

Guía de aprendizaje

Eficiencia en Entornos de Desarrollo Integrados (IDE)

Exp 2 – Semana 6

Fundamentos de Programación (PRY2201)

**Facilitador disciplinar:** Josué Oteiza

**Asesor par:** Luis Videla

# Índice

[Introducción a la semana 4](#_Toc158807474)

[Resultado de aprendizaje 5](#_Toc158807475)

[Conceptos relevantes 5](#_Toc158807476)

[Preguntas activadoras 6](#_Toc158807477)

[Actividad 6](#_Toc158807478)

[Entorno de Desarrollo Integrado (IDE) 7](#_Toc158807479)

[Integrated Development Environment 7](#_Toc158807480)

[Herramientas complementarias a Apache NetBeans 14](#_Toc158807489)

[Nuevos aspectos de Apache NetBeans 15](#_Toc158807490)

[Depuración o *Debugging* del código 18](#_Toc158807491)

[Pasos para realizar el *debugger* en NetBeans 18](#_Toc158807492)

[*Debugging Windows / Ventanas de depuración* 24](#_Toc158807493)

[Video 26](#_Toc158807494)

[Cierre de la semana 27](#_Toc158807495)

[Referencias 28](#_Toc158807496)

[Apuntes 29](#_Toc158807497)

# Introducción a la semana

Ya estás en la semana 6 de la asignatura y, como se vio en semanas anteriores, usamos algoritmos en Java para crear programas funcionales que solucionen las problemáticas planteadas, además del empleo de variables en Java que aseguren el correcto almacenamiento y manipulación de datos.

Todos estos aspectos los hemos aplicado a través del Entorno de Desarrollo Apache NetBeans, y en la cual profundizaremos para realizar procesos de depuración o debugging, componente crítico del proceso de desarrollo que implica identificar, aislar y corregir errores o "bugs" dentro de un programa. Como columna vertebral del mantenimiento de software, el debugging es vital para asegurar la funcionalidad, eficiencia y seguridad de las aplicaciones, y es una habilidad esencial en el arsenal de todo programador.

# Resultado de aprendizaje

### El estudiante será capaz de:

**RA2.** Aplica elementos básicos de Java para el desarrollo de programas que solucionen problemáticas planteadas.

**Indicador de logro:**

**IL4.** Utiliza Algoritmos en Java en la creación de programas funcionales que solucionen las problemáticas planteadas.

**IL5.** Emplea Variables en Java en la resolución de problemáticas planteadas, asegurando el correcto almacenamiento y manipulación de datos.

**IL6.** Utiliza Entornos de Desarrollo (IDEs) en la escritura, depuración y aplicación de expresiones aritméticas en programas Java que solucionen problemáticas planteadas.

# Conceptos relevantes

|  |  |  |
| --- | --- | --- |
| Control de Flujo | Eclipse | Entorno de Desarrollo Integrado (IDE) |
| Debugging | Ventana de Variables | Call Stack |

# Preguntas activadoras

* ¿Por qué es beneficioso utilizar un IDE en el desarrollo de software?
* ¿Qué herramientas de depuración has utilizado y cómo te han ayudado a identificar errores en tu código?
* ¿Cómo personalizas tu entorno de desarrollo para que se adapte a tu estilo de programación?

# Actividad

**Descripción de la actividad**

En esta sexta semana realizarás una actividad sumativa individual llamada “Depurando mi proyecto en el Entorno de Desarrollo Integrado (IDE)”, donde tendrás que brindar una solución a un caso planteado con los elementos básicos de Java como algoritmos, variables y estructuras de control. Además, tendrás que aplicar debugging en tu Entorno de Desarrollo para corregir posibles errores en tu programa.

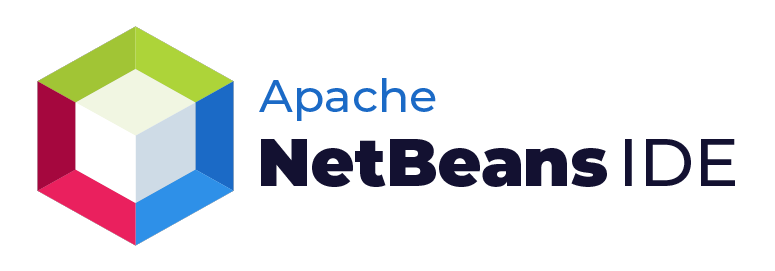
# Entorno de Desarrollo Integrado (IDE)

## Integrated Development Environment

Como su nombre lo indica, un IDE es una aplicación que proporciona a los programadores un conjunto de herramientas y características bajo una interfaz unificada para simplificar y agilizar el proceso de desarrollo de software.

Incluyen un editor de texto para escribir código, compiladores y/o intérpretes para construir y ejecutar aplicaciones, herramientas de depuración para diagnosticar y solucionar problemas, y a menudo integraciones con sistemas de control de versiones como Git.

Los IDEs pueden ser específicos de un lenguaje de programación o soportar múltiples lenguajes, y están diseñados para maximizar la productividad del desarrollador al reducir la necesidad de cambiar entre diferentes herramientas y aumentar la eficiencia del proceso de codificación, depuración y prueba.

Pero esto no es nuevo para ti, puesto que, para la creación de algoritmos, variables, estructuras de control, etc., has utilizado Apache NetBeans.

**Apache Software Foundation (ASF)**

Apache NetBeans es parte de la Apache Software Foundation (ASF), una organización sin fines de lucro que se dedica a apoyar proyectos de software de código abierto. La ASF proporciona apoyo legal, organizativo y financiero a una amplia variedad de proyectos de software libre.

ASF es responsable de una gran variedad de proyectos de software. Estos proyectos varían desde servidores web y sistemas de bases de datos hasta frameworks de desarrollo y herramientas de integración continua. A continuación, revisaremos algunos de los proyectos de software que podrán ayudarte en futuros proyectos:

**Apache Tomcat**

Apache Tomcat es un contenedor de servlets y un servidor web de código abierto, y aunque no es un Entorno de Desarrollo Integrado (IDE), juega un papel crucial en el desarrollo y despliegue de aplicaciones Java, especialmente aquellas basadas en tecnologías Java Servlet, JavaServer Pages (JSP), y Java Expression Language (EL). Algunas de sus características son:

* Arquitectura y componentes: está compuesto por varios componentes como el Conector, que maneja las solicitudes de red; el Contenedor de Servlets, que ejecuta los servlets; y el Motor JSP para páginas JSP.
* Seguridad: proporciona características para la autenticación, autorización, y aseguramiento de la confidencialidad de las comunicaciones.
* Comunidad y soporte: cuenta con una amplia comunidad de usuarios y desarrolladores que contribuyen constantemente a su mejora.

# Links de Interés

* Puedes realizar la descarga de esta herramienta a través del siguiente enlace: <https://tomcat.apache.org/>
* Y en el siguiente enlace encontrarás el Index: <https://downloads.apache.org/tomcat/>

# Videos

Además, en el siguiente video puedes revisar cómo realizar la descarga y configurarlo en Apache NetBeans: <https://www.youtube.com/watch?v=uBpo0tKkgjA>

**Apache Maven**

Se ha convertido en una de las herramientas estándar en la industria para la gestión de proyectos de software, debido a que estandariza la configuración de un proyecto en su ciclo de vida como en todas las fases de compilación y empaquetado y la instalación de mecanismos de distribución de librerías, para que puedan utilizar otros desarrolladores y equipos de desarrollo. También contempla temas relacionados con la integración continua, para poder realizar la ejecución de test unitarios y pruebas automatizadas, test de integración, etc. Dentro de sus características tenemos:

* Ciclo de vida del proyecto**:** gestiona el ciclo de vida completo del proyecto, desde la compilación, pruebas, empaquetado, hasta su despliegue.
* *Plugins* y repositorios:cuenta con una extensa colección de *plugins* y se integra con repositorios centralizados para manejar dependencias.
* Integración continua: se integra con sistemas de integración continua como Jenkins para automatizar la construcción y despliegue de aplicaciones.

# Links de Interés

* Puedes descargar la última versión a través del siguiente enlace: <https://maven.apache.org/download.cgi>
* Y en el siguiente enlace encontrarás el Index: <https://downloads.apache.org/maven/>

# Video

A través del siguiente video, aprenderás a configurar Apache Maven para la construcción de tu proyecto Java:

<https://www.youtube.com/watch?v=cL5eTtUIs2E>

**Apache Ant**

Apache Ant es una herramienta usada en [programación](https://es.wikipedia.org/wiki/Programaci%C3%B3n) para la realización de tareas mecánicas y repetitivas, normalmente durante la fase de [compilación](https://es.wikipedia.org/wiki/Compilaci%C3%B3n) y construcción (build). Es, por tanto, un software para procesos de [automatización de compilación](https://es.wikipedia.org/w/index.php?title=Automatizaci%C3%B3n_de_compilaci%C3%B3n&action=edit&redlink=1), similar a [Make](https://es.wikipedia.org/wiki/Make) pero desarrollado en [lenguaje Java](https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)) y requiere la [plataforma Java](https://es.wikipedia.org/wiki/Plataforma_Java), así que es más apropiado para la construcción de proyectos Java. Esta herramienta, hecha en este [lenguaje de programación](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n), tiene la ventaja de no depender de las órdenes del [shell](https://es.wikipedia.org/wiki/Int%C3%A9rprete_de_comandos) de cada sistema operativo, sino que se basa en archivos de configuración [XML](https://es.wikipedia.org/wiki/XML) y clases Java para la realización de las distintas tareas, siendo idónea como solución multi-plataforma. Algunas de sus características son:

* Scripts de construcción personalizables: Los scripts XML de Ant pueden personalizarse para realizar tareas específicas más allá de la compilación y empaquetado.
* Extensibilidad: Los desarrolladores pueden escribir sus propios "tasks" para extender las capacidades de Ant.
* Integración con IDEs: Se integra fácilmente con entornos de desarrollo como Eclipse y NetBeans.

# Links de Interés

* Puedes visitar la página de Apache Ant y descargar la última versión a través del siguiente enlace:

<https://ant.apache.org/bindownload.cgi>

Y en el siguiente enlace encontrarás el Index: <https://downloads.apache.org/ant/>

# Video

Aprende más sobre esta útil herramienta a través del siguiente video:

<https://www.youtube.com/watch?v=OzFvOu2UMcc>

Importante

Usar Apache Ant te permite tener un proceso de construcción reproducible y personalizable que es crucial para proyectos grandes y complejos. Aunque puede tener una curva de aprendizaje inicial, su flexibilidad y capacidad de integración lo hacen una herramienta valiosa para cualquier desarrollador que trabaje en un entorno de Java.

Ahora te presentaremos otro Entorno de Desarrollo Integrado, nos referimos a:

**Eclipse**

A diferencia de NetBeans, este IDE fue desarrollado originalmente por IBM, ahora es mantenido por la Eclipse Foundation. Tiene una comunidad muy activa y una gran cantidad de *plugins* desarrollados por la comunidad. Su interfaz es altamente personalizable, pero puede ser un poco abrumadora para los principiantes debido a su complejidad y la gran cantidad de opciones disponibles. Sin embargo, es una plataforma de desarrollo de código abierto que soporta varios lenguajes de programación, siendo especialmente popular para el desarrollo en Java.

**Figura 1**

*Menú de Eclipse*

*Nota*. Muestra de menú de Eclipse. Holger Voormann (2023). *Eclipse 2023-12 Java IDE Improvements.* [YouTube]: <https://www.youtube.com/watch?v=ilM7GY1uEYI&t=64s>

Uno de los aspectos más poderosos de Eclipse es su sistema de *plugins.* Los usuarios pueden extender la funcionalidad de Eclipse instalando *plugins* adicionales. Esto permite, por ejemplo, integrar sistemas de control de versiones, herramientas de gestión de bases de datos y frameworks específicos para el desarrollo de aplicaciones.

# Links de Interés

En su página oficial puedes descargar su paquete de escritorio, además de otras herramientas que optimizarán su funcionamiento:

<https://www.eclipse.org/downloads/>

# Videos

Además, en el siguiente video, aprenderás a descargarlo e instalarlo:

<https://www.youtube.com/watch?v=hasHQ3o8_is>

Acabamos de ver tres herramientas pertenecientes a ASF y Eclipse, que sin duda te ayudarán en tus proyectos. No te preocupes si no profundizamos en ellas, puesto que las interiorizarás a lo largo de tu carrera.

## Herramientas complementarias a Apache NetBeans

Existen dos herramientas que aportan a NetBeans con roles específicos y complementarios. Una de ellas la conoces:

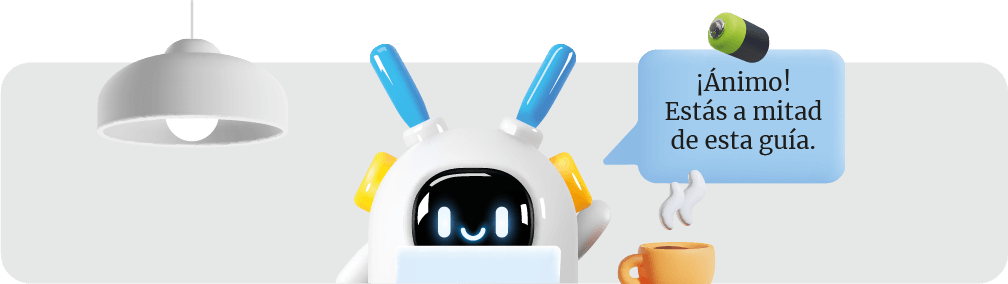
**Java Development Kit (JDK)**

Esta herramienta es esencial para el desarrollo de aplicaciones Java en NetBeans. Incluye herramientas como el compilador Java (javac), herramientas de empaquetado (jar), el generador de documentación (javadoc), y otras utilidades necesarias para desarrollar aplicaciones Java.

La segunda herramienta es:

**Java Virtual Machine (JVM)**

Es el componente de la plataforma que ejecuta programas escritos en el lenguaje Java. No interactúas directamente con la JVM en el sentido tradicional, ya que su funcionamiento es mayormente automático e invisible para el usuario. Sin embargo, comprender cómo funciona y cómo puedes influir en su comportamiento puede ser crucial para optimizar el rendimiento y la gestión de tus aplicaciones Java. Esto significa que puedes desarrollar aplicaciones en NetBeans y esperar que se comporten de manera consistente en cualquier plataforma que tenga un JVM compatible.



# Nuevos aspectos de Apache NetBeans

En este punto, ya deberíamos tener nociones de la estructura que NetBeans maneja para trabajar con programas de Java, sin embargo, no está de más que recordemos un poco al respecto:

**Figura 2**

*Estructura de proyectos en NetBeans*

*Nota*. Ejemplo de estructura de proyectos en Netbeans. Nieva, G. (2015). *Guía básica para usar Netbeans. Dcodingames*. <https://dcodingames.com/guia-basica-para-usar-netbeans/>

Esta estructura detalla cómo se organiza un proyecto de Java básico en NetBeans, especialmente cuando no se utiliza un sistema de construcción como Maven. Aquí te explicamos cómo se refleja esta estructura de manera práctica:

1. **Proyecto:** en el nivel más alto de la jerarquía está el proyecto. Cada proyecto en NetBeans se almacena en una carpeta separada.
2. **Paquetes fuente (Source Packages):** dentro de cada proyecto, hay una carpeta para los "Paquetes Fuente" donde resides tus archivos de código fuente Java (archivos .java). Además, puedes tener varios paquetes dentro de esta carpeta, y cada paquete puede contener uno o más archivos .java, y aunque cada archivo .java puede tener su propio método *main*, generalmente solo uno de ellos actúa como el punto de entrada principal de tu aplicación.
3. **Bibliotecas o Librerías (Libraries):** además de los paquetes fuente, tendrás una carpeta para las Bibliotecas o Librerías. Aquí es donde NetBeans almacena las dependencias externas y las bibliotecas necesarias para tu proyecto, como las bibliotecas de Java Runtime Environment (JRE) y cualquier otra biblioteca que hayas añadido a tu proyecto.
4. **Creación de un proyecto simple:** al crear un proyecto Java básico sin usar Programación Orientada a Objetos (POO) extensivamente, generalmente comenzarás con un solo paquete que contiene un único archivo .java. Este archivo .java incluirá tu método main, que es el punto de inicio de tu aplicación Java.
5. **Ubicación del proyecto:** por defecto, los proyectos en NetBeans se guardan en la carpeta ‘NetBeansProjects’ dentro de tu directorio Documentos (o una ubicación similar, dependiendo de tu sistema operativo y configuración), y cada proyecto tendrá su propio nombre de carpeta, que corresponde al nombre que le diste al proyecto al crearlo en NetBeans.

Para llevar lo mencionado a lo práctico, supongamos que creas un proyecto llamado ‘MiPrimerProyecto’. La estructura sería algo así:

**Figura 3**

*Estructura de proyecto*

*Nota.* Ejemplo de estructura básica en Netbeans. Lua. Org. (s.f.). *Lua the programming language* [Software]. <https://www.lua.org/>

* ‘MiClasePrincipal.java’ sería tu archivo de código que contiene el método ‘main’.
* ‘mi.paquete’ es un ejemplo de cómo podrías nombrar tu paquete fuente.
* ‘lib’ es donde se guardarían las bibliotecas externas, si las hay.
* ‘nbproject’ contiene archivos específicos de configuración de NetBeans.

Esta estructura te permite comenzar con un enfoque sencillo, ideal para aprender los fundamentos de Java antes de sumergirte en aspectos más complejos como la Programación Orientada a Objetos o el uso de herramientas de construcción como Maven.

Dicho esto, es tiempo de que comencemos con un proceso que se ejecuta en NetBeans en tiempo real llamado *debugger*. Este depurador de NetBeans es una herramienta esencial ya que te permite identificar y corregir errores en tus programas.

# Depuración o *Debugging* del código

El proceso de depuración o *debugging* es una etapa que se realiza para encontrar y corregir errores o bugs en el código fuente de un programa. Aunque los desarrolladores intentan escribir código sin errores, es inevitable que surjan problemas durante el desarrollo.

Profundizar en el funcionamiento y la importancia del *debugger* de NetBeans implica entender cómo ayuda a los desarrolladores a identificar y corregir errores en tiempo real.

El *debugger* o depurador de NetBeans permite:

* Ejecutar el código fuente paso a paso.
* Ejecutar métodos del JDK paso a paso.
* Utilizar *breakpoint* para detener la ejecución del programa y poder observar el estado de las variables.
* Conocer el valor que toma cada variable o expresión según se van ejecutando las líneas de código, y
* Modificar el valor de una variable sobre la marcha y continuar la ejecución.

Ahora revisaremos un paso a paso que nos ayudará a comprender el trabajo de depuración:

## Pasos para realizar el *debugger* en NetBeans

**Paso 1 – Abrir el código en NetBeans en modo depuración**

Si bien puedes iniciar la ejecución del Debug de varias formas, en este caso utilizaremos el menú Debug:

**Figura 4**

*Menú Debug*

*Nota.* Menú Debug y Step Into. The Apache Software Foundation. (2023). *Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

Comienza la depuración desde la primera línea del método ‘main’. El depurador se detiene esperando que decidamos el modo de depuración. Una vez iniciada la depuración y el depurador se detiene, la siguiente línea de código que se va a ejecutar aparece en verde, con una flecha verde a su izquierda:

**Figura 5**

*Depuración*

*Nota.* Ejemplo de depuración. The Apache Software Foundation. (2023). *Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

La depuración puede continuar utilizando distintas opciones:

**Tabla 1**

*Elementos de Menú Debug*

|  |  |
| --- | --- |
| Ícono | Función |
|  | **Step Over (F8):** ejecuta una línea de código. Si la instrucción es una llamada a un método, ejecuta el método sin entrar dentro del código del método. |
|  | **Step Into (F7):** ejecuta una línea de código. Si la instrucción es una llamada a un método, salta al método y continúa la ejecución por la primera línea del método. |
|  | **Step Out (Ctrl + F7):** ejecuta una línea de código. Si la línea de código actual se encuentra dentro de un método, se ejecutarán todas las instrucciones que queden del método y se vuelve a la instrucción desde la que se llamó al método. |
|  | **Run to Cursor (F4):** se ejecuta el programa hasta la instrucción donde se encuentra el cursor. |
|  | **Continue (F5):** la ejecución del programa continúa hasta el siguiente *breakpoint*. Si no existe un *breakpoint* se ejecuta hasta el final. |
|  | **Finish Debugger Session (Mayúsculas + F5):** termina la depuración del programa. |

**Paso 2 – Establecer *Breakpoints***

Un *breakpoint* o punto de interrupción es una marca que indica al depurador que debe detenerse cuando la ejecución del programa llegue a ella.

Cuando el programa se detiene en un *breakpoint,* podemos:

* Examinar los valores actuales de las variables,
* Detectar cuando se crea un objeto, y
* Continuar la depuración línea a línea del programa.

Para fijar un *breakpoint* se pulsa sobre el número de línea donde se desea colocar. La línea queda resaltada en color rojo con una marca del mismo color en el margen izquierdo.

**Figura 6**

*Fijación de un breakpoint*

*Nota.* Ejemplo de fijación de un *breakpoint*. The Apache Software Foundation. (2023). *Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

* También se puede crear seleccionando: Debug 🡪 New *Breakpoint*,
* Para eliminarlo, simplemente se pulsa sobre el cuadrado rojo.
* Para eliminarlos todos, debes presionar con el botón derecho en la ventana de *breakpoints* 🡪 Delete All.
* Para desactivarlo, con el botón derecho sobre la marca roja 🡪 *breakpoint* 🡪 desmarcar Enable.

***Figura 7***

*Desactivación de un breakpoint*

*Nota.* Ejemplo de desactivación de un *breakpoint.* The Apache Software Foundation. (2023). *Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

También puedes realizar ***Breakpoints* Condicionales,** donde puedes hacer que la depuración se detenga en un *breakpoint* sólo si se cumple una determinada condición o si la línea de código se ha ejecutado un número determinado de veces.

Para fijar un *breakpoint* condicional debes presionar con el botón derecho sobre el *breakpoint* 🡪 *Breakpoint* 🡪 Properties. La marca cambia de aspecto:

**Figura 8**

*Breakpoint condicional*

*Nota*. Ejemplo de cómo fijar un *breakpoint* condicional. The Apache Software Foundation. (2023). *Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

## *Debugging Windows / Ventanas de depuración*

En el proceso de depuración se usan distintas herramientas visuales y ventanas situadas bajo la ventana de código; algunas aparecen automáticamente. Las ventanas de depuración son paneles específicos que proporcionan información útil durante una sesión de depuración. Para mostrarlas debes presionar Windows à Debugging y seleccionar la que quieres:

**Figura 9**

*Debugging Windows*

*Nota*. Ejemplo de cómo depurar ventanas. The Apache Software Foundation. (2023). *Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

Estas ventanas son fundamentales para el proceso de depuración y las más recurrentes son:

1. **Ventanas de puntos de interrupción *(breakpoints):***lista todos los puntos de interrupción establecidos, permitiendo al desarrollador activarlos, desactivarlos o eliminarlos.
2. **Ventana de variables:** muestra las variables locales y de instancia, permitiendo al desarrollador inspeccionar y modificar sus valores en tiempo de ejecución.

**Figura 10**

*Variables*

*Nota*. Variables, su tipo y su valor actual. The Apache Software Foundation. (2023). *Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

Importante

El *debugger* permite cambiar el valor de una variable local en esta ventana y continuar la ejecución del programa usando el nuevo valor de la variable.

1. **Ventana *Watches:***permite monitorear el valor de ciertas variables o expresiones independientemente de si están en el ámbito actual o no.

Durante la depuración de un programa, para obtener el valor de una variable, basta con situar el cursor sobre la misma. El depurador mostrará entonces tanto el tipo como el valor actual de la variable:

**Figura 11**

*Variables con watch asignado*

*Nota.* Ejemplo de variables con watch asignado. The Apache Software Foundation. (2023). *Apache NetBeans (19).* [Software]. Apache NetBeans. <https://netbeans.apache.org/front/main/>

Para agregar un *watch* a una variable o una expresión, selecciona la variable o expresión a supervisar y a continuación pulsar. Debug 🡪 New Watch.

1. **Call Stack:** Muestra la pila actual de llamadas a métodos, lo que permite al desarrollador ver el camino que ha tomado el código para llegar al punto actual y navegar a través de los diferentes niveles de la pila.

# Video

Para finalizar, a través del siguiente video, puedes aprender más sobre el proceso de *debug* en Apache NetBeans: <https://www.youtube.com/watch?v=njq04Wtx5nQ>

# Cierre de la semana

Apache NetBeans es un entorno de desarrollo integrado (IDE) de código abierto que ofrece soporte para múltiples lenguajes de programación, como Java, PHP, JavaScript y más, proporcionando herramientas esenciales como resaltado de sintaxis, autocompletado y refactoring de código. Distinguido por su interfaz intuitiva y una amplia gama de funcionalidades, NetBeans facilita el desarrollo de aplicaciones robustas con herramientas integradas para el control de versiones y la construcción de software.

En el proceso de depuración, NetBeans sobresale con sus capacidades de "debugging", permitiendo a los desarrolladores realizar un seguimiento eficiente del flujo de ejecución y el estado del programa mediante puntos de interrupción, inspección de variables, y ventanas de "Watches", las cuales son cruciales para un diagnóstico preciso y una corrección efectiva de errores.

# Referencias

* Apache.org. (s.f.). Apache Project List. The Apache Software Foundation <https://www.apache.org/index.html#projects-list>
* García, E. (s.f.). *Programación Java*. Puntocomnoesunlenguaje. <https://puntocomnoesunlenguaje.blogspot.com/2014/11/netbeans-debugger-java.html>
* The Apache Software Foundation. (s.f.). *Apache NetBeans (19).* [Software] The Apache Software Foundation [https://netbeans.apache.org/front/main/](https://netbeans.apache.org/front/main/ )

# Apuntes

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



Reservados todos los derechos Fundación Instituto Profesional Duoc UC. No se permite copiar, reproducir, reeditar, descargar, publicar, emitir, difundir, de forma total o parcial la presente obra, ni su incorporación a un sistema informático, ni su transmisión en cualquier forma o por cualquier medio (electrónico, mecánico, fotocopia, grabación u otros) sin autorización previa y por escrito de Fundación Instituto Profesional Duoc UC La infracción de dichos derechos puede constituir un delito contra la propiedad intelectual.