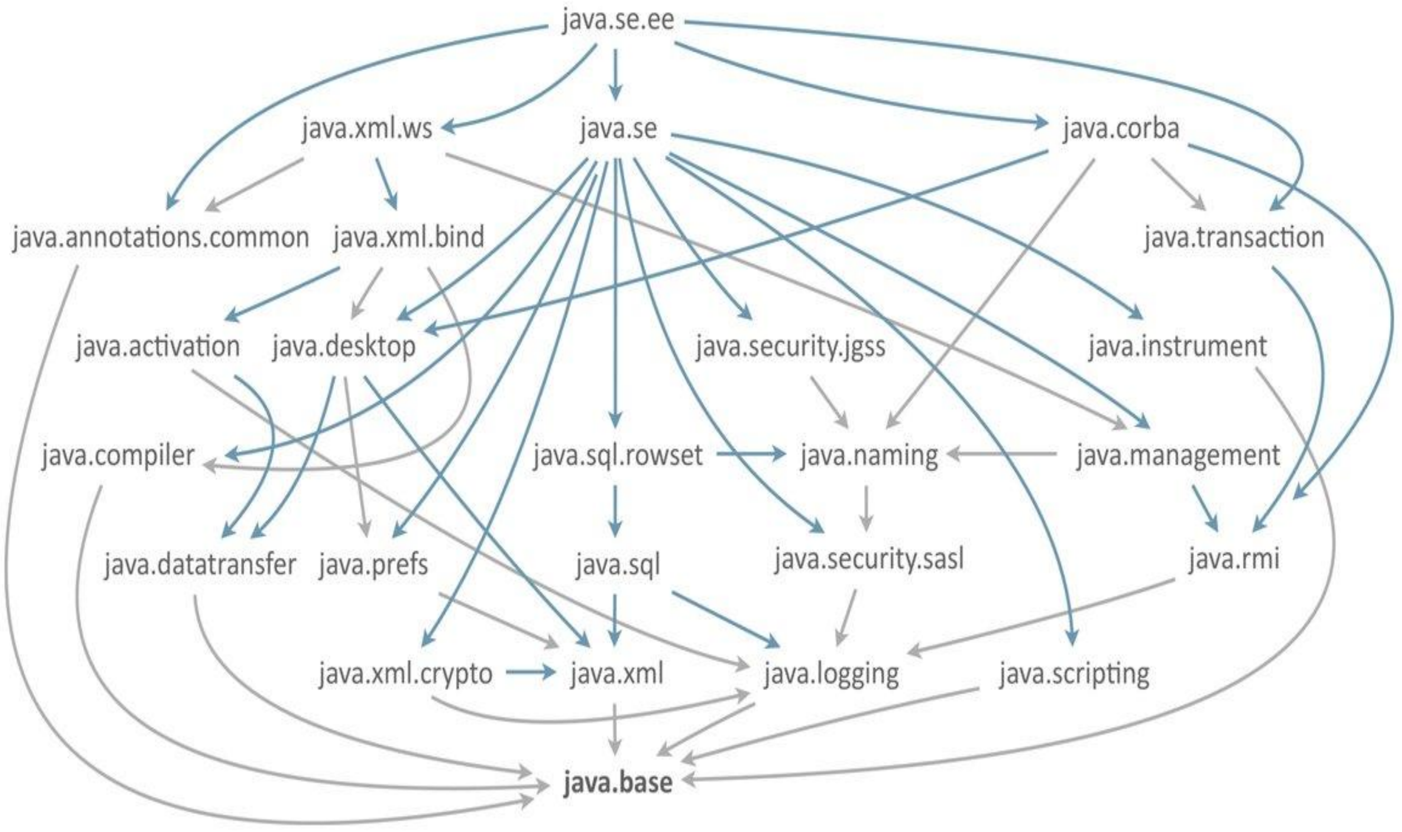
```
com.hello
  JRE System Library [jre-9]
  src
  com.hello
    com.address
      Address.java
    com.person
      Person.java
    module-info.java
```

Module System

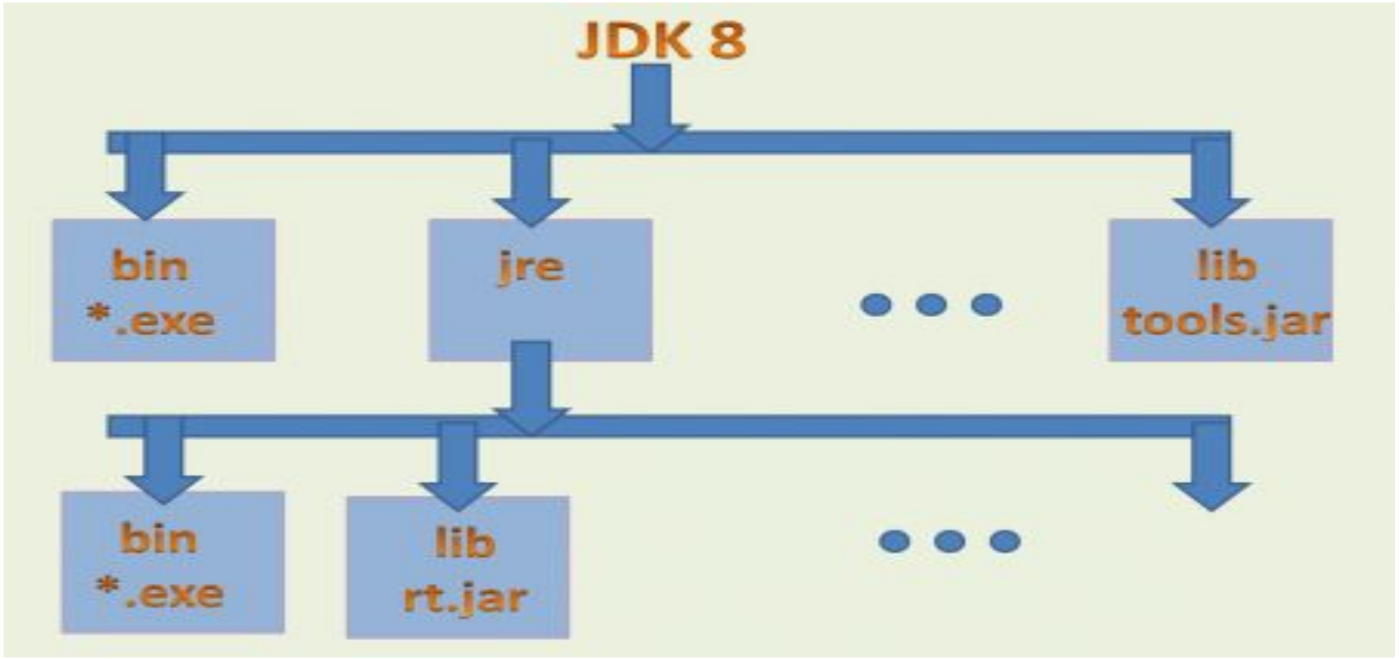
```
// module-info.java

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

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

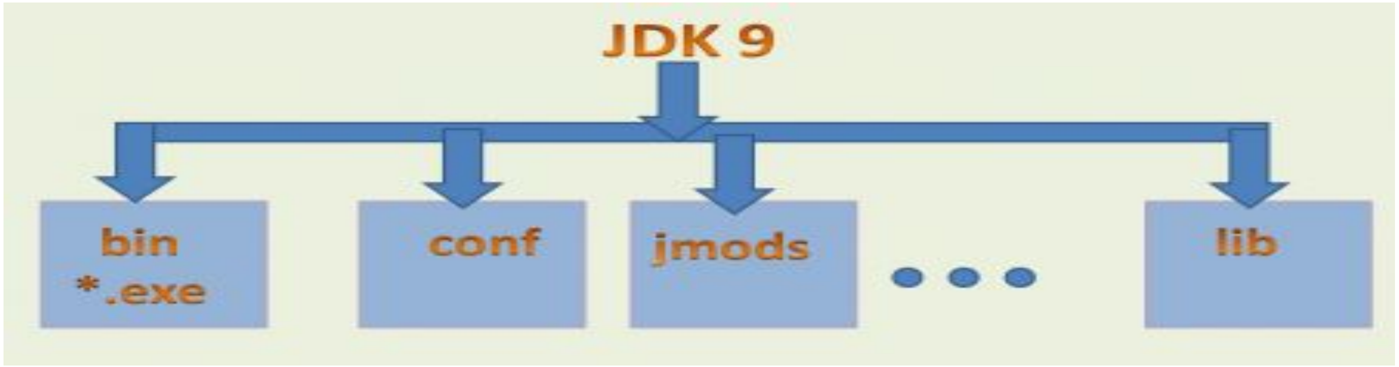
Module System

Reliable Configuration



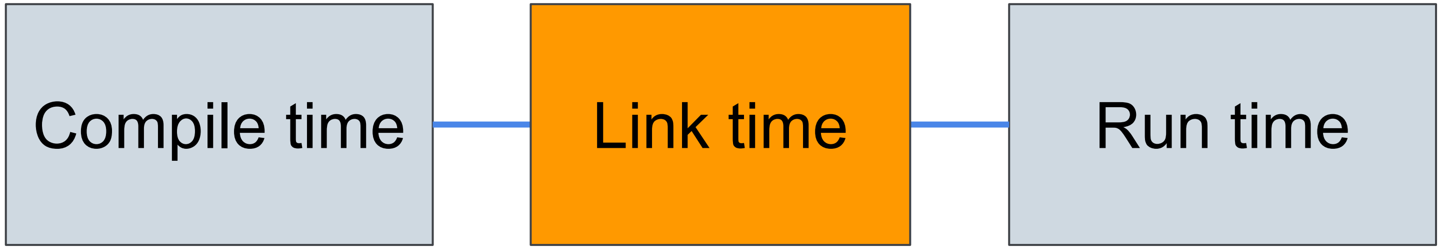
Module System



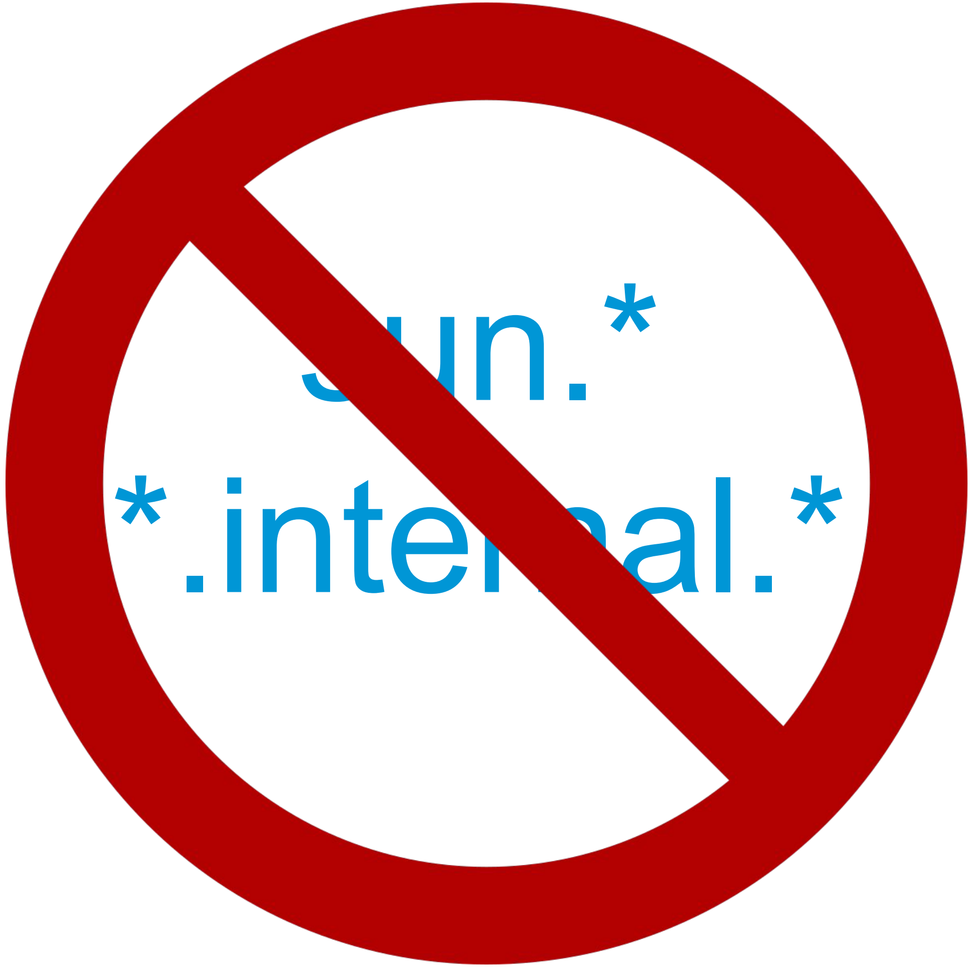


Monolithic JDK

Module System



Module System



DEMO

17

REACTIVE STREAMS

Reactive Streams

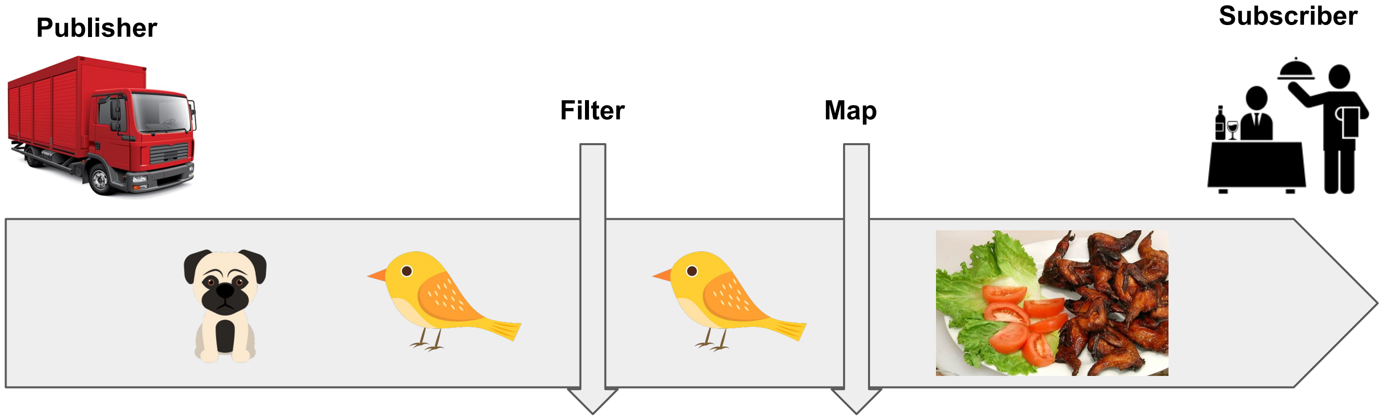
Stream

Asynchronous

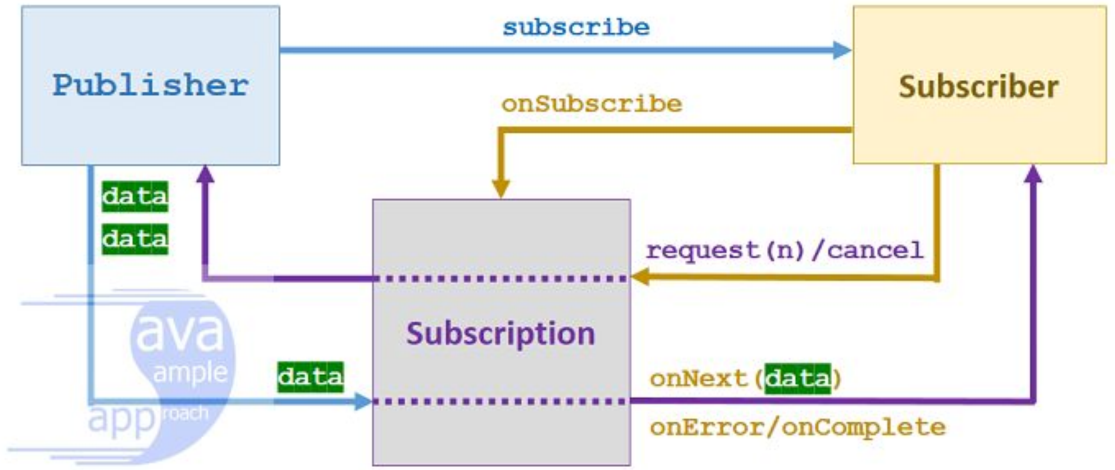
Observer

Backpressure

Reactive Streams



Reactive Streams



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

DEMO

STREAM API IMPROVEMENTS

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
```

Collections

```
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.
```

Streams

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
```

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
```

Streams

```java
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);
```

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
```

29

Optional

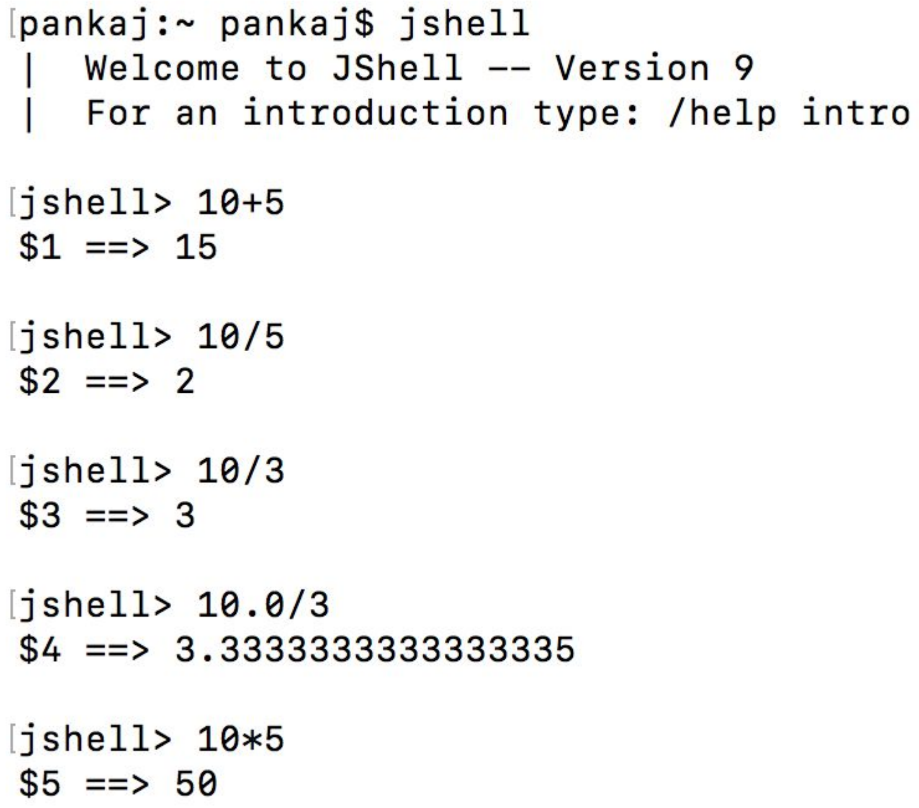
```
// Java 8
Optional<Integer> result = Optional.empty();

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")
);
```

Optional

```java
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();
    }
}
```

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
```

JAVA 10

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
```

JAVA 12

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

THANKS!

Any questions?

CHALLENGE: MIGRATE A JAVA 8 APPLICATION

Banana

Description

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

Challenge

As **manager**,

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

Acceptance Criterias

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

Information

Time: 8:00 - 11:45

Rule: The Technical Workshop Team will make final decision.

Result: Full **migrated** source code

Date of Result: 31/07/2019

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!

REFERENCES AND COURSES ONLINE

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