

# WHAT'S NEW IN JAVA 9

BILLY KORANDO @BILLYKORANDO

SOFTWARE CONSULTANT - KEYHOLE SOFTWARE

<http://bit.ly/2kVj9Hk>

## PLATINUM SPONSORS



## GOLD SPONSORS



Asynchrony Labs



deckcommerce



## SILVER SPONSORS



Conference Notebook  
& Party

Promotional Items & Attendee  
Lanyards

Attendee Swag Bags

Media





**JAVA 9 WENT GA ON  
SEPTEMBER 21ST!**

<http://bit.ly/2kVj9Hk>

# MISCONCEPTIONS ABOUT JAVA 9

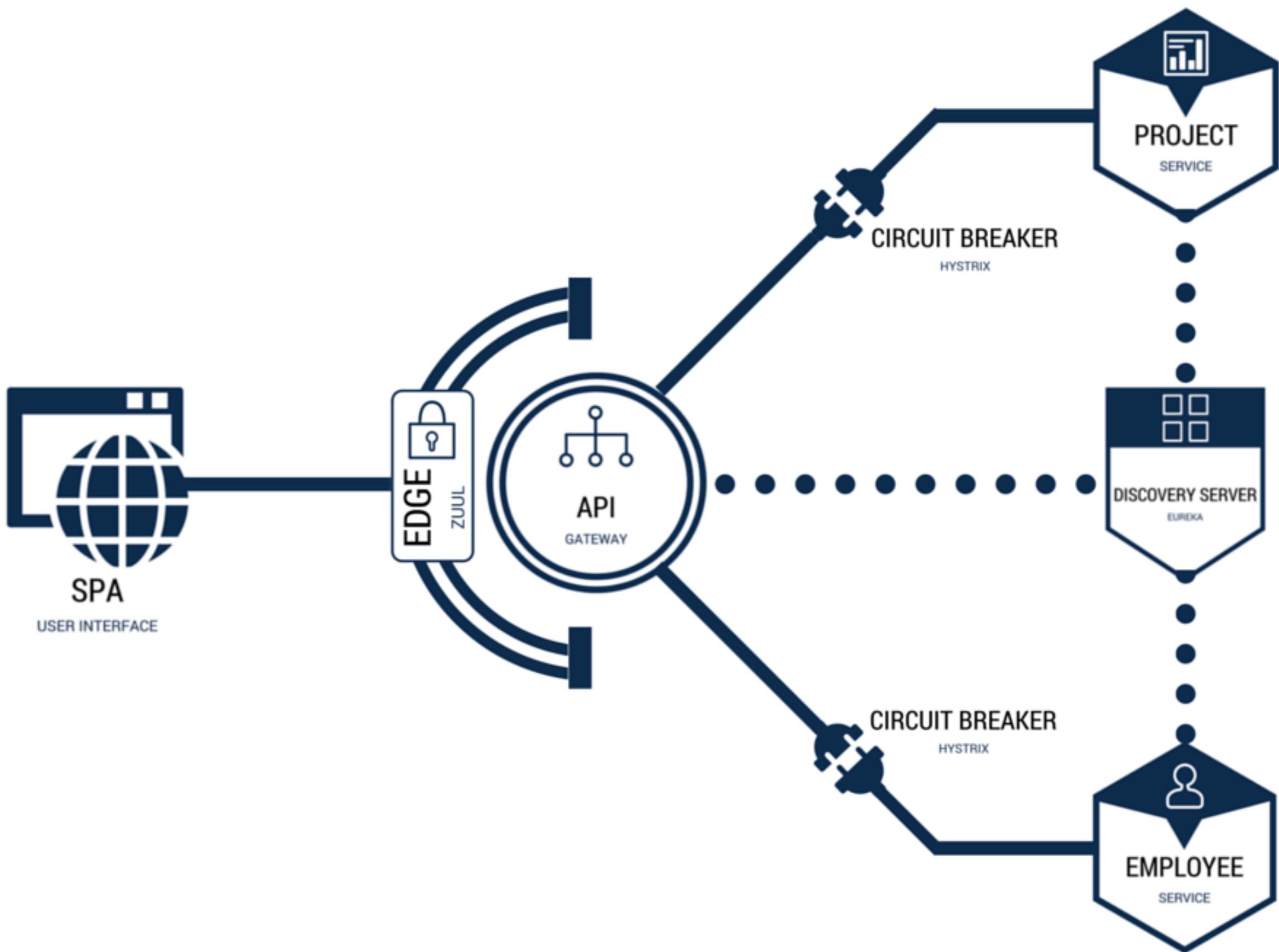
---

- 1. ~~MY BUILD TOOL WILL NOT WORK~~**
- 2. ~~I'LL HAVE TO MODULARIZE MY CODE~~**
- 3. ~~MY LIBRARIES WILL NOT WORK \*~~**

SOME LIBRARIES WILL NEED TO  
UPGRADED

# PROVE IT!

<http://bit.ly/2kVj9Hk>



<http://bit.ly/2kVj9Hk>

---

# **AN UPGRADE WITH BENEFITS**

<http://bit.ly/2kVj9Hk>

## BENEFITS OF RUNNING IN JAVA 9

---

Better Use of Memory

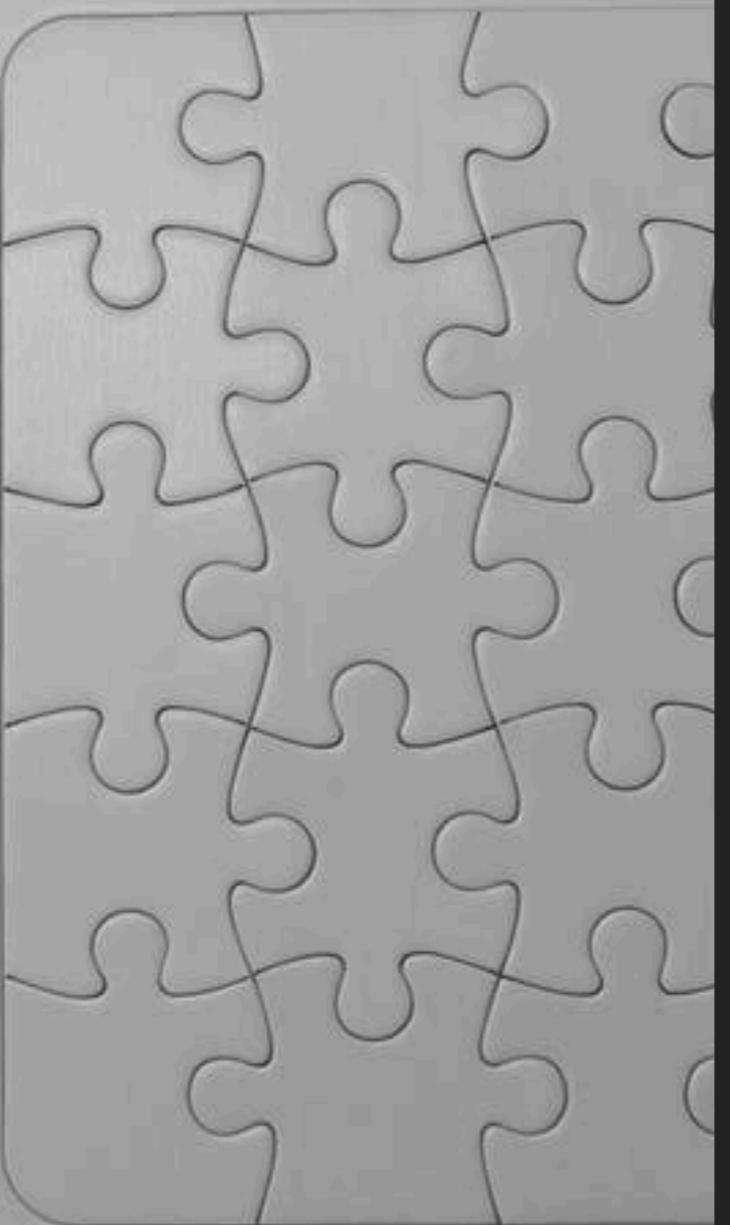
Better Performance

Better Use of Hardware

Better Documentation

Prettier Graphics

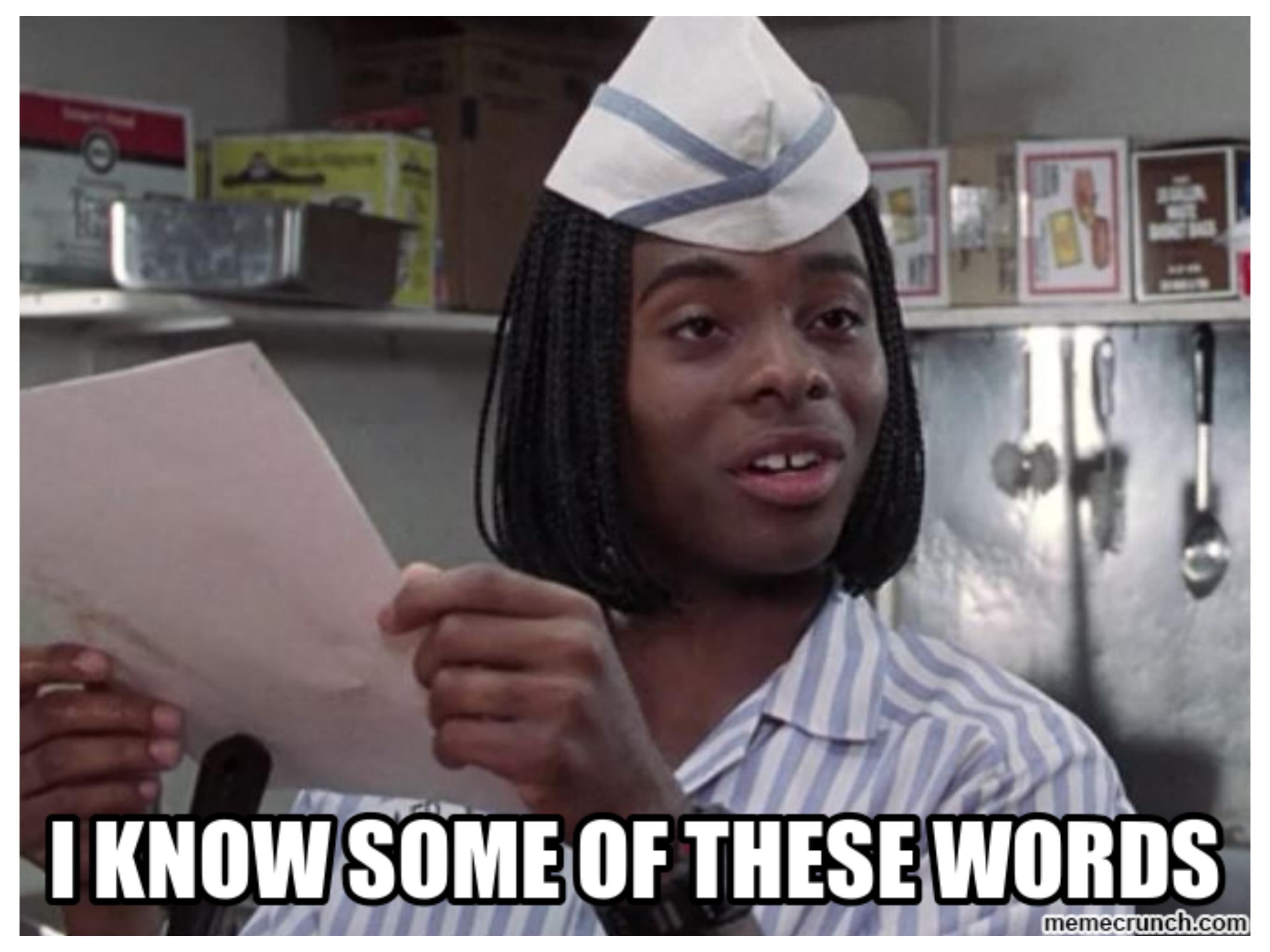
Faster Compilation



---

# JIGSAW

<http://bit.ly/2kVj9Hk>

A black and white photograph of a woman with dark hair, wearing a white sailor-style hat with blue stripes and a striped shirt. She is looking upwards and pointing her right index finger towards the top left corner of the frame. In her left hand, she holds a large, light-colored menu or piece of paper. The background shows shelves with various items, possibly in a shop or restaurant setting.

**I KNOW SOME OF THESE WORDS**

# JIGSAW GOALS

---

- ▶ Stronger encapsulation
- ▶ More reliable configuration
- ▶ Create customized JDK images (jlink tool)

# Module Types

## UNNAMED

- Jars on classpath become unnamed modules
- Exports all packages
- Requires all modules
- Cannot be declared as a dependency

## AUTOMATIC

- Jars placed on module path
- Name derived from jar name or optional  
Automatic-Module-Name in MANIFEST.MF
- Exports all packages
- Requires all modules
- Can be declared as a dependency

## EXPLICIT/NAMED

- Defined by module-info.java
- Can be declared as a dependency

## Current file structure

```
src/  
  com.foo.bar/  
    com.foo.bar.model/  
      Foo.java  
  com.foo.bar.service/  
    FooService.java  
  com.foo.bar.service.impl/  
    FooServiceImpl.java
```

# Modularized file structure

```
src/
  com.foo.bar/
    module-info.java
    com.foo.bar/
      com.foo.bar.model/
        Foo.java
      com.foo.bar.service/
        FooSvc.java
      com.foo.bar.service.impl/
        FooSvclmpl.java
```

# Dissecting module-info

Note:

New reserved words ONLY  
reserved in module-info

```
module com.foo.bar {  
    exports com.foo.bar.service;  
    exports com.foo.bar.model;  
  
    provides com.foo.bar.service.FooSvc  
        with  
        com.foo.bar.service.impl.FooSvclmpl;  
  
    uses org.evil.service.EvilService;  
  
    requires org.evil;  
    requires java.sql;  
}
```

<http://bit.ly/2kVj9Hk>

# Module Declaration

Defines the name of the module.

Additional modifiers:

“open”

exports all packages and  
allows deep reflective access

```
module com.foo.bar {
```

```
    exports com.foo.bar.service;
```

```
    exports com.foo.bar.model;
```

```
    provides com.foo.bar.service.FooSvc  
        with
```

```
        com.foo.bar.service.impl.FooSvclmpl;
```

```
    uses org.evil.service.EvilService;
```

```
    requires org.evil;
```

```
    requires java.sql;
```

```
}
```

<http://bit.ly/2kVj9Hk>

# Export Declaration

Defines a module's public API

Note:

Each exported package must  
be defined individually

Additional modifiers:

“opens” (in lieu of exports)

allows deep reflective access to  
all client modules

“to” [module name]

allows only named modules to  
have deep reflective access

```
module com.foo.bar {
```

```
    exports com.foo.bar.service;
```

```
    exports com.foo.bar.model;
```

```
    provides com.foo.bar.service.FooSvc  
        with
```

```
        com.foo.bar.service.impl.FooSvclmpl;
```

```
    uses org.evil.service.EvilService;
```

```
    requires org.evil;
```

```
    requires java.sql;
```

```
}
```

<http://bit.ly/2kVj9Hk>

## Provides

Declares the service this module supports and the implementation it supports it with.

```
module com.foo.bar {
```

```
    exports com.foo.bar.service;  
    exports com.foo.bar.model;
```

```
    provides com.foo.bar.service.FooSvc  
        with  
        com.foo.bar.service.impl.FooSvclmpl;
```

```
    uses org.evil.service.EvilService;
```

```
    requires org.evil;  
    requires java.sql;
```

```
}
```

<http://bit.ly/2kVj9Hk>

## Uses

Declares a service this module depends upon

```
module com.foo.bar {  
    exports com.foo.bar.service;  
    exports com.foo.bar.model;  
  
    provides com.foo.bar.service.FooSvc  
        with  
        com.foo.bar.service.impl.FooSvclmpl;  
  
    uses org.evil.service.EvilService;  
  
    requires org.evil;  
    requires java.sql;  
}
```

<http://bit.ly/2kVj9Hk>

# Requires

Modules this module depends upon.

Additional modifiers:

“transitive” dependent modules don’t have to additionally require this module to make use of it

Note:

`java.base` implicitly required by all modules

```
module com.foo.bar {
```

```
    exports com.foo.bar.service;
```

```
    exports com.foo.bar.model;
```

```
    provides com.foo.bar.service.FooSvc  
        with
```

```
        com.foo.bar.service.impl.FooSvclmpl;
```

```
    uses org.evil.service.EvilService;
```

```
    requires org.evil;
```

```
    requires java.sql;
```

```
}
```

<http://bit.ly/2kVj9Hk>



---

# JHELL

<http://bit.ly/2kVj9Hk>

Read

Evaluate

Print

Loop

# WHY JSHELL IS PART OF JAVA 9

---

- ▶ Explore APIs
- ▶ Tool to Teach Java

# WHY JSHELL IS PART OF JAVA 9

---

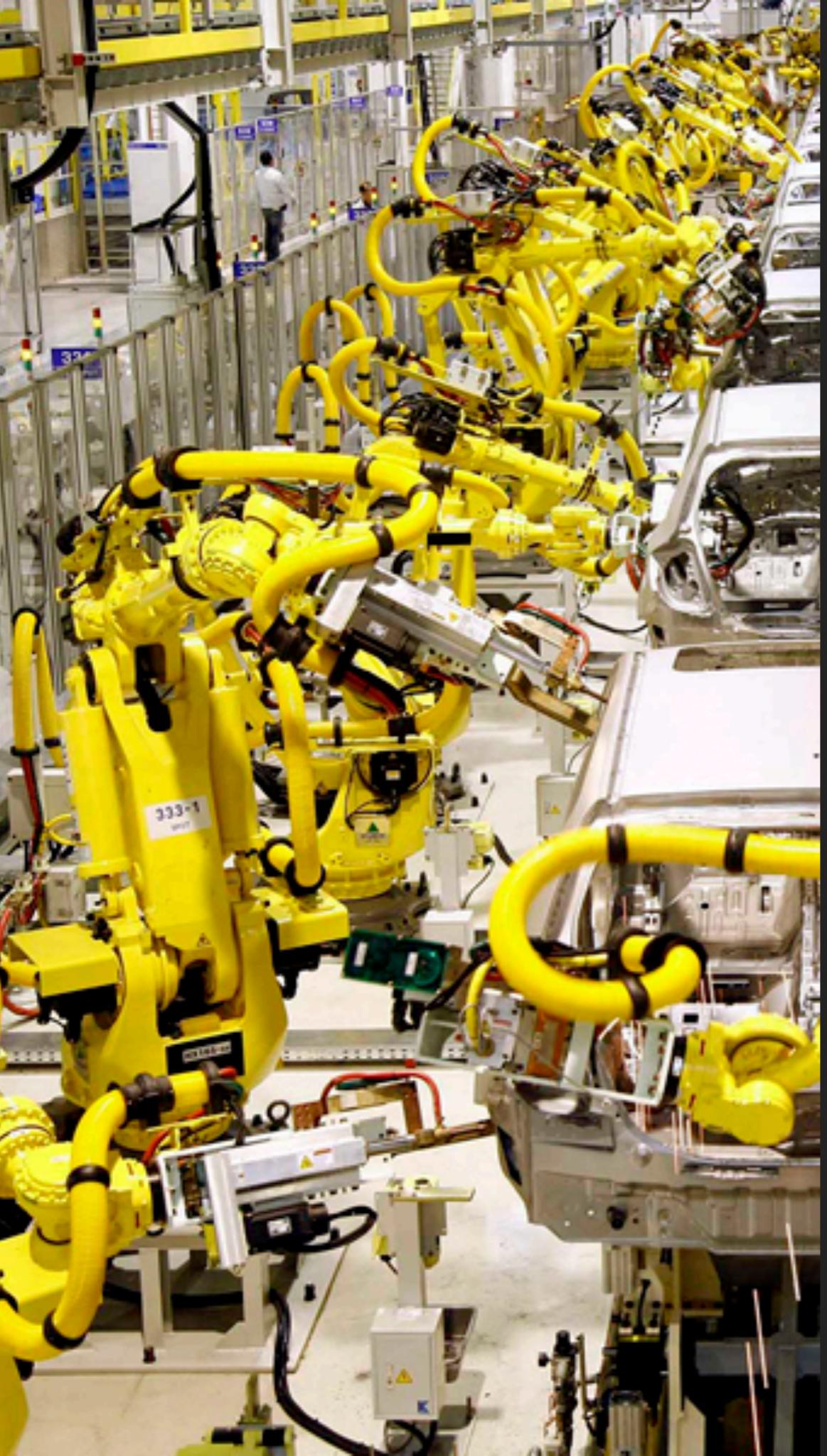
- ▶ Explore APIs
- ▶ Tool to Teach Java

```
public class HelloWorld{  
    public static void main(String args[]) {  
        System.out.println("Hello World!");  
    }  
}
```

```
public class HelloWorld{  
    public static void main(String args[]) {  
        System.out.println("Hello World!");  
    }  
}
```

# DEMO

<http://bit.ly/2kVj9Hk>



---

# COLLECTION FACTORY METHODS

<http://bit.ly/2kVj9Hk>

```
public class TestFoo{  
  
    @Test  
    public void test() {  
        Foo newFoo = new Foo();  
        List<String> list = new ArrayList();  
        list.add("Hello World!");  
        list.add("Goodbye World!");  
        newFoo.takeListOfStrings(list);  
    }  
  
}
```

```
public class TestFoo{  
  
    @Test  
    public void test() {  
        Foo newFoo = new Foo();  
        List<String> list = List.of(  
            "Hello World!", "Goodbye World!");  
        newFoo.takeListOfStrings(list);  
    }  
}
```

```
public class TestFoo{  
  
    @Test  
    public void test() {  
        Foo newFoo = new Foo();  
        Map<Integer, String> map = Map.of(  
            1, "Hello World!", 2, "Goodbye  
            World!"  
        );  
        newFoo.readAMap(map);  
    }  
}
```

```
public class TestFoo{  
  
    @Test  
    public void test() {  
        Foo newFoo = new Foo();  
        Map<Integer, String> map =  
        Map.ofEntries(  
            Map.entry( 1, "Hello World!" ),  
            Map.entry( 2, "Goodbye World!" )  
        );  
        newFoo.readAMap( map );  
    }  
}
```



---

# STREAMS UPDATES

<http://bit.ly/2kVj9Hk>

```
public class TakeKansasCitySportsTeams {  
  
    public void takeTeams () {  
        Stream.of("Chiefs", "Royals",  
                 "Sporting", "Mavericks", "", "Monarchs")  
            .takeWhile(s -> !String.isEmpty(s))  
            .forEach(String.out::print)  
    }  
  
}
```

Output: ChiefsRoyalsSportingMavericks

```
public class DropKansasCitySportsTeams {  
  
    public void dropTeams () {  
        Stream.of("", "Chiefs", "Royals",  
                "Sporting", "Mavericks")  
            .dropWhile(s -> !String.isEmpty(s))  
            .forEach(String.out::print)  
    }  
  
}
```

Output: ChiefsRoyalsSportingMavericks



---

# PRIVATE INTERFACE METHODS

<http://bit.ly/2kVj9Hk>

```
public interface Foo{  
  
    default void bar(String arg) {  
        System.out.println("Hello World!");  
    }  
  
    private default String lorem() {  
        ...  
    }  
}
```

---

# TRY-WITH-RESOURCES

# UPDATES

<http://bit.ly/2kVj9Hk>

```
public class TryWithFoo {  
  
    public void TryWithBar() {  
        try(Resource bar = new Resource()) {  
            //use bar  
        }  
    }  
}
```

```
public class TryWithFoo{  
  
    public void TryWithBar() {  
        Resource bar = new Resource();  
        try (bar) {  
            //use bar  
        }  
    }  
}  
}
```

```
public class TryWithFoo {  
  
    final Resource bar = new Resource();  
    public void TryWithBar() {  
        try (bar) {  
            //use bar  
        }  
    }  
}  
}
```



---

# MULTI-RELEASE JARS

<http://bit.ly/2kVj9Hk>

## EXAMPLE MULTI-RELEASE FILE STRUCTURE

---

jar/  
  foo.class  
  bar.class  
  evil.class  
META-INF/

  MANIFEST.MF

  -----  
  Multi-Release: true

  versions/

    9/  
      foo.class  
      bar.class

    10/  
      bar.class

Code with Java 9 artifacts

Code with Java 10 artifacts



---

# JAVADOC & DEPRECATION ENHANCEMENTS

<http://bit.ly/2kVj9Hk>

### ► Javadoc Enhancements

- searchable
- html5 compliant

### ► @Deprecation enhancements

- addition of forRemoval(boolean)
- addition of since(String)

---

# ADDITIONAL CHANGES

<http://bit.ly/2kVj9Hk>

# ADDITIONAL CHANGES

---

- ▶ Reactive updates
- ▶ G1 now default garbage collector
- ▶ Compact strings
- ▶ New version format
  - ▶ \$MAJOR.\$MINOR.\$SECURITY.\$PATCH
- ▶ HTTP/2 support (experimental)
- ▶ Unified GC Logging
- ▶ Unified JVM Logging
- ▶ Stackwalking API

---

# REMOVALS

<http://bit.ly/2kVj9Hk>

# REMOVALS

---

- ▶ jhat
- ▶ applet API
- ▶ Internal APIs (when encapsulation is turned on)
- ▶ Deprecated GC combinations
- ▶ JVM TI hprof Agent

# DEVELOPER TOOLS

<http://bit.ly/2kVj9Hk>

## DEVELOPER TOOL JAVA 9 COMPATIBILITY

---

- ▶ IntelliJ
  - ▶ IDEA 2017.1+
- ▶ Eclipse
  - ▶ Oxygen.1a (4.7.1a)
- ▶ Netbeans
  - ▶ Currently available in nightly builds, full support part of Netbeans 9
- ▶ Maven
  - ▶ 3.0+
- ▶ Gradle
  - ▶ 4.1+ (limited Java 9 compatibility)
  - ▶ Full modules support coming soon

# SPRING FRAMEWORK JAVA 9 SUPPORT

---



- ▶ SPRING 4.X (CURRENT)
  - LIMITED AND ONLY FOR 4.3.10
- ▶ SPRING 5.X (CURRENTER)
  - STABLE AUTOMATIC MODULE NAMES
- ▶ SPRING 6.X (2019+)
  - FULL JIGSAW SUPPORT

# SPRING LIBRARIES JAVA 9 SUPPORT

---



- ▶ SPRING BOOT 1.X (CURRENT)
  - NO FORMAL JAVA 9 SUPPORT
- ▶ SPRING BOOT 2.X (SOONISH)
  - FORMAL JAVA 9 SUPPORT
- ▶ OTHER LIBRARIES WILL VARY

# WHO TO FOLLOW FOR JAVA 9 NEWS

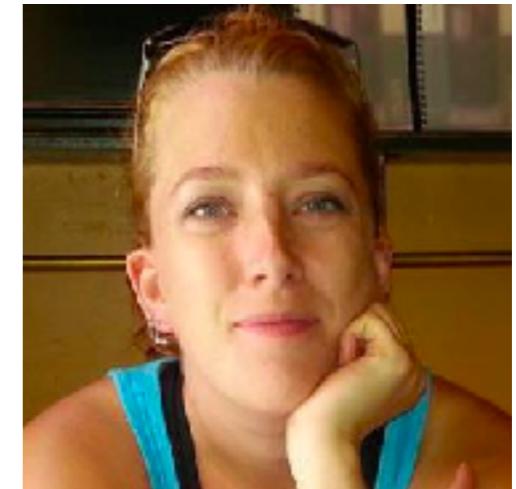
---



**@PBAKKER**  
PAUL BAKKER



**@SANDER\_MAK**  
SANDER MAK



**@TRISHA\_GEE**  
TRISHA GEE



**@MREINHOLD**  
MARK REINHOLD



**@NIPAFX**  
NICOLAI PARLOG

# RESOURCES

---

Source code: <https://github.com/wkorando/java-9-microservices-demo>

Real World Java 9 by Trisha Gee: <https://www.youtube.com/watch?v=GkP83hvdeMk>

AskArch DevoxxUK: [https://www.youtube.com/watch?v=ac1v5kF\\_FGs](https://www.youtube.com/watch?v=ac1v5kF_FGs)

Java 9 Modularity in Action: <https://www.youtube.com/watch?v=oy3202OFPpM>

Full Java 9 feature list: <http://openjdk.java.net/projects/jdk9/>

What's in Java 9: <https://dzone.com/articles/java-9-besides-modules>

---

**GET STARTED!**

**HTTP://JDK.JAVA.NET/9/**