

UD2: Instalación y uso de Entornos de Desarrollo

Entornos de desarrollo
M^a Carmen Safont Richarte

INDICE

1. Entornos de Desarrollo

- 1.1. ¿Qué es un IDE?
- 1.2. Evolución histórica
- 1.3. Componentes o herramientas de un IDE
- 1.4. Tipos de IDE's

2. Instalación de un Entorno de Desarrollo

- 2.1. Proceso de instalación
- 2.2. Interfaz de Apache Netbeans
- 2.3. Personalización del entorno.

3. Crear un proyecto Java con Apache Netbeans

- 3.1. Creación de proyectos
- 3.2. Estructura de un proyecto Java
- 3.3. Importar un proyecto Java
- 3.4. Otras funcionalidades



1. ENTORNOS DE DESARROLLO

CONTENIDOS

- 1.1. ¿Qué es un IDE?
- 1.2. Componentes o herramientas de un IDE
- 1.3. Evolución histórica
- 1.4. Tipos de IDE's
 - 1.4.1. Entornos de Desarrollo online (en la nube)
 - 1.4.2. Entornos de Desarrollo libres
 - 1.4.3. Entornos de Desarrollo propietarios

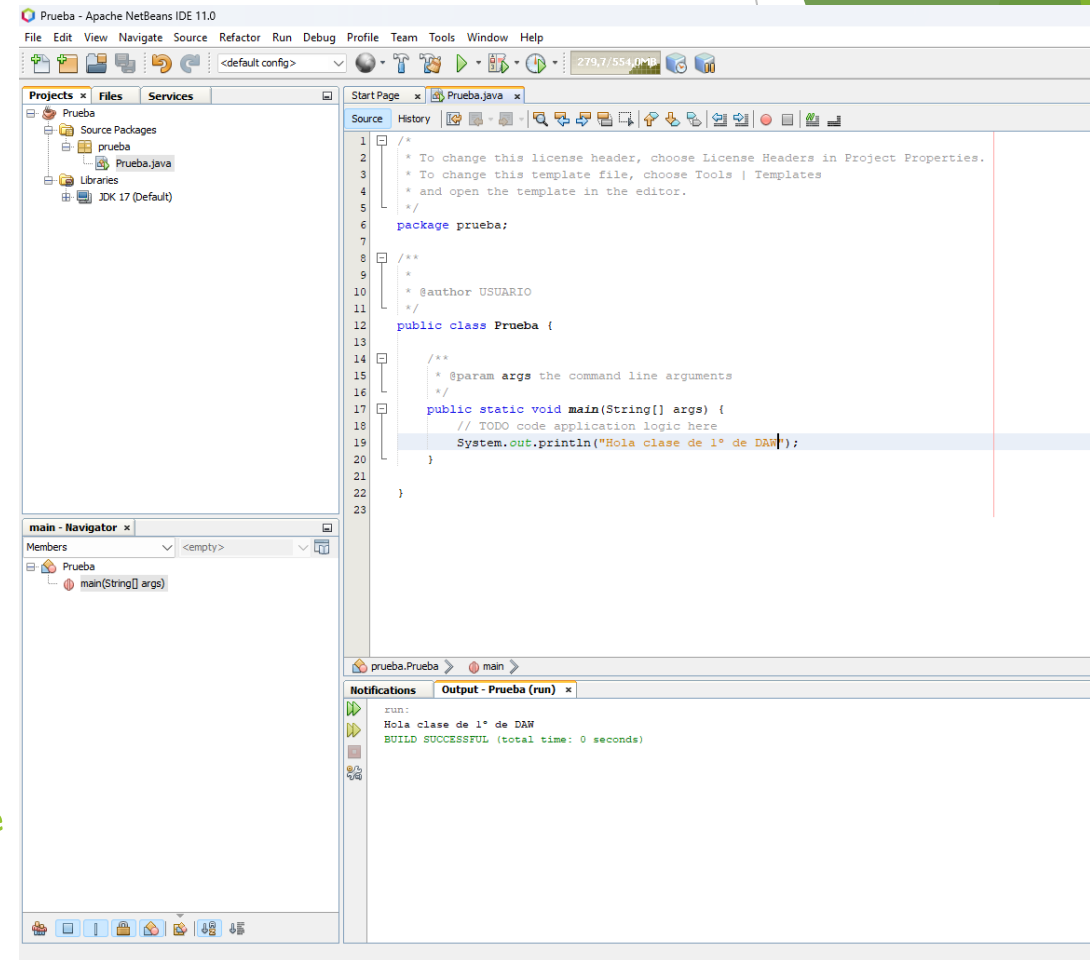
1. ENTORNOS DE DESARROLLO

1.1. ¿Qué es IDE?

DEFINICIÓN

Un entorno de desarrollo integrado es un conjunto de herramientas que facilitan la escritura, generación y depuración de código.

NOTA: La cantidad de herramientas incluidas es configurable dependiendo de las características de las aplicaciones que se quieran crear.



1. ENTORNOS DE DESARROLLO

1.2. Componentes o herramientas de un IDE

| | |
|----------------------------|---|
| Editor | <ul style="list-style-type: none">• Facilita la escritura organizando las instrucciones atendiendo a un formato, colores.• Permite la escritura automática, autocompletar, además de las clásicas funciones de buscar, reemplazar, cortar, pegar, etc. |
| Compilador / Intérprete | <ul style="list-style-type: none">• Herramientas para generar código ejecutable |
| Depurador | <ul style="list-style-type: none">• Permite la ejecución de una aplicación instrucción a instrucción, examinar el valor de las variables, etc. Para localizar errores. |
| Asistente GUI | <ul style="list-style-type: none">• Permite crear ventanas, botones, campos de texto, ... |
| Extensiones | <ul style="list-style-type: none">• Permite la instalación de extensiones (plugins) que añaden nuevas funcionalidades (control de versiones, generación de documentación, realización de conjunto de pruebas o testeo,) |

1. ENTORNOS DE DESARROLLO

1.3. Evolución histórica

Antes del IDE, los programadores escribían sus programas en **editores de texto**. Esto implicaba escribir y guardar una aplicación en el editor de texto antes de ejecutar el compilador, escribir cualquier mensaje de error y luego volver al editor de texto para revisar su código, y eso lleva mucho tiempo.

En 1983, la empresa danesa [Borland](#) lanzó un editor de código y compilador para el lenguaje de programación [Pascal](#) llamado [Turbo Pascal](#).

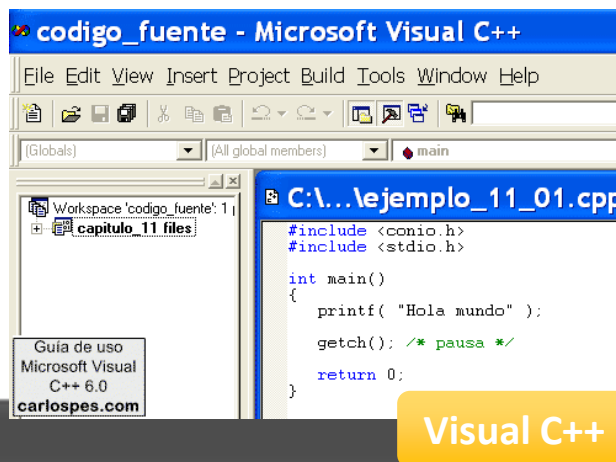
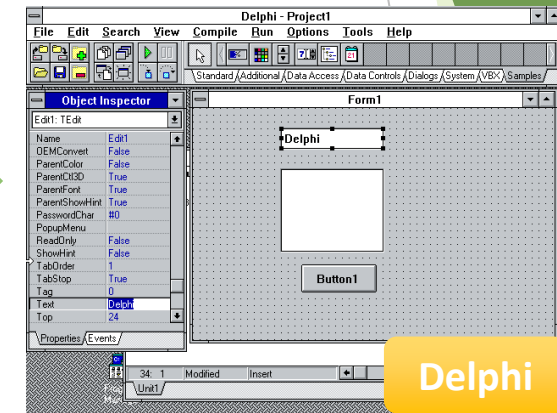
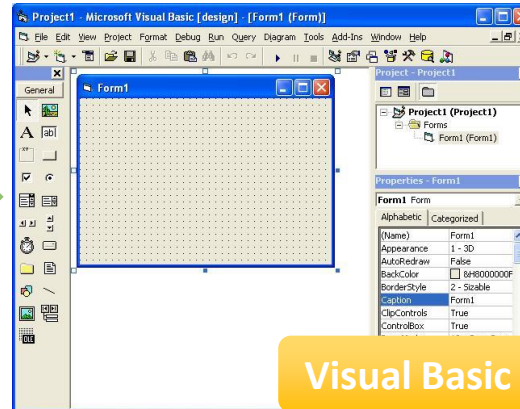
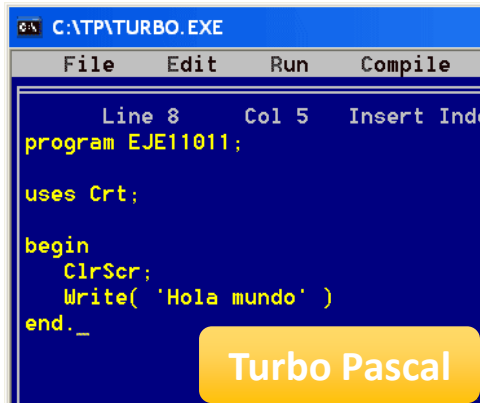
El desarrollo del compilador Turbo Pascal, por parte de Borland, fue de gran importancia en el éxito del lenguaje Pascal, ya que no se trataba de un simple compilador, sino de un *entorno de desarrollo donde se podía construir y depurar el código*, lo que supuso una novedad para el tiempo.

Turbo Pascal lanzó la idea de un entorno de desarrollo integrado, pero muchos creen que [Visual Basic \(VB\)](#) de Microsoft, lanzado en 1991, fue en realidad el primer IDE real de la historia.

Dato curioso: Según datos, casi todas las personas que empiezan a programar utilizan un editor de textos y un compilador-depurador instalado en su equipo. Sin embargo, prácticamente todas acaban utilizando un entorno de desarrollo.

1. ENTORNOS DE DESARROLLO

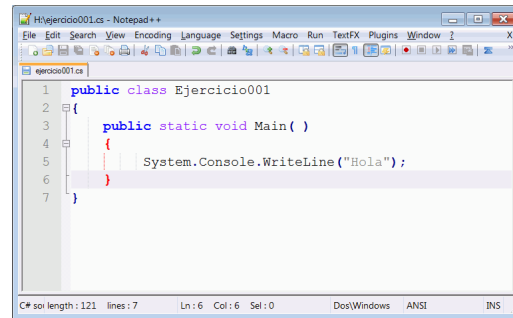
1.3. Evolución histórica



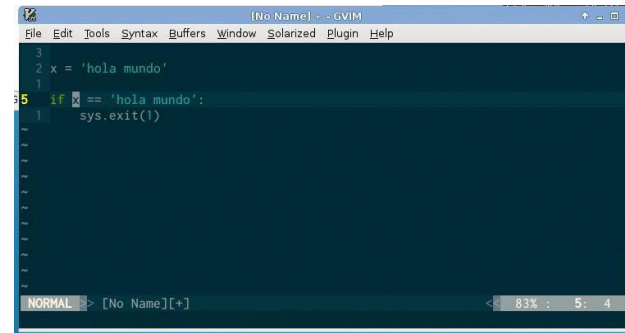
1. ENTORNOS DE DESARROLLO

1.3. Evolución histórica

- Los editores de código pueden ser editores de texto simples, como [Notepad++](#) o [VIM](#), por ejemplo, pero no mejoran el proceso de edición de código.



```
1 public class Ejercicio001
2 {
3     public static void Main( )
4     {
5         System.Console.WriteLine("Hola");
6     }
7 }
```



```
3
2 x = 'hola mundo'
5 if x == 'hola mundo':
1     sys.exit(1)
```

- Sin embargo, existen algunos editores de código con varias funcionalidades integradas, plugins y terminales adjuntos que facilitan mucho el desarrollo. Como ejemplo podemos mencionar el famoso [Visual Studio Code](#).

1. ENTORNOS DE DESARROLLO

1.3. Evolución histórica

¿Cuál es la diferencia entre un editor de código y un IDE?

- ▶ Los editores de código son editores de texto con potentes funciones integradas y funciones especializadas para simplificar y acelerar el proceso de edición de código.
- ▶ Un IDE, por otro lado, es un conjunto de herramientas de desarrollo de software diseñadas para facilitar la codificación. En otras palabras, un IDE tiene un editor de código, un depurador, un compilador y otras funciones importantes, todo en una sola herramienta.

1. ENTORNOS DE DESARROLLO

StackOverflow



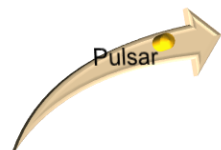
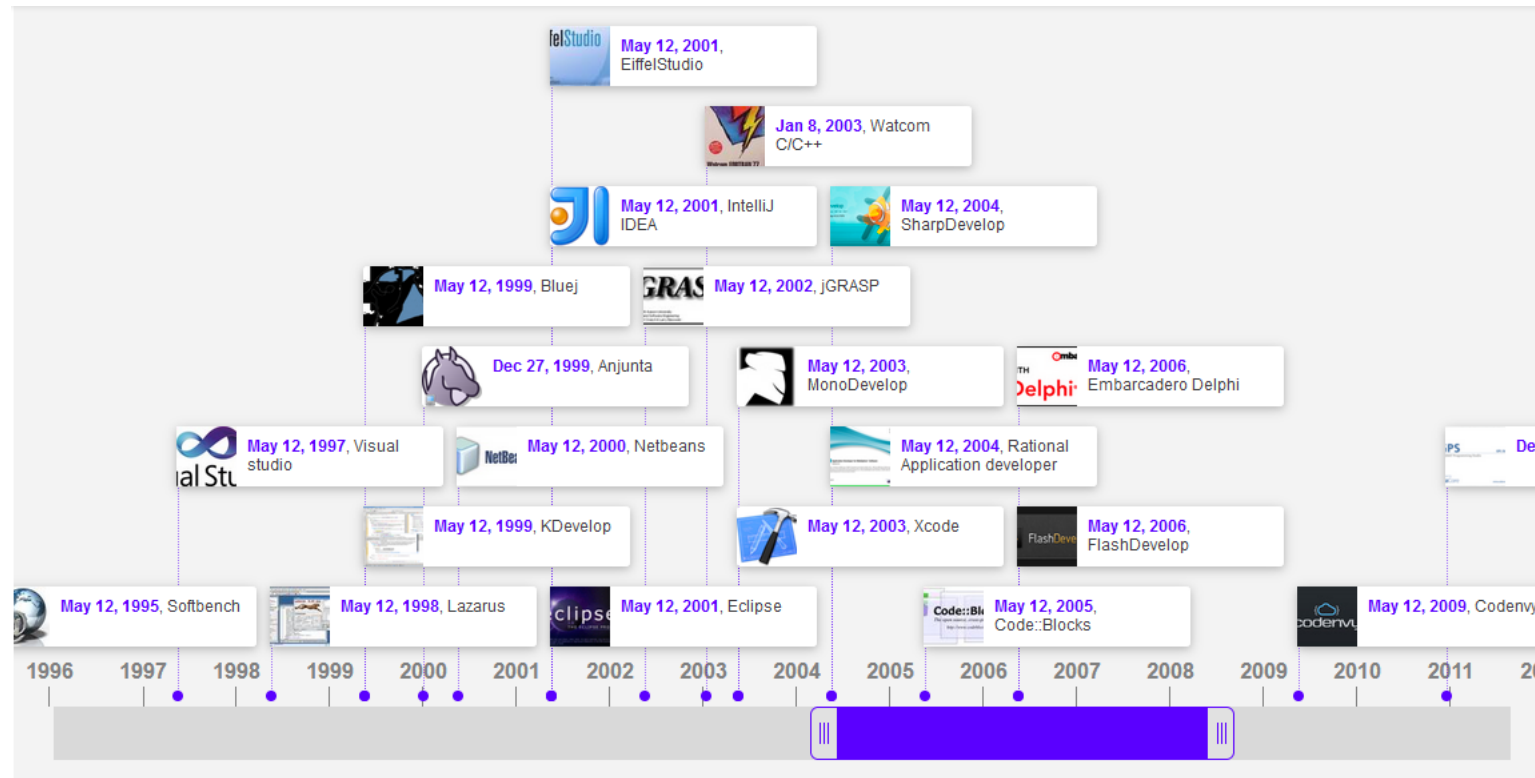
1.3. Evolución histórica

A día de hoy, hay diferentes tipos de IDE, algunos de grandes organizaciones, como [Visual Studio](#) de Microsoft, [Xcode](#) de Apple y [Android Studio](#) de Google. Los IDE populares son [Visual Studio](#), [Eclipse](#), [IntelliJ IDEA](#), [NetBeans](#), [PyCharm](#) entre otros.

La mayoría de ellos pueden ser utilizados para programar en diferentes lenguajes de programación, basta con añadir el plugin correspondiente.

1. ENTORNOS DE DESARROLLO

1.3. Evolución histórica



TimeLine evolución IDE's

1. ENTORNOS DE DESARROLLO

1.4. Tipos de IDE's

1.4.1. Entornos de Desarrollo online

- ▶ Muchas empresas están apostando por usar este tipo de entornos.
- ▶ Poseen casi las mismas funcionalidades de un IDE convencional.
- ▶ La mayoría de estos entornos tienen un estilo parecido a Visual Studio Code
- ▶ Casi todos tienen planes de precio gratuitos.
- ▶ Ejemplos: [AWS Cloud9](#), [Codeanywhere](#).



VENTAJAS

- Trabajo colaborativo
- Repositorios comunes
- Trabajar desde cualquier dispositivo.

INCONVENIENTES

- Menor potencia

1. ENTORNOS DE DESARROLLO

1.4. Tipos de IDE's

1.4.2. Entornos de Desarrollo libres

Son aquellos con **licencia de uso público**.

No hay que pagar por ellos, y aunque los más conocidos y utilizados son **Eclipse** y **NetBeans**, hay bastantes más.

| IDE | Lenguajes que soporta | SO |
|------------|---|---------------------|
| NetBeans | C/C++, Java, JavaScript, PHP, Python, HTML5, CSS, ... | Windows, Linux, Mac |
| Eclipse | Java, C++, JavaScript, Python o PHP entre otros. | Windows, Linux, Mac |
| Gambas | Gambas (basado en Basic) | Linux |
| Geany | C/C++, Java | Windows, Linux, Mac |
| JDeveloper | Java | Windows, Linux, Mac |

1. ENTORNOS DE DESARROLLO

1.4. Tipos de IDE's

1.4.2. Entornos de Desarrollo libres

Son aquellos entornos integrados de desarrollo que **necesitan licencia**. No son free software, hay que pagar por ellos. El más conocido y utilizado es Microsoft Visual Studio, que usa el framework .NET y es desarrollado por Microsoft.

| IDE | Lenguajes que soporta | SO |
|------------------|---|--|
| MS Visual Studio | C/C++, Java, JavaScript, PHP, Python, ... | Windows, Mac y Linux para la versión Code. |
| Xcode | C/C++, Java | Mac OS X |
| IntelliJ IDEA | Java, JavaScript, PHP, Python, HTML5, ... | Windows, Mac y Linux |
| JBuilder | Java | Windows, Mac y Linux |
| Jcreator | Java | Windows |

1. ENTORNOS DE DESARROLLO

¿Hay algún editor o IDE que sea el mejor?

- Para los que programan en Java, IntelliJ, NetBeans o Eclipse son excelentes recomendaciones.
- Para los que desarrollan con Javascript, Visual Studio Code y Sublime.
- La elección de editor o IDE es personal y depende mucho del lenguaje o área de especialización. Además, hay personas que se adaptan mejor a un entorno que a otro.

Conclusión

El IDE y/o el editor de código ayudan a las personas a desarrollar. Por lo tanto, conocer la herramienta y sus atajos puede acelerar el desarrollo.

2. INSTALACIÓN DE UN IDE

CONTENIDOS

- 2.1. Proceso de instalación
- 2.2. Interfaz gráfica de Apache Netbeans 19
- 2.3. Personalización del entorno

2. INSTALACIÓN DE UN IDE

2.1. Proceso de instalación

PASOS PREVIOS

La instalación del IDE Apache NetBeans, ya sea en Linux, Windows o Mac, requiere la instalación previa del JDK (Java Developer KIT) compatible con la versión de NetBeans que se quiera instalar.

En nuestro caso vamos a instalar Apache Netbeans 19, y el JDK 17 (el JDK 21 no es compatible con esta versión)

¿Cómo saber la versión de JVM instalada en nuestro equipo?

Bastará con ejecutar el siguiente comando desde la consola:

• **java --version**

2. INSTALACIÓN DE UN IDE

2.1. Proceso de instalación

Instalación del JDK 17 en Linux

Podéis consultar el manual de instalación en el apartado de Recursos de Aules

Instalación de Apache Netbeans 19 en Linux

Podéis consultar el manual de instalación en el apartado de Recursos de Aules.

Instalación del JDK 17 en Windows.

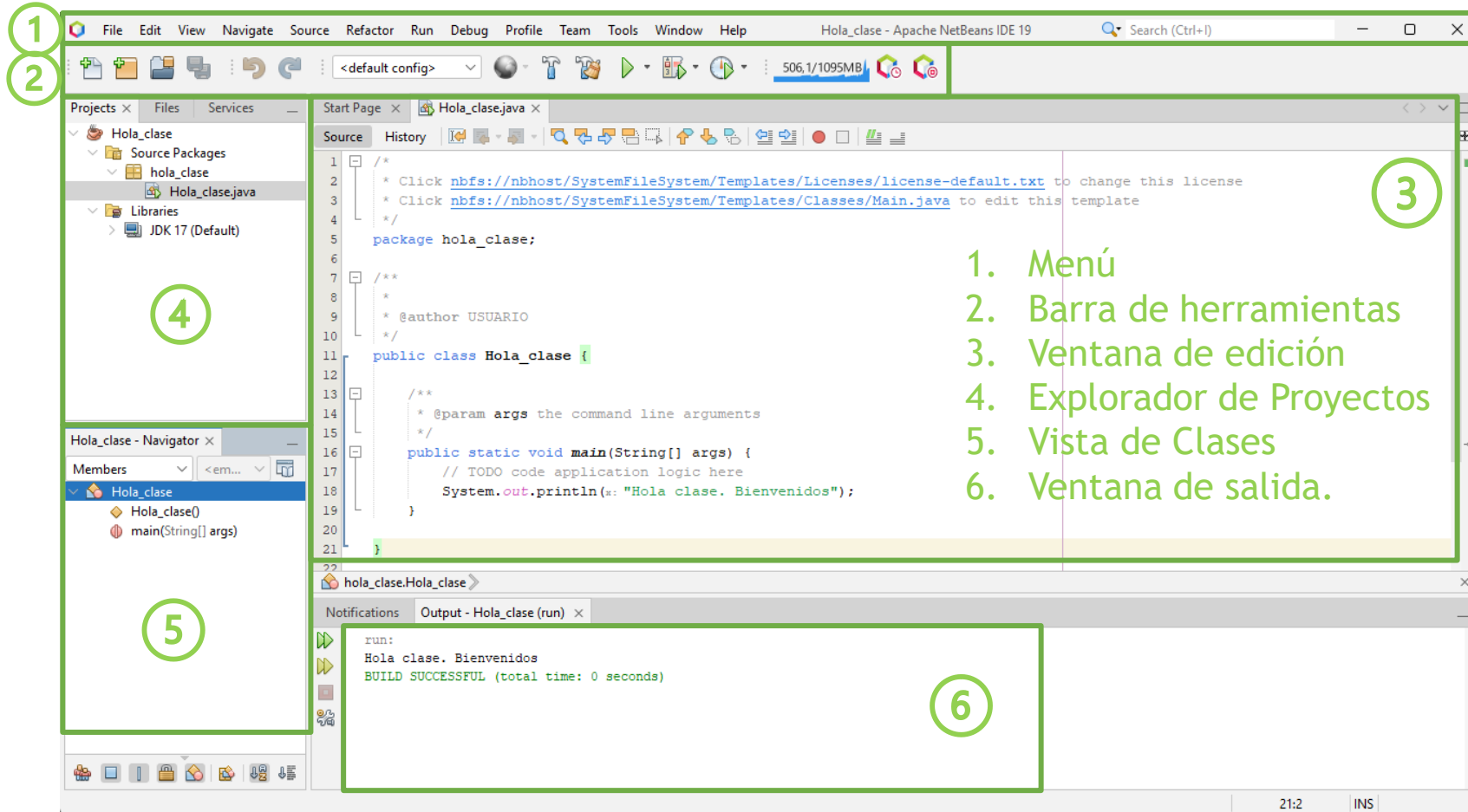
Podéis consultar el manual de instalación en el apartado de Recursos de Aules.

Instalación de Apache Netbeans 19 en Windows

Podéis consultar el manual de instalación en el apartado de Recursos de Aules.

2. INSTALACIÓN DE UN IDE

2.2. Interfaz gráfica de Apache NetBeans



NetBeans ofrece una interfaz de usuario intuitiva y repleta de características para ayudarte a desarrollar aplicaciones de manera eficiente.

2. INSTALACIÓN DE UN IDE

2.1. Interfaz gráfica de Apache Netbeans

1. Barra de menú

- Opciones principales de NetBeans, como "Archivo," "Editar," "Ver," "Proyectos," "Herramientas," y "Ayuda."

2. Barra de herramientas

- Accesos rápidos a funciones comunes como guardar, compilar y depurar.

3. Ventana de edición

- Donde editas tu código fuente y archivos. Puedes tener múltiples ventanas de edición abiertas al mismo tiempo.

4. Explorador de proyectos

- Muestra una jerarquía de tus proyectos y sus archivos asociados.

5. Ventana de salida

- Muestra mensajes del sistema, información de depuración y resultados de compilación.

6. Ventana de clases

- Muestra las clases de nuestro proyecto.

7. Ventana de servicios

- La ventana de servicios proporciona acceso a bases de datos, servidores y otros recursos. Puedes configurar conexiones de bases de datos y servidores desde aquí. (es una pestaña del Explorador de proyectos)

2. INSTALACIÓN DE UN IDE

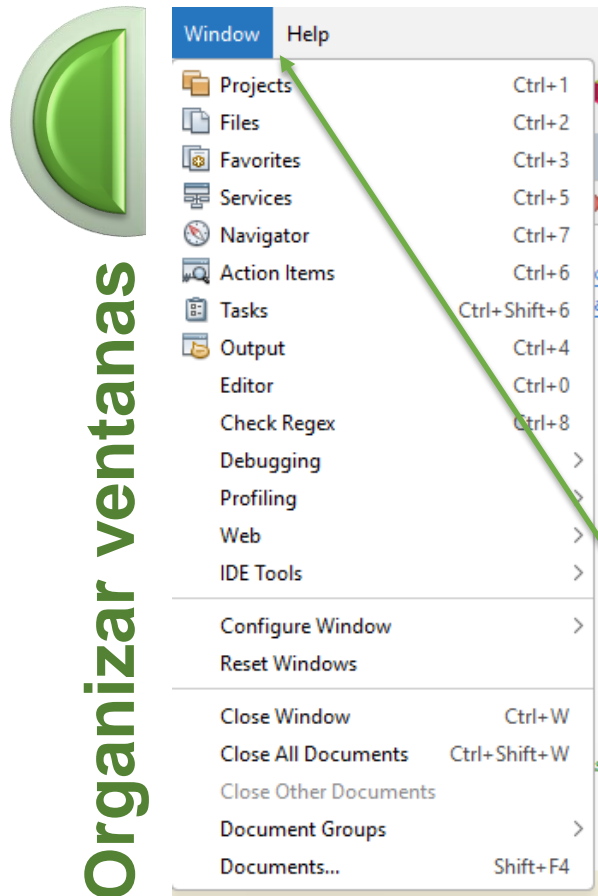
2.2. Personalización de Apache Netbeans

Podemos personalizar completamente nuestro entorno de trabajo añadiendo elementos de ayuda en nuestro IDE.



2. INSTALACIÓN DE UN IDE

2.2. Personalización de Apache Netbeans



Puedes arrastrar y soltar ventanas para ajustar su posición en la interfaz de NetBeans, según tus preferencias.

Ejemplo: Supongamos que prefieres tener la ventana de propiedades a la izquierda en lugar de a la derecha. Para hacerlo, simplemente arrastra la ventana de propiedades desde su ubicación actual y suéltala en la parte izquierda de la interfaz. NetBeans te permite organizar y anclar las ventanas en las ubicaciones que desees.

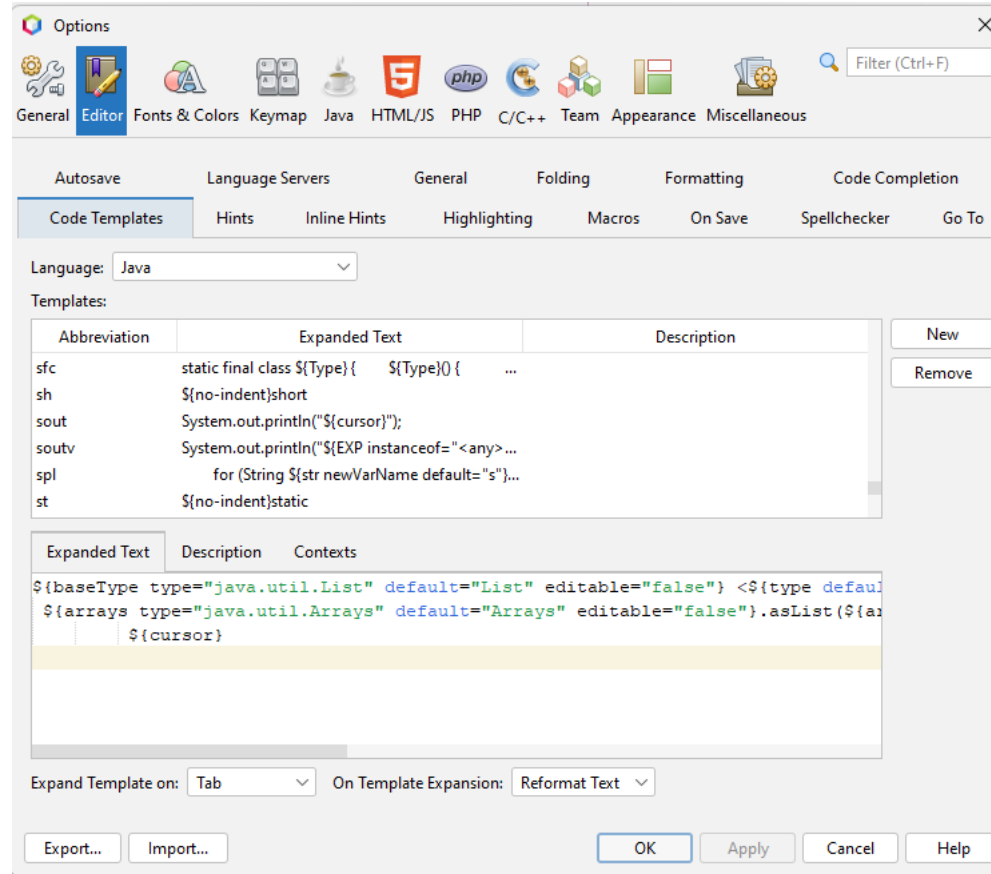
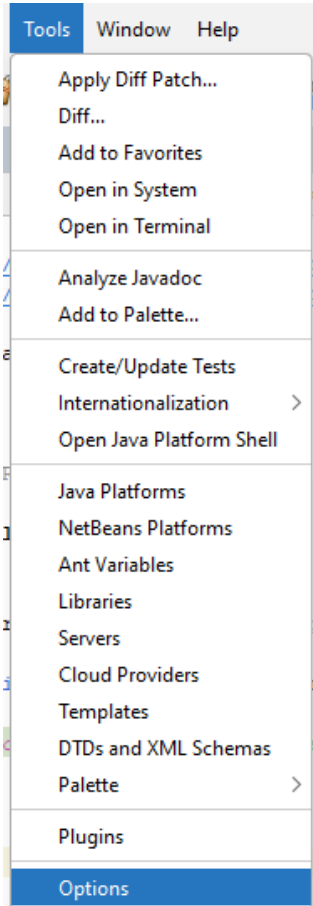
Para añadir ventanas podemos hacerlo desde la barra de menú **Window**.

2. INSTALACIÓN DE UN IDE

2.2. Personalización de Apache Netbeans



Atajos de teclado



NetBeans permite personalizar atajos de teclado para acciones comunes, para aumentar la productividad.

Ejemplo: Crear un atajo de teclado personalizado para compilar proyectos, asignando la siguiente combinación de teclas:

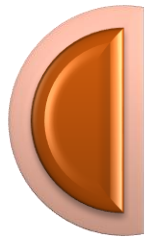
Ctrl + Shift + C.

Para realizar esta personalización, accede a:

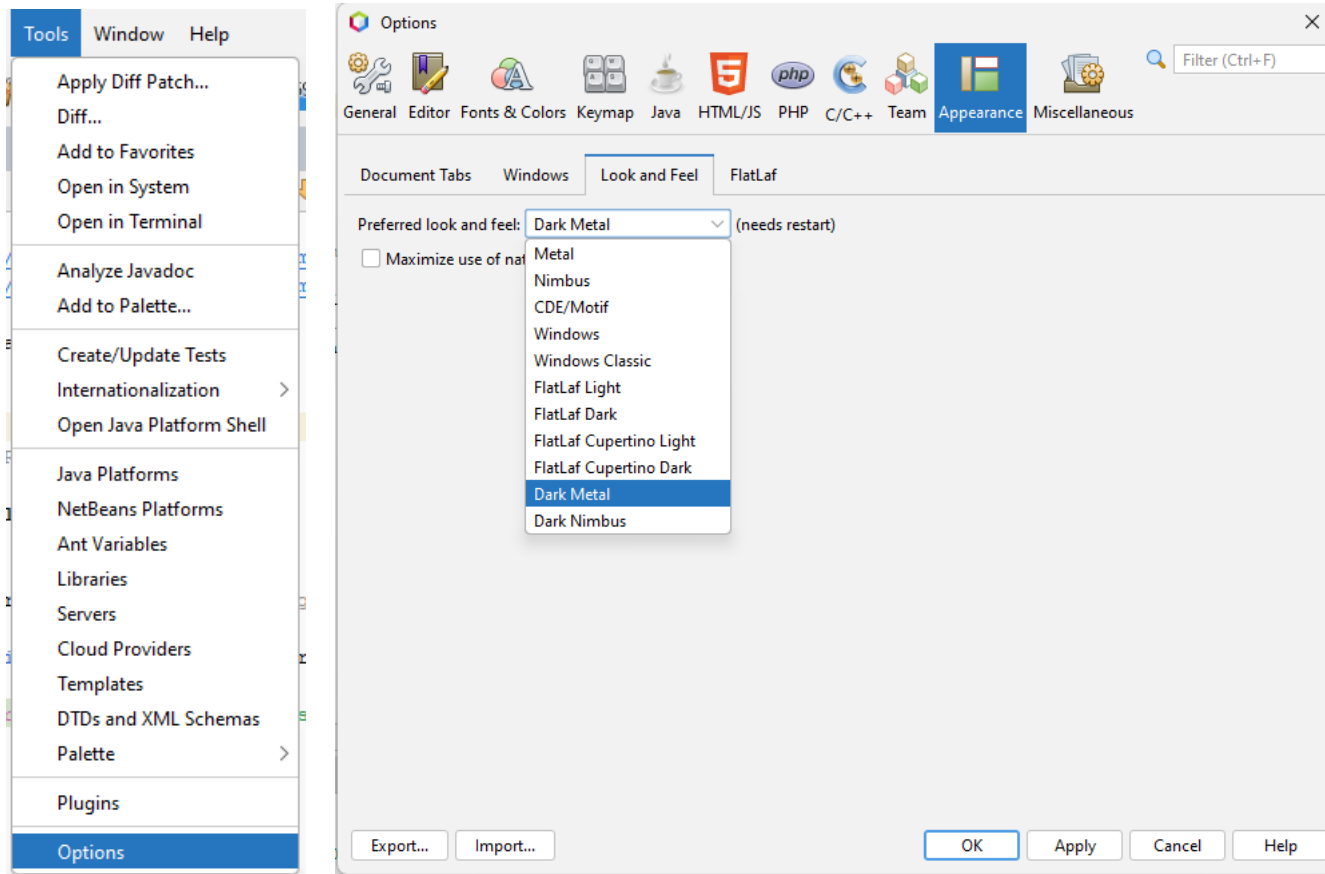
Tools/Options/Editor/Code Templates

2. INSTALACIÓN DE UN IDE

2.2. Personalización de Apache Netbeans



Temas y esquemas de colores



Puedes cambiar el tema y el esquema de colores de NetBeans para que se adapten a tus preferencias visuales.

Ejemplo: Si prefieres un esquema de colores oscuro en lugar de uno claro, NetBeans te permite cambiar el tema. Puedes personalizar el tema (y crearte tu propia plantilla) y eligiendo entre las opciones disponibles desde **Tools/Options/Apparrance**.

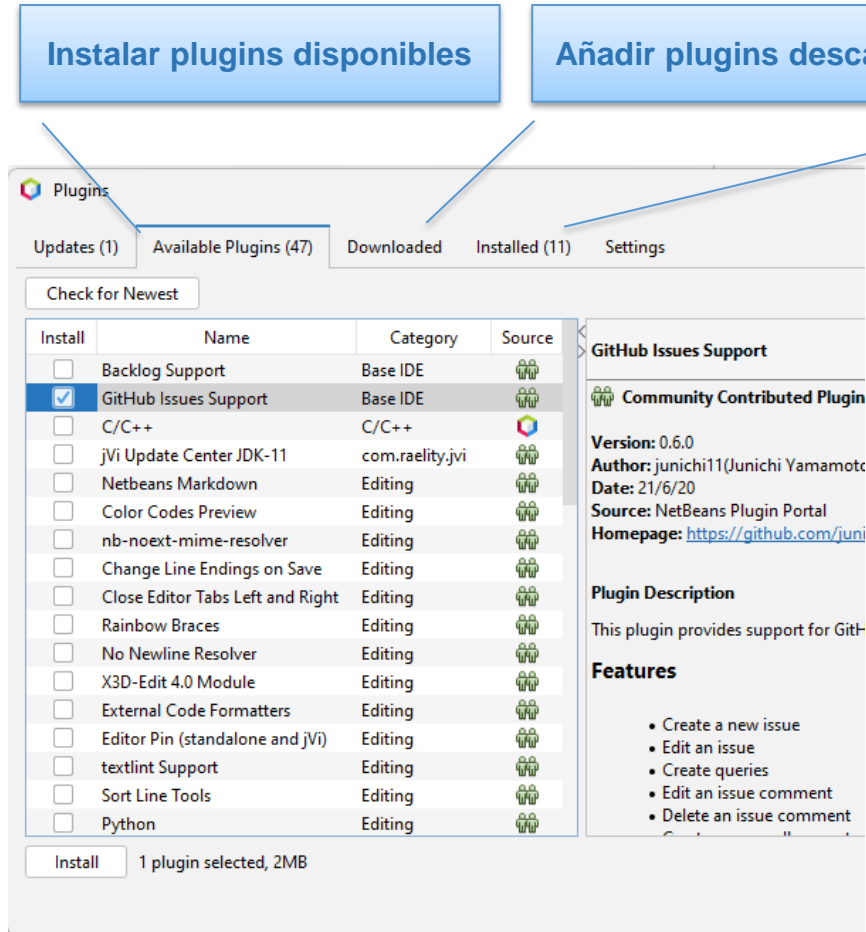
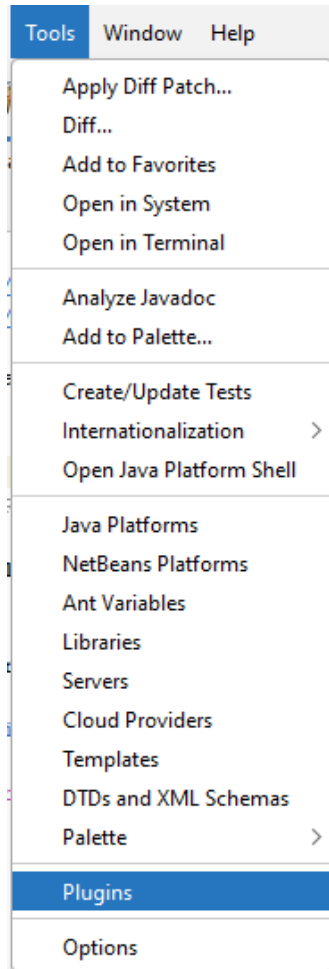
O añadir una plantilla ya formateada como puede ser **Dracula** (lo veremos en el siguiente apartado).

2. INSTALACIÓN DE UN IDE

2.2. Personalización de Apache Netbeans



Plugins y extensiones



Los plugins y extensiones proporcionan características adicionales y mejoras en la interfaz de usuario.

Están disponibles a través del Centro de Actualizaciones.

Ejemplo: añadir funcionalidades para integrar control de versiones como Git.

Para añadir ventanas podemos hacerlo desde la barra de menú **Tools**.

2. INSTALACIÓN DE UN IDE

2.2. Personalización de Apache Netbeans

La personalización de la interfaz no solo te permite mejorar la apariencia, sino también optimizar tu flujo de trabajo y aumentar tu productividad.

Al adaptar NetBeans a tus preferencias, puedes trabajar de manera más eficiente y centrarte en la programación sin distracciones innecesarias.

Experimenta con estas opciones de personalización y ajusta NetBeans para que se adapte a tus necesidades específicas.

3. PROYECTOS JAVA CON NETBEANS

CONTENIDOS

- 3.1. Creación de proyectos
- 3.2. Estructura de un proyecto Java
- 3.3. Importar un proyecto Java
- 3.4. Otras funcionalidades

3. PROYECTOS JAVA CON NETBEANS

3.1. Creación de proyectos

Hay muchas actividades involucradas en el desarrollo de software, como compilar el código fuente, empaquetar el código binario, ejecutar pruebas automatizadas e implementar en producción. También es importante crear documentación y notas de lanzamiento.

Para hacer estas actividades simples y fáciles, los desarrolladores usan diferentes herramientas de software.

Ant, **Maven** y **Gradle** son herramientas de construcción y gestión de proyectos utilizadas en el desarrollo de software, especialmente en el contexto de proyectos Java.

3. PROYECTOS JAVA CON NETBEANS

3.1. Creación de proyectos

Apache Ant

Descripción

Herramienta de **construcción** basada en XML desarrollada por Apache Software Foundation.

Características:

Se basa en archivos XML (build.xml) para definir tareas y procesos de construcción.

Proporciona flexibilidad para personalizar la construcción del proyecto y ejecutar tareas específicas, como compilar, empaquetar y probar.

Uso común:

Útil en proyectos donde se necesita un alto grado de personalización y control sobre las tareas de construcción.

Es especialmente valioso cuando se trabaja en proyectos no basados en Java o cuando se requieren tareas de construcción personalizadas.



Apache Maven



Descripción:

Herramienta de **gestión de proyectos** y **construcción** desarrollada por la Apache Software Foundation.

Características:

Se basa en un archivo XML llamado "pom.xml" (Project Object Model) para definir la estructura del proyecto, las dependencias y las tareas de construcción.

Proporciona una estructura de proyecto predefinida y una gestión automatizada de dependencias.

Uso común:

Ampliamente utilizado en proyectos Java donde se busca una estructura de proyecto estandarizada y una gestión de dependencias sencilla.

Es especialmente útil para proyectos basados en Java EE y aplicaciones web.

Gradle



Descripción:

Herramienta de **construcción y gestión de proyectos** basada en Groovy o Kotlin.

Proporciona una forma más flexible de definir tareas de construcción.

Características:

Utiliza un DSL (Domain Specific Language) basado en Groovy o Kotlin para definir scripts de construcción en archivos "build.gradle".

Combina la flexibilidad de Ant con la automatización y la gestión de dependencias de Maven.

También se centra en la eficiencia y el rendimiento.

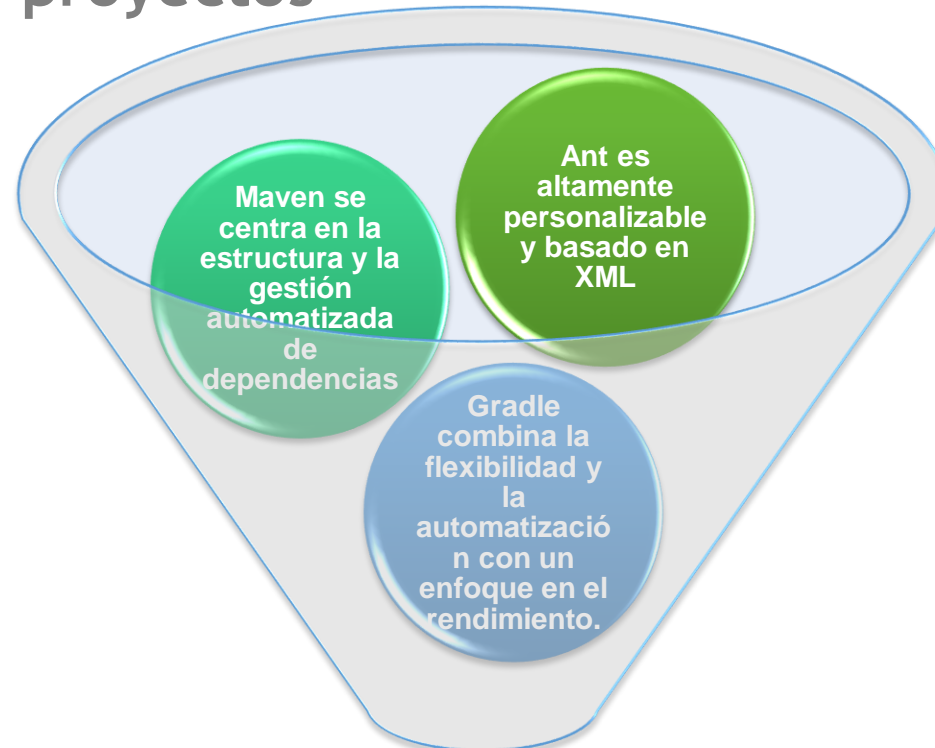
Uso común:

Ampliamente utilizado en proyectos Java y no Java para ofrecer un alto grado de personalización en la construcción de proyectos.

Es especialmente eficiente en proyectos multiproyecto y proyectos grandes.

3. PROYECTOS JAVA CON NETBEANS

3.1. Creación de proyectos



La elección de la herramienta depende de las necesidades específicas del proyecto y las preferencias del equipo de desarrollo. Cada una de estas herramientas es valiosa en su contexto adecuado.



Para empezar, nos bastará con crear proyectos Ant

3. PROYECTOS JAVA CON NETBEANS

3.1. Creación de proyectos

Un proyecto, es un "**contenedor**" global donde podemos incluir o crear todos los archivos que de una u otra forma vayan a ser utilizados en nuestro programa, clases, interfaces, archivos de texto, imágenes, paquetes, etc.

Nos permite dar un orden y una clasificación a nuestro trabajo y a nuestros programas, evitando así que estemos desarrollando algo con los archivos de diferentes programas mezclados en un mismo lugar.

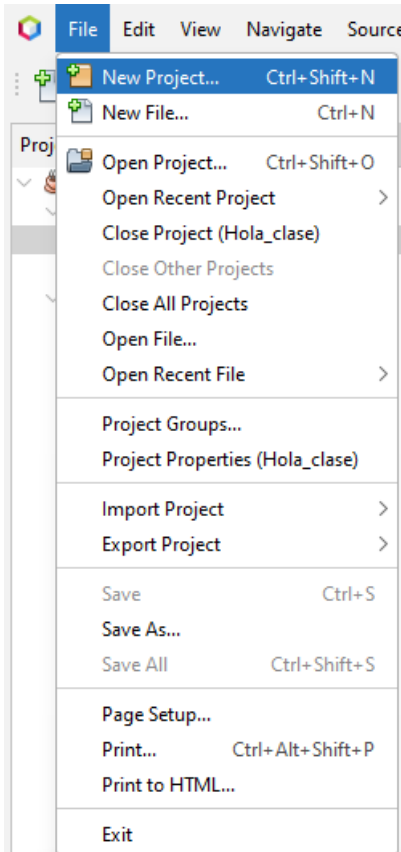
Los proyectos nos permiten mantener ordenado nuestro espacio de trabajo permitiéndonos ser más ágiles y evitar errores.

3. PROYECTOS JAVA CON NETBEANS

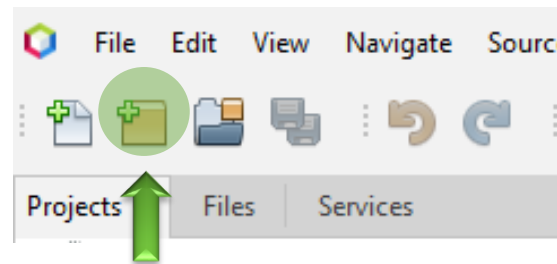
Elegir tipo proyecto

3.1. Creación de proyectos

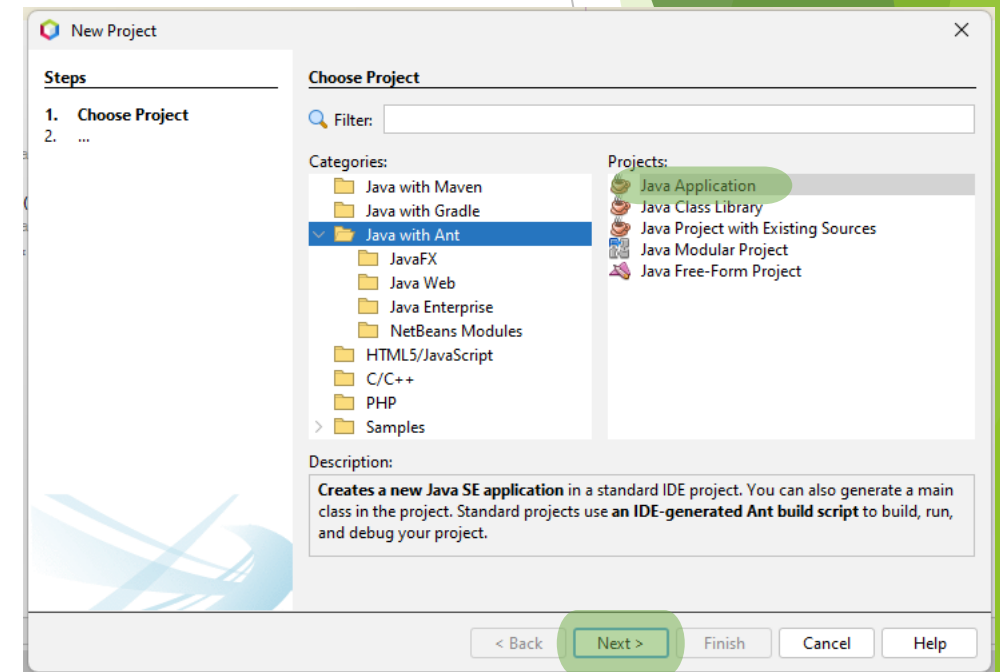
BARRA DE MENÚ



BARRA DE HERRAMIENTAS



ATAJO DE TECLADO



3. PROYECTOS JAVA CON NETBEANS

3.1. Creación de proyectos

Nombrar
proyecto

New Java Application

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name:

Project Location:

Project Folder:

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

☒ Create Main Class

< Back Next > **Finish** Cancel Help

Nombre del proyecto

Ubicación del
proyecto

Marcar esta opción para que nos cree
un archivo main donde
desarrollaremos el código de
nuestro proyecto.

3. PROYECTOS JAVA CON NETBEANS

3.1. Creación de proyectos

The screenshot displays the NetBeans IDE interface. On the left, the 'Projects' pane shows the project structure: 'MiPrimeraAplicacionJava' (labeled 'Paquetes') contains 'Source Packages' (labeled 'Clases') with 'miprimeraaplicacionjava' and 'Clase2.java'. Below this, the 'Libraries' section shows 'JDK 17 (Default)'. The 'MiPrimeraAplicacionJava - Navigator' pane lists members: 'MiPrimeraAplicacionJava()' and 'main(String[] args)'. The main editor shows the source code of 'MiPrimeraAplicacionJava.java' with a blue selection box around the class definition and its 'main' method. The 'Output - MiPrimeraAplicacionJava (run)' pane at the bottom shows the execution results: 'run: Hola clase' and 'BUILD SUCCESSFUL (total time: 0 seconds)' (labeled 'Resultados'). A pink box labeled 'Notificaciones' points to the 'Notifications' tab in the output pane.

Paquetes

Clases

Código de la clase

Notificaciones : errores o problemas detectados durante la ejecución

Resultados

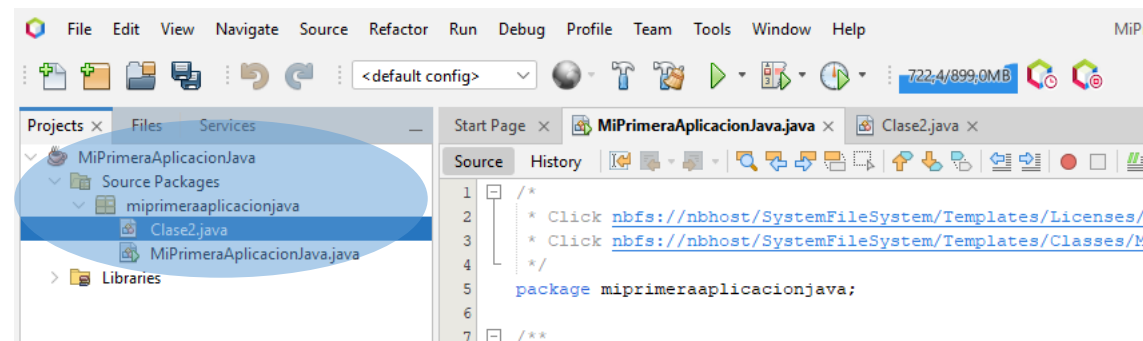
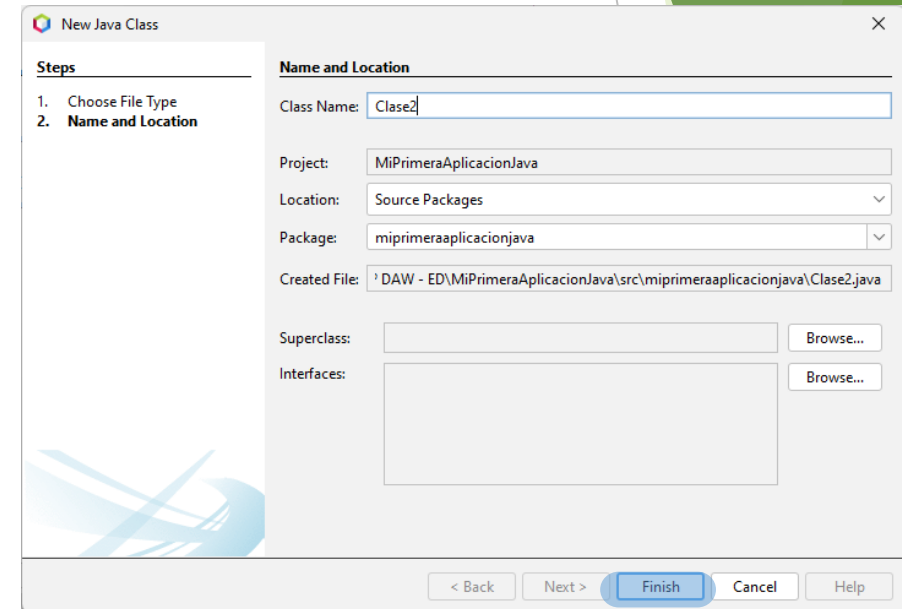
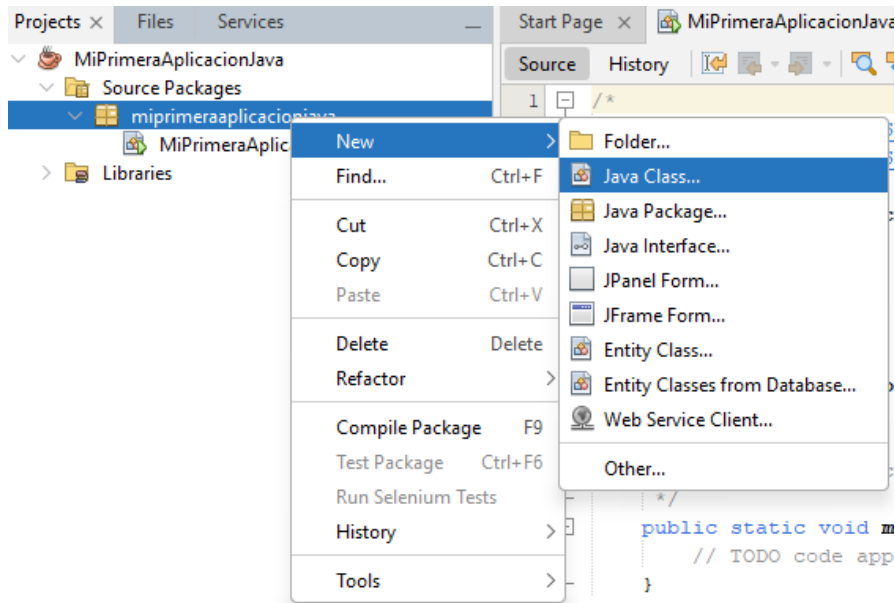
```
1 package miprimeraaplicacionjava;
2
3 /**
4  * @author USUARIO
5  */
6 public class MiPrimeraAplicacionJava {
7
8     /**
9      * @param args the command line arguments
10     */
11     public static void main(String[] args) {
12         // TODO code application logic here
13         System.out.println("Hola clase");
14     }
15 }
```

run:
Hola clase
BUILD SUCCESSFUL (total time: 0 seconds)

3. PROYECTOS JAVA CON NETBEANS

3.1. Creación de proyectos

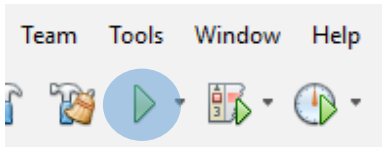
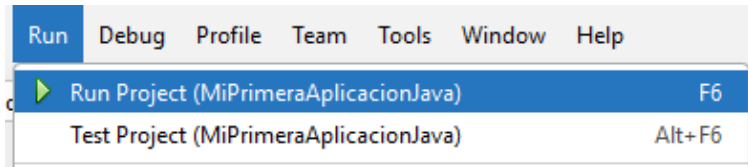
Añadir
clases



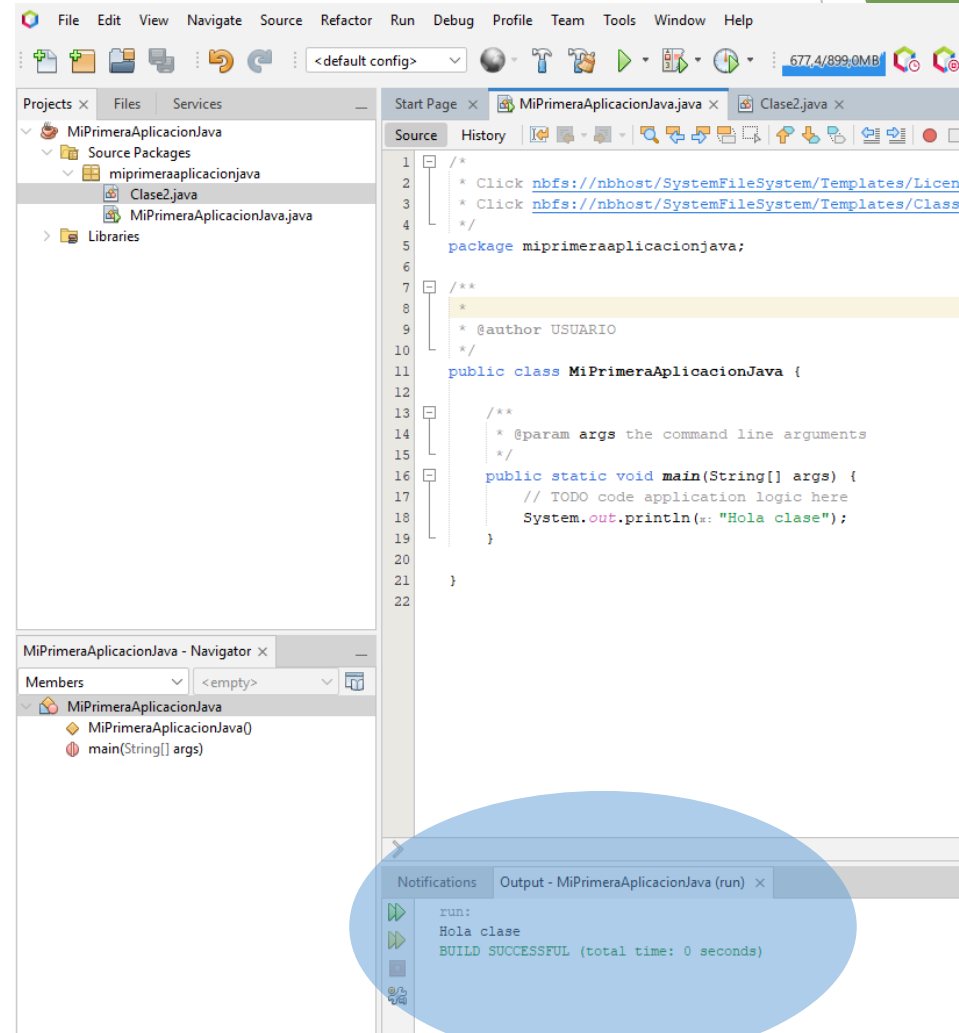
3. PROYECTOS JAVA CON NETBEANS

Ejecutar
proyecto

3.1. Creación de proyectos



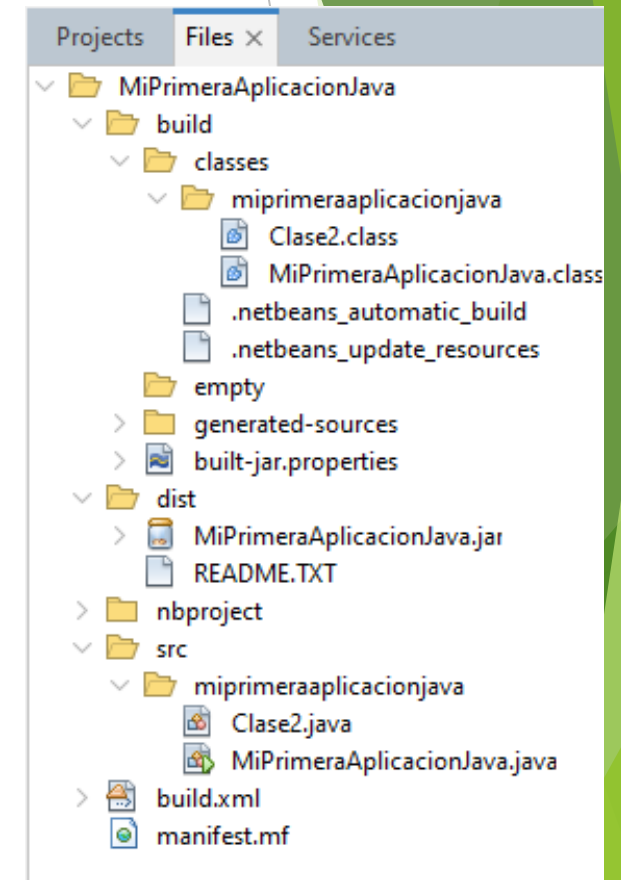
F6



3. PROYECTOS JAVA CON NETBEANS

3.2. Estructura de un proyecto Java

Un proyecto Java podemos considerarlo como contenedor organizado en carpetas de acuerdo con **una lógica para mantener organizado el código**. Un proyecto suele constar de archivos .java, archivos .class, archivos .jar, librerías, documentación...



Los archivos .jar, pueden ejecutarse directamente desde consola:

```
java -jar c:/rutadelarchivo.jar
```

3. PROYECTOS JAVA CON NETBEANS

3.2. Estructura de un proyecto Java

Archivos y directorios más comunes

build.xml

Archivo de construcción principal de Apache Ant. Contiene las instrucciones para compilar, empaquetar y ejecutar el proyecto. Puedes personalizar este archivo para agregar tareas de construcción específicas de tu proyecto.

nbproject

Contiene los archivos de configuración específicos de NetBeans para el proyecto, como `project.properties` y `project.xml`. No es necesario modificar estos archivos manualmente a menos que tengas requisitos de configuración específicos.

src

Aquí se encuentran tus archivos fuente del proyecto Java. Puedes organizar tus clases Java en paquetes y subdirectorios dentro de esta carpeta. Por ejemplo, las clases Java de tu proyecto se ubicarán en `src/com/miproyecto/`.

test

Este directorio contiene clases de prueba (unitarias) para tu proyecto. Siguiendo una estructura similar a `src`, aquí puedes organizar las clases de prueba relacionadas con las clases del proyecto.

lib

En este directorio, puedes colocar las bibliotecas externas (JARs) necesarias para tu proyecto. Estas bibliotecas se incluirán en el proceso de construcción y se usarán para compilar y ejecutar tu proyecto.

dist

Después de la construcción, los archivos generados (como archivos JAR o WAR, si es una aplicación web) se colocarán en este directorio.

build

Es el directorio donde Apache Ant realiza la construcción del proyecto. Aquí se generan archivos intermedios y de salida durante el proceso de construcción.

build-impl.xml

Este archivo es un script Ant generado por NetBeans que contiene tareas específicas de construcción y configuración para tu proyecto. Por lo general, no necesitas modificarlo manualmente, a menos que necesites personalizar tareas específicas de construcción.

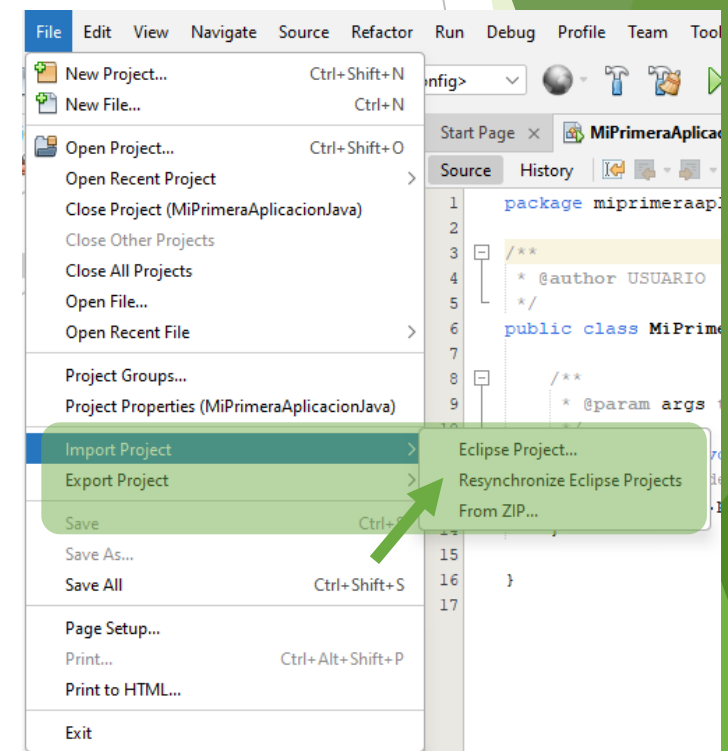
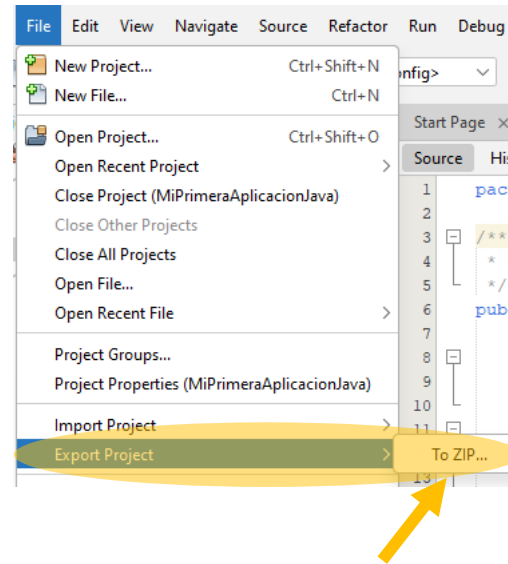
3. PROYECTOS JAVA CON NETBEANS

3.3. Importar un proyecto Java

En ocasiones, es posible que estemos trabajando con equipos de personas distintos o entre diversas compañías.

Cada programador o compañía, puede usar su IDE preferido. Es por ello, que nos puede resultar útil realizar una **importación** o **exportación** adecuada de proyectos.

Ambas opciones, las podemos desplegar desde nuestro menú principal.



3. INGENIERIA DEL SW

3.4. Otras funcionalidades

NetBeans proporciona una variedad de funcionalidades para simplificar y mejorar el proceso de desarrollo de proyectos Java.



3. INGENIERIA DEL SW

3.4. Otras funcionalidades

Depuración

Depuración interactiva:

- Potente capacidad de depuración que te permite establecer puntos de interrupción en tu código Java y seguir la ejecución del programa paso a paso.
- Puedes inspeccionar variables, ver el estado de la pila de llamadas y evaluar expresiones en tiempo real.

Depuración remota:

- Admite la depuración remota, lo que te permite depurar aplicaciones que se ejecutan en servidores remotos o dispositivos embebidos.

Monitoreo de variables:

- Puedes rastrear y examinar el valor de las variables mientras tu aplicación se ejecuta, lo que facilita la identificación y corrección de errores en tu código.

Documentación

Generación de Javadoc:

- Proporciona herramientas para generar documentación Javadoc de manera automática a partir de tus comentarios en el código.
- Esto te permite mantener una documentación clara y actualizada de tus clases y métodos.

Navegación de código:

- Puedes navegar por tu código fuente de manera eficiente utilizando atajos de teclado y la función de búsqueda de NetBeans.
- Esto facilita la búsqueda y comprensión de tu código, lo que es especialmente útil cuando trabajas en proyectos grandes.

Gestión de Versiones

Integración con sistemas de control de versiones:

- Se integra con sistemas de control de versiones como Git, SVN y Mercurial, lo que facilita el seguimiento de cambios en el código, la gestión de ramas y la colaboración en proyectos con otros desarrolladores.

Detección de cambios:

- Puede detectar cambios en tus archivos y proyectos y proporcionar una interfaz para confirmar y enviar cambios a tu sistema de control de versiones.

Pruebas

Pruebas unitarias:

- Es compatible con pruebas unitarias.
- Puedes escribir y ejecutar pruebas unitarias de manera efectiva utilizando marcos de prueba como JUnit o TestNG.

Integración de pruebas:

- Proporciona integración con herramientas de pruebas, lo que facilita la ejecución y supervisión de pruebas de regresión y pruebas de integración.

Análisis de cobertura de código:

- Puedes utilizar herramientas de análisis de cobertura de código para evaluar la efectividad de tus pruebas y determinar qué partes de tu código no están siendo probadas.



Fin

Entornos de desarrollo
M^a Carmen Safont Richarte