

UD 2.1

INSTALACIÓN Y USO DE ENTORNOS IDE

ENTORNOS DE DESARROLLO

PARTE 1 DE 4: INTEGRATED DEVELOPMENT ENVIRONMENT (IDE) E INTRODUCCIÓN A JAVA

Autor: Sergio Badal

Modificado: Raúl Palao

Fecha: 15-10-2023

Licencia Creative Commons

versión 1.0



Reconocimiento - NoComercial - CompartirIgual (by-nc-sa): No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

1. ¿QUÉ ES UN ENTORNO DE DESARROLLO INTEGRADO (IDE)?

Se entiende por **Entorno de Desarrollo Integrado** o Integrated Development Environment (IDE) una aplicación informática que tiene el objetivo de asistir al programador en la tarea de diseñar un software mediante la inclusión de múltiples herramientas destinadas para dicha tarea” (Casado, 2012:46). En definitiva, es ~~un programa~~ una aplicación informática que ayuda a programar.

Los elementos básicos de un IDE son:

- Un **editor de código**: para escribir el código fuente
- Un **compilador**: para generar el código objeto y el ejecutable de lenguajes no interpretados
- Un **intérprete**: para traducir el código
- Un **depurador**: que ayuda a corregir errores

Sus características principales son

- Pueden ser para uno o más lenguajes de programación
- Ayudan en la visualización del código fuente
- Permite moverse rápidamente entre los ficheros de la aplicación

Usar IDE, por tanto, supone una ayuda a la programación. Sin embargo, como se debe acostumbrarse a su funcionamiento acaban generando dependencia. Además, consumen más recursos y algunos son de pago.

Hoy en día existen editores de texto que realizan dichas funciones básicas (Notepad++, Texpad, KLite...). Todos ellos identifican las palabras reservadas y los elementos clave coloreándolos.

2. ¿POR QUÉ USAR UN IDE?

Sin embargo, un IDE ofrece esto y mucho más. Principalmente, el **autocompletado de código**: cada vez que comenzamos a escribir una palabra reservada aparece una ayuda contextual indicando las opciones con las que seguir. También permite crear estructuras de clases o instrucciones de bucles automáticamente. Además, ofrecen herramientas para refactorizar, para ejecutar depurando más opciones que ayudan a la programación y que hacen más corto el ciclo de vida del software.

Otra ventaja de usar IDEs es que son “altamente configurables”. Esto quiere decir que cada programador puede configurarlo como más le agrada: con más o menos barras de herramientas, con atajos de teclado, con comandos personalizados, con distintas ubicaciones física de los distintos paneles, etc. Pero también permite configurar la depuración y la compilación de proyectos, convirtiéndose así en una herramienta cómoda y útil.

Además, como editores de programas que son, permiten la virtualización de la ejecución del código, de modo que se puede probar la funcionalidad del programa sin tener que desplegarla realmente.

Finalmente, los IDE permiten el **desarrollo colaborativo**. Esto es, desarrollar el software de manera descentralizada y distribuida, a través de la integración con el control de versiones. Así, varios programadores pueden trabajar sobre el mismo código mejorándolo, tal y como ocurre en el software libre. El control de versiones se lleva a cabo creando repositorios para que los clientes puedan descargar y subir código. Así, se trabaja sobre la versión anterior, y no sobre el proyecto inicial que se ha subido al repositorio. Los IDE permiten un fácil manejo de las versiones y una rápida sincronización.

En la siguiente figura podemos ver algunas de las funciones más destacadas:

Depurador	• Analiza elementos del programa y revisa errores
GUI	• (Graphical User Interface) Crea ventanas, botones...
Control de versiones	• Guarda el código fuente sin perder versiones anteriores
Plantillas	• Crea programas en base a uno creado como modelo
Prueba	• Prueba programas
Multiplataforma	• Uso en distintos sistemas operativos
Multilenguaje	• Uso con distintos lenguajes de programación
Libre/pago	• Permite la opción de uso pagando/sin pagar
Plugins	• Instala nuevas funciones
Bases de datos	• Interactúa con bases de datos
Tomcat	• Puede incluir el servidor de aplicaciones web interactivas Tomcat
UML	• Realizar gráficos con el lenguaje de modelado UML
Documentación	• Documentar el código
Refactorización	• Optimiza el código
Formateo de código	• Organiza el código fuente para que sea más legible

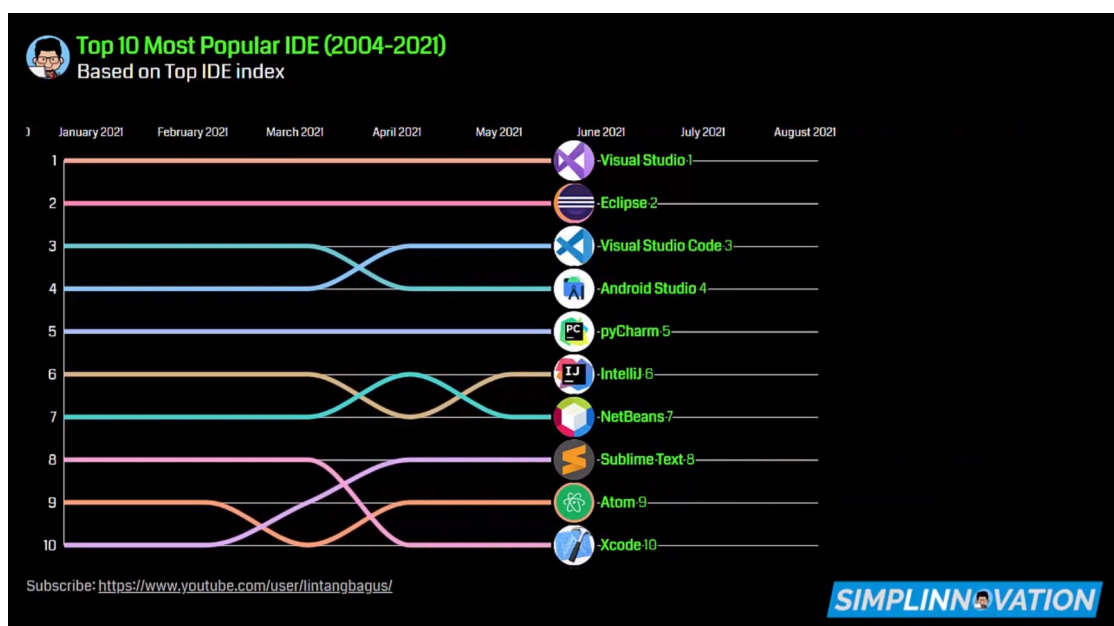
A la hora de elegir un IDE debemos tener en cuenta varias cosas:

- **Sistema operativo:** Hoy en día la gran mayoría de IDEs son multiplataforma.
- **Lenguaje de programación:** un IDE puede soportar uno o varios lenguajes de programación.
- **Framework:** la plataforma de trabajo también puede variar (por ejemplo, si vamos a desarrollar con Visual Basic con la plataforma .NET en Linux, deberíamos usar Mono Develop en vez de VB).
- **Herramientas:** también se elige un IDE por el surtido de herramientas que posee. No todos los IDE tienen las mismas funciones.

3. EVOLUCIÓN DE LOS IDE

Existen cientos de IDE casi todos especializados en uno o varios lenguajes. Para Java, los más usados son NetBeans y Eclipse (los que veremos en ese módulo) pero existen muchos más. En este vídeo puedes ver su evolución:

- <https://youtu.be/Eoslfa4N3Gw?si=Sg7pWoMY0bdyemk5>



4. EDITORES DE CÓDIGO

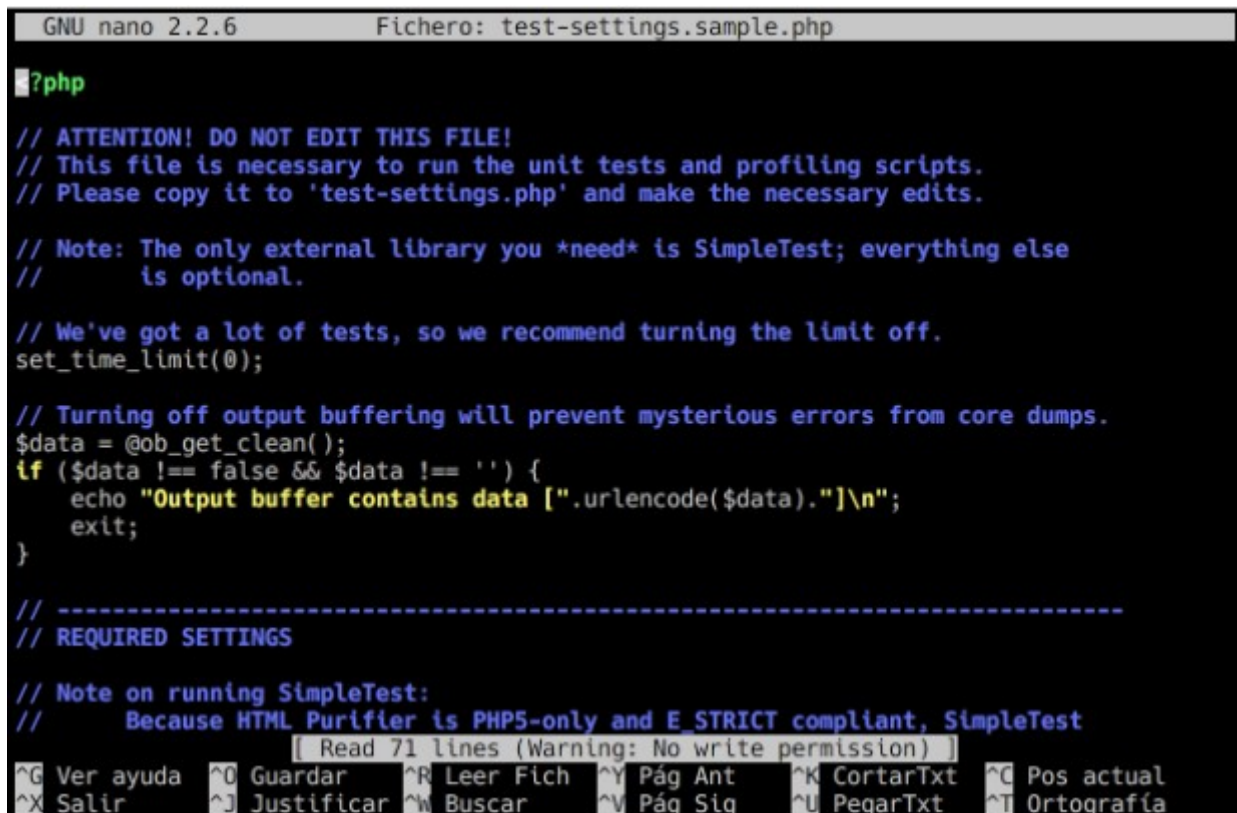
Los editores de código son aplicaciones que no tienen tantas características como los IDE. Normalmente solo disponen de una ventana donde se escribe el código fuente. El proceso de compilación es externo al editor. Aunque es cierto que presentan una serie de ventajas:

- Rapidez.
- Simplicidad.
- Menor consumo de recursos.

Por otro lado, también presentan desventajas:

- Solo podemos trabajar con ficheros concretos, en lugar de proyectos enteros.
- Disponen de menos cantidad de plugins.

El editor de texto más conocido en Windows es el **bloc de notas**. En las distribuciones Linux disponemos de editores gráficos como **Gedit** y editores por consola como “vi” o “nano”.



```
GNU nano 2.2.6 Fichero: test-settings.sample.php

?php

// ATTENTION! DO NOT EDIT THIS FILE!
// This file is necessary to run the unit tests and profiling scripts.
// Please copy it to 'test-settings.php' and make the necessary edits.

// Note: The only external library you *need* is SimpleTest; everything else
//       is optional.

// We've got a lot of tests, so we recommend turning the limit off.
set_time_limit(0);

// Turning off output buffering will prevent mysterious errors from core dumps.
$data = @ob_get_clean();
if ($data !== false && $data !== '') {
    echo "Output buffer contains data [" . urlencode($data) . "]\n";
    exit;
}

// -----
// REQUIRED SETTINGS

// Note on running SimpleTest:
//   Because HTML Purifier is PHP5-only and E_STRICT compliant, SimpleTest
//   [ Read 71 lines (Warning: No write permission) ]

^G Ver ayuda ^O Guardar ^R Leer Fich ^Y Pág Ant ^K CortarTxt ^C Pos actual
^X Salir ^J Justificar ^W Buscar ^V Pág Sig ^U PegarTxt ^T Ortografía
```

5. JAVA

Java es un lenguaje de programación y una plataforma informática que fue comercializada por primera vez en 1995 por Sun Microsystems.

El lenguaje de programación Java fue desarrollado originalmente por James Gosling, de Sun Microsystems (constituida en 1983 y posteriormente adquirida el 27 de enero de 2010 por la compañía Oracle), y publicado en 1995 como un componente fundamental de la plataforma Java de Sun Microsystems.

En sus dos grandes versiones podemos distinguir:

- **Java SE:** es la versión estándar de la plataforma, siendo esta plataforma base para todo entorno de desarrollo en Java en cuanto a aplicaciones cliente, de escritorio o web.
- **Java EE:** es la versión más grande de Java y se utiliza en general para crear aplicaciones grandes de cliente / servidor y para desarrollo de WebServices.

Cuando compilamos con JAVA no generamos directamente código ejecutable, sino que se generan byte-codes en un fichero con extensión .class.

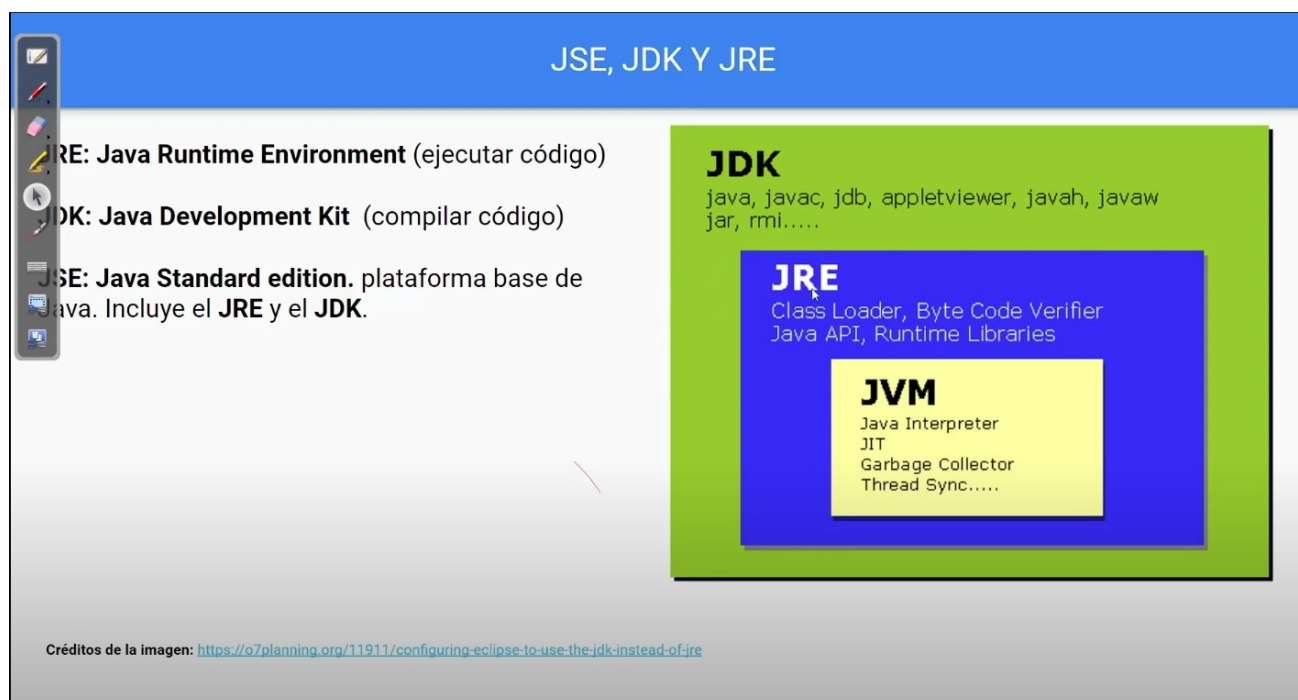
El fichero ejecutable en JAVA puede tener extensión:

- **JAR: (Java Archive):** es un formato de archivo independiente de la plataforma que permite que varios archivos puedan ser encapsulados dentro de uno solo, de modo que éste pueda ser una aplicación completa de fácil movilidad y ejecución.
- **WAR: (Web Application archive):** es un archivo JAR (con la extensión WAR) usado para distribuir una colección de archivos JSP, servlets, clases Java, archivos XML y contenido web estático (HTML). En conjunto constituyen una aplicación Web.
- **EAR: (Enterprise Archive File):** es un formato para empaquetar varios módulos en un solo archivo. Permite desplegar varios módulos de éstos en un servidor de aplicaciones. Contiene archivos XML, llamados descriptores de despliegue, que describen cómo efectuar esta operación (EAR = JAR + WAR).

5.1 HERRAMIENTAS DE JAVA

JAVA dispone de varias herramientas dentro de su plataforma, entre las que se encuentran las siguientes:

- **JRE (Java Runtime Environment):** incluye la Máquina Virtual de Java (JVM), que es responsable de ejecutar el código Java, así como bibliotecas y clases estándar necesarias para la ejecución de aplicaciones Java. Cuando desarrollas una aplicación en Java, la mayoría de las personas que la utilizarán no necesitan instalar un JDK (Java Development Kit), que es lo que utilizas para programar en Java. En su lugar, solo necesitan tener el JRE instalado en sus sistemas para poder ejecutar la aplicación.
- **JDK (Java Development Kit):** es un conjunto de herramientas y software necesario para desarrollar aplicaciones en el lenguaje de programación Java. El JDK proporciona todo lo que un desarrollador de Java necesita para escribir, compilar, depurar y ejecutar aplicaciones Java.
- **JSE/JEE:** como hemos descrito en el apartado anteriores son las versiones de JAVA. Incluyen el JRE y el JDK.



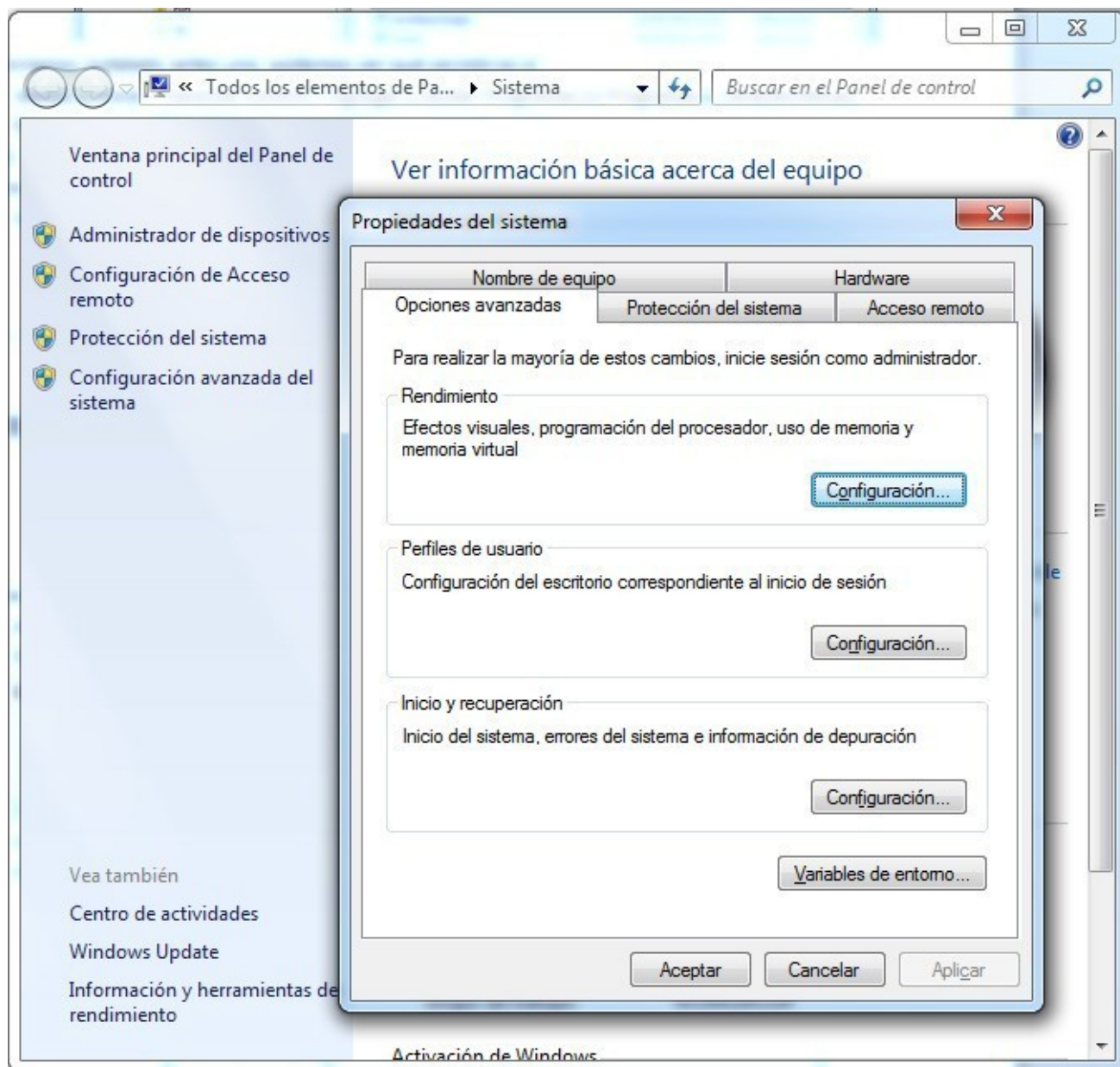
5.2 EJEMPLO DE COMPILACIÓN EN JAVA

Podemos comprobar la versión de nuestro JAVA abriendo la terminal y escribiendo `java --version`:

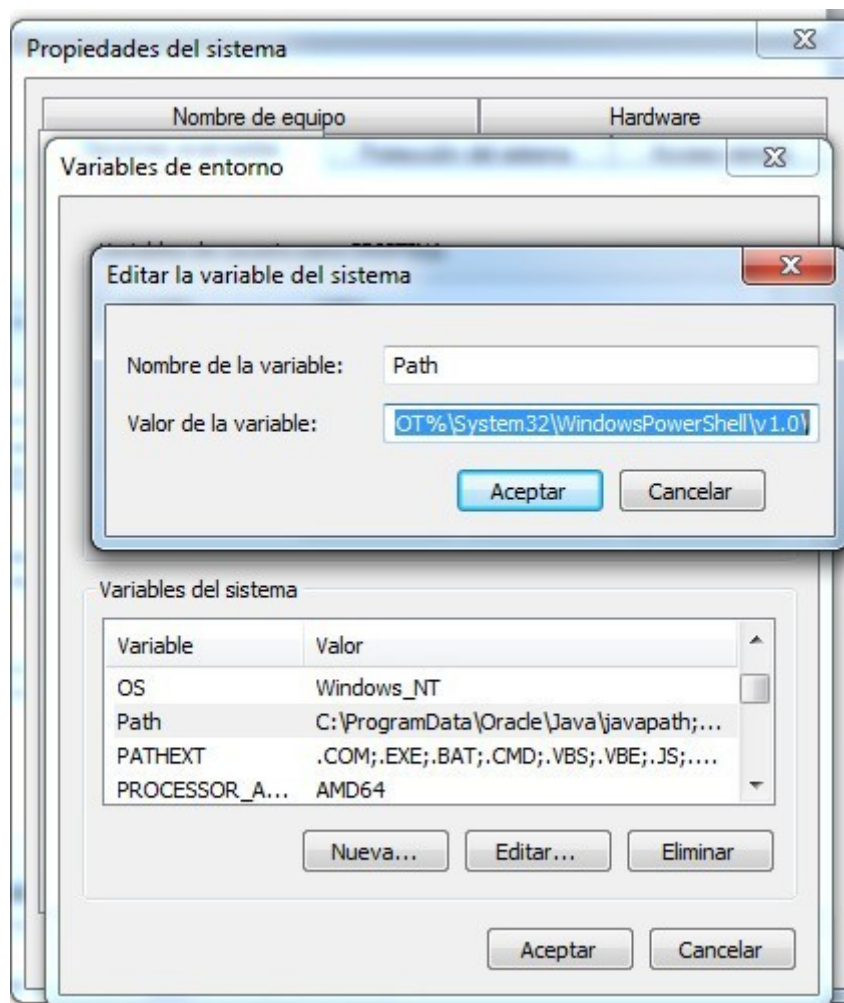
```
Administrador: Windows Pow x + v
PS C:\Archivos de programa\Java\jdk-21\bin> java --version
java 21.0.1 2023-10-17 LTS
Java(TM) SE Runtime Environment (build 21.0.1+12-LTS-29)
Java HotSpot(TM) 64-Bit Server VM (build 21.0.1+12-LTS-29, mixed mode, sharing)
PS C:\Archivos de programa\Java\jdk-21\bin>
```

Antes de comenzar a compilar debemos establecer las variables de entorno en el caso de Windows. Esto le dirá al compilador donde se encuentran los ficheros que nos permiten compilar y ejecutar nuestros programas JAVA:

- Vamos a inicio y hacemos botón derecho sobre “Mi PC” > Propiedades > Configuración avanzada del sistema > Opciones avanzadas > Variables de entorno.



- Buscamos en las variables de sistema la variable PATH y le damos a editar:



- Nos ponemos al final del Valor de Variable y AÑADIMOS lo siguiente ;C:\Program Files\Java\VERSIÓN_DE_TU_JAVA\bin, en el caso de este ejemplo, sería ;C:\Program Files\Java\jdk1.8.0_20\bin (OJO: ten en cuenta que si cambiaste la instalación de Java a otra carpeta deberás poner aquí la ruta correcta)
- Pulsamos "Aceptar" y pulsamos el botón "Nueva..." para crear una nueva variable del sistema, que se llamará Classpath y tendrá como valor la dirección de nuestro src.zip, que por defecto es;C:\Program Files\Java\VERSIÓN_DE_TU_JAVA\src.zip, en el caso de este ejemplo sería ;C:\Program Files\Java\jdk1.8.2.0_20\src.zip

5.3 EJEMPLO DE COMPILACIÓN, EJECUCIÓN Y EMPAQUETADO

Vamos a probar con un ejemplo:

- Abrimos Notepad.exe
- Escribimos el siguiente código:

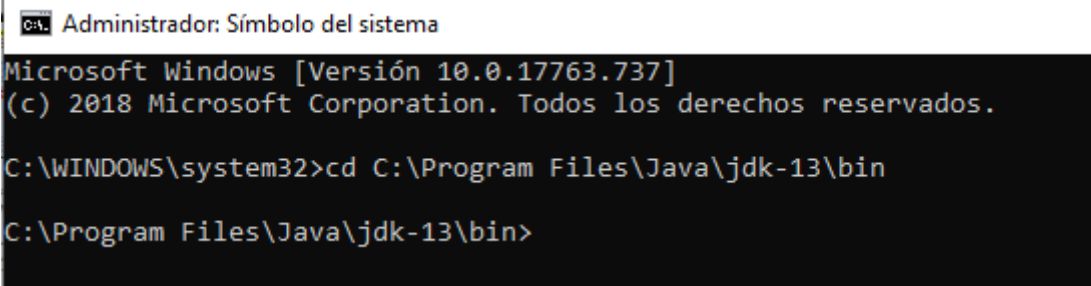
```
public class HolaMundo{  
    public static void main(String[] args){  
        System.out.println("Hola Mundo");  
    }  
}
```

Lo guardamos como HolaMundo.java (**OJO**, debes seleccionar otras extensiones para que no lo guarde como .txt!). Lo teneis que guardar en la misma carpeta donde esta el fichero javac

- Vamos a ejecutarlo. Para ello accedemos a la línea de comandos (cmd).

En mi caso C:\Program Files\Java\jdk-13\bin

- Nos dirigimos al directorio donde están los ejecutables del JDK (donde hemos guardado nuestro fichero), así que en la terminal escribimos (en mi caso es: cd C:\Program Files\Java\jdk-13\bin)



```
Administrador: Símbolo del sistema  
Microsoft Windows [Versión 10.0.17763.737]  
(c) 2018 Microsoft Corporation. Todos los derechos reservados.  
C:\WINDOWS\system32>cd C:\Program Files\Java\jdk-13\bin  
C:\Program Files\Java\jdk-13\bin>
```

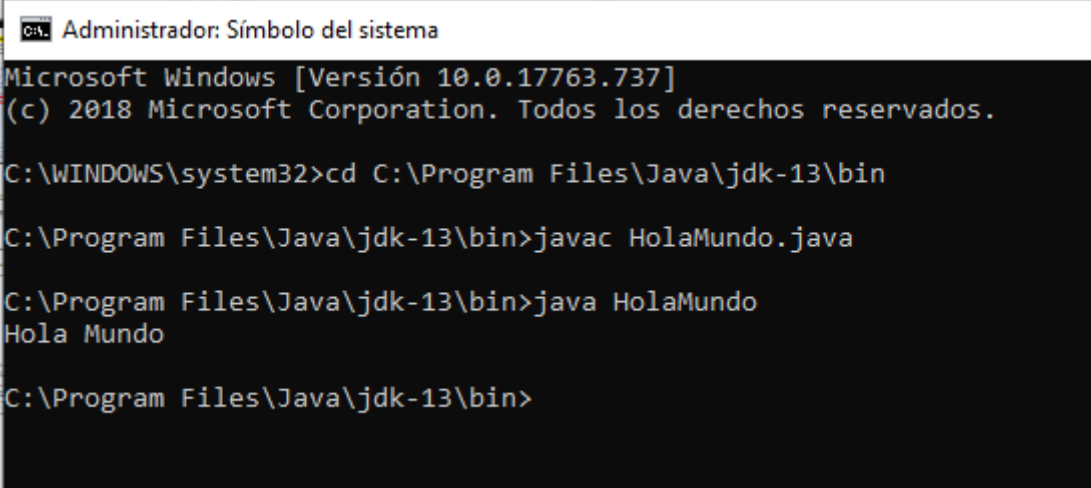
Compilamos mediante el siguiente comando:

```
javac HolaMundo.java
```

Después ejecutamos con:

```
java HolaMundo
```

De esta forma se mostrará por pantalla el texto.



```
Administrador: Símbolo del sistema  
Microsoft Windows [Versión 10.0.17763.737]  
(c) 2018 Microsoft Corporation. Todos los derechos reservados.  
C:\WINDOWS\system32>cd C:\Program Files\Java\jdk-13\bin  
C:\Program Files\Java\jdk-13\bin>javac HolaMundo.java  
C:\Program Files\Java\jdk-13\bin>java HolaMundo  
Hola Mundo  
C:\Program Files\Java\jdk-13\bin>
```

En este caso solamente tenemos un fichero .java y su correspondiente .class, pero en la mayoría de los proyectos JAVA tendremos un gran número de ficheros con código fuente, es por esto que debemos empaquetar nuestro código en único fichero. En el caso de JAVA en un fichero .jar.

Los ficheros JAR (Java ARchives) permiten recopilar en un solo fichero varios ficheros diferentes almacenándolos en un formato comprimido para que ocupen menos espacio. Es por tanto algo similar a un fichero .zip. La particularidad de los ficheros .jar es que no necesitan ser descomprimidos para ser usados.

Para crear un JAR primero es necesario crear el Manifest, un archivo de texto que contiene información sobre la versión del programa, el creador y la clase principal a ejecutar.

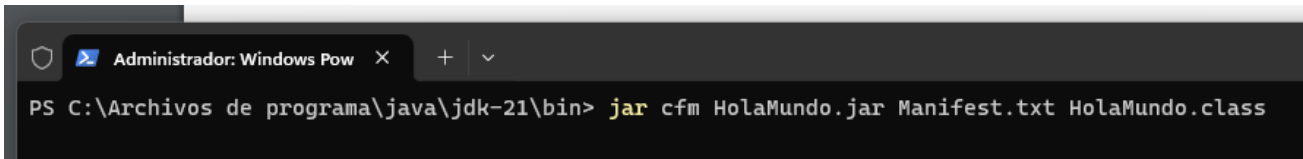
Con el bloc de notas creamos un nuevo archivo de texto llamado Manifest.txt con el texto de abajo, y lo guardamos en la misma carpeta del archivo HolaMundo.java.

Manifest-version: 1.0
Created-By: 1.0 (DAW)
Main-Class: HolaMundo

Una vez creado el fichero Manifest.txt podemos crear el empaquetado .jar con el siguiente comando:

```
C:\Users\raulp>jar cfm HolaMundo.jar Manifest.txt HolaMundo.class
```

Ahora para ejecutar el fichero .jar empaquetado podemos ejecutar el siguiente comando:



```
PS C:\Archivos de programa\java\jdk-21\bin> jar cfm HolaMundo.jar Manifest.txt HolaMundo.class
```

6. BIBLIOGRAFÍA

- i. Escobedo, J. G. (2012, 20 agosto). *Depuración de programas con NetBeans*. Sitio Web de Javier García Escobedo (javiergarciaescobedo.es).
<https://javiergarciaescobedo.es/programacion-en-java/28-programacion-estructurada/114-depuracion-de-programas-con-netbeans>
- ii. Iglesias, C. C. (2020). Entornos de Desarrollo (GRADO SUPERIOR). RA-MA S.A. Editorial y Publicaciones.
- iii. Aldarias, F. (2012): Apuntes de Entornos de Desarrollo, CEEDCV