



Tema 3: Entornos de desarrollo (IDE)

¿Qué aprenderás?

- Identificar qué es un IDE y para qué sirve.
- Reconocer el uso básico de un entorno de desarrollo.
- Diferenciar las herramientas que puedes encontrar en un IDE.
- Definir los conceptos sobre la instalación de un entorno de desarrollo.

¿Sabías que...?

- Es posible programar con programas independientes sin usar un IDE.
- La necesidad básica que un IDE debe cubrir es editar programas y convertir el código fuente en código ejecutable.

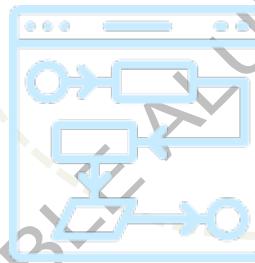


3.1. ¿Qué es un IDE?

3.1.1. Definición de Entorno de Desarrollo Integrado (IDE)

La tarea del programador la podríamos resumir (de forma muy simplificada) de la siguiente manera a partir de las siguientes tareas:

- Codificar con un lenguaje de programación determinado.
- Asegurarse que el código fuente sea válido y tenga sentido antes de ser compilado.
- Compilar el código.
- Verificar que el ejecutable funciona correctamente.



Todas estas tareas las podríamos hacer con un editor de código, un generador del ejecutable y un compilador. Todo por separado.

Un IDE es un programa que nos ofrece todas estas herramientas que necesitamos para programar en un único programa.

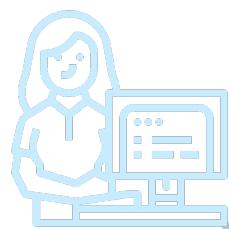
Todo en un único ecosistema pensado para que los programadores puedan codificar un software de forma cómoda incluyendo una serie de herramientas. Eclipse y NetBeans son algunos de los IDE más populares.



3.1.2. ¿Cuál es el objetivo de un IDE?

El objetivo principal es ayudar al programador en la tarea de desarrollar y diseñar software.

Esta ayuda es gracias a las múltiples herramientas que vienen incluidas. Por ejemplo el editor de código, en el que vamos a escribir el código fuente. El mismo editor de código dispone de una serie de herramientas como la función de autocompletar, resaltar posibles errores y advertencias.



Tal vez consideremos que es algo normal a día de hoy, pero de tener estas herramientas a no tenerlas existe una gran diferencia. Por muy bien que se nos de la programación hemos de reconocer que es una herramienta que nos facilita mucho la tarea durante la codificación.

En algunos lenguajes de programación más antiguos como COBOL o PLI/1 en el que estas herramientas no estaban disponibles (actualmente existen algunos entornos que ayudan bastante), la dificultad de aprendizaje y la programación que planteaban sin duda todo un desafío.

Esta ayuda que aporta el IDE está basada en parte en estas funcionalidades añadidas. Si sumamos todas estas herramientas a nuestra experiencia y habilidad, veremos como mejora nuestra experiencia al programar y que aumentaremos considerablemente la productividad en la fabricación del código.



3.1.3. ¿En qué nos puede ayudar un IDE?

Programar de una forma más cómoda

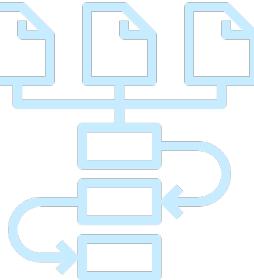
Al incluir todo un arsenal de herramientas altamente configurables y personalizables. Como las barras de menú o las diferentes vistas los atajos de teclado.

Mejorar nuestro rendimiento

Todas estas herramientas que están dentro de un mismo programa que vienen integradas unas con otras ganaremos tiempo. Que todas estas herramientas estén vinculadas es una gran comodidad a la hora de programar.

Nos ayuda a no depender de diferentes programas

El IDE nos ahorra la necesidad de usar diferentes programas. Ya que nos lo proporciona todo junto e integrado.





3.1.4. ¿Qué criterios hemos de tener en cuenta para escoger un IDE?

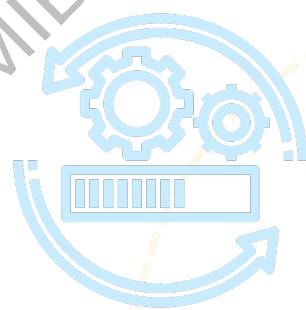
Muchas veces, y sobre todo cuando vayamos a trabajar de programadores para una empresa, nos dirán qué entorno de desarrollo hacen servir. Lo más normal es que nos tengamos que adaptar y aprender a hacerlo servir. Eso es debido a que alguien (un experto o experta) ha decidido qué es la mejor opción para llevar a cabo ese proyecto.

La decisión de escoger un IDE no es algo arbitrario o aleatorio. Tampoco vale escoger el que más te guste. Es necesario escogerlo basándose en una serie de criterios.

Los criterios que vamos a tener en cuenta para escoger el ID serán los siguientes:

3.1.4.1. El sistema operativo en el que vamos a trabajar

El sistema operativo es el primer software que está en contacto con el hardware. Por lo tanto **es el que va a comunicar el código máquina con el hardware.**



Por este motivo cuando descargamos Eclipse o NetBeans normalmente nos pregunta qué versión de sistema operativo vamos a usar ya sea Windows o Linux de 32 o 64 bits. Esto determina parte de la arquitectura interna del IDE.



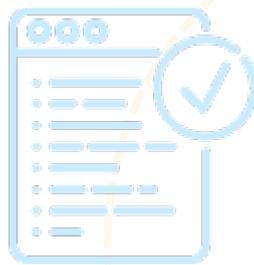
3.1.4.2. Para qué sistema operativo se va a desarrollar nuestro software

Hemos de tener en cuenta en **el sistema operativo en el que se va a ejecutar nuestra aplicación** (esto puede ser independiente al sistema operativo en el cual desarrollamos el software). Por ejemplo podemos trabajar en un sistema operativo Windows y generar ejecutables para Windows y Linux. Siempre que el IDE lo permita.

Hemos de tener en cuenta que el ejecutable que va a tener que generar el IDE debe estar adaptado al sistema operativo en particular que se va a ejecutar. A no ser que se trate de un programa que se ejecute a través de la máquina virtual como pasa en Java.

3.1.4.3. El lenguaje de programación

Para escoger el lenguaje de programación no nos podemos guiar por un lenguaje que nos resulte más fácil, con el que hemos aprendido o que nos guste más. Esta decisión se basa en las necesidades y requisitos que tiene nuestro programa.



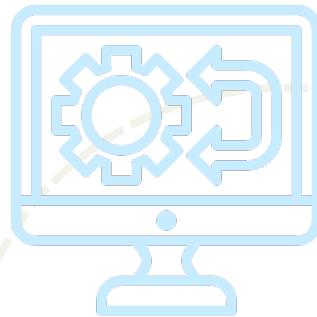
Está claro que es una decisión técnica que toma algún experto que decide cuál es el lenguaje de programación adecuado para el proyecto.

En cuanto al IDE, **no todos los entornos de desarrollo soportan todos los lenguajes**, aunque en general un IDE soporta varios lenguajes de programación.



3.1.4.4. Las herramientas que dispone el IDE

Este criterio es más relativo a la personalización y a las funcionalidades que puedan aportar a los programadores y que les puedan resultar más útiles para trabajar.



Hemos de investigar qué herramientas dispone el IDE y ver si se adecuan a nuestras necesidades o nos pueden resultar prácticas. Por ejemplo, a Eclipse se le pueden añadir herramientas que se integran fácilmente a modo de plugins.



3.1.5. Las diferentes herramientas que podemos encontrar en un IDE

Los componentes básicos comunes que se suelen encontrar en un IDE son el editor de texto, el compilador, el intérprete, el depurador y el cliente (para el control de versiones).

3.1.5.1. Compilador

Es la herramienta que permite compilar el código fuente de un lenguaje de programación a lenguaje máquina con tal de que pueda ser interpretado por el procesador. El compilador del IDE es el que realiza las diferentes fases del análisis del código.

3.1.5.2. Ejecutar de forma virtual

El IDE normalmente permite ejecutar el programa de forma virtual (RUN). Gracias a esto no va a ser necesario generar un ejecutable final.

The screenshot shows a Java application console window. The tabs at the top are 'Problems', '@ Javadoc', 'Declaration', 'Console', and 'X'. The 'Console' tab is selected. The text in the console is:

```
<terminated> EjemploProyectoJava (1) [Java Application] C:\Progr
Hola mundo!
Siguiiente instrucción
0
1
2
3
4
```

Vamos a poder hacer simulaciones del funcionamiento del programa antes de generar un ejecutable. Esto facilita mucho la tarea de testing pruebas y nos ahorra mucho tiempo. Para ejecutar de forma virtual nuestro código no debe tener errores de compilación.



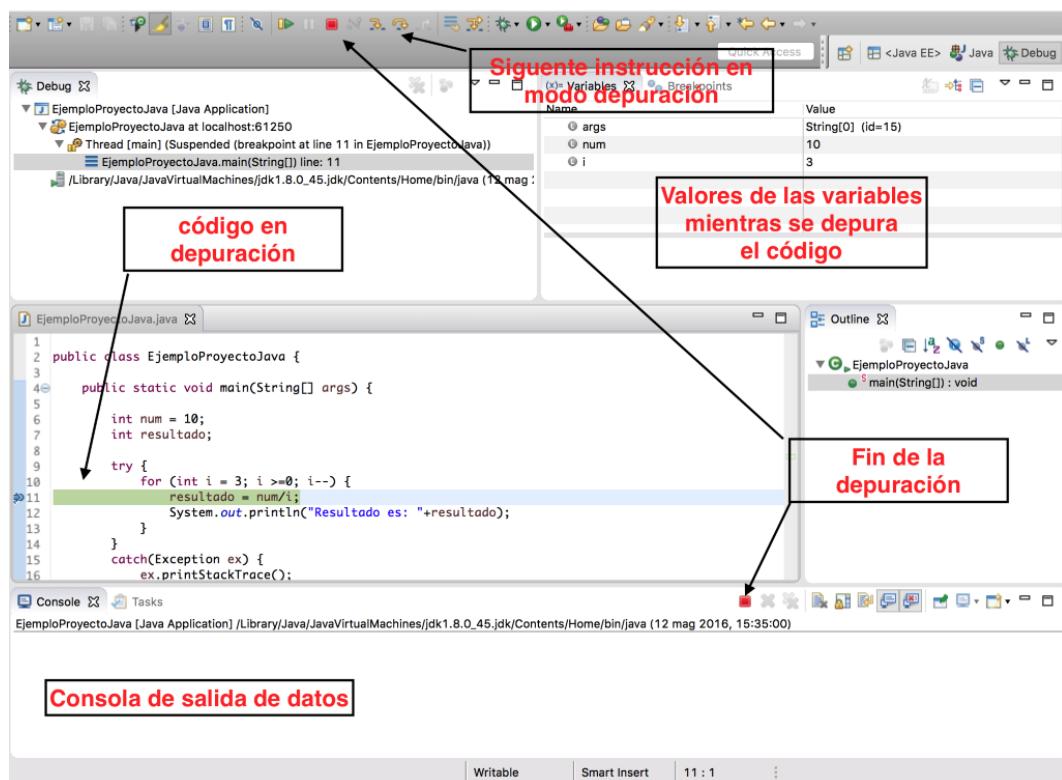
3.1.5.3. Depurador

Es la herramienta que nos permite probar y depurar el código fuente del programa.

También nos permite detectar errores.

Gracias al depurador podemos:

- Ejecutar el código línea a línea.
- Pausar el programa en un momento determinado.
- Manipular los valores de las variables modificar partes del programa mientras se ejecuta.



Un Breakpoint es un punto que permite parar la ejecución de un programa a una línea en concreto:

- Es un punto de interrupción en el código de un programa.
- Pueden haber varios.
- Añadir/Quitar un Breakpoint: a través de un click en la línea de código.



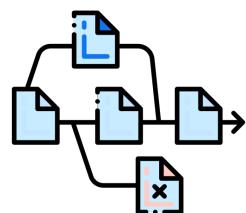
The screenshot shows the Eclipse IDE interface. On the left is the Package Explorer view, showing a project named 'EjemploProyectoJava' with a 'src' folder containing 'EjemploProyecto.java'. The main editor window displays the Java code for 'EjemploProyecto.java'. A red callout box highlights line 11 of the code, which contains a breakpoint annotation: '11 public static void main(String[] args) {'. The code implements a simple loop to divide 10 by decreasing values of i, printing the result each time. The bottom status bar indicates the file is open ('EjemploProyectoJava.java - EjemploProyecto/src') and the application is terminated ('terminated> EjemploProyectoJava [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_45.jdk/Contents/Home/bin/java (12 mag 201').

Punto de interrupción
(se activa/desactiva clicando dos veces)

```
1 public class EjemploProyectoJava {  
2     public static void main(String[] args) {  
3         int num = 10;  
4         int resultado;  
5         try {  
6             for (int i = 3; i >=0; i--) {  
7                 resultado = num/i;  
8                 System.out.println("Resultado es: "+resultado);  
9             }  
10        } catch(Exception ex) {  
11            ex.printStackTrace();  
12        }  
13    }  
14 }  
15  
16  
17  
18  
19  
20  
21 }  
22 }
```

3.1.5.4. Control de versiones

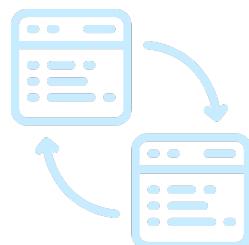
Es la herramienta que hace posible tener un registro histórico de las tareas hechas o versiones del código fuente. La vemos con más detalle más adelante en el apartado del uso básico de un IDE.





3.1.5.5. Refactorización

La refactorización es una técnica que permite reestructurar el código fuente mejorarlo sin alterar la funcionalidad. El IDE puede ayudarnos a refactorizar el código una vez codificado.



3.1.5.6. Documentación

Los IDE nos van a ayudar para generar la documentación del código que lo vamos construyendo como por ejemplo pasa con JavaDoc.

3.1.5.7. El gestor de proyectos

Nos va a permitir ajustar las dependencias del código además de todas las opciones de compilador.

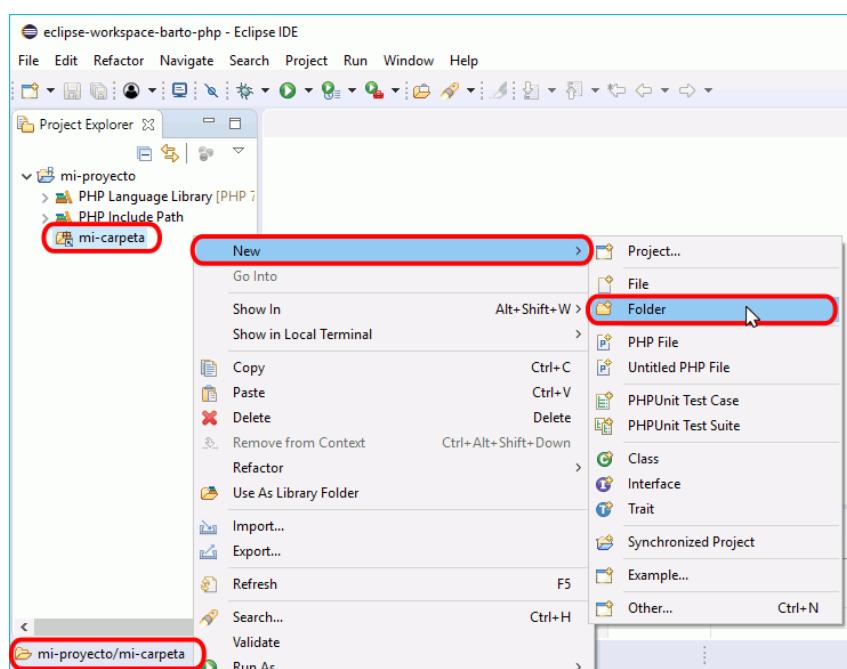
The screenshot shows the Eclipse IDE interface. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The left sidebar is the Package Explorer, showing two projects: 'OtroProyecto' and 'ProyectoPrincipal'. 'ProyectoPrincipal' has a 'src' folder containing a '(default package)'. The right side is the code editor for 'EjemploProyectoJava.java'. The code is:

```
1 public class EjemploProyectoJava {  
2     public static void main(String[] args) {  
3         // TODO Auto-generated method stub  
4         System.out.println("Hola mundo!");  
5         System.out.println("Siguiente instrucción");  
6     }  
7 }  
8  
9  
10  
11  
12 }
```



Gracias al gestor de proyectos se nos generaran de forma automatica todas las **dependencias de nuestro código fuente**. Tambien nos ayudará a tenerlas **localizadas**.

Por ejemplo cuando generamos una nueva clase en Eclipse, se generará dentro de un proyecto, el gestor se asegurará que quede debidamente enlazado haciendo las llamadas e imports pertinentes.



Es posible ajustar las dependencias de nuestro programa según lo que necesitemos.



3.1.5.8. El editor de texto

Es la ventana que nos permite crear y editar el texto del código fuente. Normalmente es un editor de texto plano.

Es posible tener más de un editor de texto abierto a la vez.

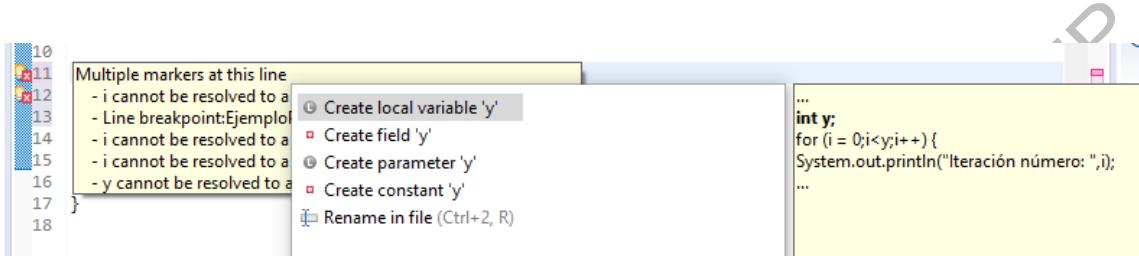
```
1 public class EjemploProyectoJava {
2     public static void main(String[] args) {
3         // TODO Auto-generated method stub
4         System.out.println("Hola mundo!");
5         System.out.println("Siguiente instrucción");
6         var x = y;
7         for (i=0;i<y;i++) {
8             i cannot be resolved to a variable
9             intln("Iteración número: ",i);
10        }
11    }
12 }
13
14
15
16
17 }
```

El editor de texto puede disponer de una serie de herramientas.



• Autocompletado de código

Una herramienta del editor de texto predice la palabra que el usuario intenta codificar después de solo introducir uno unos pocos caracteres, se anticipa a lo que vamos a escribir y además nos hace sugerencias con cierta lógica.



• Coloreado de sintaxis

Resalta las palabras del código fuente con diferentes colores o formato (negrita o cursiva) para mejorar la legibilidad del código. Normalmente se marcan con color diferente las palabras reservadas.

```
11 int y = 5;
12 for (int i = 0;i<y;i++) {
13     System.out.println(i);
14 }
15
```

• Inspecciones de clases y objetos

El inspector nos muestra los componentes de los objetos de forma jerárquica. Es muy útil porque visualmente nos permite ver la estructura de los objetos y estructura de los componentes.



3.1.5.9. Vistas

Las vistas son unas ventanas auxiliares que nos sirven para mostrar diferentes tipos de contenidos como por ejemplo el valor de las variables, el árbol de directorios del proyecto, la vista debug...

También es posible modificar las propiedades de cada componentes.

3.1.5.10. Añadir y modificar la barra de herramientas

Una perspectiva es un conjunto de ventanas de editores y vistas relacionadas entre si.

Según la perspectiva que tengamos en nuestro IDE tendremos una serie de herramientas u otras. Por ejemplo, no es lo mismo estar en la vista Java que en la vista debug.

Todas las barras de herramientas que aparecen, se pueden personalizar. De tal modo que podemos mostrar más herramientas, ocultarlas o cambiar de ubicación.

3.1.5.11. Configurar diferentes interfaces

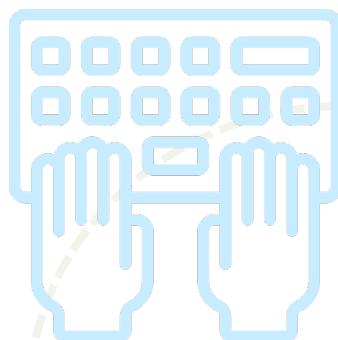
No es lo mismo la vista o interfaz que usamos para codificar que la vista que usamos para depurar el código. Necesitaremos algunas herramientas según la acción que estemos haciendo en aquel momento.

Los IDE nos van a permitir configurar diferentes interfaces con distintas herramientas posición de ventanas mostrar ocultar herramientas o utilidades.



3.1.5.12. Comandos personalizados y atajos de teclado

Los comandos personalizados y los atajos de teclado nos permiten ejecutar tareas de forma más rápida a partir de una combinación de teclas.



Los programadores profesionales suelen dominar bastantes atajos de teclado, ya que con estas pequeñas acciones se gana tiempo, que si se va sumando, al largo del proyecto, se pueden ganar minutos y horas.

Por ejemplo, en eclipse es posible crear un *metodo main* mediante el siguiente atajo de teclado: Escribir main, pulsar CTRL + ESPACIO y seleccionar *main method*.



3.1.6. El uso básico de un IDE

El uso que pueden hacer los programadores de un IDE lo podemos resumir de forma rápida en codificar programas en un determinado lenguaje de programación con tal de obtener un ejecutable que haga una determinada función.

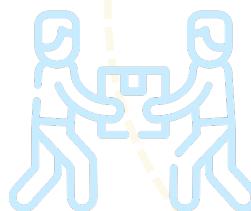
En definitiva, el uso básico podría ser:

- Codificar mediante el editor de código fuente.
- Realizar pruebas y verificar el código con el depurador.
- Compilar el código fuente.
- Trabajar colaborativamente.

Recuerda que estas herramientas que dispone el IDE de forma conjunta también las podríamos encontrar por separado. Por ejemplo, podríamos usar un editor independiente, compilar en otro programa generar ejecutable o enlazar con librerías de otro programa. Pero todo eso puede llegar a ser una tarea bastante compleja y perder la cohesión entre estas herramientas implicaría más tiempo en hacerlo con otras.

3.1.6.1. El desarrollo colaborativo

Los proyectos de software casi siempre requieren desarrollo colaborativo, es decir más de un programador va dedicarse a codificar en el mismo proyecto.





El programa de control de versiones que viene integrado en el IDE es una herramienta asíncrona que nos va a permitir controlar y gestionar las fuentes y versiones del código que guardar en un repositorio.

- Son aplicaciones asíncronas
- Permiten controlar y gestionar las versiones del código
- Constan de servidor y cliente

El funcionamiento básico de un control de versiones lo podemos resumir en:

- Descargar el código del repositorio.
- Trabajar sobre el código en local.
- Actualizar los cambios al repositorio.

Por ejemplo: un programador a puede estar trabajando en un código fuente uno...

Si más de un programador trabajamos sobre un mismo código fuente es muy probable que si no controlamos las distintas versiones de este código podemos eliminar cambios que se han hecho previamente mientras el programador estaba editando.

La herramienta del control de versiones normalmente está descentralizada, no depende de un único servidor, y cada usuario puede tener una copia del código con lo que todos los cambios. En el caso que un programador actualice cada nuevo código se notificará al servidor que tiene todo el código y este se encargará de notificar y mantener todas las versiones existentes del código.

Gracias al control de versiones vamos a poder saber qué archivos actualizar, qué cambios podemos omitir para no pisar trabajo de otros compañeros y viceversa además de saber qué archivos han cambiado y obtener directamente sobre nuestro código todos estos nuevos cambios en nuestro actual desarrollo.



En muchos casos se hacen servir también herramientas externas, como por ejemplo git.

El gestor de versiones consta de un cliente y un servidor. El servidor de forma centralizada tiene todos los códigos y se va a encargar de que en cada cliente venga una versión actualizada de todo el código.

El repositorio es todo el conjunto de código que está guardado en el servidor para que los clientes puedan descargar y subir código.



3.1.7. La instalación de un IDE

Cada IDE tiene sus propias características y funcionalidades específicas. En esta documentación, no vamos a ver la instalación de un IDE en concreto, ya que dependiendo del programa o incluso de la versión va a ser diferente. Lo que vamos a ver aquí son algunos conceptos básicos relativos a la instalación de Eclipse para codificar en Java. Para la instalación de Eclipse disponéis de otra documentación.

Normalmente la instalación de un IDE se podrá hacer de forma sencilla siguiendo un asistente. La parte más compleja es al añadir plugins o funcionalidades específicas.

Desarrollar en Java en Eclipse

Para poder desarrollar programas Java la primera cosa es tener el "Java Development Kit" (JDK) instalado.



De una forma resumida, el JDK nos permite ejecutar y desarrollar aplicaciones Java y está formado por:

- **JRE:** Es la máquina virtual de Java.
- **Java SE:** las librerías (códigos) Java necesarios para ejecutar programas de escritorio.

The screenshot shows the Oracle Java Technology Network download page for Java SE. The top navigation bar includes links for 'Ménu', 'ORACLE', 'Resumen', 'Descargas', 'Documentación', 'Comunidad', 'Tecnología', and 'Formación'. The main content area is titled 'Descargas de Java SE' and features two download options: 'Java Platform (JDK) 8u111 / 8u112' (with a 'DOWNLOAD +' button) and 'NetBeans con JDK 8' (with a 'DOWNLOAD +' button). A sidebar on the right is titled 'Java SDKs and Tools' and lists links for Java SE, Java EE and Glassfish, Java ME, Java Card, NetBeans IDE, and Java Mission Control. Another sidebar titled 'Java Resources' lists links for Java APIs, Technical Articles, Demos and Videos, Events, Java Magazine, and Developer Training. At the bottom, a note states: 'Java SE 8u111 incluye importantes soluciones de seguridad. Oracle recomienda que todos los usuarios de Java SE 8 actualicen esta versión. Java SE 8u112 es una actualización con un conjunto de parches, que incluye todas las características adicionales de 8u111 (descritas en las notas de la versión)'.



Diferentes implementaciones de JDK:

- **J2SE** para aplicaciones de escritorio.
- **J2EE** para aplicaciones distribuidas y webs (Servlets/JSP, RMI, EJB, ...).
- **J2ME** para plataformas con recursos más reducidos como móviles o PDAs.

VERSIÓN IMPRIMIBLE ALUMNO LINKIAFP



Recursos y enlaces

- [Atajos para eclipse](#)



- [Información sobre el JDK de Java](#)



Conceptos clave

- **IDE:** Entorno de desarrollo integrado.
- **Objetivo IDE:** Facilitar la tarea del programador.
- **Criterios para escoger un IDE:** El sistema operativo en el que vamos a trabajar, para qué sistema operativo se va a desarrollar nuestro software, el lenguaje de programación, las herramientas que dispone el IDE.
- **Herramientas:** compilador, ejecutar de forma virtual, depurador, control de versiones, refactorización, documentación, gestor de proyectos, editor de texto, vistas, barras de herramientas, interfaces, atajos de teclado...
- **Desarrollo colaborativo:** cuando más de un programador va dedicarse a codificar en el mismo proyecto. Con el control de versiones herramienta asíncrona que nos va a permitir controlar y gestionar las fuentes y versiones del código que guardar en un repositorio.



Test de autoevaluación

1. ¿En qué nos puede ayudar un IDE?
 - a) A programar de una forma más cómoda.
 - b) A mejorar nuestro rendimiento.
 - c) En no depender de otros programas.
 - d) Todas las anteriores.
2. El desarrollo colaborativo...
 - a) Es una herramienta asíncrona que nos va a permitir controlar y gestionar las fuentes y versiones del código que guardar en un repositorio.
 - b) Es una herramienta síncrona que nos va a permitir controlar y gestionar las fuentes y versiones del código que guardar en un repositorio.
 - c) Permite ejecutar y desarrollar aplicaciones Java y está formado por JRE y Java SE.
 - d) Permite ejecutar y desarrollar aplicaciones Java y está formado por el J2SE, J2EE y J2ME.
3. De los criterios hemos de tener en cuenta para escoger un IDE, ¿cuál debería ser el primer a tener en cuenta?
 - a) El sistema operativo en el que vamos a trabajar.
 - b) Para qué sistema operativo se va a desarrollar nuestro software.
 - c) El lenguaje de programación.
 - d) Las herramientas que dispone el IDE.



Ponlo en práctica

Actividad 1

Instala eclipse en tu sistema operativo y haz capturas de pantalla de las siguientes perspectivas:

- Depurador.
- Java

Muestra tambien la herramienta que te permite personalizar la perspectiva.



SOLUCIONARIOS

Test de autoevaluación

1. ¿En qué nos puede ayudar un IDE?
 - a) A programar de una forma más cómoda.
 - b) A mejorar nuestro rendimiento.
 - c) En no depender de otros programas.
 - d) **Todas las anteriores.**
2. El desarrollo colaborativo...
 - a) **Es una herramienta asíncrona que nos va a permitir controlar y gestionar las fuentes y versiones del código que guardar en un repositorio.**
 - b) Es una herramienta síncrona que nos va a permitir controlar y gestionar las fuentes y versiones del código que guardar en un repositorio.
 - c) Permite ejecutar y desarrollar aplicaciones Java y está formado por JRE y Java SE.
 - d) Permite ejecutar y desarrollar aplicaciones Java y está formado por el J2SE, J2EE y J2ME.
3. De los criterios hemos de tener en cuenta para escoger un IDE, ¿cuál debería ser el primer a tener en cuenta?
 - a) **El sistema operativo en el que vamos a trabajar.**
 - b) Para qué sistema operativo se va a desarrollar nuestro software.
 - c) El lenguaje de programación.
 - d) Las herramientas que dispone el IDE.



Ponlo en práctica

Actividad 1

Instala eclipse en tu sistema operativo y haz capturas de pantalla de las siguientes perspectivas:

- Depurador.

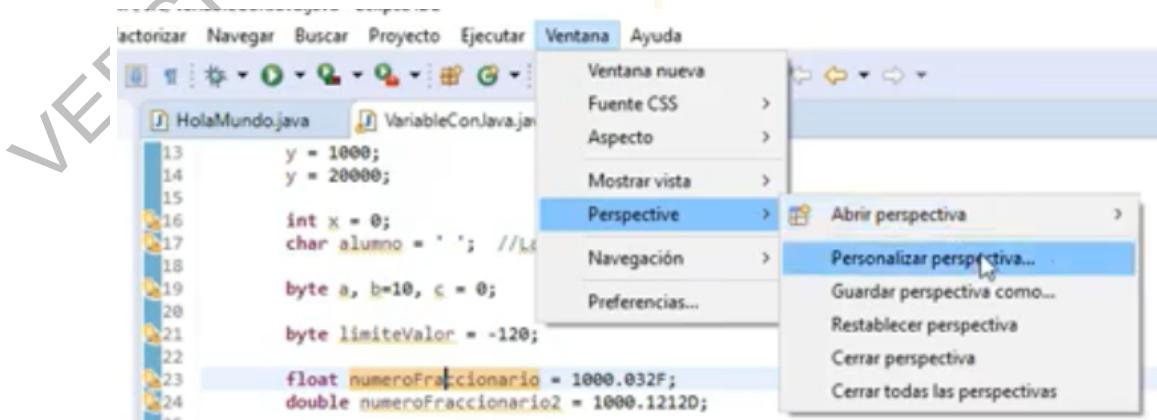
```
workspace - PRIMER_PROYECTO_JAVA/src/VariableConJava.java - Eclipse IDE
Archivo Editar Código fuente Refactorizar Navegar Buscar Proyecto Ejecutar Ventana Ayuda
Depurar Project Explorer
13     y = 1000;
14     y = 20000;
15
16     int x = 0;
17     char alumno = ' ';
18
19     byte a, b=10, c = 0;
20
21     byte limiteValor = -120;
22
23     float numeroFraccionario = 1000.032F;
24     double numeroFraccionario2 = 1000.1212D;
25
26     boolean existe = true;
27
28     boolean entero = false;
29
30
31     int comparador1 = 1001;
32     int comparador2 = 1000;
33
34     boolean resultado = (comparador1 > comparador2); //==, >, <, >=, <=
35
36     System.out.println("El resultado es: " + resultado);
37
38 }
39 }
```

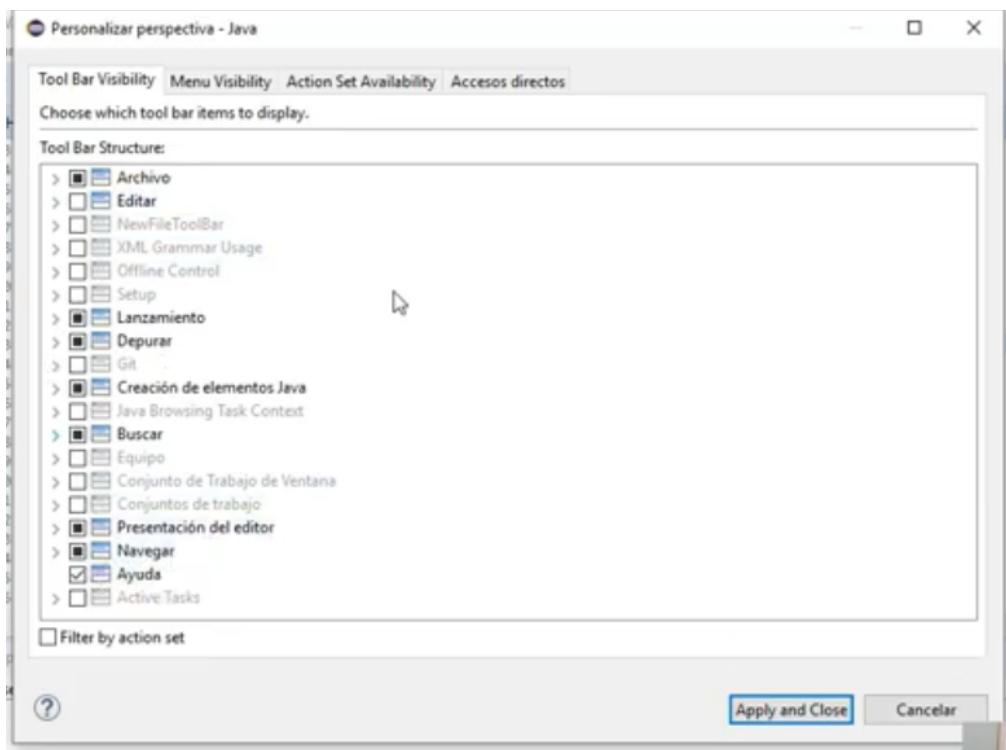
Consola Problems Debug Shell
No se pueden visualizar consolas en este momento.

- Java

Muestra también la herramienta que te permite personalizar la perspectiva.

Solución:





VERSIÓN IMPRIMIBLE ALU