Desde Java 8 hasta Java 14

Víctor Orozco - @tuxtor

18 de octubre de 2020

Nabenik

¿Como se hace Java?

De Java 8 hasta Java 14

Java 9

Java 10

Java 11

Java 12

Java 13

Java 14

Mundo real

¿Como se hace Java?

¿Java?

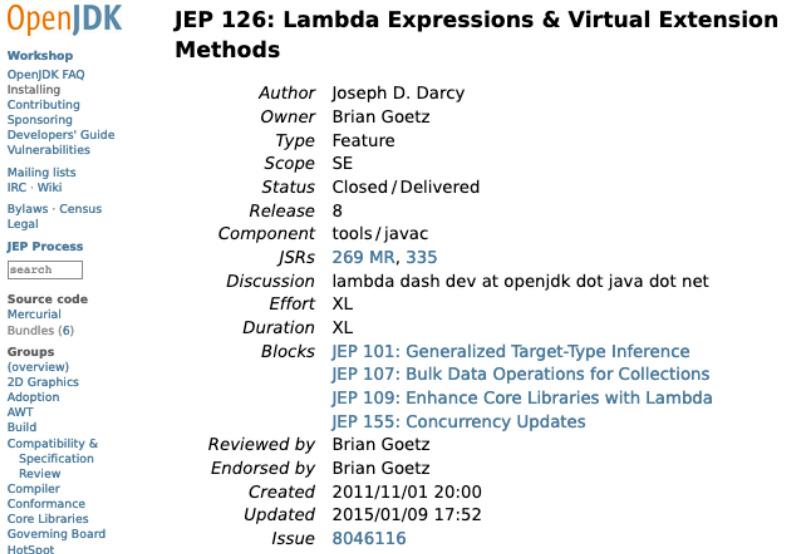
- Lenguaje
- VM
- Bibliotecas/API

El conjunto es la plataforma Java (TM)

¿Como se actualiza Java?

- JCP - Java Community Process
- JSR - Java Specification Request
- JEP - Java Enhancement Proposal
- JCK - Java Compatibility Kit

¿Como se actualiza Java? - Java Enhancement Proposal



OpenJDK

Workshop
OpenJDK FAQ
Installing
Contributing
Sponsoring
Developers' Guide
Vulnerabilities

Mailing lists
IRC · Wiki

Bylaws · Census
Legal

JEP Process
search

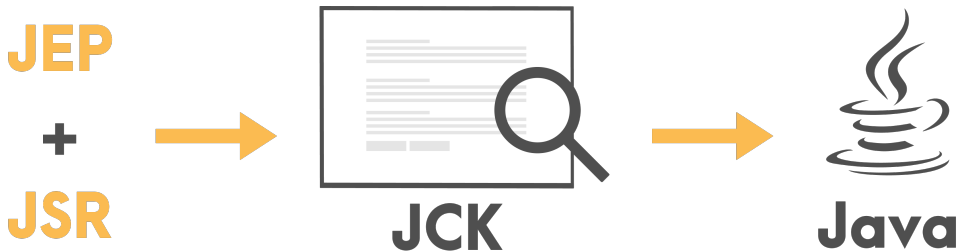
Source code
Mercurial
Bundles (6)

Groups
(overview)
2D Graphics
Adoption
AWT
Build
Compatibility & Specification Review
Compiler
Conformance
Core Libraries
Governing Board
HotSpot

JEP 126: Lambda Expressions & Virtual Extension Methods

Author	Joseph D. Darcy
Owner	Brian Goetz
Type	Feature
Scope	SE
Status	Closed / Delivered
Release	8
Component	tools / javac
JSRs	269 MR, 335
Discussion	lambda dash dev at openjdk dot java dot net
Effort	XL
Duration	XL
Blocks	JEP 101: Generalized Target-Type Inference JEP 107: Bulk Data Operations for Collections JEP 109: Enhance Core Libraries with Lambda JEP 155: Concurrency Updates
Reviewed by	Brian Goetz
Endorsed by	Brian Goetz
Created	2011/11/01 20:00
Updated	2015/01/09 17:52
Issue	8046116

¿Como se actualiza Java? - Java Compatibility Kit



¿Java libre?

Java **es libre y gratis**.

Algunas empresas ofrecen soporte en su ”versión”de Java.

De Java 8 hasta Java 14

¿Que recibo con cada versión nueva de Java?

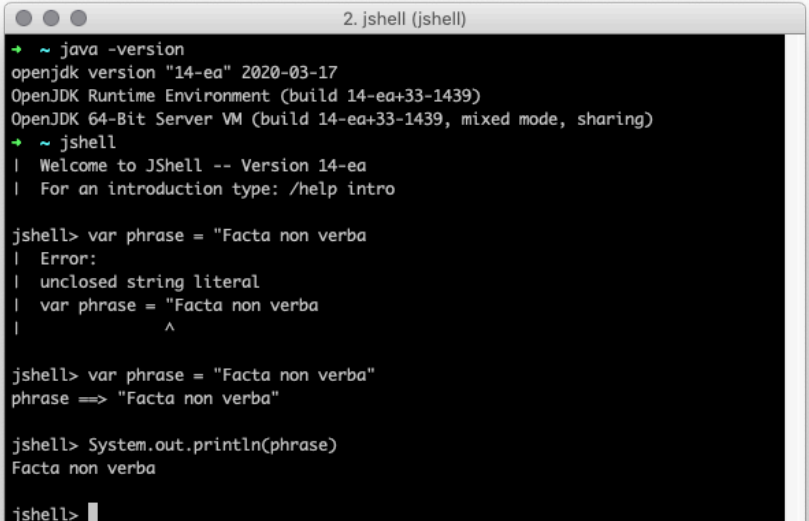
- Java - **Lenguaje**
- Java - Bibliotecas e APIs
- Java - Maquina Virtual de Java

Java - Las mejoras que resaltan

- Java 9
  - Modulos
  - JShell
  - HTTP/2
  - Factory methods
- Java 10
  - Type Inference
  - Class Data Sharing
  - Time based release
- Java 11
  - String methods
  - File methods
  - Direct .java execution
- Java 12
  - Switch expressions
- Java 13
  - Text blocks
- Java 14
  - Pattern matching
  - Records
  - Helpfull NPE

Java 9

JEP 222: jshell: The Java Shell (Read-Eval-Print Loop)



JEP 110: HTTP/2 Client

```
1  HttpRequest request = HttpRequest.newBuilder()
2      .uri(new URI("https://swapi.co/api/starships/9"))
3      .GET()
4      .build();
5
6  HttpResponse<String> response = HttpClient.newHttpClient()
7      .send(request, BodyHandlers.ofString());
8
9  System.out.println(response.body());
```

JEP 269: Convenience Factory Methods for Collections

Antes

```
1 Set<String> set = new HashSet<>();
2 set.add("a");
3 set.add("b");
4 set.add("c");
5 set = Collections.unmodifiableSet(set);
```

”Pro”

```
1 Set<String> set = Collections.unmodifiableSet(new HashSet<>(
      Arrays.asList("a", "b", "c")));
```

Ahora

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

JEP 213: Milling Project Coin - Private methods in interfaces

Antes

```
1  public interface Vehicle{
2      public void move();
3  }
```

Ahora

```
1  public interface Vehicle{
2      public default void makeNoise(){
3          System.out.println("Making noise!");
4          createNoise();
5      }
6
7      private void createNoise(){
8          System.out.println("Run run");
9      }
10 }
```

JEP 213: Milling Project Coin - Try-with-resources

Antes

```
1  BufferedReader reader = new BufferedReader(new FileReader("
       langs.txt"));
2
3  try(BufferedReader innerReader = reader){
4      System.out.println(reader.readLine());
5  }
```

Ahora

```
1  BufferedReader reader = new BufferedReader(new FileReader("
       langs.txt"));
2
3  try(reader){
4      System.out.println(reader.readLine());
5  }
```

Java 10

286: Local-Variable Type Inference
296: Consolidate the JDK Forest into a Single Repository
304: Garbage-Collector Interface
307: Parallel Full GC for G1
310: Application Class-Data Sharing
312: Thread-Local Handshakes
313: Remove the Native-Header Generation Tool (javah)
314: Additional Unicode Language-Tag Extensions
316: Heap Allocation on Alternative Memory Devices
317: Experimental Java-Based JIT Compiler
319: Root Certificates
322: Time-Based Release Versioning

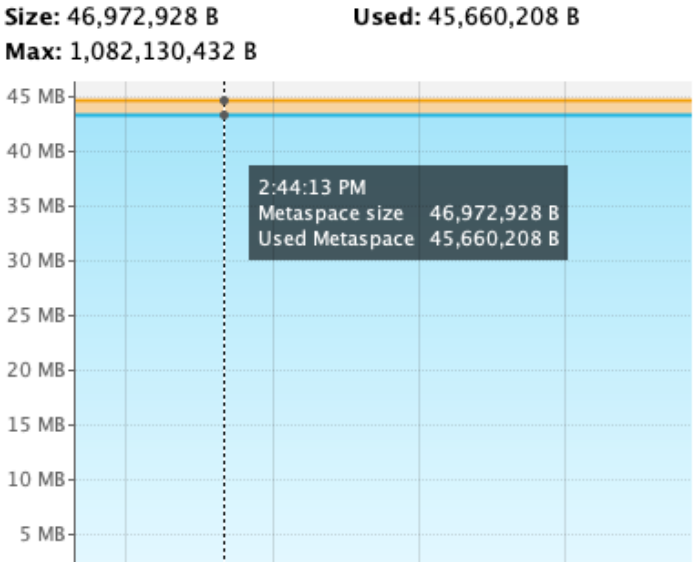
JEP 286: Local-Variable Type Inference

```
1 public static void main(String args[]){
2     var localValue = 99;
3     System.out.println(++localValue);
4     //localValue = "Foo"
5 }
```

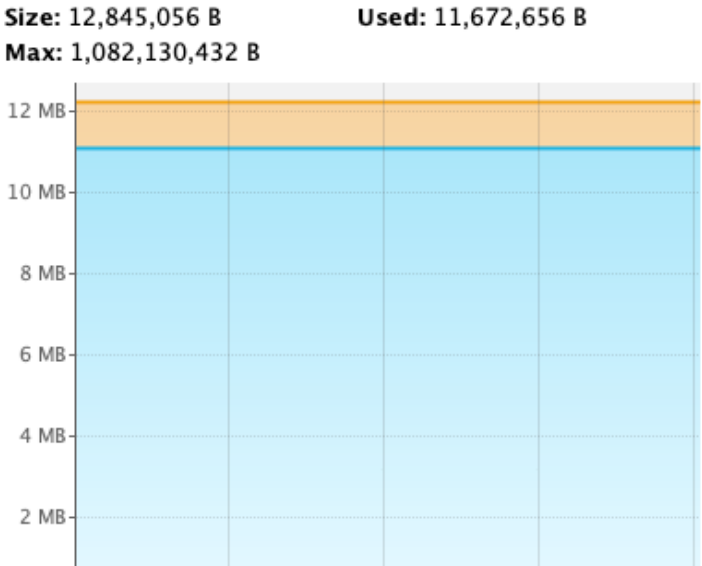
JEP 310: Application Class-Data Sharing

```
java -XX:ArchiveClassesAtExit=app-cs.jsa -jar payara-micro-5.192.jar
java -XX:SharedArchiveFile=app-cs.jsa -jar fpjava.jar
```

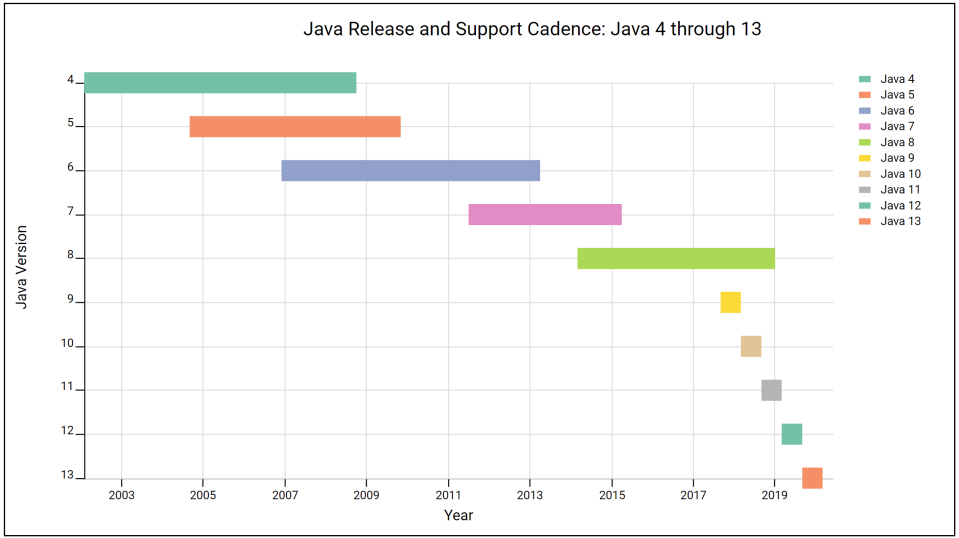

JEP 310: Application Class-Data Sharing



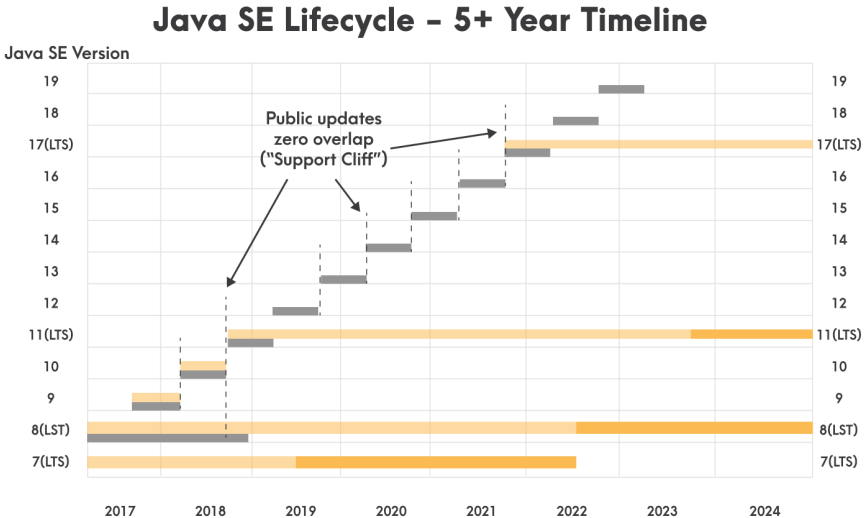
JEP 310: Application Class-Data Sharing



JEP 322: Time-Based Release Versioning



JEP 322: Time-Based Release Versioning



Java 11

Java 11

181: Nest-Based Access Control
309: Dynamic Class-File Constants
315: Improve Aarch64 Intrinsics
318: Epsilon: A No-Op Garbage Collector
320: Remove the Java EE and CORBA Modules
321: HTTP Client (Standard)
323: Local-Variable Syntax for Lambda Parameters
324: Key Agreement with Curve25519 and Curve448
327: Unicode 10
328: Flight Recorder
329: ChaCha20 and Poly1305 Cryptographic Algorithms
330: Launch Single-File Source-Code Programs
331: Low-Overhead Heap Profiling
332: Transport Layer Security (TLS) 1.3
333: ZGC: A Scalable Low-Latency Garbage Collector (Experimental)
335: Deprecate the Nashorn JavaScript Engine
336: Deprecate the Pack200 Tools and API

JEP 323: Local-Variable Syntax for Lambda Parameters

Antes

```
1 BiPredicate<String,String> demoPredicate =
2     (String a, String b) -> a.equals(b);
3 BiPredicate<String,String> demoPredicate =
4     (a, b) -> a.equals(b);
```

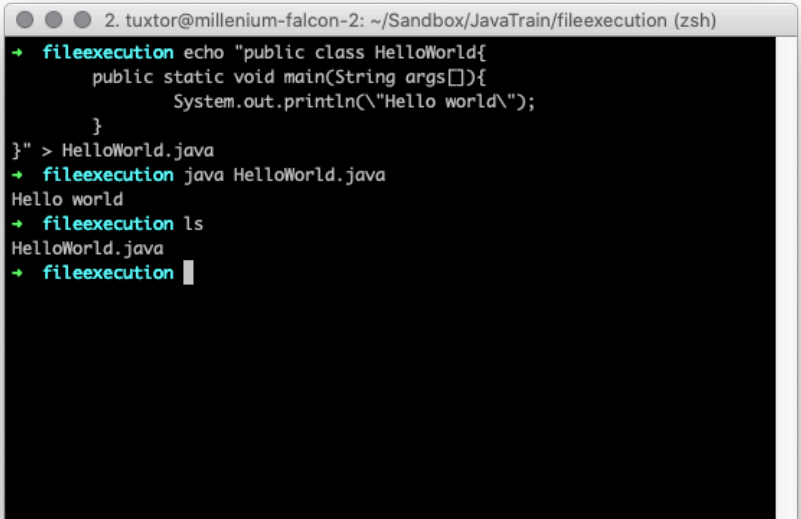
Ahora

```
1 BiPredicate<String,String> demoPredicate =
2     (var a, var b) -> a.equals(b);
```

Posibilidades

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

JEP 330: Launch Single-File Source-Code Programs



Java 12

189: Shenandoah: A Low-Pause-Time Garbage Collector (Experimental)
230: Microbenchmark Suite
325: Switch Expressions (Preview)
334: JVM Constants API
340: One AArch64 Port, Not Two
341: Default CDS Archives
344: Abortable Mixed Collections for G1
346: Promptly Return Unused Committed Memory from G1

325: Switch Expressions (Preview)

Antes

```
1  String langType = "";
2  switch (args[0]) {
3      case "Java":
4      case "Scala":
5      case "Kotlin":
6          langType = "Static typed";
7          break;
8      case "Groovy":
9      case "JavaScript":
10         langType = "Dynamic typed";
11         break;
12 }
13 System.out.println(langType);
```

325: Switch Expressions (Preview)

Ahora

```
1  String langType = switch (args[0]) {
2      case "Java", "Scala", "Kotlin" -> "Static typed";
3      case "Groovy", "JavaScript" -> "Dynamic typed";
4      default -> {
5          System.out.println("This meant to be a processing
               block");
6          yield "Probably LISP :)";
7      }
8  };
9  System.out.println(langType);
```

Java 13

Java 13

350: Dynamic CDS Archives
351: ZGC: Uncommit Unused Memory
353: Reimplement the Legacy Socket API
354: **Switch Expressions (Preview)**
355: **Text Blocks (Preview)**

355: Text Blocks (Preview)

Antes

```
1 String html = "<html>\n" +
2 "    <body>\n" +
3 "        <p>Hello, world</p>\n" +
4 "    </body>\n" +
5 "</html>\n";
```

Ahora

```
1 String html = """
2 <html>
3     <body>
4         <p>Hello, world</p>
5     </body>
6 </html>
7 """;
```

Java 14

Java 14

305: Pattern Matching for instanceof (Preview)
343: Packaging Tool (Incubator)
345: NUMA-Aware Memory Allocation for G1
349: JFR Event Streaming
352: Non-Volatile Mapped Byte Buffers
358: Helpful NullPointerExceptions
359: Records (Preview)
361: Switch Expressions (Standard)

362: Deprecate the Solaris and SPARC Ports
363: Remove the Concurrent Mark Sweep (CMS) Garbage Collector
364: ZGC on macOS
365: ZGC on Windows
366: Deprecate the ParallelScavenge + SerialOld GC Combination
367: Remove the Pack200 Tools and API
368: Text Blocks (Second Preview)
370: Foreign-Memory Access API (Incubator)

JEP 359: Records (Preview)

Data carrier

```
1 record Person(String name, String email, int age) {}
```

Uso

```
1 Person foo = new Person("Marco", "example@mail.com",99);
2 System.out.println(foo);
3 //foo.name = "Polo";
```

305: Pattern Matching for instanceof (Preview)

Antes

```
1  if(o instanceof Person){
2      Person p = (Person)o;
3      System.out.println("Hello " + p.name());
4  }else{
5      System.out.println("Unknown object");
6  }
```

Ahora

```
1  if(o instanceof Person p){
2      System.out.println("Hello " + p.name());
3  }else{
4      System.out.println("Unknown object");
5  }
```

Mundo real

Mundo real

Mi ”mundo real”

- ERP - 10 modulos (1 EAR, 9 EJB, 1 WAR), JBoss/Wildfly
- Venta/Geocerca (5 WAR) Payara Application Server
- POS - JavaFX y Windows D:

El rompe cabezas

- Módulos
- sun.misc.unsafe
- Corba y Java EE
- JavaFX
- IDE
- Licenciamiento

Mundo real

El rompe cabezas

- Módulos
- sun.misc.unsafe
- Corba y Java EE
- JavaFX
- IDE
- Licencia

Estrategia

1. Probar la compatibilidad del runtime/servidor/framework
2. Múltiples JVM en desarrollo con cambio fácil
3. Actualizar el compilador de Maven
4. Actualizar las bibliotecas
5. Incluir los módulos EE en los war/jar
6. Actualizar el IDE
7. Preparar el proyecto para módulos en el caso de JavaFX
8. Determinar previamente el Java que necesito
9. Ejecutar distintas versiones de Java en producción

Compatibilidad runtime

Compatibilidad con Java 11

- Tomcat
- Spring
- Micronaut
- Vert.x
- JakartaEE (JBoss/Wildfly, OpenLiberty, Payara, WebLogic)

Varias JVMs

jEnv

Manage your Java environment

Bibliotecas

Generación dinámica de Bytecode

- ByteBuddy
- ASM
- glib
- Spring
- Java EE
- Hibernate
- Mockito

Maven

- Maven 3.5.0
- Compiler 3.8.0
- surefire 2.22.0
- failsafe 2.22.0
- release version 11.0

JAF (java.activation)

```
<dependency>
    <groupId>com.sun.activation</groupId>
    <artifactId>javax.activation</artifactId>
    <version>1.2.0</version>
</dependency>
```

CORBA = RIP

JAXB (java.xml.bind)

```
<!-- API -->
<dependency>
    <groupId>jakarta.xml.bind</groupId>
    <artifactId>jakarta.xml.bind-api</artifactId>
    <version>2.3.2</version>
</dependency>

<!-- Runtime -->
<dependency>
    <groupId>org.glassfish.jaxb</groupId>
    <artifactId>jaxb-runtime</artifactId>
    <version>2.3.2</version>
</dependency>
```

JAX-WS (java.xml.ws)

```xml
<!-- API -->
<dependency>
    <groupId>jakarta.xml.ws</groupId>
    <artifactId>jakarta.xml.ws-api</artifactId>
    <version>2.3.2</version>
</dependency>

<!-- Runtime -->
<dependency>
    <groupId>com.sun.xml.ws</groupId>
    <artifactId>jaxws-rt</artifactId>
    <version>2.3.2</version>
</dependency>
```

Common Annotations (java.xml.ws.annotation)

```
<dependency>
    <groupId>javax.annotation</groupId>
    <artifactId>javax.annotation-api</artifactId>
    <version>1.3.1</version>
</dependency>
```

IDEs

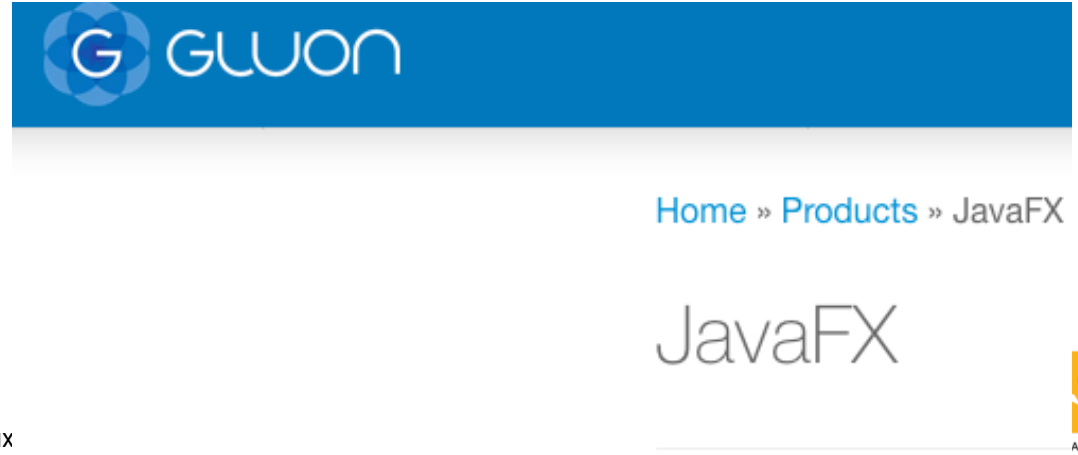
IDEs compatibles con Java 11

- Eclipse
- NetBeans
- IntelliJ IDEA

Algunos plug-ins problemáticos

1. Glassfish
2. WebLogic
3. Icefaces

JavaFX es un módulo independiente del JDK a partir de Java 11, compatible con JPMS, casi todos usan la compilación de Gluon



¿Cual Java necesito?

Obligatorios por contrato

- Software comercial de Oracle (HotSpot)
- Software comercial de SAP (SAP VM)
- Software comercial de Red Hat (OpenJDK + RHEL)
- Software comercial de IBM (J9)

Algunos otros ”Javas”

- AdoptOpenJDK (soporte de IBM en J9)
- Correto
- Azul Zulu
- Java en Linux

Varias JVM en producción

Linux

- Docker
- RHEL
- Debian
- Gentoo

Windows

- Docker
- Variables de entorno en proyecto/runtime
- Lo importante es la salud

Víctor Orozco

- vorozco@nabenik.com
- @tuxtor
- http://vorozco.com
- http://tuxtor.shekalug.org