

El futuro de Java SE



Aurelio Garcia-Ribeyro
Director Senior
Grupo de Plataforma Java, Oracle
@aureliog

JavaYourNext
(Cloud)

Ago 2019



Disculpas a los puristas

Algunos términos en esta presentación no han sido traducidos del inglés.

Términos que el uso y costumbre –al menos en algunos países hispanohablantes- permiten usar en inglés, nombres propios, y conceptos que no supimos traducir sin cambiar el significado han sido dejados en el idioma original.

Por política de Oracle Descargos de Responsabilidad Legal no pueden ser traducidos.

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Introducción

Aurelio Garcia-Ribeyro

- Director Senior – Gerencia de Producto, Grupo de Plataforma Java, Oracle
- A cargo de los requerimientos de JDK Oracle desde que ingresé a Oracle a través de la adquisición de Sun Microsystems en el 2010
- Antes de unirme a Oracle trabajé en Sun Microsystems para el grupo de gerencia de producto de Java SE
- MBA de MIT Sloan y Bachiller en Ingeniería de Sistemas de la Universidad de Lima





Abierto



Evolucionando



Ágil

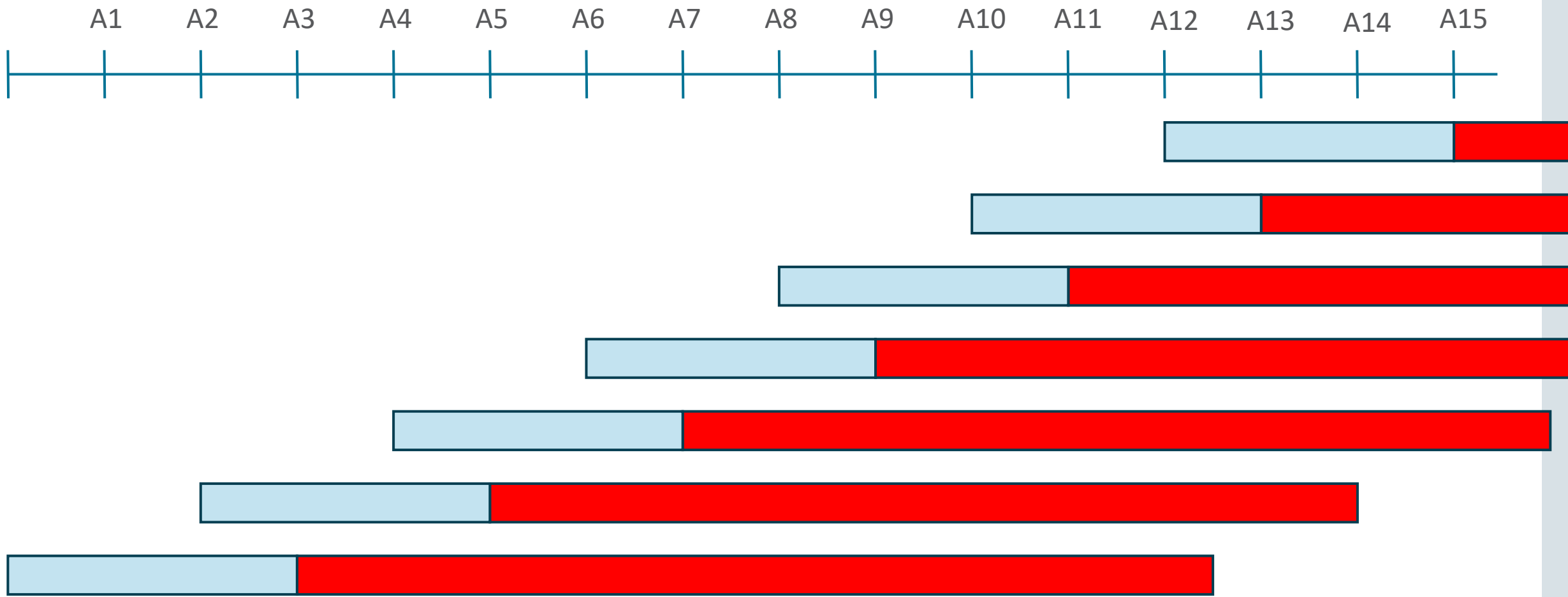
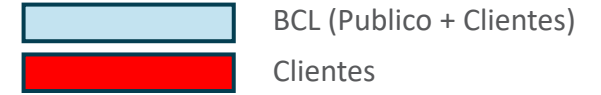


Escalable

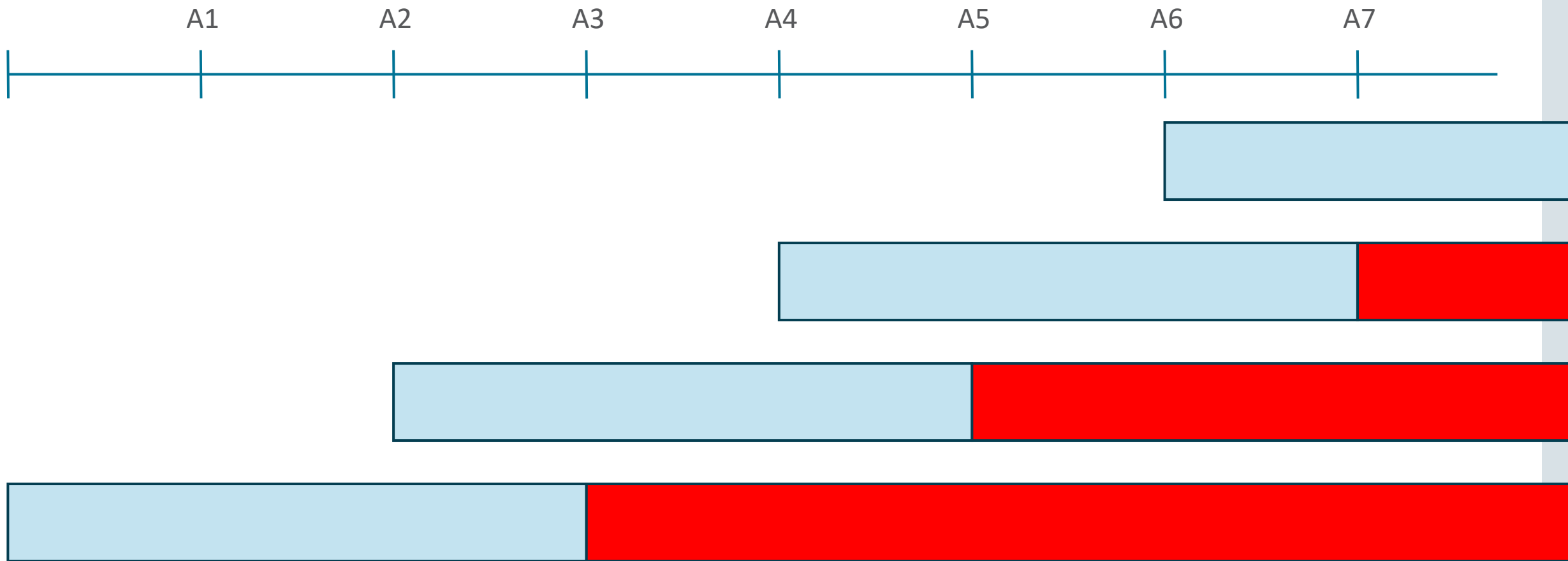
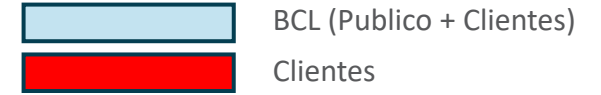
El nuevo modelo de Desarrollo

De Por-Objetivos a Tiempo-Fijo

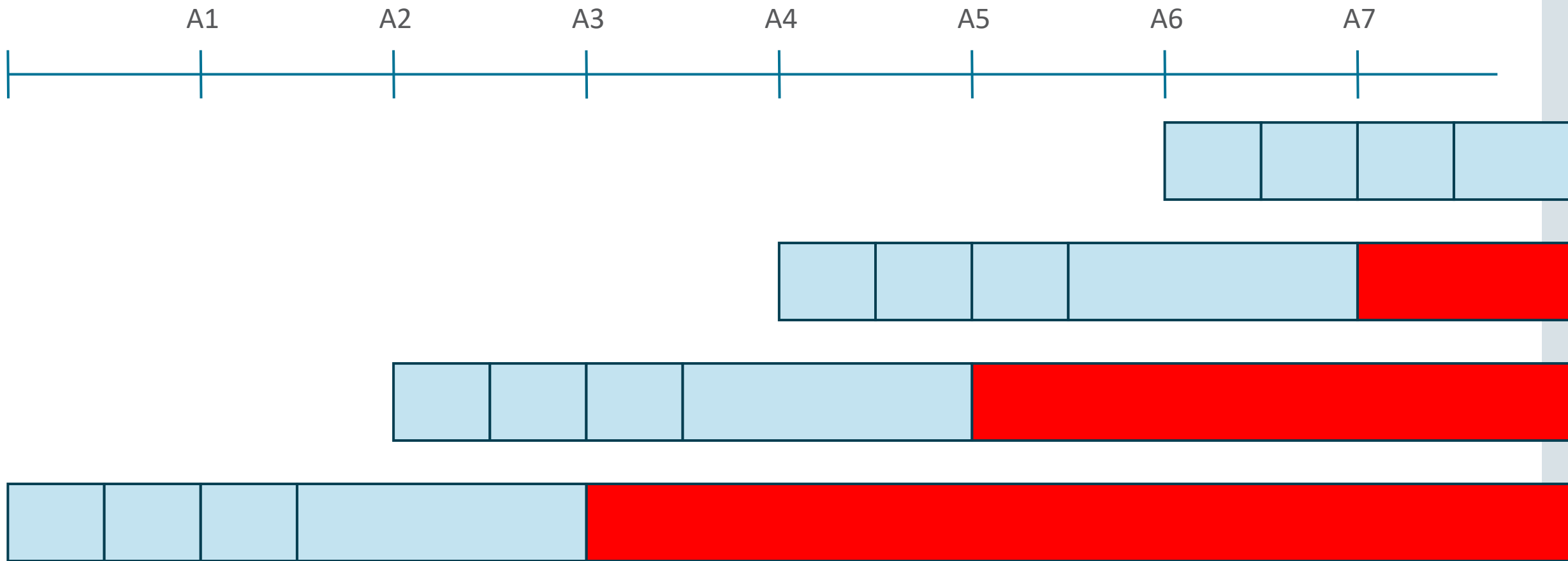
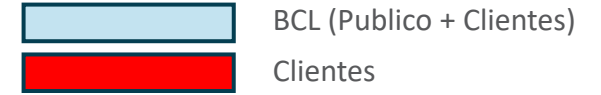
Modelo anterior de desarrollo de JDK

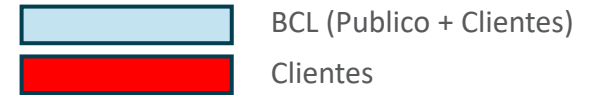


Modelo anterior de desarrollo de JDK

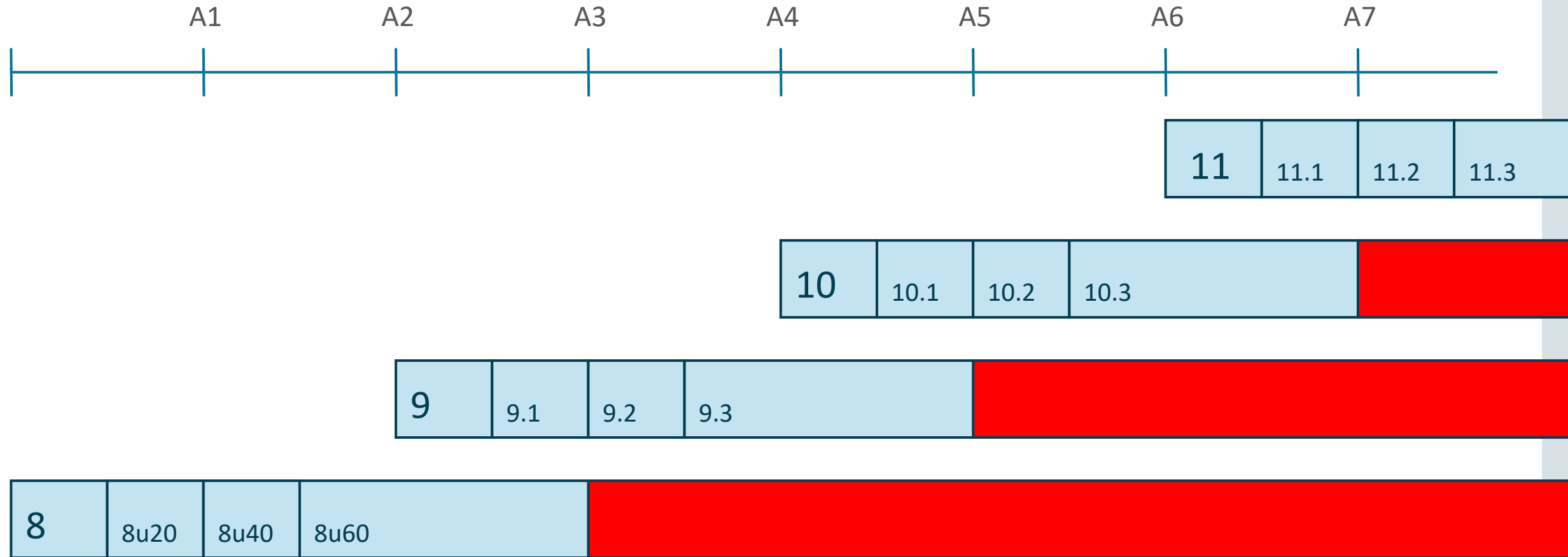


Modelo anterior de desarrollo de JDK







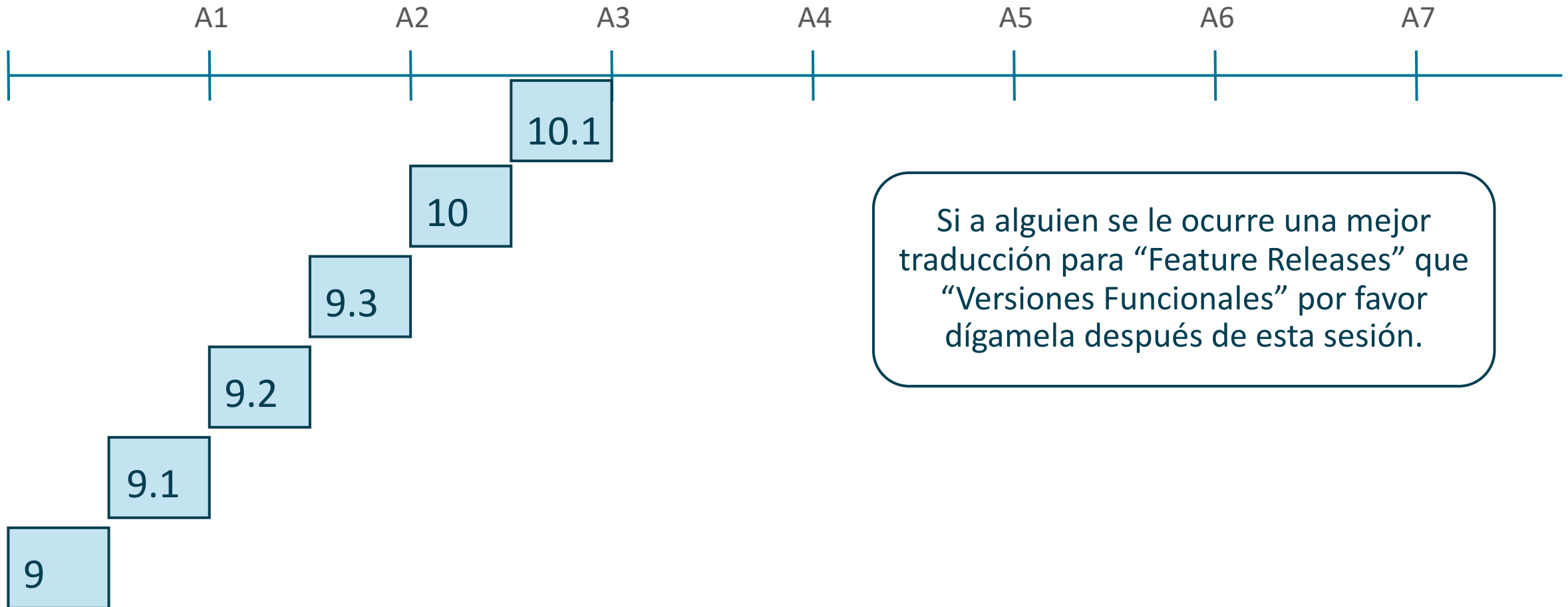
Modelo anterior de desarrollo de JDK



Nuevo Modelo de desarrollo de JDK

Versiones Funcionales cada 6 meses

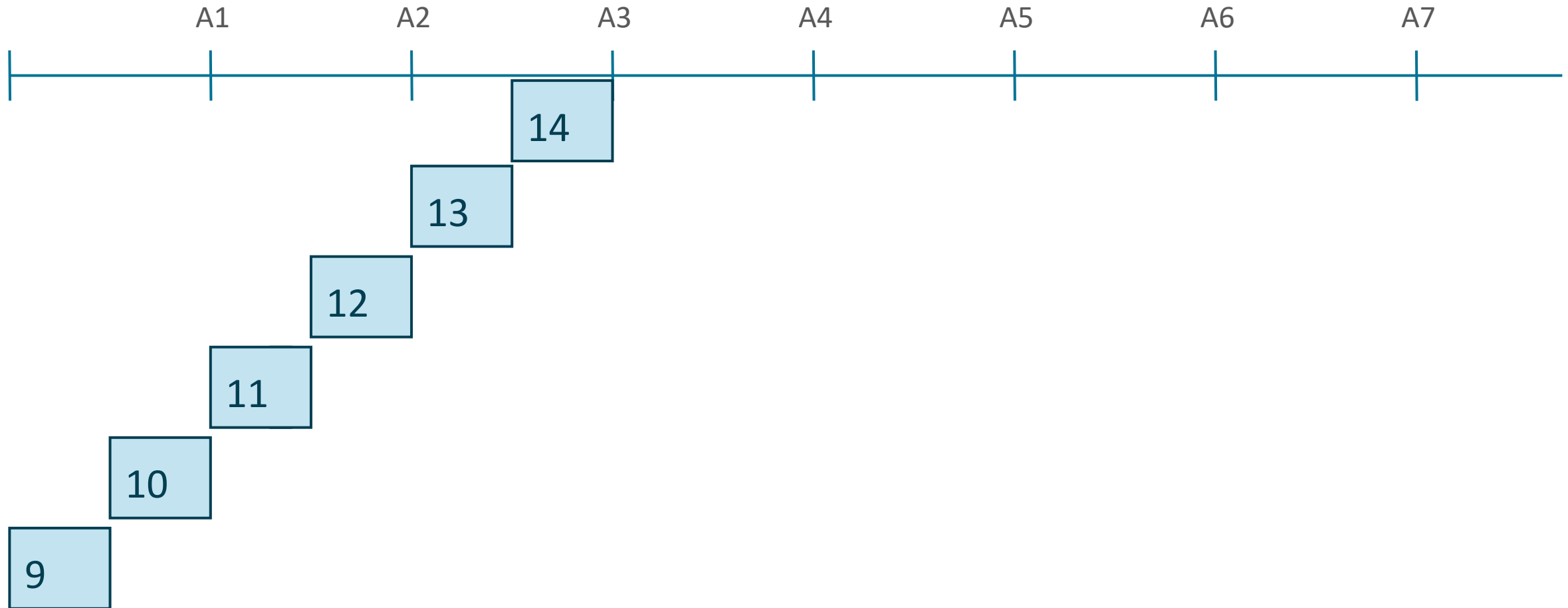
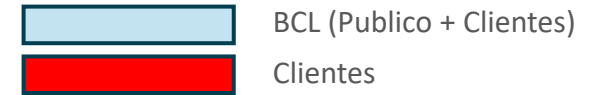
 BCL (Publico + Clientes)
 Clientes



Si a alguien se le ocurre una mejor traducción para “Feature Releases” que “Versiones Funcionales” por favor dígamela después de esta sesión.

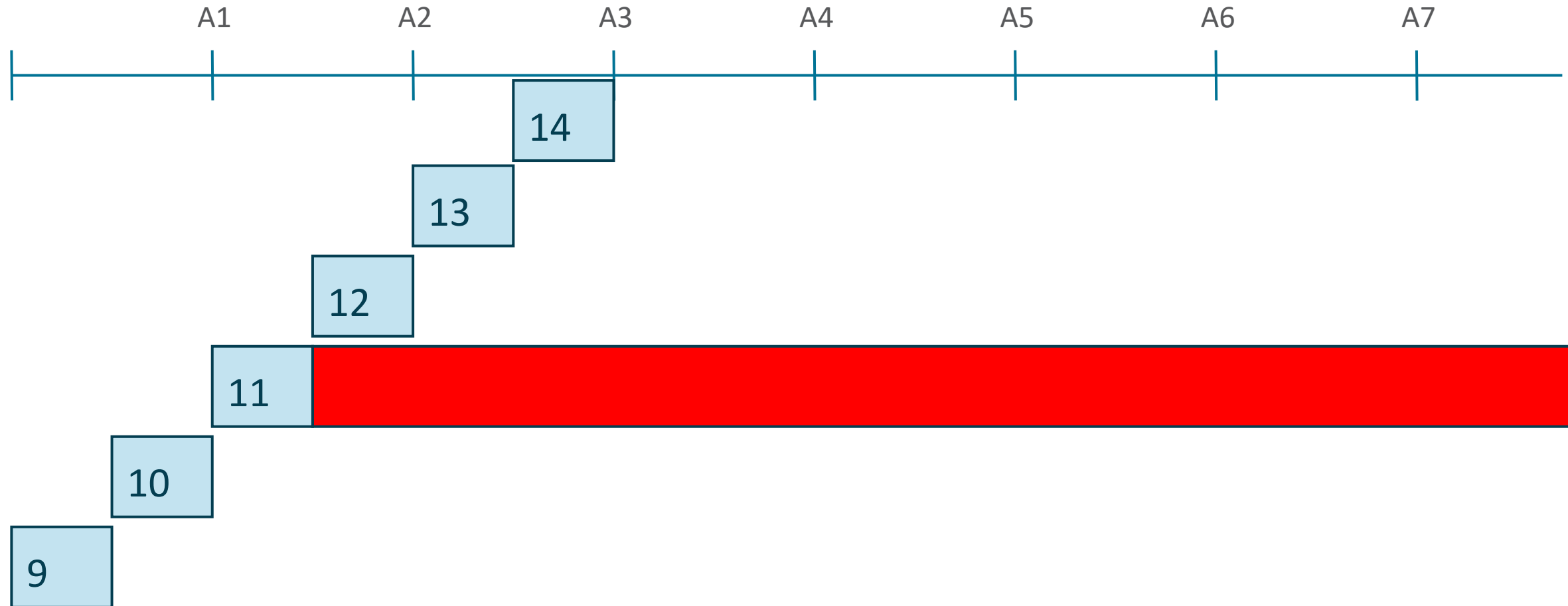
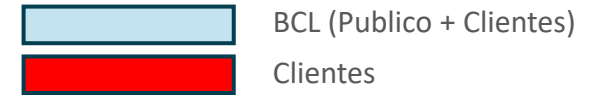
Nuevo Modelo de desarrollo de JDK

Versiones con Soporte a Largo Plazo (LTS)





Nuevo Modelo de desarrollo de JDK

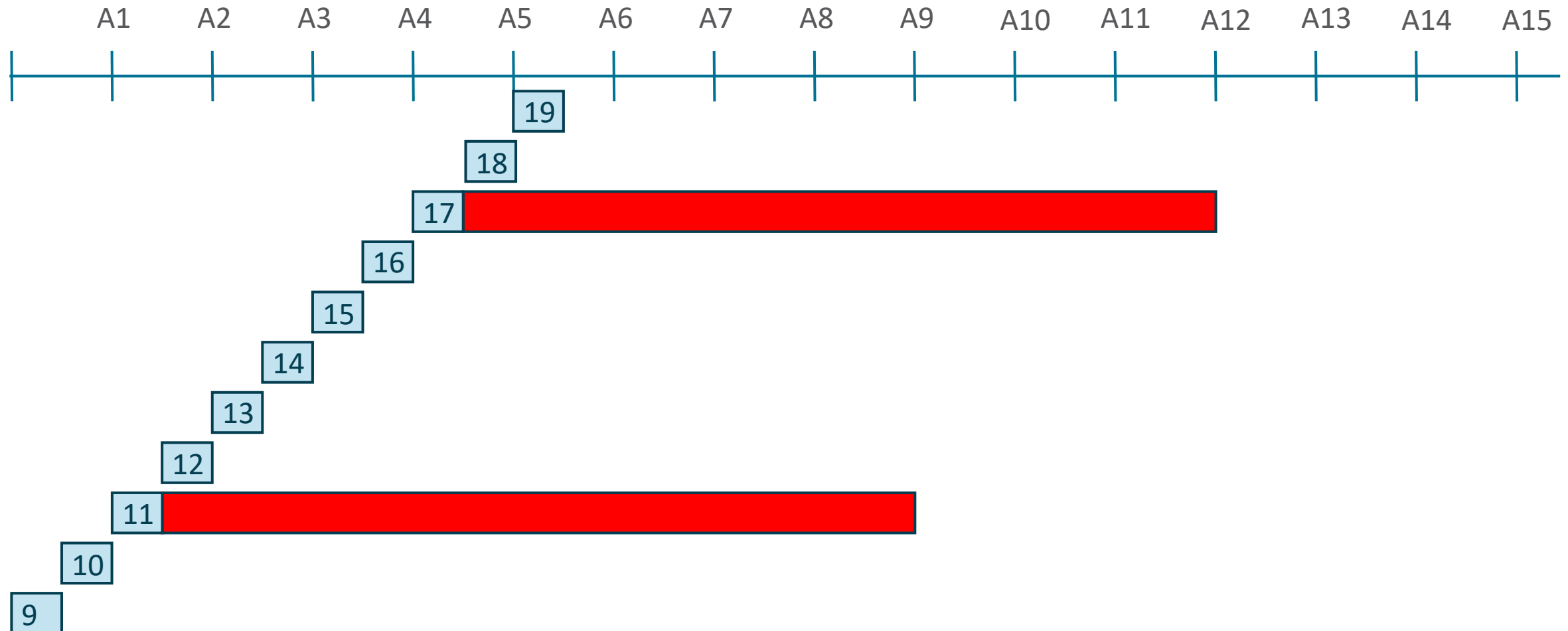
Versiones con Soporte a Largo Plazo (LTS)



Nuevo Modelo de desarrollo de JDK



Versiones con Soporte a Largo Plazo (LTS)

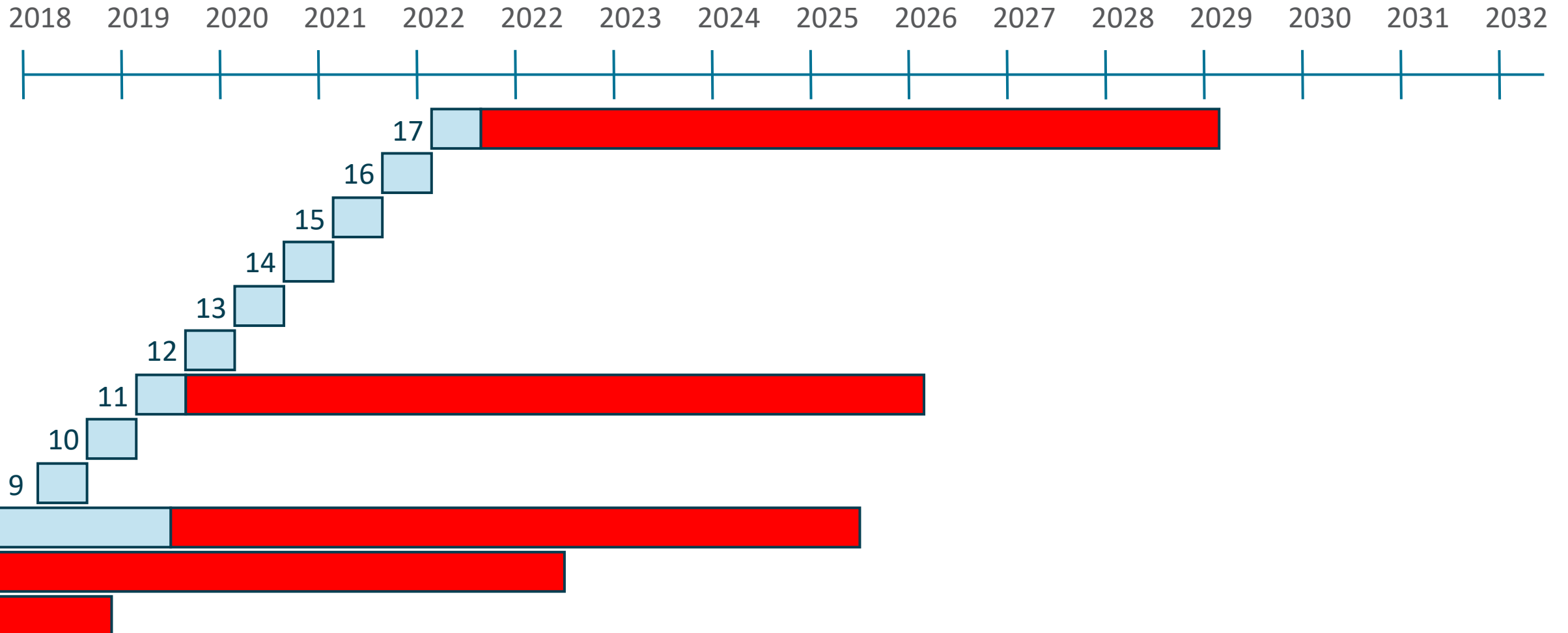
 BCL (Publico + Clientes)
 Clientes



Nuevo Modelo de desarrollo de JDK

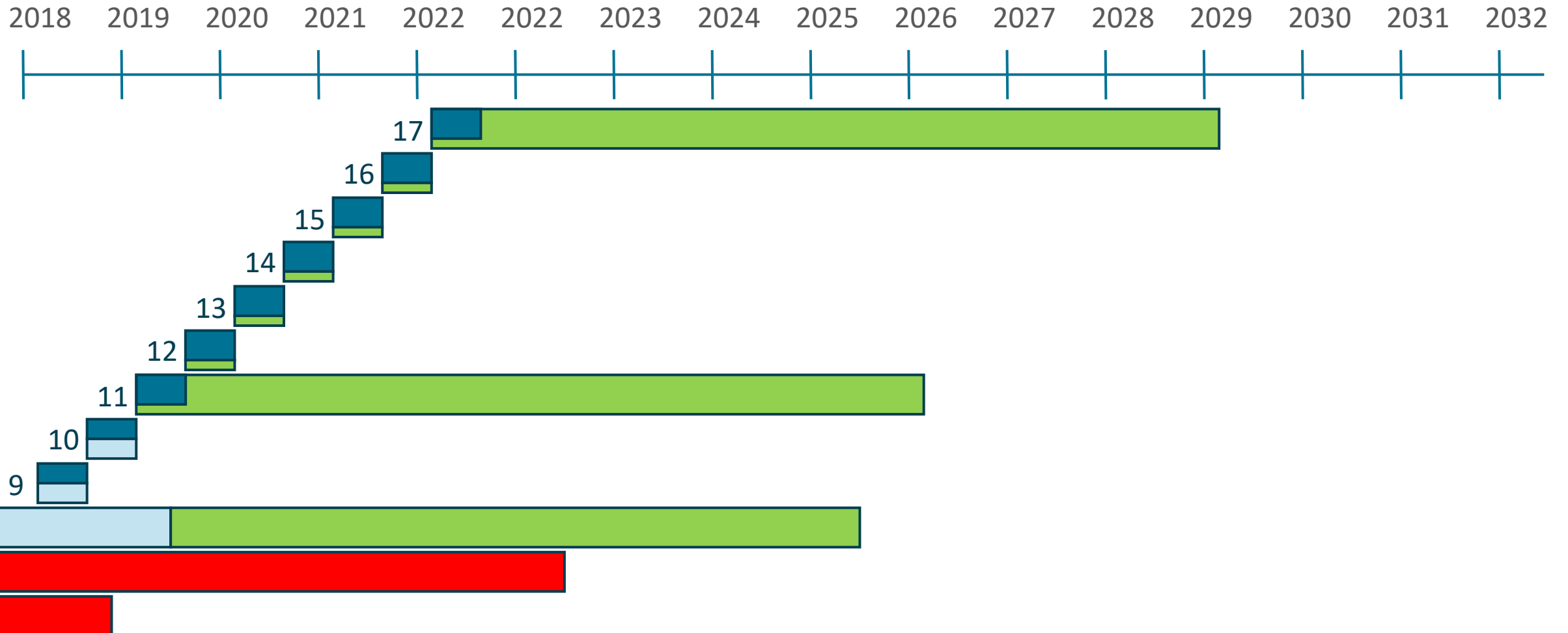
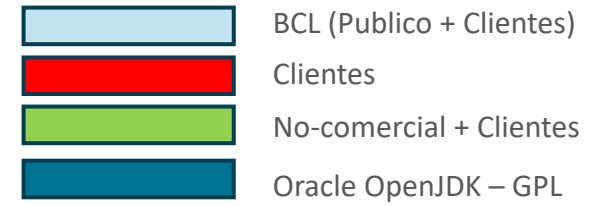
Versiones con Soporte a Largo Plazo (LTS)

 BCL (Publico + Clientes)
 Clientes



OpenJDK por Oracle

Bajo Licencia GPL



Cambios desde JDK 8

Desde marzo 2014

JDK 9

- GA Septiembre 2017
- La ultima versión “Mayor”
 - Release Driver: Jigsaw
 - 100+ mejoras
- Tiempo desde JDK 8: 3 años y 6 meses

Más información sobre cualquier JEP:
<http://openjdk.java.net/jeps/{JEP#}>

OpenJDK **JDK 9**

The goal of this Project is to produce an open-source reference implementation of the Java SE 9 Platform defined by JSR 379 in the Java Community Process.

The schedule and features of this release are proposed and tracked via the JEP Process, as amended by the JEP 2.0 proposal.

Schedule

2016/05/26	Feature Complete
2016/12/22	Feature Extension Complete
2017/01/05	Rampdown Start
2017/02/09	All tests Run
2017/02/16	Zero Bug Bounce
2017/03/16	Rampdown Phase Two
2017/06/22	Initial Release Candidate
2017/07/06	Final Release Candidate
2017/09/21	General Availability

8M, 16D

Status

We are now in the final phase of the release, in which we aim to fix only those bugs that are truly showstoppers to the success of the release. Please see the Release Candidate page for details.

Quick links

Release-Candidate Phase	[candidate bugs]
Rampdown Phase Two	[candidate bugs]
Rampdown Phase One	[candidate bugs]
Bug-deferral process (RDP 1 and later)	[pending requests]
Fix-request process (RDP 2 and later)	[pending requests]
Feature-Complete extension request process	[pending requests]

Features

- 102: Process API Updates
- 110: HTTP 2 Client
- 143: Improve Contended Locking
- 158: Unified JVM Logging
- 165: Compiler Control
- 193: Variable Handles
- 197: Segmented Code Cache
- 199: Smart Java Compilation, Phase Two
- 200: The Modular JDK

JDK 9

- GA Septiembre 2017
- La última versión “Mayor”
 - Release Driver: Jigsaw
 - 100+ mejoras
- Tiempo desde JDK 8: 3 años y 6 meses

Más información sobre cualquier JEP:
<http://openjdk.java.net/jeps/{JEP#}>

264: Platform Logging API and Service
265: Marlin Graphics Renderer
266: More Concurrency Updates
267: Unicode 8.0
268: XML Catalogs
269: Convenience Factory Methods for Collections
270: Reserved Stack Areas for Critical Sections
271: Unified GC Logging
272: Platform-Specific Desktop Features
273: DRBG-Based SecureRandom Implementations
274: Enhanced Method Handles
275: Modular Java Application Packaging
276: Dynamic Linking of Language-Defined Object Models
277: Enhanced Deprecation
278: Additional Tests for Humongous Objects in G1
279: Improve Test-Failure Troubleshooting
280: Indify String Concatenation
281: HotSpot C++ Unit-Test Framework
282: jlink: The Java Linker
283: Enable GTK 3 on Linux
284: New HotSpot Build System
285: Spin-Wait Hints
287: SHA-3 Hash Algorithms
288: Disable SHA-1 Certificates
289: Deprecate the Applet API
290: Filter Incoming Serialization Data
291: Deprecate the Concurrent Mark Sweep (CMS) Garbage Collector
292: Implement Selected ECMAScript 6 Features in Nashorn
294: Linux/s390x Port
295: Ahead-of-Time Compilation
297: Unified arm32/arm64 Port
298: Remove Demos and Samples
299: Reorganize Documentation

Milestone definitions

The milestone definitions for JDK 9 are the same as those for JDK 8, with the addition of:

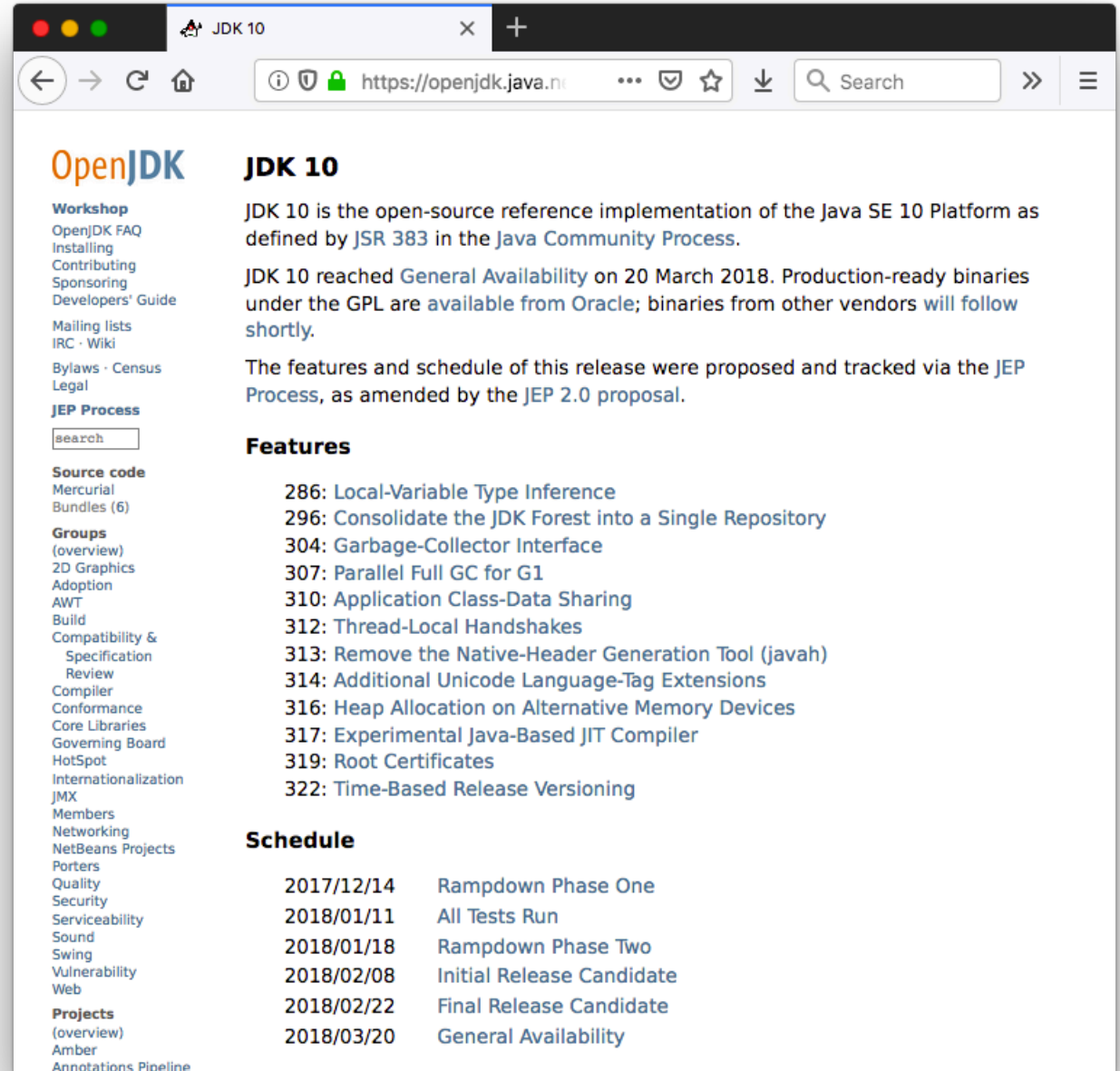
- *Feature Extension Complete* — The date by which JEPs and small enhancements that have been granted extensions via the FC extension-request process must be integrated into the master forest.
- *Initial Release Candidate* — The date on which the first release candidate is built and submitted for testing.

Last update: 2017/6/26 20:57 UTC



JDK 10 – Marzo 2018

- Primera Versión Funcional
- 12 JEPs (Java Enhancement Proposals)
- 6 meses después de Java 9



The screenshot shows the OpenJDK website for JDK 10. The page is titled "JDK 10" and provides information about the release. The left sidebar contains a navigation menu with categories like "Workshop", "Source code", "Groups", and "Projects". The main content area includes a description of JDK 10, a list of features, and a release schedule.

OpenJDK

JDK 10

JDK 10 is the open-source reference implementation of the Java SE 10 Platform as defined by JSR 383 in the [Java Community Process](#).

JDK 10 reached **General Availability** on 20 March 2018. Production-ready binaries under the GPL are available from Oracle; binaries from other vendors will follow shortly.

The features and schedule of this release were proposed and tracked via the [JEP Process](#), as amended by the [JEP 2.0 proposal](#).

Features

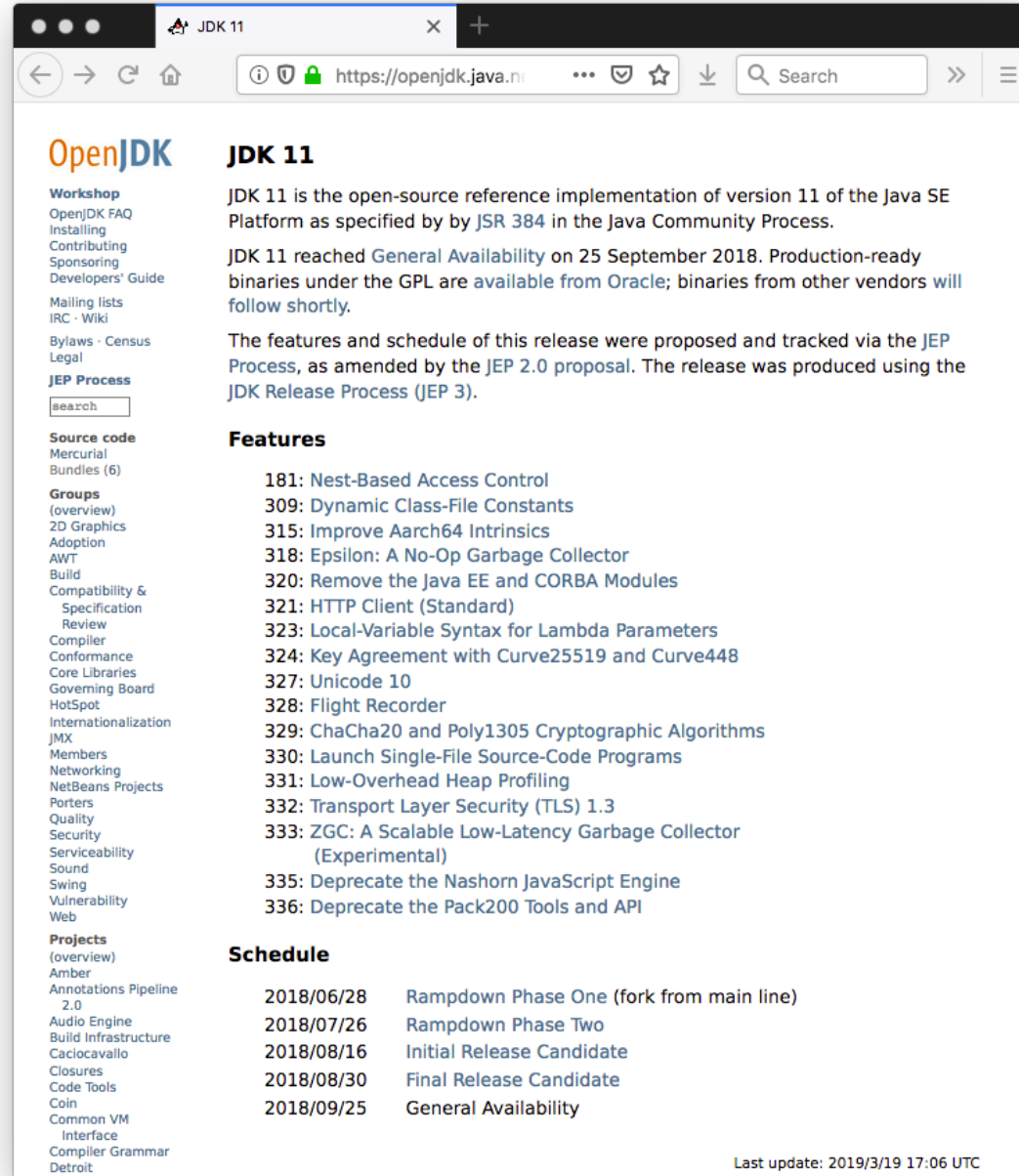
- 286: Local-Variable Type Inference
- 296: Consolidate the JDK Forest into a Single Repository
- 304: Garbage-Collector Interface
- 307: Parallel Full GC for G1
- 310: Application Class-Data Sharing
- 312: Thread-Local Handshakes
- 313: Remove the Native-Header Generation Tool (javah)
- 314: Additional 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

Schedule

2017/12/14	Rampdown Phase One
2018/01/11	All Tests Run
2018/01/18	Rampdown Phase Two
2018/02/08	Initial Release Candidate
2018/02/22	Final Release Candidate
2018/03/20	General Availability

JDK 11 – Septiembre 2018

- 17 JEPs
 - TLS 1.3, la menor demora entre la aprobación de una especificación y su inclusión en una versión de JDK
 - Definido por RFC 8446 en Ago 2018
- Primera versión LTS bajo el nuevo modelo
- 6 meses después de Java 10



The screenshot shows the OpenJDK 11 website. The page title is "JDK 11". The main content area contains the following text:

JDK 11 is the open-source reference implementation of version 11 of the Java SE Platform as specified by JSR 384 in the Java Community Process.

JDK 11 reached **General Availability** on 25 September 2018. Production-ready binaries under the GPL are available from Oracle; binaries from other vendors will follow shortly.

The features and schedule of this release were proposed and tracked via the JEP Process, as amended by the JEP 2.0 proposal. The release was produced using the JDK Release Process (JEP 3).

Features

- 181: Nest-Based Access Control
- 309: Dynamic Class-File Constants
- 315: Improve Aarch64 Intrinsic
- 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

Schedule

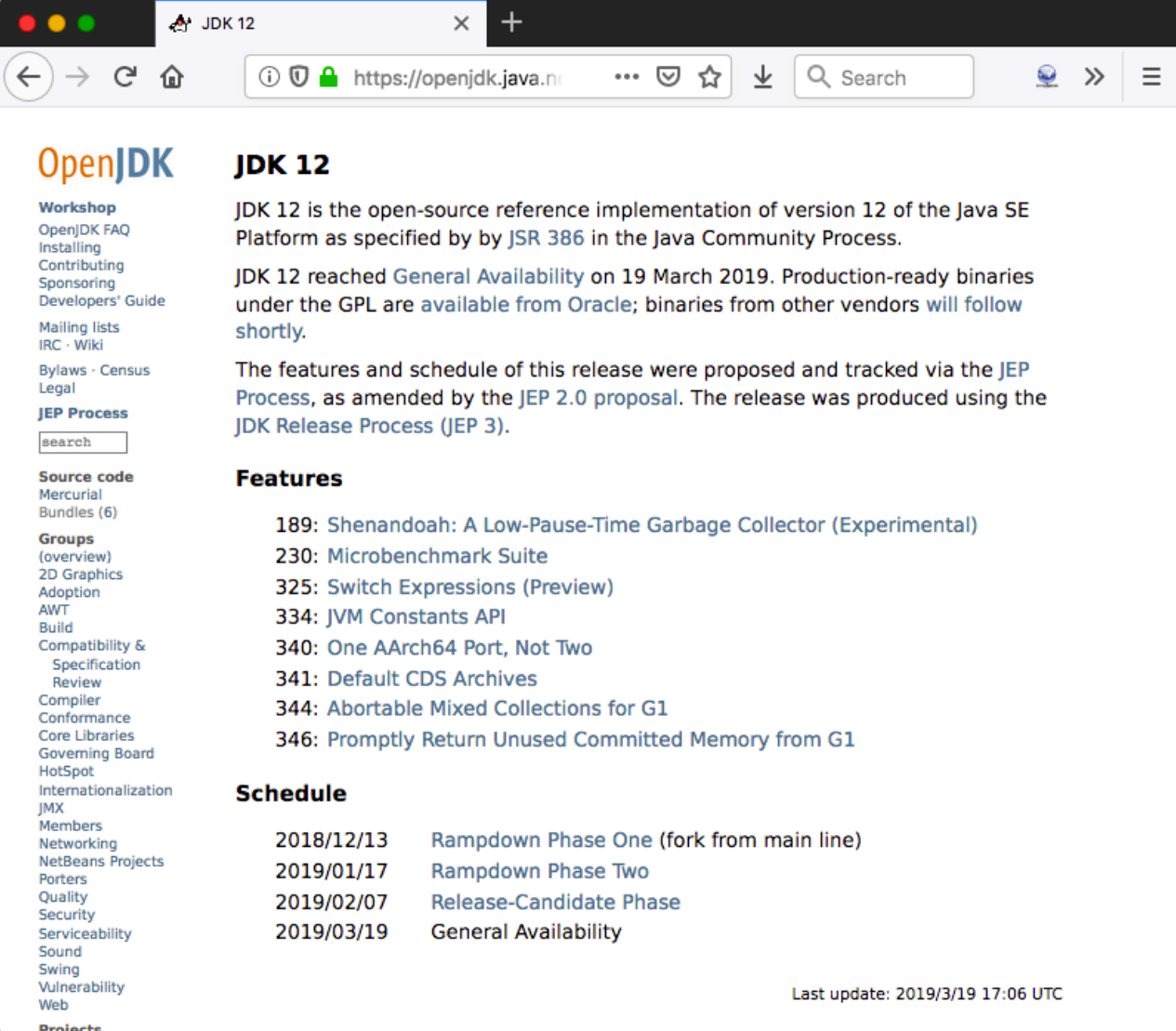
2018/06/28	Rampdown Phase One (fork from main line)
2018/07/26	Rampdown Phase Two
2018/08/16	Initial Release Candidate
2018/08/30	Final Release Candidate
2018/09/25	General Availability

Last update: 2019/3/19 17:06 UTC

JDK 12 – Marzo 2019

- 8 JEPs
 - Primera Funcionalidad Preliminar (Switch Expressions)
- 6 meses después de Java 11

¿Alguien empieza a notar un patrón aquí?



The screenshot shows the OpenJDK website for JDK 12. The page is titled "JDK 12" and provides information about the release. The main content includes a description of JDK 12 as the open-source reference implementation of version 12 of the Java SE Platform, announced on 19 March 2019. It lists features such as Shenandoah garbage collector, Microbenchmark Suite, Switch Expressions, JVM Constants API, AArch64 port, Default CDS Archives, Abortable Mixed Collections for G1, and Promptly Return Unused Committed Memory from G1. A schedule table shows the release timeline from 2018/12/13 to 2019/03/19. The page also includes a sidebar with navigation links and a search bar.

OpenJDK

JDK 12

JDK 12 is the open-source reference implementation of version 12 of the Java SE Platform as specified by by JSR 386 in the Java Community Process.

JDK 12 reached **General Availability** on 19 March 2019. Production-ready binaries under the GPL are available from Oracle; binaries from other vendors will follow shortly.

The features and schedule of this release were proposed and tracked via the **JEP Process**, as amended by the **JEP 2.0 proposal**. The release was produced using the **JDK Release Process (JEP 3)**.

Features

- 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

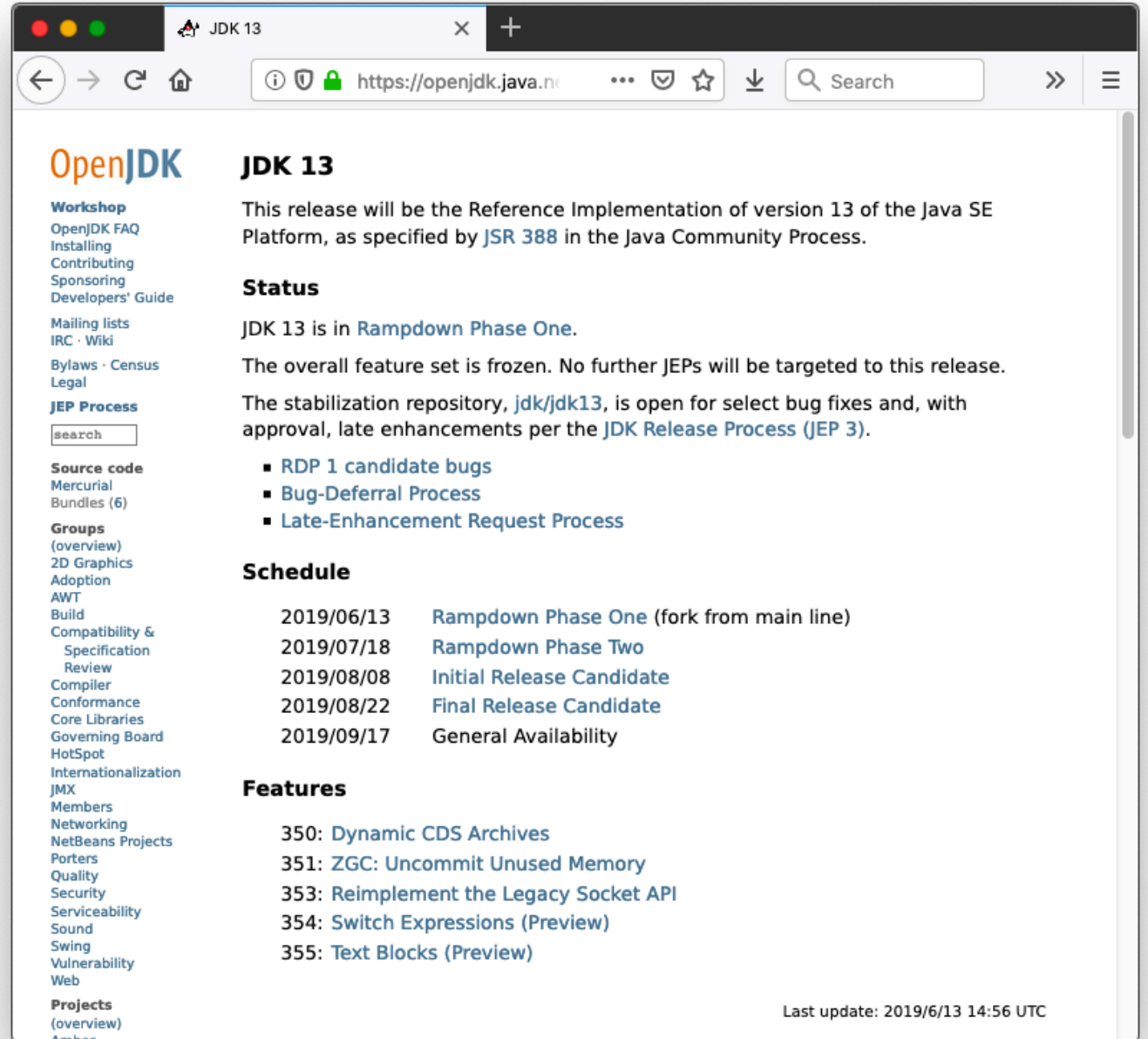
Schedule

2018/12/13	Rampdown Phase One (fork from main line)
2019/01/17	Rampdown Phase Two
2019/02/07	Release-Candidate Phase
2019/03/19	General Availability

Last update: 2019/3/19 17:06 UTC

JDK 13 – Septiembre 2019

- 5 JEPs
- 2 Funcionalidades Preliminares
 - Switch Expression continua siendo preliminar con algunos cambios
 - Text Blocks
- 6 meses después de Java 12



The screenshot shows the OpenJDK website for JDK 13. The page is titled "JDK 13" and provides information about the release, its status, schedule, and features. The left sidebar contains a navigation menu with links to various sections like Workshop, Source code, Groups, and Projects. The main content area includes a description of the release, its status (Rampdown Phase One), a schedule table, and a list of features.

OpenJDK

JDK 13

This release will be the Reference Implementation of version 13 of the Java SE Platform, as specified by [JSR 388](#) in the Java Community Process.

Status

JDK 13 is in [Rampdown Phase One](#).

The overall feature set is frozen. No further JEPs will be targeted to this release.

The stabilization repository, [jdk/jdk13](#), is open for select bug fixes and, with approval, late enhancements per the [JDK Release Process \(JEP 3\)](#).

- [RDP 1 candidate bugs](#)
- [Bug-Deferral Process](#)
- [Late-Enhancement Request Process](#)

Schedule

2019/06/13	Rampdown Phase One (fork from main line)
2019/07/18	Rampdown Phase Two
2019/08/08	Initial Release Candidate
2019/08/22	Final Release Candidate
2019/09/17	General Availability

Features

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

Last update: 2019/6/13 14:56 UTC

Quien hace Java: Cambios resueltos en JDK 11



Quien hace Java: Cambios resueltos en JDK 12

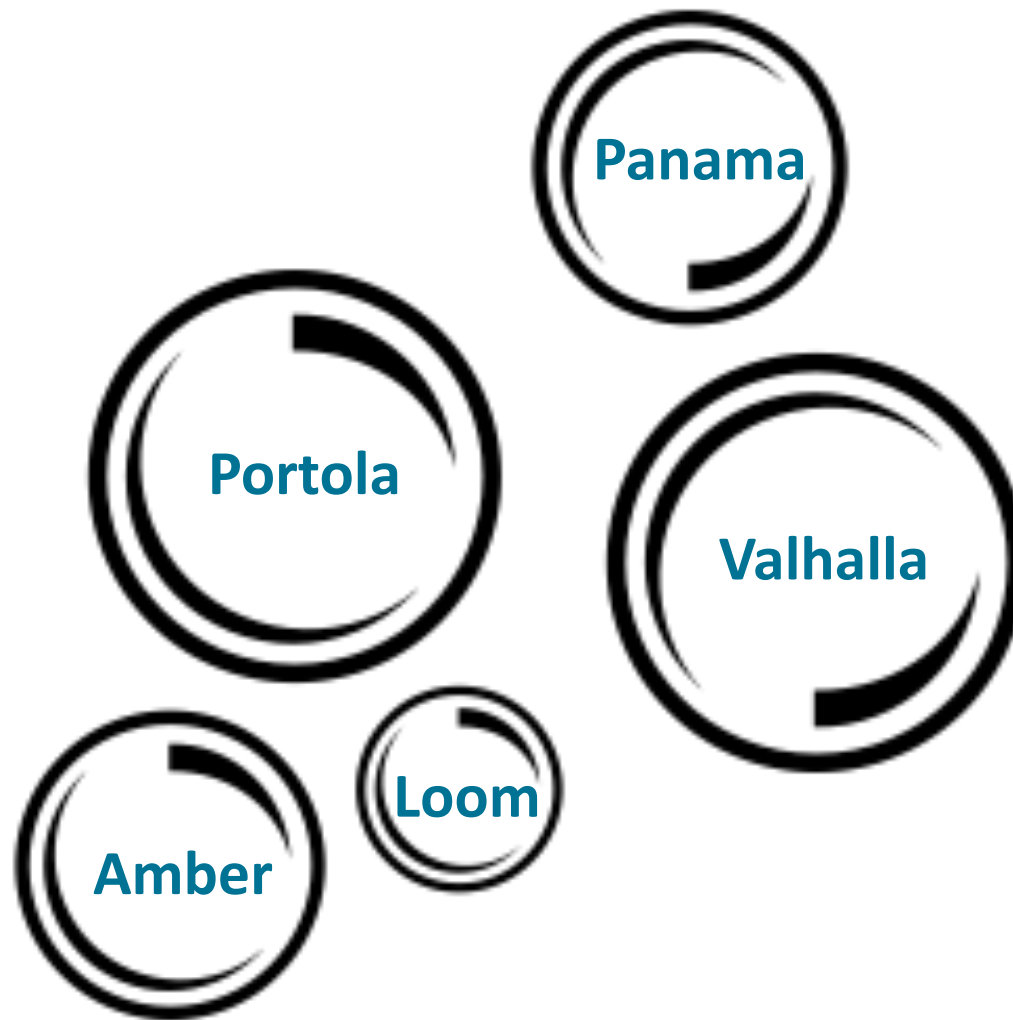


Proyectos en OpenJDK

Trabajando en las mejoras para las siguientes versiones

Siguientes ~~Desafíos~~ Oportunidades

- Contenedores
- Predictibilidad
- Performance
- Optimización de Data
- Aceleración de HW
- Escalabilidad
- Mejoras continuas al lenguaje



openjdk.java.net

Proyecto Amber

- Completos

- Local-Variable Type Inference (var) (JDK 10)
- Local-Variable Syntax for Lambda Parameters (JDK 11)
- Switch Expressions [Funcionalidad Preliminar] (JDK 12)
- Text Blocks [Funcionalidad Preliminar] (JDK 12)

- En desarrollo

- Mejoras para Lambda
- Pattern Matching
- Concise Method Bodies

```
URL url = new URL("http://www.oracle.com/"); ]
```

```
URLConnection conn = url.openConnection();
```

```
Reader reader = new BufferedReader(  
    new InputStreamReader(conn.getInputStream()));
```

Proyecto Amber

- Completos

- Local-Variable Type Inference (var) (JDK 10)
- Local-Variable Syntax for Lambda Parameters (JDK 11)
- Switch Expressions [Funcionalidad Preliminar] (JDK 12)
- Text Blocks [Funcionalidad Preliminar] (JDK 12)

- En desarrollo

- Mejoras para Lambda
- Pattern Matching
- Concise Method Bodies

```
var url = new URL("http://www.oracle.com/"); ]
```

```
var conn = url.openConnection();
```

```
var reader = new BufferedReader(  
    new InputStreamReader(conn.getInputStream()));
```

Proyecto Amber

- Completos

- Local-Variable Type Inference (var) (*JDK 10*)
- Local-Variable Syntax for Lambda Parameters (JDK 11)
- **Switch Expressions [Funcionalidad Preliminar] (JDK 12)**
- Text Blocks [Funcionalidad Preliminar] (JDK 12)

- En desarrollo

- Mejoras para Lambda
- Pattern Matching
- Concise Method Bodies

```
int numLetters;
switch (day) {
    case MONDAY:
    case FRIDAY:
    case SUNDAY:
        numLetters = 6;
        break;
    case TUESDAY:
        numLetters = 7;
        break;
    case THURSDAY:
    case SATURDAY:
        numLetters = 8;
        break;
    case WEDNESDAY:
        numLetters = 9;
        break;
    default:
        throw new IllegalStateException("wat: " + day);
}
```

Proyecto Amber

- Completos

- Local-Variable Type Inference (var) (JDK 10)
- Local-Variable Syntax for Lambda Parameters (JDK 11)
- **Switch Expressions [Funcionalidad Preliminar] (JDK 12)**
- Text Blocks [Funcionalidad Preliminar] (JDK 12)

- En desarrollo

- Mejoras para Lambda
- Pattern Matching
- Concise Method Bodies

```
int numLetters = switch (day) {  
    case MONDAY, FRIDAY, SUNDAY -> 6;  
    case TUESDAY                 -> 7;  
    case THURSDAY, SATURDAY     -> 8;  
    case WEDNESDAY              -> 9;  
};
```

/*The cases of a switch expression must be *exhaustive*; for all possible values there must be a matching switch label.

In practice this normally means that a default clause is required (except for enums)*/

Proyecto Amber

- Completos

- Local-Variable Type Inference (var) (JDK 10)
- Local-Variable Syntax for Lambda Parameters (JDK 11)
- Switch Expressions [Funcionalidad Preliminar] (JDK 12)
- **Text Blocks [Funcionalidad Preliminar] (JDK 12)**

- En desarrollo

- Mejoras para Lambda
- Pattern Matching
- Concise Method Bodies

```
String html = "<html>\n" +  
    "<body>\n" +  
    "<p>Hello World.</p>\n" +  
    "</body>\n" +  
    "</html>\n";
```

Proyecto Amber

- Completos

- Local-Variable Type Inference (var) (JDK 10)
- Local-Variable Syntax for Lambda Parameters (JDK 11)
- Switch Expressions [Funcionalidad Preliminar] (JDK 12)
- **Text Blocks [Funcionalidad Preliminar] (JDK 12)**

- En desarrollo

- Mejoras para Lambda
- Pattern Matching
- Concise Method Bodies

```
String html = """
    <html>
        <body>
            <p>Hello World.</p>
        </body>
    </html>
    """;
```

Proyecto Amber

- Completos

- Local-Variable Type Inference (var) (JDK 10)
- Local-Variable Syntax for Lambda Parameters (JDK 11)
- Switch Expressions [Funcionalidad Preliminar] (JDK 12)
- **Text Blocks [Funcionalidad Preliminar] (JDK 12)**

- En desarrollo

- Mejoras para Lambda
- Pattern Matching
- Concise Method Bodies

```
String html = """  
.....<html>  
.....<body>  
.....<p>Hello World.</p>  
.....</body>  
.....</html>  
.....""";
```

Proyecto Amber

- Completos

- Local-Variable Type Inference (var) (JDK 10)
- Local-Variable Syntax for Lambda Parameters (JDK 11)
- Switch Expressions [Funcionalidad Preliminar] (JDK 12)
- Text Blocks [Funcionalidad Preliminar] (JDK 12)

- En desarrollo

- Mejoras para Lambda
- **Pattern Matching**
- Concise Method Bodies

```
String formatted = "unknown";
if (constant instanceof Integer) {
    int i = (Integer) constant;
    formatted = String.format("int %d", i);
}
else if (constant instanceof Byte) {
    byte b = (Byte) constant;
    formatted = String.format("byte %d", b);
}
else if (constant instanceof Long) {
    long l = (Long) constant;
    formatted = String.format("long %d", l);
}
else if (constant instanceof Double) {
    Double d = (Double) constant;
    formatted = String.format("double %f", d);
}
// Short, Character, Float, Boolean
else if (constant instanceof String) {
    String s = (String) constant;
    formatted = String.format("String %s", s);
}
```

Proyecto Amber

- Completos

- Local-Variable Type Inference (var) (JDK 10)
- Local-Variable Syntax for Lambda Parameters (JDK 11)
- Switch Expressions [Funcionalidad Preliminar] (JDK 12)
- Text Blocks [Funcionalidad Preliminar] (JDK 12)

- En desarrollo

- Mejoras para Lambda
- **Pattern Matching**
- Concise Method Bodies

```
String formatted;
switch (constant) {
    case Integer i:
        formatted = String.format("int %d", i);
        break;
    case Byte b:
        formatted = String.format("byte %d", b);
        break;
    case Long l:
        formatted = String.format("long %d", l);
        break;
    case Double d:
        formatted = String.format("double %f", d);
        break;
    case String s:
        formatted = String.format("String %s", s);
        break;
    // Short, Character, Float, Boolean
    default:
        formatted = "unknown";
}
```

Proyecto Amber

- Completos

- Local-Variable Type Inference (var) (JDK 10)
- Local-Variable Syntax for Lambda Parameters (JDK 11)
- **Switch Expressions [Funcionalidad Preliminar] (JDK 12)**
- Text Blocks [Funcionalidad Preliminar] (JDK 12) }

- En desarrollo

- Mejoras para Lambda
- **Pattern Matching**
- Concise Method Bodies

```
String formatted =
    switch (constant) {
        case Integer i -> String.format("int %d", i);
        case Byte b     -> String.format("byte %d", b);
        case Long l     -> String.format("long %d", l);
        case Double d   -> String.format("double %f", d);
        case String s   -> String.format("String %s", s);
        // Short, Character, Float, Boolean
        default -> "unknown";
    }
```

Advertencia:

La siguiente sección no ha sido traducida, y la veremos a la velocidad del estornudo....
(toda la información esta en OpenJDK 9)

Sección bonus

Solo para los que todavia tienen Java 9 en el futuro

Behind the scenes improvements

Goodness you get for free just by updating to JDK 9

No need for user to change anything to benefit from these

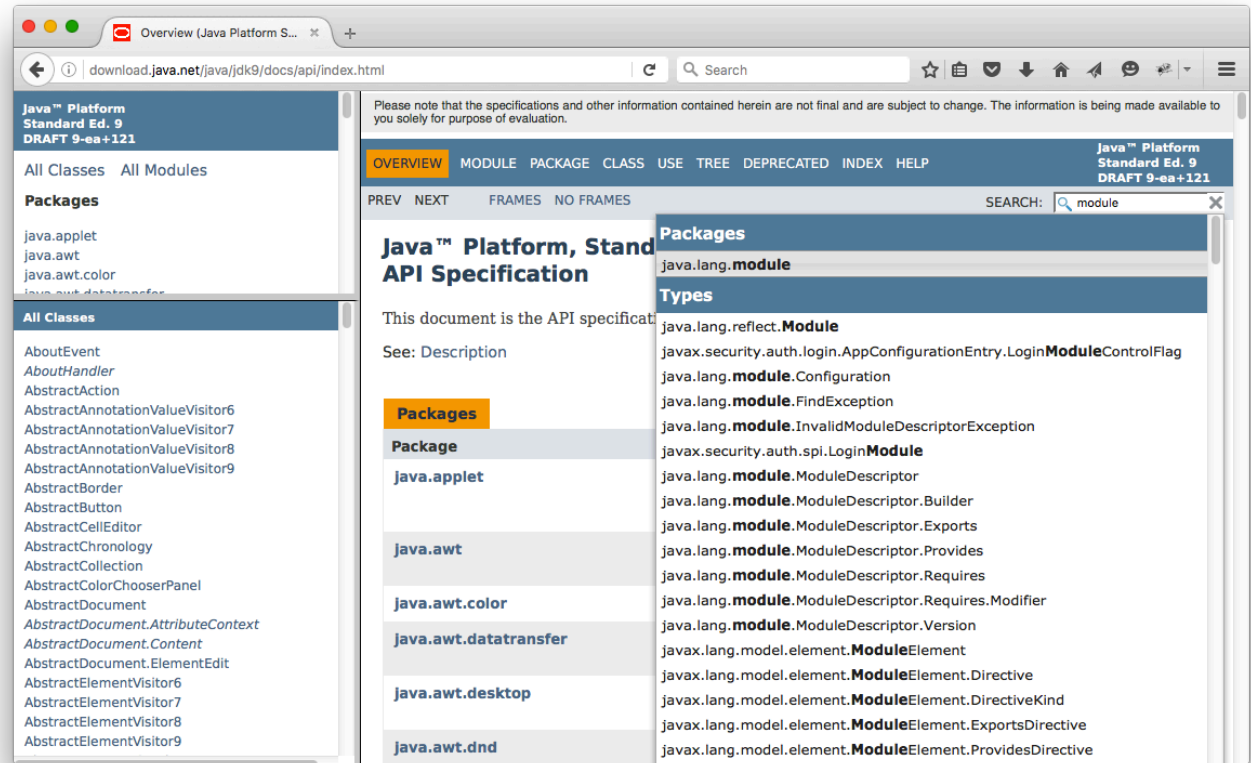
JEP 254: Compact Strings

core-libs / java.lang

- Adopt a more space-efficient internal representation for strings
- **Less memory used for storing strings**
- String class stores characters in a char array, using two bytes (sixteen bits) for each character
- Change the internal representation of the String class to a byte array plus an encoding-flag field

JEP 225: Javadoc Search tools / javadoc(tool)

- Add a search box to generated API documentation that can be used to search for program elements and tagged words and phrases within the documentation



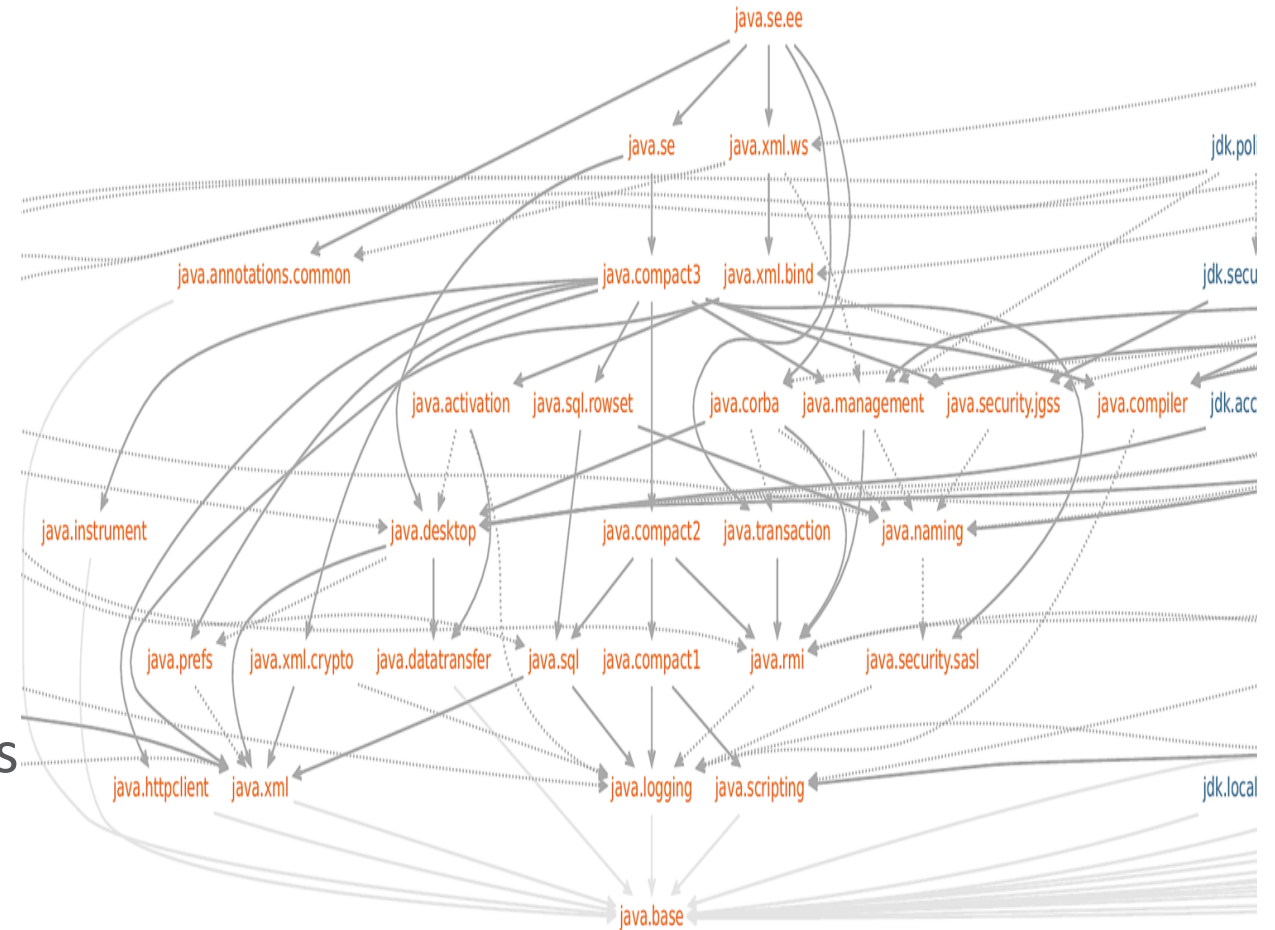
New features and functionality

**New tools and capabilities likely to be useful to most developers
Will have to choose to use these**

Project Jigsaw

Modularize the Java Platform

- JEP 261: Module System
- JEP 200: The Modular JDK
- JEP 201: Modular Source Code
- JEP 220: Modular Run-Time Images
- Plus
 - JEP 260: Encapsulate Most Internal APIs
 - JEP 282: jlink: The Java Linker



JEP 282: jlink: The Java Linker

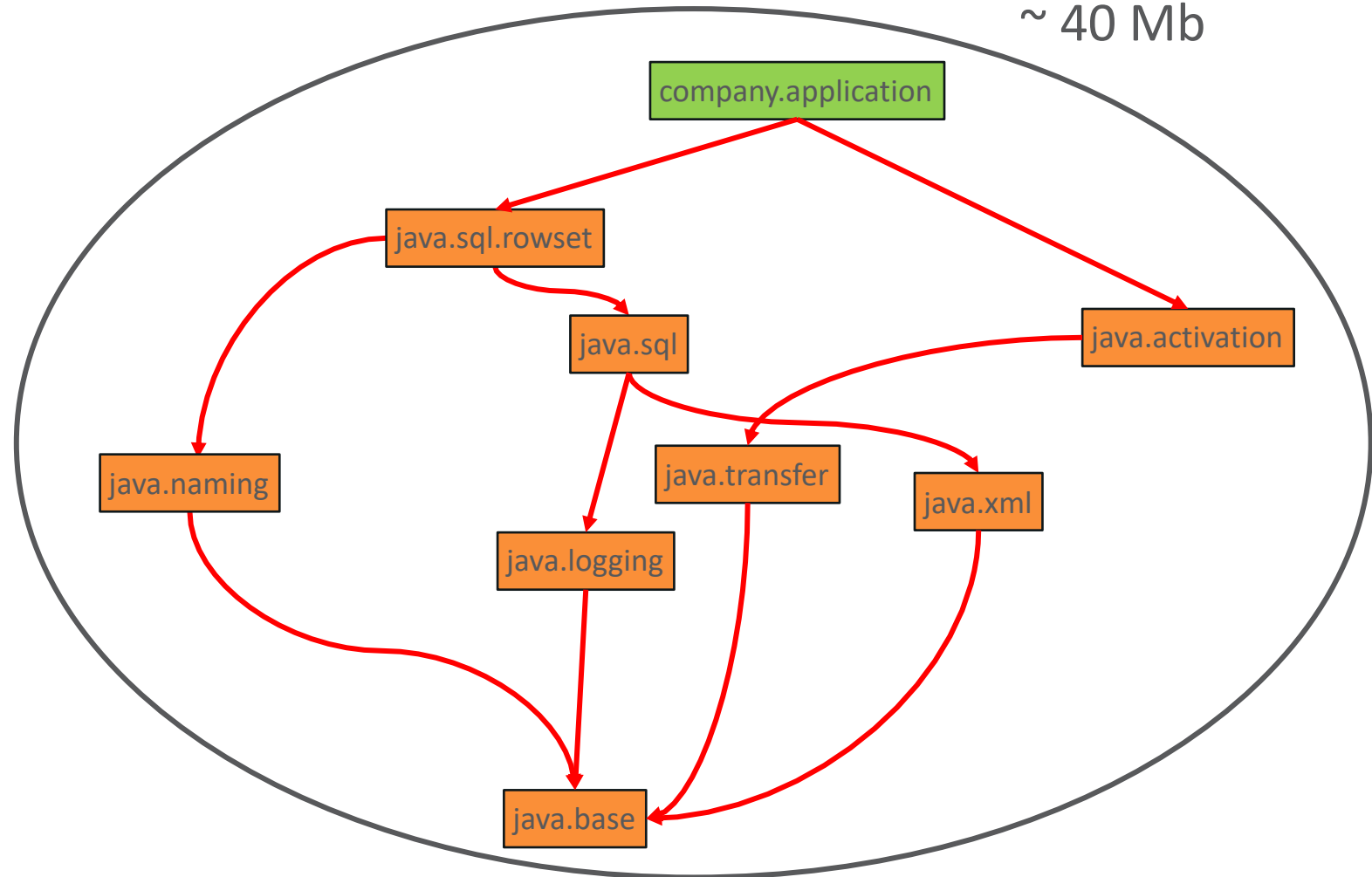
tools / jlink

- Create a tool that can assemble and optimize a set of modules and their dependencies into a custom run-time image as defined in JEP 220. Define a plugin mechanism for transformation and optimization during the assembly process, and for the generation of alternative image formats
- **Create a custom runtime optimized for a single program**
- JEP 261 defines *link time* as an optional phase between the phases of compile time and run time. Link time requires a linking tool that will assemble and optimize a set of modules and their transitive dependencies to create a run-time image or executable

Java Custom Runtime

Custom Image
~ 40 Mb

- Includes the Modular Application



JEP 277: Enhanced Deprecation

core-libs / java.lang

`@Deprecated(since=9, forRemoval=true)`

- Revamp the deprecation annotation, and provide tools to strengthen the API life cycle
- Provide better information about the status and intended disposition of APIs in the specification
- Provide a tool to analyze an application's static usage of deprecated APIs
- Provide a tool to detect an application's dynamic usage of of deprecated APIs in order to emit warnings at runtime

JEP 269: Convenience Factory Methods for Collections

core-libs / java.util:collections

- Define library APIs to make it convenient to create instances of collections and maps with small numbers of elements, so as to ease the pain of not having collection literals in the Java programming language
- **Decrease the amount of code needed for creating small collections and maps**

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

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

tools / jshell

- Provide an interactive tool to evaluate declarations, statements, and expressions of the Java programming language, together with an API so that other applications can leverage this functionality
- A Read-Eval-Print Loop (REPL) is an interactive programming tool which loops, continually reading user input, evaluating the input, and printing the value of the input or a description of the state change the input caused. Scala, Ruby, JavaScript, Haskell, Clojure, and Python all have REPLs and all allow small initial programs. JShell adds REPL functionality to the Java platform

Adopting new standards

JDK 9 keeping up with improvements in the industry

JEP 287: SHA-3 Hash Algorithms

`security-libs / java.security`

- Implement the SHA-3 cryptographic hash functions (BYTE-only) specified in NIST FIPS 202

JEP 224: HTML5 Javadoc

tools / javadoc(tool)

- Enhance the javadoc tool to allow generating HTML5 markup.

JEP 110: HTTP/2 Client

core-libs / java.net

- Define a new HTTP client API that implements HTTP/2 and WebSocket, and can replace the legacy HttpURLConnection API
- For JDK 9 this API will be on the incubator modules (JEP 11) with the goal of adding them to the standard on a later release.



- Store Interned Strings in CDS Archives
- Improve Contended Locking
- Compact Strings
- Improve Secure Application Performance
- Leverage CPU Instructions for GHASH and RSA
- Tiered Attribution for javac
- Javadoc Search
- Marlin Graphics Renderer
- HiDPI Graphics on Windows and Linux
- Enable GTK 3 on Linux
- Update JavaFX/Media to Newer Version of GStreamer

Behind the scenes

- **Jigsaw – Modularize JDK**
- Enhanced Deprecation
- Stack-Walking API
- Convenience Factory Methods for Collections
- Platform Logging API and Service
- jshell: The Java Shell (Read-Eval-Print Loop)
- Compile for Older Platform Versions
- Multi-Release JAR Files
- Platform-Specific Desktop Features
- TIFF Image I/O
- Multi-Resolution Images

New functionality

- Process API Updates
- Variable Handles
- Spin-Wait Hints
- Dynamic Linking of Language-Defined Object Models
- Enhanced Method Handles
- More Concurrency Updates
- Compiler Control

Specialized

- HTTP 2 Client
- Unicode 8.0
- UTF-8 Property Files
- Implement Selected ECMAScript 6 Features in Nashorn
- Datagram Transport Layer Security (DTLS)
- OCSP Stapling for TLS
- TLS Application-Layer Protocol Negotiation Extension
- SHA-3 Hash Algorithms
- DRBG-Based SecureRandom Implementations
- Create PKCS12 Keystores by Default
- Merge Selected Xerces 2.11.0 Updates into JAXP
- XML Catalogs
- HarfBuzz Font-Layout Engine
- HTML5 Javadoc

New standards

- Parser API for Nashorn
- Prepare JavaFX UI Controls & CSS APIs for Modularization
- Modular Java Application Packaging
- New Version-String Scheme
- Reserved Stack Areas for Critical Sections
- Segmented Code Cache
- Ahead-of-Time Compilation
- Indify String Concatenation
- Unified JVM Logging
- Unified GC Logging
- Make G1 the Default Garbage Collector
- Use CLDR Locale Data by Default
- Validate JVM Command-Line Flag Arguments
- Java-Level JVM Compiler Interface
- Disable SHA-1 Certificates
- Simplified Doclet API
- Deprecate the Applet API
- Process Import Statements Correctly
- Annotations Pipeline 2.0
- Elide Deprecation Warnings on Import Statements
- Milling Project Coin
- Filter Incoming Serialization Data
- Remove GC Combinations Deprecated in JDK 8
- Remove Launch-Time JRE Version Selection
- Remove the JVM TI hprof Agent
- Remove the jhat Tool

Housekeeping

Gone





Y ahora.. regresamos a nuestra programa original...

Proyecto Portola

Java en un mundo de contenedores

- Las características de Java lo hacen ideal para ambientes con contenedores
 - Seguro y protegido, alta performance, confiable, rico ecosistema
- Estamos comprometidos en mantener a Java como la mejor opción para aplicaciones en la nube



Proyecto Panama

- Funciones y Data nativos/en otro lenguaje
- Remplazo simple, seguro y de alta performance para JNI
- Acceso directo a funciones de hardware de bajo nivel usando Java
 - Instrucciones Vectoriales, tipos especial de memoria (ej. memoria no-volátil)
- Big Data, Machine Learning

Proyecto Skara

- Investiga alternativas de software de control de versiones para el código fuente del JDK
- Las opciones investigadas están basadas en Git en lugar de Mercurial
- El objetivo es permitir que los colaboradores en OpenJDK sean mas productivos, tanto los expertos como los novatos

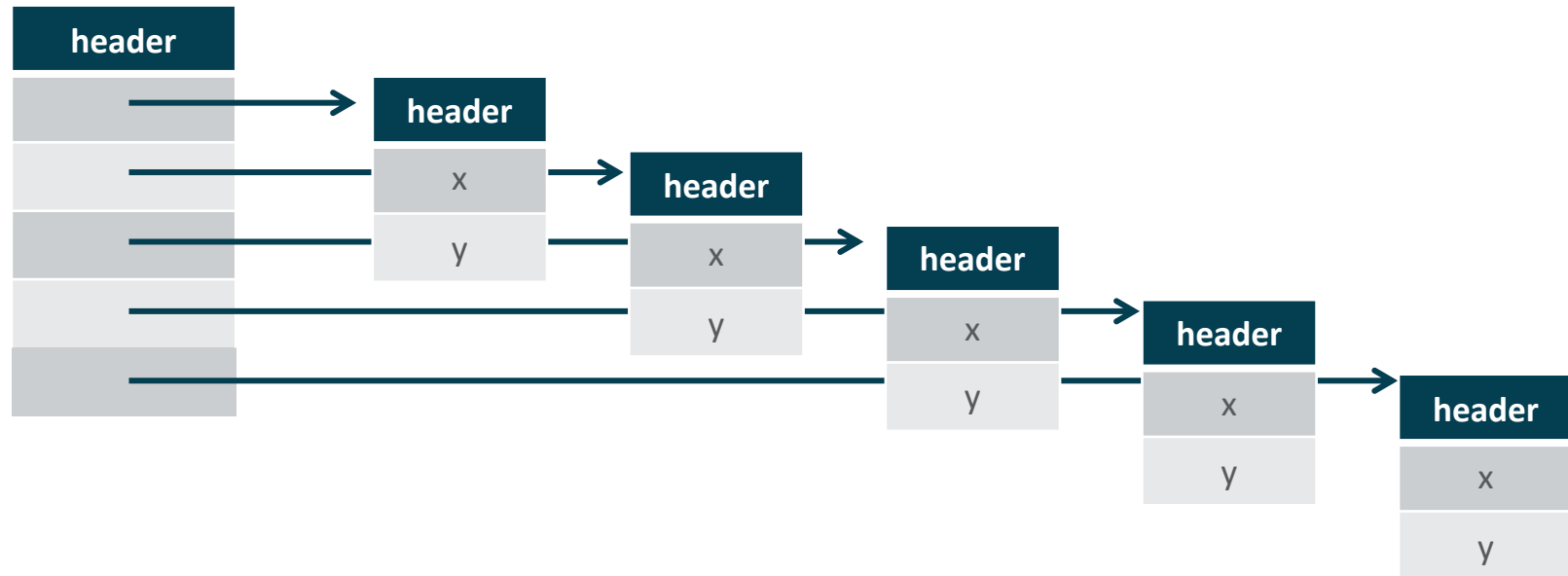


Proyecto Valhalla

Data en Memoria con Java hoy

```
final class Point {  
    final int x;  
    final int y;  
}
```

Point[] pts =



Proyecto Valhalla

Data en Memoria como nos gustaría

```
inline class Point {  
    int x;  
    int y;  
}
```

`Point[] pts =`

header
x
y
x
y
x
y
x
y

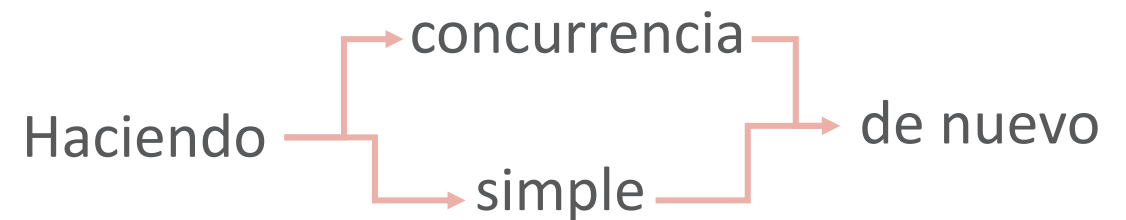
Value Types:

Tan sencillo de usar como una clase, con la performance de una primitiva.

¿Qué haría int?

Proyecto Loom

- Modelo para concurrencia mas sencillo y escalable
 - Hacer llamadas bloqueantes prácticamente sin costo
- “Fibras” (como hilos pero menos costosos) y continuaciones
 - Millones de fibras pueden ser creadas en una sola instancia del JVM



OpenJDK

versiones preliminares

- Futuras Versiones Funcionales
- Proyectos de OpenJDK



The screenshot shows a web browser window with the URL `jdk.java.net`. The page content includes the following text:

jdk.java.net

Java Development Kit builds, from Oracle

Ready for use: JDK 12

Early access: JDK 14, JDK 13, jpackage, OpenJFX, Panama, Valhalla, & JMC

Reference implementations: Java SE 12, 11, 10, 9, 8, & 7

At the bottom left is the ORACLE logo. At the bottom right is the copyright notice: © 2019 Oracle Corporation and/or its affiliates. Terms of Use · Privacy · Trademarks.

Resumen

- El desarrollo de la plataforma Java en OpenJDK se esta volviendo mas abierto
- JDK 11, la primera versión LTS , incluye innovación y mejores en seguridad, productividad y performance
- Oracle ha introducido una suscripción de Java para hacerlo mas asequible y mantener los costos predecibles
- Continuamos trabajando en diversos frentes para traer mejoras a Java cada seis meses

ÚNETE y conviértete en un **COLLABORDOR** de **OpenJDK**

<https://openjdk.java.net>

MANTENTE INFORMADO

<https://blogs.oracle.com/java-platform-group>

[@OpenJDK](#) [@gsaab](#) [@mreinhold](#) [@BrianGoetz](#)

[@BTraTra](#) [@DonaldOJDK](#) [@MikaelVidstedt](#)

[@Sharat_Chander](#) [@robilad](#)

[@aureliog](#) [@SimmsUpNorth](#)

Q & A

