

MP0485
Programación
UF2. Introducción a Java

2.1. Primeros pasos con Java

Índice

☰	Objetivos	3
☰	Clasificación de los lenguajes de programación	4
☰	Características del lenguaje Java	6
☰	La plataforma Java	8
☰	Instalación del kit de desarrollo Java	9
☰	Configuración de las variables de entorno	12
☰	Crear el primer programa Java ("Hola Mundo")	20
☰	Parámetros en línea de comandos	25
☰	Resumen	28

Objetivos

Con esta unidad perseguimos los siguientes objetivos:

- 1 Conocer las características principales del lenguaje de programación Java.
- 2 Conocer el concepto de "plataforma Java" como el entorno de ejecución en el que corren las aplicaciones Java.
- 3 Descargar e instalar el kit de desarrollo de Java.
- 4 Crear el primer programa Java con los mínimos recursos: el bloc de notas y el kit de desarrollo de Java.
- 5 Conocer el mecanismo de paso de argumentos por línea de comandos.

¡Ánimo y adelante!

Clasificación de los lenguajes de programación

Antes de ponernos a hablar de Java queremos que conozcas algunas características de los lenguajes de programación en general para que puedas encuadrar Java dentro de un contexto más amplio.

Paradigma de programación

Un paradigma de programación es un conjunto de técnicas y estándares que podemos utilizar para desarrollar nuestras aplicaciones. Cada lenguaje de programación nos permite trabajar en base a uno o varios de los paradigmas. Estos son los tres paradigmas más importantes:

Paradigma de programación estructurada

Es el que basa la creación de programas en las tres estructuras básicas: secuencial, alternativa y repetitiva.

Paradigma de programación modular

Es el que divide un programa complejo en varios programas más pequeños o módulos. Este paradigma no es incompatible con el paradigma de la programación estructurada, ya que cada módulo puede estar basado en las tres estructuras básicas que dicta el paradigma de la programación estructurada.

Paradigma de orientación a objetos

Utiliza métodos de análisis que examinan los requerimientos del sistema desde la perspectiva de objetos cooperantes extraídos directamente del vocabulario original del problema. Por ejemplo, un programa que tiene como finalidad comprobar el resultado de un partido de fútbol tendría un funcionamiento basado en un objeto *partido*, dos objetos *equipo* y varios objetos *jugador*. Cada objeto encapsularía una funcionalidad. La programación orientada a objetos es también modular.

Intérpretes, compiladores y montadores

Un ordenador solamente es capaz de comprender información binaria, es decir, una codificación basada en conjuntos de valores 0 y 1. El ordenador, compuesto de circuitos, solo puede trabajar con dos estados; entrada de corriente (representado con un 1) y falta de corriente (representado con un 0).

Cuando escribimos un programa con ayuda de un lenguaje de programación, escribimos lo que se llama **código fuente**. Este código fuente tiene que ser convertido a código binario (a base de ceros y unos) para que pueda ser ejecutado por el ordenador. El modo en el que se realiza esta conversión da lugar a tres tipos de lenguajes:

1

Lenguajes interpretados: el código fuente es interpretado línea a línea a la vez que va siendo ejecutado. Este proceso se repite cada vez que se ejecuta dicho programa. Es el caso de JavaScript, donde el navegador web se encarga de interpretar cada línea de código, traducirla a código máquina y ejecutarla.

2

Lenguajes compilados: el código fuente ha de ser convertido íntegramente a código máquina, generándose otro fichero denominado **código objeto**. Este proceso se denomina **compilación**. Es el caso del lenguaje C.

3

Lenguajes compilados e interpretados a la vez: este es el caso de Java, cuyos programas fuentes deben ser compilados, pero no para crear código máquina directamente, sino para crear un código que queda a medio camino entre el código fuente y el código máquina, al que denominamos **bytecode** (o **bycode**). El **bytecode** es un tipo de codificación parecida al ensamblador que deberá ser interpretada por una **máquina virtual** línea a línea según se vaya ejecutando.



Por otro lado, es importante saber que una aplicación puede estar formada por varios programas que han sido compilados por separado y es necesaria alguna herramienta que une todas las piezas para crear lo que se denomina un **programa ejecutable**. Esta es labor de los programas **montadores o linkadores**.

Características del lenguaje Java

En este apartado veremos las principales características del lenguaje de programación Java.

1

Es un lenguaje de propósito general, lo que significa que nos permite desarrollar cualquier tipo de aplicación. En cuanto a la interfaz de usuario, podemos desarrollar programas de escritorio a base de ventanas, aplicaciones para la web a las que el usuario accede con su navegador, aplicaciones para ejecutar desde el símbolo del sistema, etc. Con Java podemos desarrollar desde programas sencillos hasta grandes proyectos empresariales que acceden a bases de datos y a recursos distribuidos.

2

Fue comercializado por primera vez en 1995 por Sun Microsystems, actualmente perteneciente a Oracle Corporation. La última versión es "Java 9 Standard Edition" y salió en **septiembre de 2017**.

3

Es un lenguaje orientado a objetos. Toda instrucción que escribimos tiene que estar encerrada dentro de una clase. Cada clase es como un modelo o plantilla para la construcción de objetos de dicha clase.

4

Es un lenguaje medio compilado y medio interpretado. Compilado porque los ficheros fuente que escribimos deben ser compilados para obtener código *bytecode*, un código intermedio que todavía no llega a ser código máquina. Tras la compilación se obtiene un fichero con extensión *.class* que será interpretado por la máquina virtual de Java.

5

Su sintaxis está basada en el lenguaje C++, pero se han suprimido las características más complejas, de modo que su curva de aprendizaje es menor.

6

Hay disponibles muchas librerías. Las librerías son componentes software reutilizables que podemos emplear en nuestras aplicaciones.

7

Admite aplicaciones distribuidas, ya que nos aporta la infraestructura necesaria para desarrollar aplicaciones en red con acceso a recursos remotos.

8

Posee su propio mecanismo para la gestión del uso de la memoria denominado **recolector de basura**.

9

Es **independiente de la plataforma** gracias a que el código fuente Java es compilado creando un código intermedio (el *bytecode*), que es independiente de la plataforma. Podemos ejecutar la misma aplicación Java en entornos muy diferentes como un Mac, Windows, Linux, etc. Cada entorno debe disponer de una máquina virtual de Java adaptada, que será la encargada de interpretar y ejecutar el *bytecode*.

La plataforma Java

Llamamos plataforma Java a la infraestructura mínima necesaria para poder ejecutar aplicaciones Java.

Esta infraestructura necesitará dos componentes:

1

La máquina virtual de Java (JVM): funciona como un procesador virtual capaz de interpretar y ejecutar *bytecode* (código que resulta de la compilación de código fuente Java). La JVM se encarga de traducir el *bytecode* en código máquina nativo compatible con cada plataforma, es decir, si el programa ha de ejecutarse en un Mac, la JVM se encargará de ir traduciendo a código máquina para Mac y ejecutando el resultado de la conversión.

2

Conjunto de librerías básicas: se trata de las bibliotecas o módulos que contienen los componentes necesarios (clases) para que nuestros programas funcionen.

Esta infraestructura o plataforma Java también se denomina **JRE (Java Runtime Environment)**.

Instalación del kit de desarrollo Java

El kit de desarrollo de Java está compuesto por la JVM (máquina virtual de Java) y el conjunto de herramientas que necesitan los programadores para elaborar sus programas: compilador, analizador sintáctico, depurador de código, etc.

El **JDK**, del inglés *Java Development Kit* (kit de desarrollo de Java), es el software de Oracle que un programador necesita para comenzar a desarrollar aplicaciones Java.

No hay que confundir el **JRE** (*Java Runtime Environment*) con el JDK. El JRE es el entorno de ejecución para aplicaciones Java, compuesto por la máquina virtual de Java y el conjunto de librerías y componentes necesarios para ejecutar un programa Java, lo que en el apartado anterior llamamos "La plataforma Java".

El JRE es lo único que se necesita si solo vamos a ejecutar programas Java y no vamos a desarrollar nuestras propias aplicaciones. El JRE es más pequeño que el JDK, ya que el JDK incluye el JRE y otras herramientas de desarrollo.

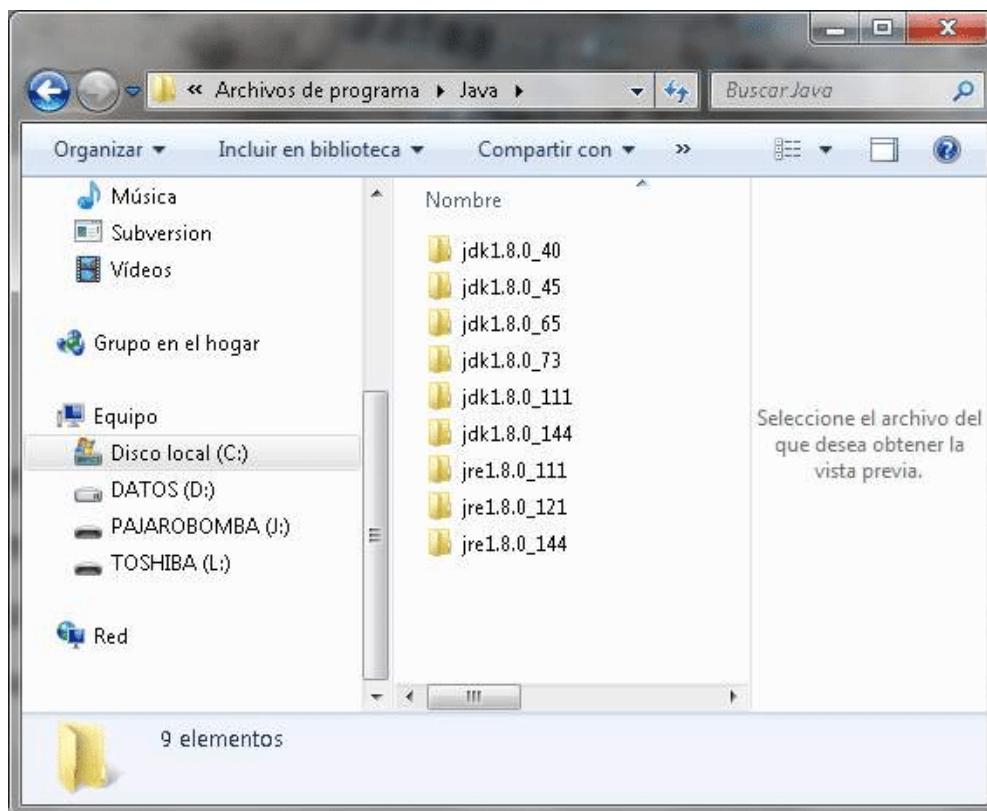
Puedes obtener la versión 8 del JDK gratuitamente en la [siguiente web](#).

Recomendamos en este curso la versión 8 porque es más estable. La versión 9 es demasiado moderna.

En el [siguiente vídeo](#) puedes ver paso a paso cómo instalar en tu equipo el JDK.

Puedes comprobar si has completado todos los pasos de la instalación dirigiéndote mediante el explorador de archivos a la ruta "C:\Program Files\Java". Esta ruta puede variar de un equipo a otro. Dentro de la carpeta Java tendremos una subcarpeta para cada versión de Java que hayamos instalado.

Es posible incluso que tengas instaladas en tu equipo varias versiones de Java. La siguiente imagen muestra la carpeta Java de un equipo que tiene varias versiones instaladas.



Carpeta Java.

Por la imagen anterior comprobamos que el equipo tiene 6 versiones distintas del kit de desarrollo, que corresponden a las carpetas cuyo nombre comienza por *jdk*. También observamos que tiene 3 versiones de JRE (Java Runtime Environment).

i CURIOSIDAD: las aplicaciones que están dentro de la carpeta *C:\Program Files* (Archivos de programa) son aplicaciones de 64 bits y las aplicaciones que están dentro de la carpeta *C:\Program Files x86* (Archivos de programas x86) son aplicaciones de 32 bits compatibles con el antiguo Windows XP. Es posible que incluso tengas otra carpeta Java con más versiones en tu carpeta *C:\Program Files x86*.

Y con tanta versión de Java, ¿cuál de ellas ejecutan nuestros programas?

En los siguientes apartados aprenderás a configurar el sistema para decidir qué versión de Java es la activa.

Configuración de las variables del entorno

Ahora vamos a configurar nuestro equipo para decidir, entre otras cosas, con qué versión de Java se ejecutarán nuestros programas.

¿Qué son las variables de entorno?

Son valores de configuración del sistema en formato de texto al que pueden acceder determinados programas para saber cómo comportarse en determinadas circunstancias.

Para comenzar a trabajar con Java necesitamos configurar tres variables de entorno:

PATH

Esta variable con seguridad ya existe, ya que es la que contiene las rutas donde el intérprete de comandos (símbolo del sistema) debe buscar los programas a ejecutar.

JAVA_HOME

Esta variable de entorno contiene la ruta donde se encuentra la versión de Java en uso. La que contiene la ruta al JRE que ejecutará nuestros programas.

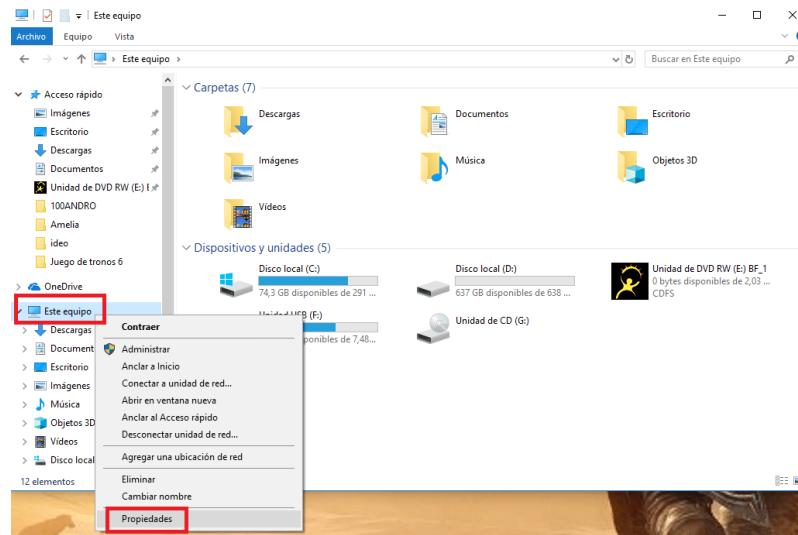
CLASSPATH

Ruta donde se encuentran nuestras clases Java, es decir, nuestros programas.

Sigue estos pasos para configurar la variable de entorno Path

1. Acceder a las propiedades del equipo (con Windows 10)

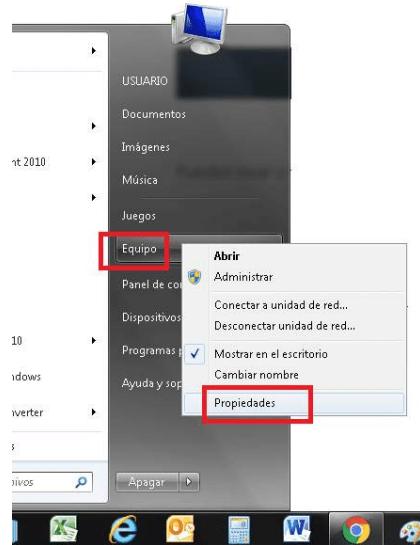
Si trabajas con Windows 10, para acceder a las propiedades del equipo puedes abrir el explorador de archivos y a la izquierda, en la lista de unidades y carpetas, seleccionas "Este equipo", luego haces clic derecho y seleccionas **Propiedades** en el menú contextual.



1. Acceder a las propiedades del equipo (con Windows 7)

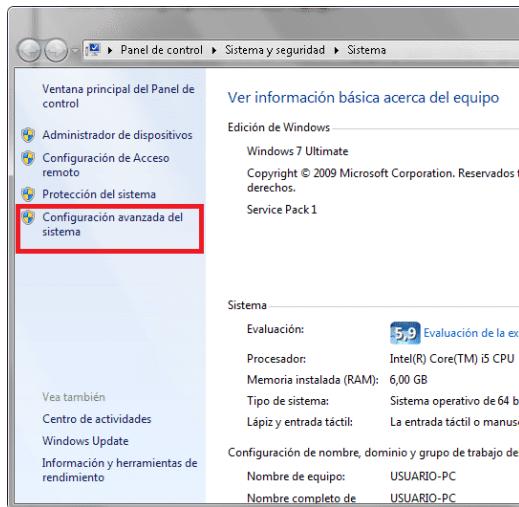
Si trabajas con Windows 7, para acceder a las propiedades del equipo abre el menú **Inicio**, haz clic derecho en **Equipo** y selecciona **Propiedades** en el menú contextual.

Dependiendo de la versión de Windows que utilices el acceso a las propiedades del equipo será diferente. Si utilizas una versión de Windows distinta a Windows 10 o Windows 7, tendrás que buscar en Internet cómo acceder a las propiedades del equipo.



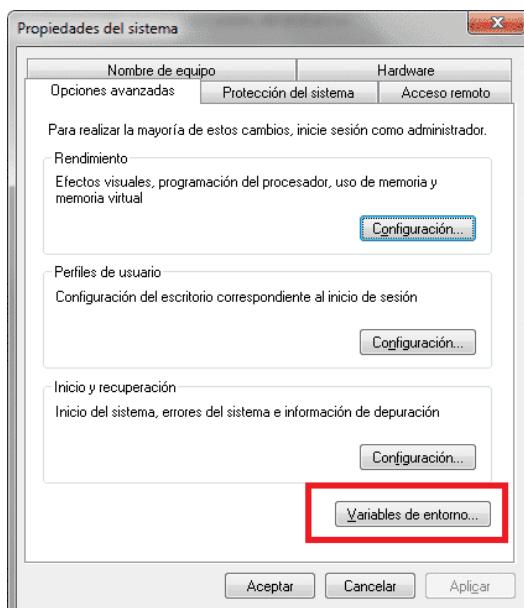
2. Configuración avanzada del sistema

Haz clic en el vínculo de la izquierda "Configuración avanzada del sistema".



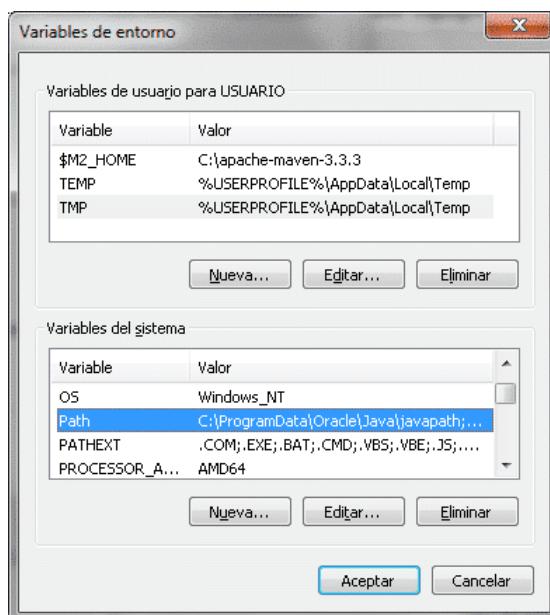
3. Variables de entorno

Haz clic en el botón "Variables de entorno...".



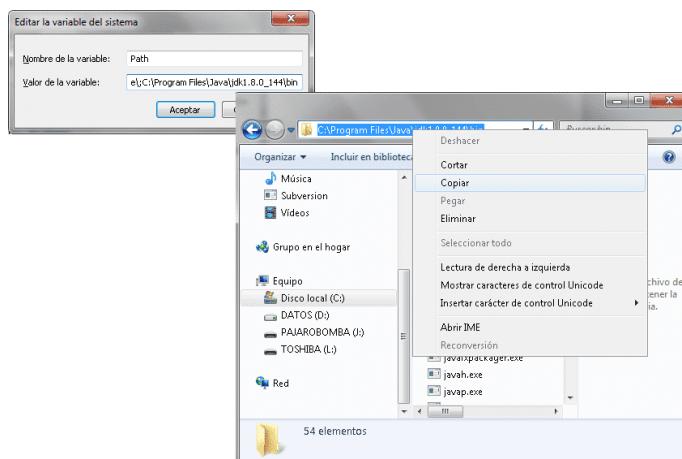
4. Variables del sistema: Path

Existen dos grupos de variables de entorno: las variables de usuario, solo para la sesión de un usuario concreto, y las variables del sistema, para todos los usuarios. Ahora vamos a modificar la variable de entorno **Path**, así que debes editarla donde se encuentre. En el caso de la imagen ya existe una variable *Path* para el sistema, así que **respetaremos el texto que contiene y escribiremos al final**. Debes hacer doble clic sobre el nombre de la variable *Path* para editarla.



5. Escribe la nueva ruta en la variable Path

Escribe la ruta hasta la carpeta bin de la nueva instalación de Java. Para no equivocarte, lo mejor es que te sitúes por medio del explorador de archivos de Windows en la carpeta *bin*. En la barra de dirección seleccionas la ruta, haces clic derecho y seleccionas copiar en el menú contextual. Luego escribe **punto y coma** al final del valor de la variable *Path* y pega (CTRL+V) el texto que antes has copiado.

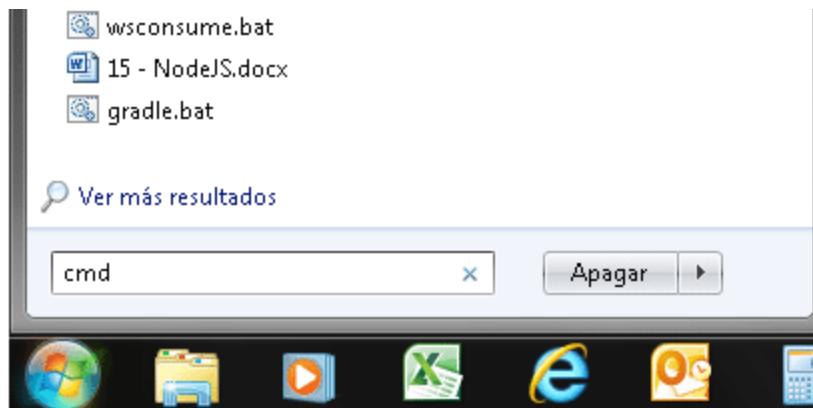


(i) IMPORTANTE: si trabajas con un sistema operativo distinto de Windows, por ejemplo Linux o Mac, tendrás que investigar cómo se accede a las variables de entorno, ya que para cada sistema operativo el acceso a la configuración del sistema es diferente.

Con estos pasos has logrado que el "símbolo de sistema" sea capaz de reconocer desde cualquier ubicación los programas situados en el kit de desarrollo de Java. Para ponerlo en práctica puedes comenzar por un sencillo comando que te dice la versión de Java activa. Sigue estos pasos:

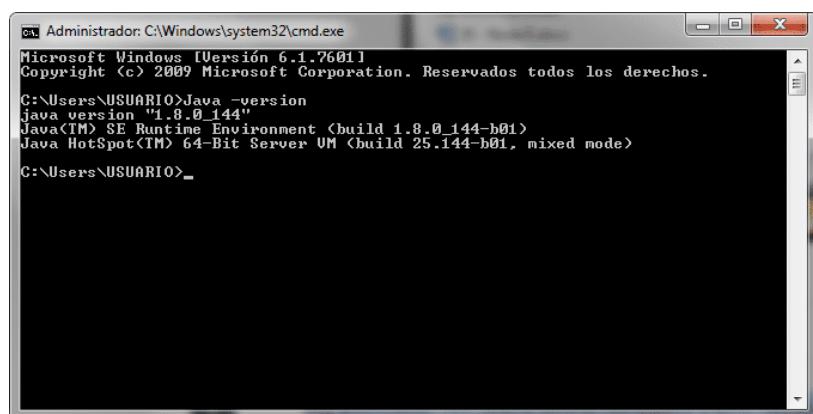
1

En el cuadro "Buscar programas y archivos" escribe cmd para ir al símbolo del sistema y pulsa enter.



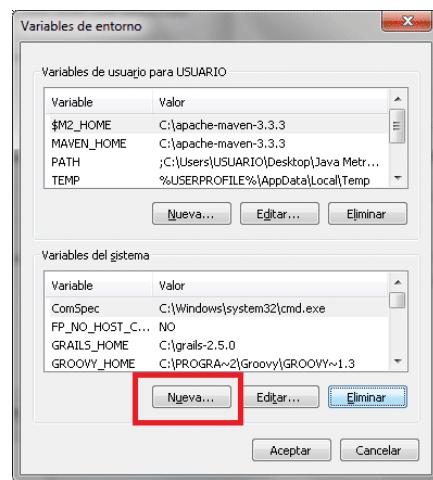
2

En la línea de comandos escribe Java -version, como puedes ver en la imagen:



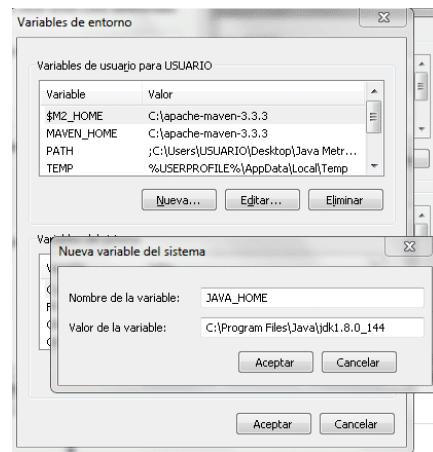
Sigue los pasos para configurar la variable JAVA_HOME

Recuerda que para acceder tenías que ir seleccionando: *Menú inicio / clic derecho en Equipo / Configuración avanzada del sistema / Variables de entorno*. Luego tendrás que hacer clic en el botón "Nueva" para crear una nueva variable del sistema. También puedes crear la variable como variable para el usuario.



2. Crea la variable JAVA_HOME

Una vez que has pulsado el botón "Nuevo" para crear la nueva variable, se abrirá un pequeño cuadro de diálogo, donde debes escribir el nombre de la variable (JAVA_HOME) y el valor de la variable, donde escribirás la ruta hasta la instalación de Java, tal como puedes apreciar en la imagen. Por último, pulsa el botón *Aceptar* y ve cerrando todas las ventanas que se han ido abriendo.

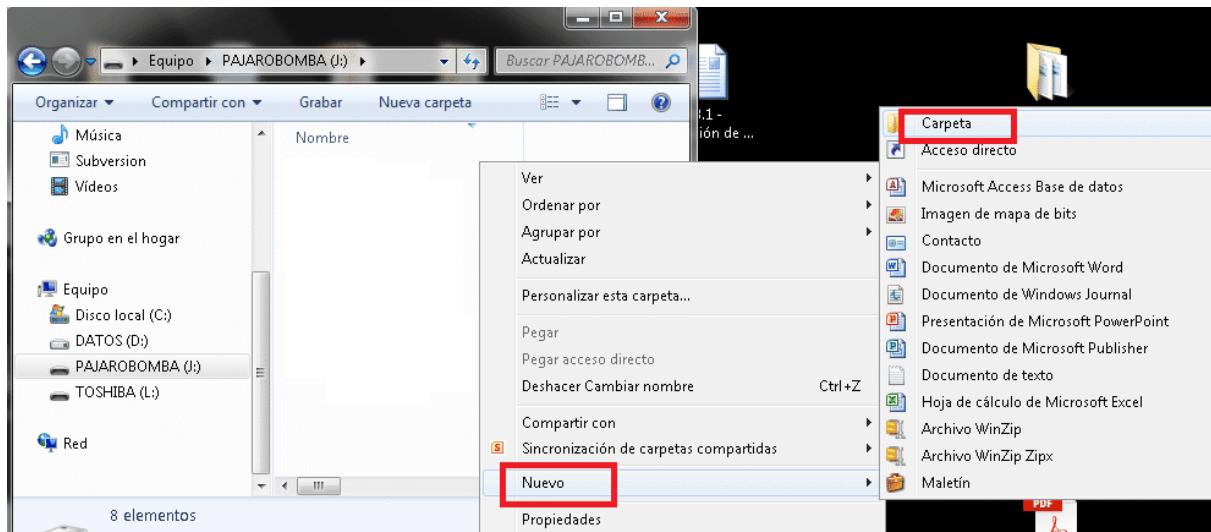


Sigue estos pasos para configurar la variable CLASSPATH

1. Decide cuál será la ruta donde guardarás tus ficheros Java

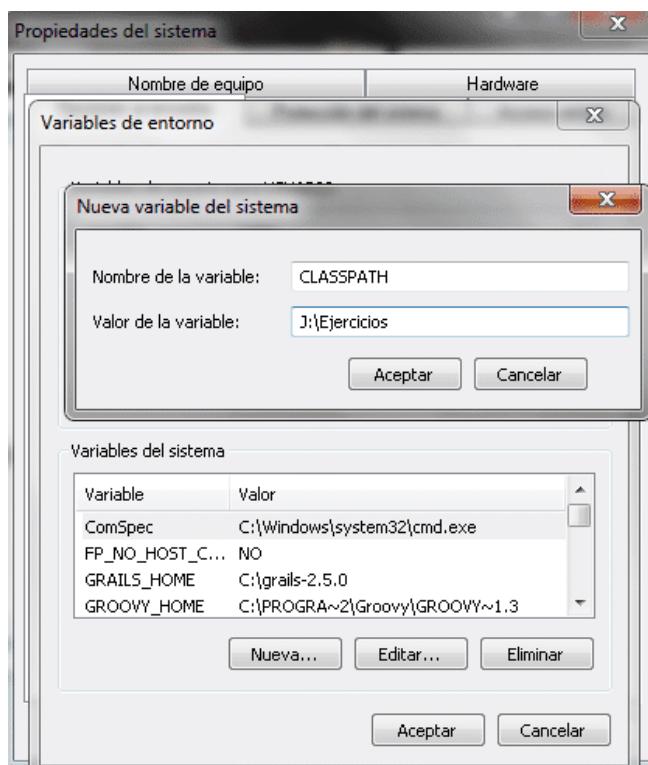
La variable de entorno *CLASSPATH* debe apuntar a la ruta donde guardarás tus ficheros de código fuente Java, que serán ficheros con extensión *.java*. Antes de nada tienes que decidir cuál será esa ruta. Para este ejemplo utilizaremos una carpeta llamada *Ejercicios*, que hemos creado en una memoria USB, que en nuestro sistema es la unidad J. De modo que nuestro *CLASSPATH* debería apuntar a la ruta *J:\Ejercicios*.

Si te sitúas en la unidad J (o la unidad que desees), con ayuda del explorador de archivos puedes hacer **clic derecho** y seleccionar **Nuevo / Carpeta** en el menú contextual.



2. Configura la variable de entorno CLASSPATH

Tal como has aprendido anteriormente, accede a las variables de entorno y configura la variable *CLASSPATH* con la ruta donde guardarás tu código fuente Java.



Es muy posible que tengas que reiniciar el equipo para que las variables de entorno tengan efecto. Si creaste variables de usuario, puede que sea suficiente con iniciar sesión.

Crear el primer programa Java ("Hola Mundo")

En este apartado crearás tu primer programa Java. Por el momento el clásico programa "Hola Mundo".

Para este primer programa no debes preocuparte por entender todo el código, el objetivo es que aprendas a utilizar el "kit de desarrollo de Java" para compilar y ejecutar.

Sigue los pasos que se detallan a continuación:

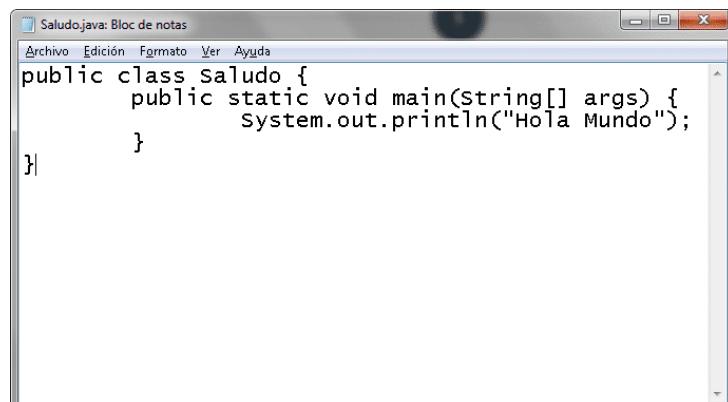
1

Escribe el código fuente

Con ayuda de cualquier editor de textos, por ejemplo el bloc de notas de Windows, escribe el siguiente código y guárdalo en `J:\Ejercicios` o en la ruta que hayas elegido como carpeta de trabajo. El nombre del archivo será `Saludo.java`.

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

Programa Java "Hola Mundo".



Bloc de notas.

Observa que la primera línea de código pone "public class Saludo". *Saludo* es el nombre de la clase Java que estás creando y es complemento obligatorio que el fichero que crees se llame *Saludo.java*, es decir, el nombre de la clase debe coincidir con el nombre del archivo. Respeta incluso que la *S* de *Saludo* va en mayúsculas y el resto en minúsculas.

Recuerda que poco a poco irás comprendiendo todo el código. Ahora el objetivo es que puedas compilar y ejecutar.

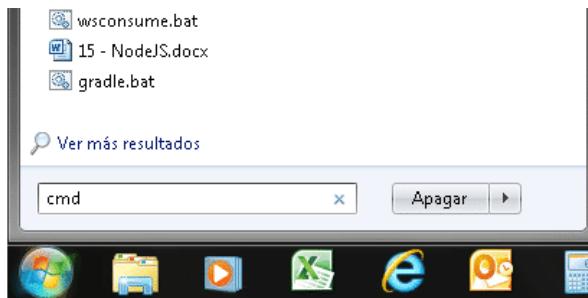


IMPORTANTE: has de saber que Java es "**case sensitive**", lo que significa que distingue entre mayúsculas y minúsculas. Debes escribir tu código exactamente igual que aparece en la imagen. En la próxima unidad se darán pautas a la hora de asignar nombre a variable, constantes, métodos, etc.

2

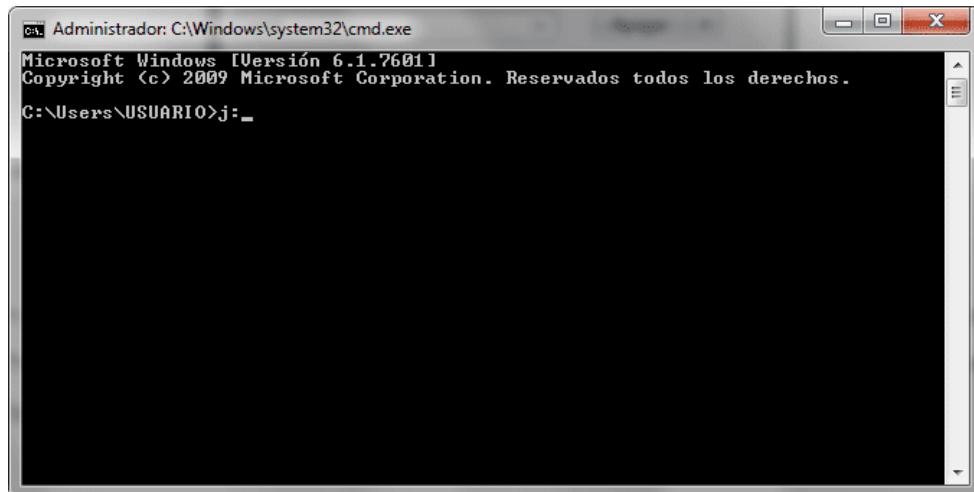
Compila el código fuente desde el símbolo del sistema

Escribe *cmd* en el cuadro de búsqueda de archivos y carpetas para acceder al símbolo del sistema.



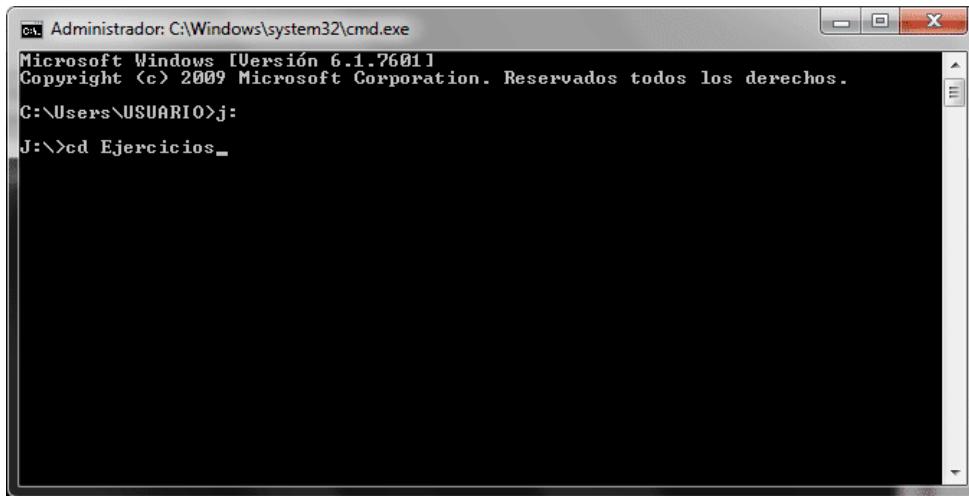
Cuadro de búsqueda de archivos y carpetas.

En la ventana de símbolo del sistema cambia a la unidad de disco que hayas elegido para guardar tus archivos Java. Para el ejemplo yo utilizaré la unidad *J:*, así que escribiré *J:* en la línea de comando y pulsaré *enter*.



Símbolo del sistema.

Luego escribe "cd Ejercicios" o la ruta que deseas y pulsa *enter*. Cd es un comando que permite cambiar la carpeta en uso.

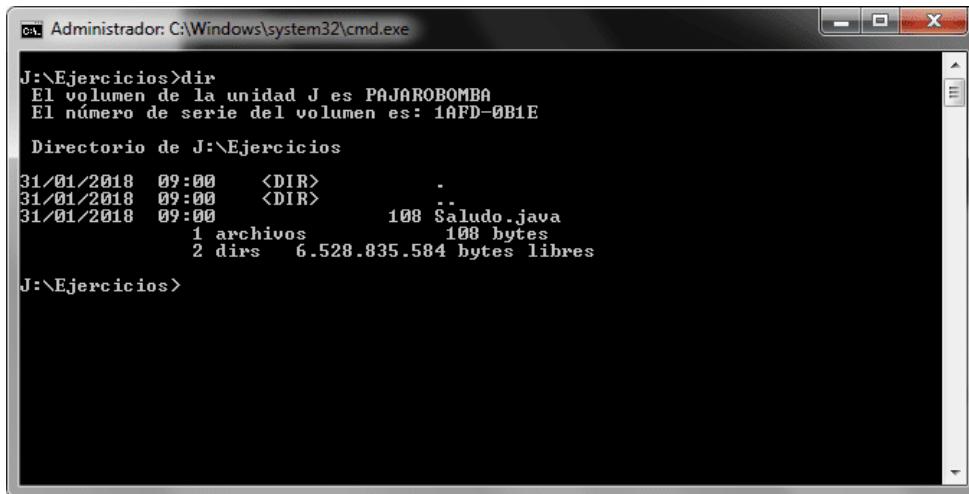


```
ca Administrador: C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 6.1.7601]
Copyright <c> 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\USUARIO>j:
J:\>cd Ejercicios
```

Cambio de ruta o carpeta.

Ahora estás en la carpeta *Ejercicios*. Puedes ejecutar el comando "Dir" para realizar un listado de los archivos y comprobarás que tu archivo *Saludo.java* se encuentra allí.



```
ca Administrador: C:\Windows\system32\cmd.exe
J:\Ejercicios>dir
El volumen de la unidad J es PAJAROBOMBA
El número de serie del volumen es: 1AFD-0B1E

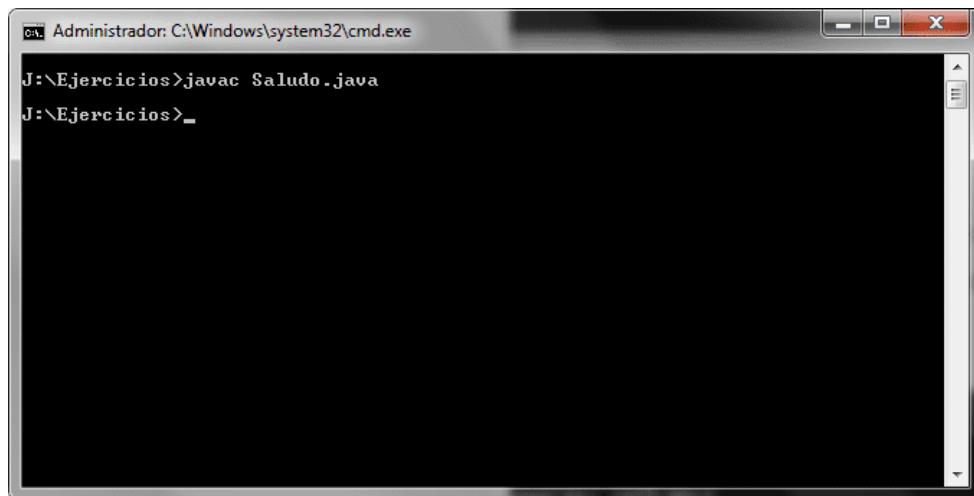
Directorio de J:\Ejercicios

31/01/2018  09:00    <DIR>      .
31/01/2018  09:00    <DIR>      ..
31/01/2018  09:00           108 Saludo.java
                           1 archivos        108 bytes
                           2 dirs   6.528.835.584 bytes libres

J:\Ejercicios>
```

Uso del comando Dir.

Ya está todo preparado para que puedas compilar tu primer programa. Para compilar debes escribir "*javac Saludo.java*" en la línea de comandos. Javac ejecuta el compilador de Java.

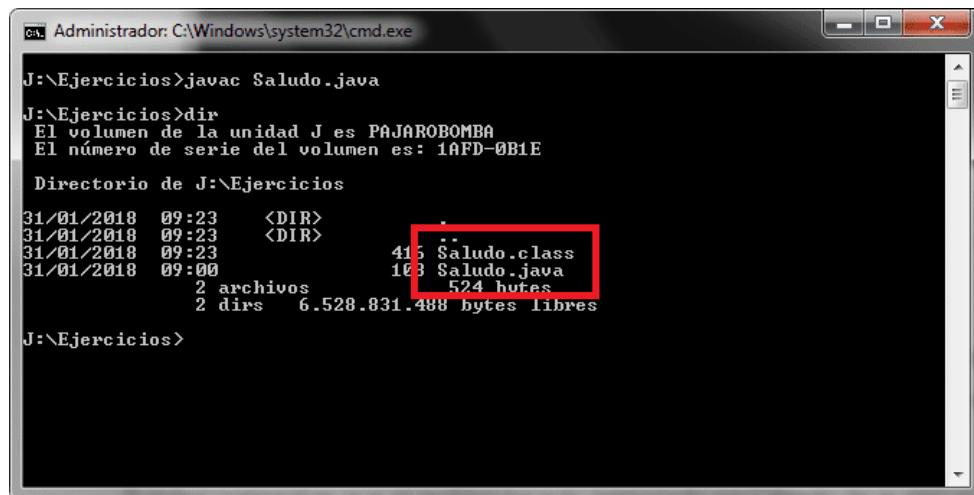


```
Administrator: C:\Windows\system32\cmd.exe
J:\Ejercicios>javac Saludo.java
J:\Ejercicios>
```

Usando el compilador de Java (*javac*).

Si no aparece ningún mensaje como respuesta será que todo ha salido bien y el programa se ha compilado sin errores, de lo contrario aparecerá un mensaje de error en pantalla.

Puedes comprobar que el archivo ha sido compilado ejecutando de nuevo el comando *Dir* para ver la lista de archivos. El fichero compilado (en formato *bytecode*) tendrá extensión *.class*.



```
Administrator: C:\Windows\system32\cmd.exe
J:\Ejercicios>javac Saludo.java
J:\Ejercicios>dir
El volumen de la unidad J es PAJAROBOMBA
El n mero de serie del volumen es: 1AFD-0B1E
Directorio de J:\Ejercicios

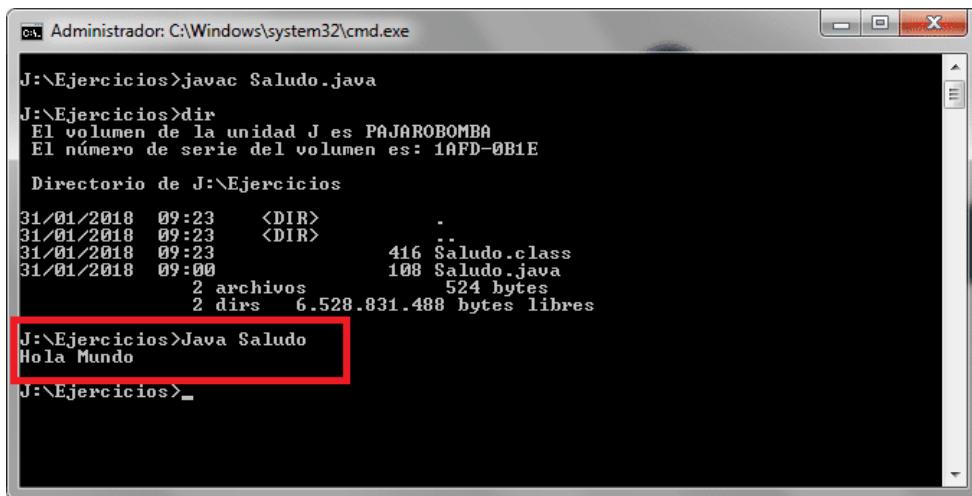
31/01/2018 09:23    <DIR>
31/01/2018 09:23    <DIR>
31/01/2018 09:23              .
31/01/2018 09:00          415 Saludo.class
31/01/2018 09:00          103 Saludo.java
                           524 bytes
                           2 archivos
                           2 dirs   6.528.831.488 bytes libres

J:\Ejercicios>
```

3

Ejecuta tu primer programa

Desde el símbolo del sistema escribe "Java Saludo", como puedes ver en la imagen.



The screenshot shows a Windows Command Prompt window titled "Administrador: C:\Windows\system32\cmd.exe". The command line shows the following sequence of actions:

```
J:\>Ejercicios>javac Saludo.java
J:\>Ejercicios>dir
El volumen de la unidad J es PAJAROBOMBA
El número de serie del volumen es: 1AFD-0B1E
Directorio de J:\>Ejercicios
31/01/2018 09:23 <DIR> .
31/01/2018 09:23 <DIR> ..
31/01/2018 09:23 416 Saludo.class
31/01/2018 09:00 108 Saludo.java
2 archivos 524 bytes
2 dirs 6.528.831.488 bytes libres
J:\>Ejercicios>Java Saludo
Hola Mundo
J:\>Ejercicios>
```

The output "Hola Mundo" is highlighted with a red box.

Ejecutando el primer programa.

Parámetros en línea de comandos

En este apartado vamos a modificar un poco el código de nuestro programa para que reciba un argumento en la línea de comandos.

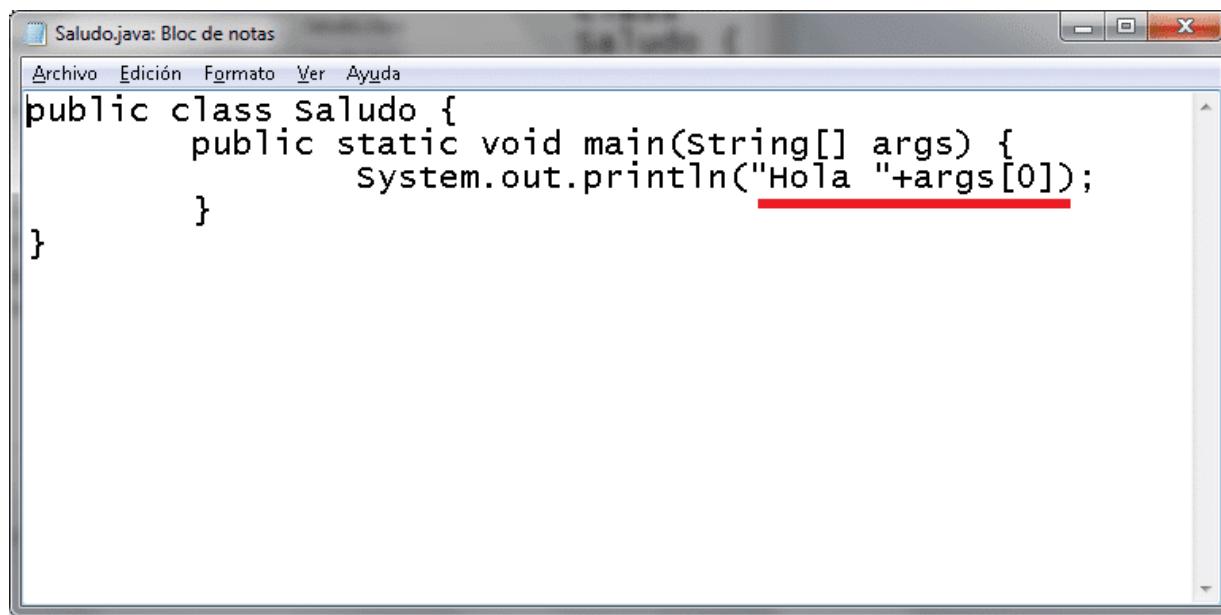
Observa que una de las líneas de código de tu programa es:

```
public static void main(String[] args)
```

Se trata de la cabecera de la función *main*, función principal que arranca el programa. Esta función tiene un parámetro llamado *args* que es un *array* que recibe los argumentos en línea de comandos. El primer argumento estará en *args[0]*, el segundo en *args[1]*, el tercero en *args[2]* y así sucesivamente.

Modificaremos el programa para que en lugar de decir "Hola Mundo" pueda recibir el nombre del usuario en la línea de comandos y lo utilice en la ejecución.

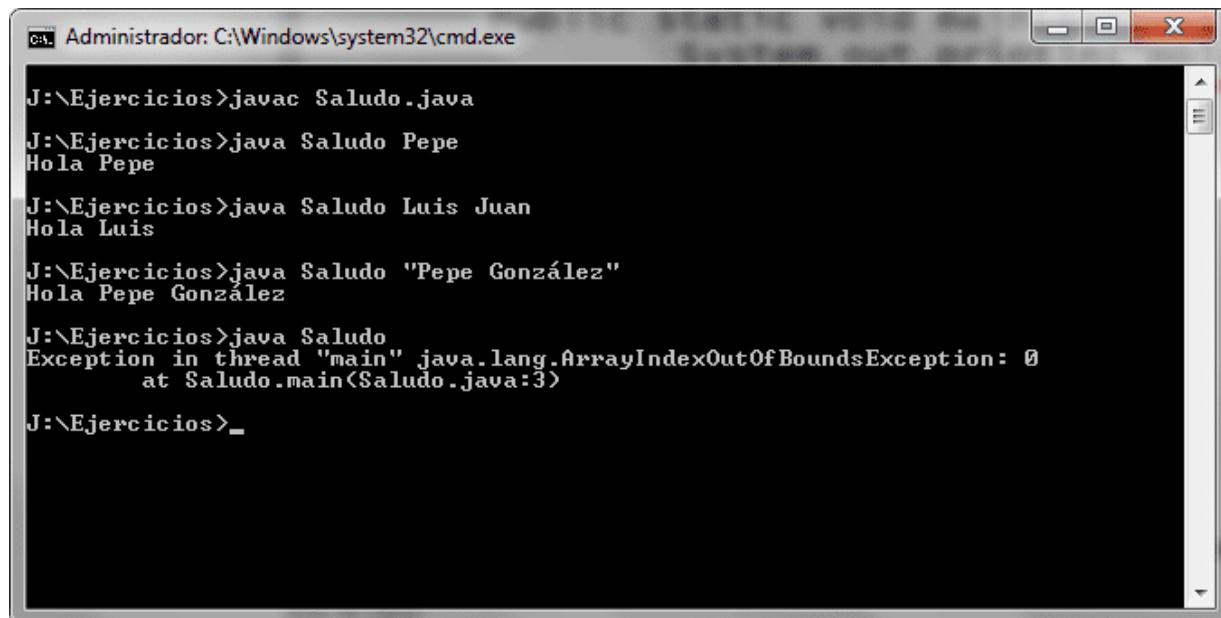
Edita tu archivo y realiza el cambio que ves en la imagen:



```
public class Saludo {  
    public static void main(String[] args) {  
        System.out.println("Hola "+args[0]);  
    }  
}
```

Bloc de notas.

Lo primero será volver a compilar el programa para que los cambios se reflejen en el archivo *Saludo.class*. Luego ejecutaremos varias veces desde la línea de comandos para comprobar el nuevo comportamiento del programa.



```
J:\Ejercicios>javac Saludo.java  
J:\Ejercicios>java Saludo Pepe  
Hola Pepe  
J:\Ejercicios>java Saludo Luis Juan  
Hola Luis  
J:\Ejercicios>java Saludo "Pepe González"  
Hola Pepe González  
J:\Ejercicios>java Saludo  
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 0  
    at Saludo.main(Saludo.java:3)  
J:\Ejercicios>_
```

Ejecución desde el símbolo del sistema.

Observa la anterior imagen. Primero he compilado el programa de nuevo y luego he realizado varias ejecuciones con diferentes argumentos de entrada. Después de "Java Saludo" lo que va son los argumentos. Analicemos lo que ha ocurrido con cada una de las ejecuciones:

- **java Saludo Pepe**

El programa ha recibido el texto *Pepe* como primer argumento (*args[0]*) y lo ha concatenado con la palabra *Hola* para que la respuesta del programa sea "*Hola Pepe*".

- **java Saludo Luis Juan**

El programa ha recibido el texto *Luis* como primer argumento (*args[0]*) y el texto *Juan* como segundo argumento (*args[1]*). Cada palabra es recogida como un argumento diferente. Pero el programa solamente utiliza el primer argumento, de modo que el segundo ha sido ignorado, mostrando como resultado el texto "*Hola Luis*".

- **java Saludo "Pepe González"**

Un texto encerrado entre comillas será recibido por el programa como un solo argumento, de modo que "*Pepe González*" es el valor recibido en *args[0]* y el programa muestra como resultado el texto "*Hola Pepe González*".

- **java Saludo**

En la tercera ejecución no se ha especificado ningún argumento y como el programa intenta acceder al primer argumento (*args[0]*) que no existe, provoca un error en tiempo de ejecución.

El objetivo de este apartado es que comprendas el mecanismo de paso de argumentos durante la ejecución en la línea de comandos. Por ahora no te preocupes de comprender absolutamente todo el código.

Resumen

Has terminado la sesión, vamos a ver los puntos más importantes que hemos tratado.

- El **JRE (Java Runtime Environment)** es el entorno de ejecución para aplicaciones Java. Está compuesto por la máquina virtual de Java y el conjunto de librerías y componentes necesarios para ejecutar un programa Java. Es suficiente si solo vamos a ejecutar programas, pero no es suficiente si vamos a desarrollar nuestras propias aplicaciones Java.
- El **JDK**, del inglés **Java Development Kit** (kit de desarrollo de Java) es el software de Oracle que un programador necesita para comenzar a desarrollar aplicaciones Java. Incluye el JRE y otras herramientas de desarrollo para los programadores.
- Para comenzar a trabajar con Java necesitamos configurar las variables de entorno **PATH**, **JAVA_HOME** y **CLASSPATH**.



PROEDUCA