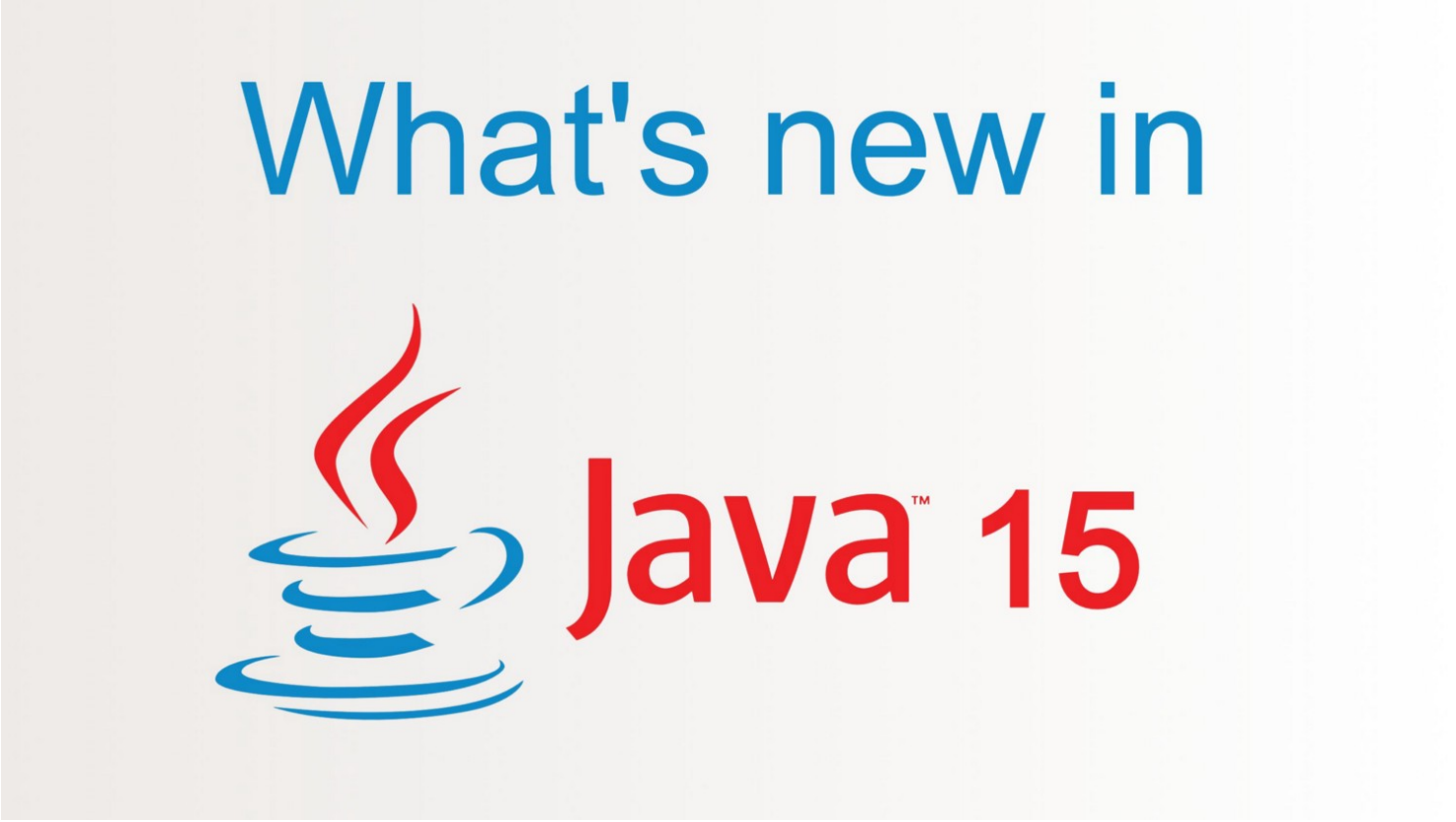
What's new in Java 15

In this article, we will check what was added and removed in Java 15.

Dmytro Timchenko Follow
Nov 21, 2020 · 7 min read ★



JDK 15 is the open-source reference implementation of version 15 of the Java SE Platform, as specified by JSR 390 in the Java Community Process.

JDK 15 reached General Availability on 15 September 2020. 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).

Removals

Solaris and SPARC Ports

The source code and build support for the Solaris/SPARC, Solaris/x64, and Linux/SPARC ports were removed. These ports were deprecated for removal in JDK 14 with the express intent to remove them in a future release.

> JEP 381: Remove the Solaris and SPARC Ports
>
> Remove the source code and build support for the Solaris/SPARC, Solaris/x64, and Linux/SPARC ports. These ports were...
>
> openjdk.java.net

Nashorn Javascript Engine

The Nashorn JavaScript script engine and APIs, and the `jjs` tool were removed. The engine, the APIs, and the tool were deprecated for removal in Java 11 with the express intent to remove them in a future release.

> JEP 372: Remove the Nashorn JavaScript Engine
>
> Remove the Nashorn JavaScript script engine and APIs, and the jjs tool. The engine, the APIs, and the tool were...
>
> openjdk.java.net

Deprecations

RMI Activation

RMI Activation mechanism is deprecated. RMI Activation is an obsolete part of RMI that has been optional since Java 8. No other part of RMI will be deprecated.

> JEP 385: Deprecate RMI Activation for Removal
>
> Deprecate the RMI Activation mechanism for future removal. RMI Activation is an obsolete part of RMI that has been...
>
> openjdk.java.net

What's new

Helpful NullPointerExceptions

Usability of `NullPointerException` has been improved. Messages generated by the JVM are describing precisely which variable was `null` .

Before Java 15

In Java 15

```
a().b().c.doSomething();
```

```
Exception in thread "main" java.lang.NullPointerException:
   Cannot invoke "Example$C.doSomething()" because
   "Example$A.b().c" is null
```

> JEP 358: Helpful NullPointerExceptions
>
> Improve the usability of NullPointerExceptions generated by the JVM by describing precisely which variable was null...
>
> openjdk.java.net

Text Blocks

A text block is a multi-line string literal that avoids the need for most escape sequences, automatically formats the string in a predictable way, and gives the developer control over the format when desired.

Before Java 15

```
String json = "{\n  \"firstName\": \"Sander\",\n   \"lastName\"
```

```
String json = "{\n" +
        "  \"firstName\": \"Sander\",\n" +
        "  \"lastName\": \"Mak\"\n" +
        "}";
```

```
String json = new StringBuilder()
        .append("{\n")
        .append("  \"firstName\": \"Sander\",\n")
        .append("  \"lastName\": \"Mak\"\n")
        .append("}")
        .toString();
```

In Java 15

```
String json = """
          {
             "firstName": "Sander",
             "lastName": "Mak"
          }""";
```

> JEP 378: Text Blocks
>
> Add text blocks to the Java language. A text block is a multi-line string literal that avoids the need for most escape...
>
> openjdk.java.net

JVM Improvements

ZGC Garbage Collector

The Z Garbage Collector (ZGC) is a scalable low latency garbage collector. ZGC performs all expensive work concurrently, without stopping the execution of application threads for more than 10ms, which makes it suitable for applications that require low latency and/or use a very large heap (multi-terabytes).

At a glance, ZGC is:

- Concurrent
- Region-based
- Compacting
- NUMA-aware
- Using colored pointers
- Usando barreras de carga

En esencia, ZGC es un recolector de basura **concurrente** , lo que significa que todo el trabajo pesado se realiza mientras los **subprocesos de Java continúan ejecutándose** . Esto limita en gran medida el impacto que tendrá la recolección de basura en el tiempo de respuesta de su aplicación.

A partir de Java 15, es un GC listo para producción.

> JEP 333: ZGC: un recolector de basura escalable de baja latencia (experimental)
>
> El recolector de basura Z, también conocido como ZGC, es un recolector de basura escalable de baja latencia. Los tiempos de pausa de GC no deberían ...
>
> openjdk.java.net

Clases ocultas

Clases que no pueden ser utilizadas directamente por el código de bytes de otras clases. Las clases ocultas están pensadas para que las utilicen los marcos que generan clases en tiempo de ejecución y las usan indirectamente, a través de la reflexión. Una clase oculta

puede definirse como miembro de un nido de control de acceso y puede descargarse independientemente de otras clases.

> JEP 371: Clases ocultas
>
> Introduzca clases ocultas, que son clases que no pueden ser utilizadas directamente por el código de bytes de otras clases. Oculto...
>
> openjdk.java.net

EdDSA

EdDSA es un esquema de firma de curva elíptica moderno que tiene varias ventajas sobre los esquemas de firma existentes en el JDK.

> JEP 339: Algoritmo de firma digital Edwards-Curve (EdDSA)
>
> Implemente firmas criptográficas utilizando el algoritmo de firma digital Edwards-Curve (EdDSA) como lo describe RFC ...
>
> openjdk.java.net

API de DatagramSocket

Reemplazo de las implementaciones subyacentes de las API `java.net.DatagramSocket` y `java.net.MulticastSocket` con implementaciones más simples y modernas que son fáciles de mantener y depurar. Las nuevas implementaciones son fáciles de adaptar para trabajar con subprocesos virtuales, y actualmente se están explorando en Project Loom .

> JEP 373: Reimplementar la API de DatagramSocket heredada
>
> Reemplace las implementaciones subyacentes de las API java.net.DatagramSocket y java.net.MulticastSocket con más simples...
>
> openjdk.java.net

Funciones de vista previa

Una función de vista previa es una nueva función cuyo diseño, especificación e implementación están completos, pero que no es permanente, lo que significa que la función puede existir en una forma diferente o no existir en las futuras versiones de JDK.

La coincidencia de patrones

La coincidencia de patrones implica probar si un objeto tiene una estructura particular y luego extraer datos de ese objeto si hay una coincidencia. Ya puedes hacer esto con Java ; sin embargo, la coincidencia de patrones introduce nuevas mejoras de lenguaje que le permiten extraer datos condicionalmente de objetos con un código más conciso y sólido.

Más específicamente, JDK 15 extiende el `instanceof` operador: puede especificar una *variable de enlace* ; si el resultado del `instanceof` operador es `true` , entonces el objeto que se está probando se asigna a la variable de enlace.

```
if (object instanceof BigDecimal) {
   BigDecimal b = (BigDecimal) object;
   return b.precision();
}
```

```
if (object instanceof BigDecimal b) {
   return b.precision();
}
```

JEP 375: Coincidencia de patrones, por ejemplo, de (Segunda vista previa)

Mejore el lenguaje de programación Java con coincidencia de patrones para la instancia del operador. La coincidencia de patrones permite lo común ...

openjdk.java.net

Registros

Introducidas como una función de vista previa en Java SE 14, las clases de registros ayudan a modelar agregados de datos sin formato con menos ceremonia que las clases normales. Java SE 15 amplía la función de vista previa con capacidades adicionales como clases de registros locales.

Una clase de registro declara una secuencia de campos, y después los correspondientes **descriptores de acceso, constructores `equals`, `hashCode` y `toString` los métodos se crean automáticamente** . Los campos son definitivos porque la clase está destinada a servir como un simple "soporte de datos". Significa que los **registros son inmutables .**

```
public record Person(String name, int age, String hobby) {
}
```

JEP 384: Registros (segunda vista previa)

Mejore el lenguaje de programación Java con registros, que son clases que actúan como portadores transparentes para inmutables ...

openjdk.java.net

Clases Selladas

Las clases e interfaces selladas restringen qué otras clases o interfaces pueden extenderlas o implementarlas.

Uno de los propósitos principales de la herencia es la reutilización del código: cuando desea crear una nueva clase y ya existe una clase que incluye parte del código que desea, puede derivar su nueva clase a partir de la clase existente. Al hacer esto, puede reutilizar los campos y métodos de la clase existente sin tener que escribirlos (y depurarlos) usted mismo.

Antes de clases selladas

```
public abstract class Option<T> {

    private Option() {}

    public final static class Some<T> extends Option<T> { ... }
```

```
    public final static class Empty extends Option<Void> { ... }

    // ...
}
```

Con clases selladas

```
public sealed class Option<T> permits Some, Empty {
  // ..
}
```

```
public final class Empty extends Option<Void> {
  // ..
}
```

```
public final class Some<T> extends Option<T> {
  // ..
}
```

> JEP 360: clases selladas (versión preliminar)
>
> Mejore el lenguaje de programación Java con clases e interfaces selladas. Las clases e interfaces selladas restringen qué ...
>
> openjdk.java.net

Conclusión

En este artículo, hemos comprobado qué se agregó y quitó en Java 15. Y cómo todos los cambios que se entregaron con Java 15 pueden mejorar sus proyectos existentes.

¿Te ha resultado útil este artículo? ¡Sígueme (Dmytro Timchenko) en Medium y mira mis otros artículos a continuación! ¡Por favor 👏 este artículo para compartirlo!

Concurrencia de Java: bloqueos

En este artículo, cubriremos los bloqueos en la programación multiproceso de Java y vamos a utilizarlos.

medium.com

Pruebas de integración con Spring Boot, TestContainers, Liquibase y JUnit 5

En este artículo, cubriremos cómo hacer pruebas de integración en la aplicación Spring Boot con PostgreSQL DB, Liquibase ...

medium.com

Concurrencia de Java: punto muerto

En este artículo, cubriremos situaciones de interbloqueo en la programación multiproceso de Java y trataremos de evitarlas.

medium.com

Recursos:

JDK 15: https://openjdk.java.net/projects/jdk/15/

Cambios en Java: https://docs.oracle.com/en/java/javase/15/language/java-language-changes.html#GUID-6459681C-6881-45D8-B0DB-395D1BD6DB9B

Novedades en Java java 15: https://app.pluralsight.com/library/courses/java-15-whats-new/table-of-contents

Pluralsight también ofrece un **fin de semana gratuito** , por lo que puede consultar este curso GRATIS.