WHAT'S NEW IN JAVA 9

BILLY KORANDO @BILLYKORANDO

SOFTWARE CONSULTANT - KEYHOLE SOFTWARE

http://bit.ly/2woEb1S

PLATINUM SPONSORS

KEYHOLE SOFTWARE

SYLLOGISTEKS

Daugherty BUSINESS SOLUTIONS

POLARIS SOLUTIONS

VALOREM

ArchitectNow

Couchbase

GOLD SPONSORS

Ungerboeck Software International

Asynchrony Labs

deckcommerce

GREENFIELD CREATIVE

Expium

VantageLinks

Covenant Technology Partners

Byrne Software Technologies, Inc.

CENTRIQ Training

ADAPTIVE Solutions Group

Progress

TEKsystems

Microsoft

SILVER SPONSORS

LogicNP Software The power of Components

GrapeCity

redgate

Octopus Deploy

JetBrains

Saint Louis University

Robert Half Technology

Saint Louis .NET User Group

St. Louis Silver Linings

Conference Notebook & Party

Daugherty BUSINESS SOLUTIONS

Promotional Items & Attendee Lanyards

GREENFIELD CREATIVE

Attendee Swag Bags

Asynchrony Labs

Media

InfoQ

http://bit.ly/2woEb1S

JAVA 9 WENT GA ON SEPTEMBER 21ST!

http://bit.ly/2woEb1S

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

http://bit.ly/2woEb1S

PROVE IT!

http://bit.ly/2woEb1S

AN UPGRADE WITH BENEFITS

http://bit.ly/2woEb1S

BENEFITS OF RUNNING IN JAVA 9

Better Use of Memory

Better Performance

Better Use of Hardware

Better Documentation

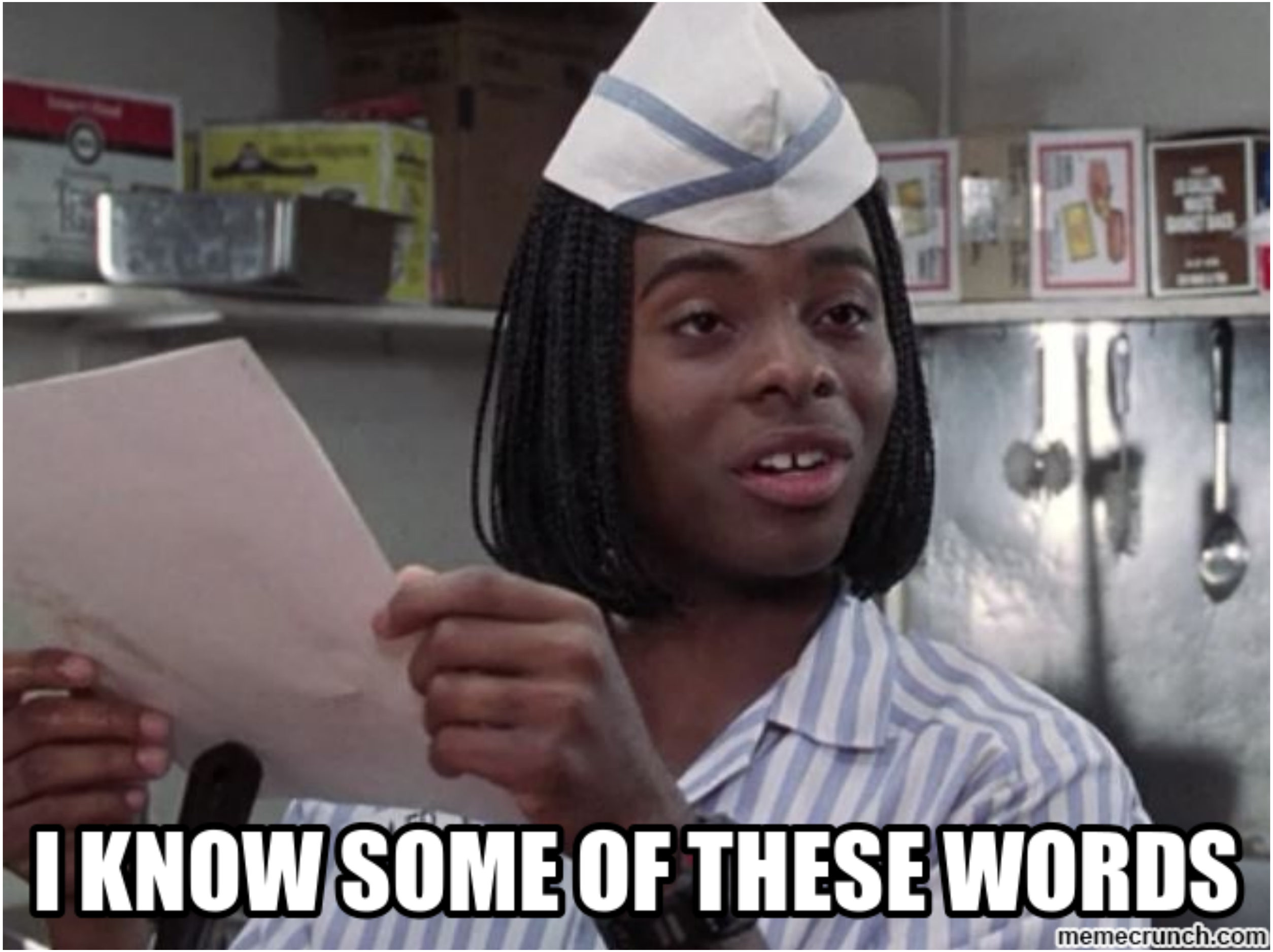
Prettier Graphics

Faster Compilation

http://bit.ly/2woEb1S

JIGSAW

http://bit.ly/2woEb1S


I KNOW SOME OF THESE WORDS
memecrunch.com

JIGSAW GOALS

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

http://bit.ly/2woEb1S

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
```

http://bit.ly/2woEb1S

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/
        FooSvcImpl.java
```

http://bit.ly/2woEb1S

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.FooSvcImpl;

    uses org.evil.service.EvilService;

    requires org.evil;
    requires java.sql;

}
```

http://bit.ly/2woEb1S

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.FooSvcImpl;

    uses org.evil.service.EvilService;

    requires org.evil;
    requires java.sql;

}
```

http://bit.ly/2woEb1S

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.FooSvcImpl;

    uses org.evil.service.EvilService;

    requires org.evil;
    requires java.sql;

}
```

http://bit.ly/2woEb1S

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.FooSvcImpl;

    uses org.evil.service.EvilService;

    requires org.evil;
    requires java.sql;

}
```

http://bit.ly/2woEb1S

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.FooSvcImpl;

    uses org.evil.service.EvilService;

    requires org.evil;
    requires java.sql;

}
```

http://bit.ly/2woEb1S

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.FooSvcImpl;

    uses org.evil.service.EvilService;

    requires org.evil;
    requires java.sql;

}
```

http://bit.ly/2woEb1S

JSHELL

http://bit.ly/2woEb1S

JSHELL

Read

Evaluate

Print

Loop

http://bit.ly/2woEb1S

WHY JSHELL IS PART OF JAVA 9

- Explore APIs
- Tool to Teach Java

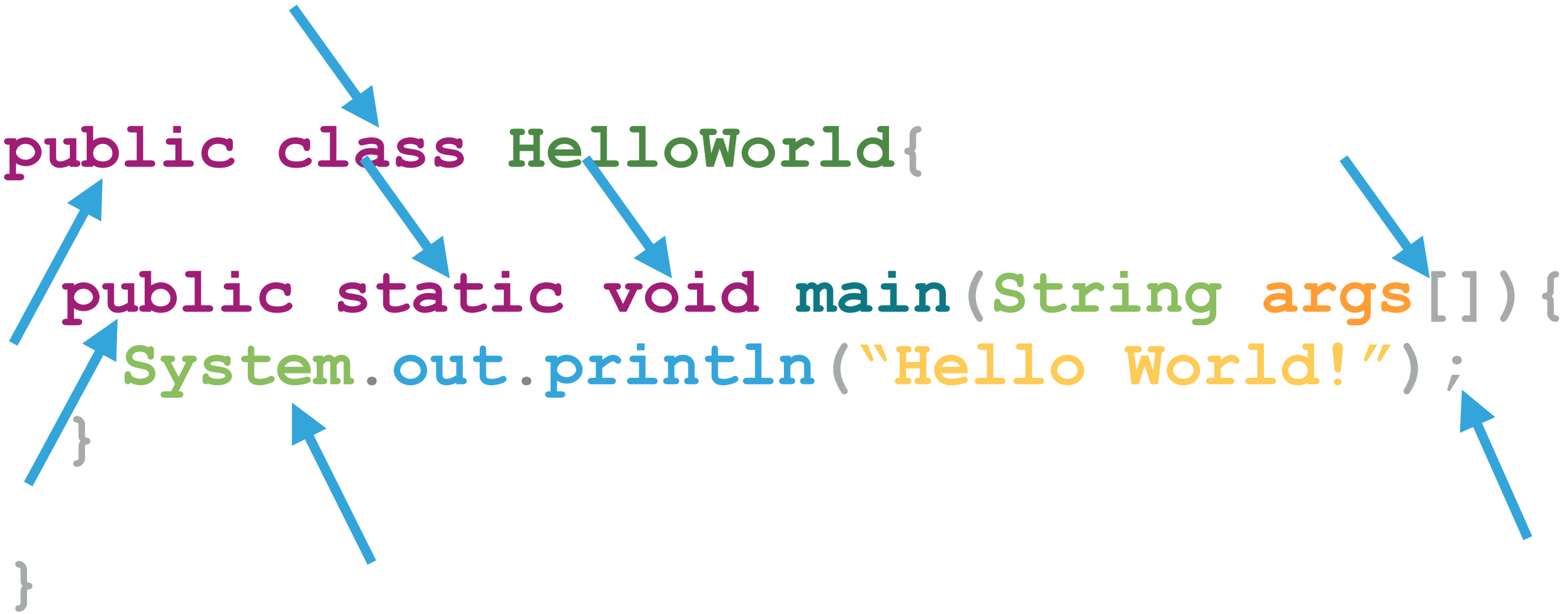
http://bit.ly/2woEb1S

WHY JSHELL IS PART OF JAVA 9

- Explore APIs

- Tool to Teach Java

http://bit.ly/2woEb1S



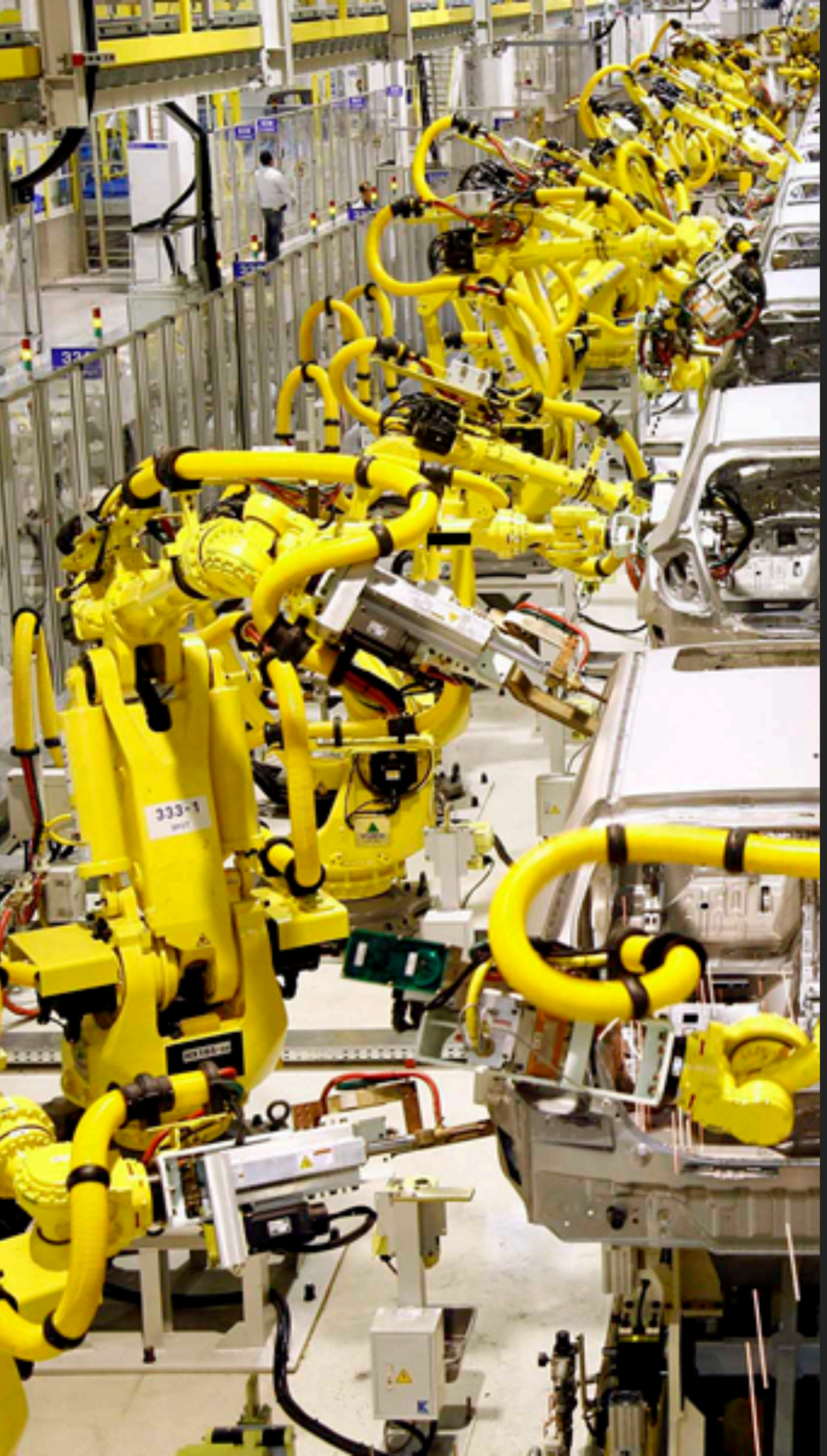
http://bit.ly/2woEb1S

```
public class HelloWorld{

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

}
```

http://bit.ly/2woEb1S

DEMO

http://bit.ly/2woEb1S



COLLECTION FACTORY METHODS

http://bit.ly/2woEb1S

```
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);
  }

}
```

http://bit.ly/2woEb1S

```
public class TestFoo{

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

}
```

http://bit.ly/2woEb1S

```
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);
  }

}
```

http://bit.ly/2woEb1S

```
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);
  }


}
```

http://bit.ly/2woEb1S

STREAMS UPDATES

http://bit.ly/2woEb1S

```
public class TakeKansasCitySportsTeams{

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

}
```

```
Output: ChiefsRoyalsSportingMavericks
```

http://bit.ly/2woEb1S

```
public class DropKansasCitySportsTeams{

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

}
```

```
Output: ChiefsRoyalsSportingMavericks
```

http://bit.ly/2woEb1S

PRIVATE INTERFACE METHODS

http://bit.ly/2woEb1S

```
public interface Foo{

  default void bar(String arg){
    System.out.println(“Hello World!”);
  }

  private default String lorem(){
    ...
  }
}
```

http://bit.ly/2woEb1S

TRY-WITH-RESOURCES UPDATES

http://bit.ly/2woEb1S

```
public class TryWithFoo{

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

}
```

http://bit.ly/2woEb1S

```
public class TryWithFoo{

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

}
```

http://bit.ly/2woEb1S

```
public class TryWithFoo{

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

}
```

http://bit.ly/2woEb1S

MULTI-RELEASE JARS

http://bit.ly/2woEb1S

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/2woEb1S

JAVADOC & DEPRECATION ENHANCEMENTS

- Javadoc Enhancements
  - searchable
  - html5 compliant
- @Deprecation enhancements
  - addition of forRemoval(boolean)
  - addition of since(String)

http://bit.ly/2woEb1S

ADDITIONAL CHANGES

http://bit.ly/2woEb1S

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

http://bit.ly/2woEb1S

REMOVALS

http://bit.ly/2woEb1S

REMOVALS

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

http://bit.ly/2woEb1S

DEVELOPER TOOLS

http://bit.ly/2woEb1S

DEVELOPER TOOL JAVA 9 COMPATIBILITY

- Intellij
  - IDEA 2017.1+
- Eclipse
  - Requires Oxygen, currently in beta, full support coming in October
- 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

http://bit.ly/2woEb1S

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

http://bit.ly/2woEb1S

SPRING LIBRARIES JAVA 9 SUPPORT

- SPRING BOOT 1.X (CURRENT)
  - NO SUPPORT
- SPRING BOOT 2.X (NOVEMBER 2017)
  - SUPPORTS RUNNING IN JAVA 9
- OTHER LIBRARIES WILL VARY

http://bit.ly/2woEb1S

WHO TO FOLLOW FOR JAVA 9 NEWS

@PBAKKER
PAUL BAKKER

@SANDER_MAK
SANDER MAK

@NIPAFX
NICOLAI PARLOG

http://bit.ly/2woEb1S

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

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

http://bit.ly/2woEb1S

GET STARTED!

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