

El JDK de Java

El JDK de Java está compuesto por una serie de programas para muy variadas acciones relacionadas con el desarrollo. De todos estos programas existentes, a nosotros nos interesan dos en particular: Javac y Java.

Javac es el programa compilador de Java (Javac significa *Java Compiler*) y su función es crear archivos con extensión .class los cuales pueden ser ejecutados por la Java Virtual Machine (JVM). El comando Java ejecuta dichos archivos de extensión .class en la JVM.

Para poder ejecutar estos dos comandos, es necesario en primer lugar, hacer una configuración de la variable PATH. La variable PATH, es una variable de entorno en la cual se especifican las rutas en las cuales el intérprete de comandos (También conocido como consola, shell, terminal, símbolo del sistema o msdos en el caso específico de windows) debe buscar los programas a ejecutar. Esta variable debe contener todos los directorios en los que se quiera que el intérprete busque programas.

Para compilar y ejecutar un programa escrito en Java, es necesario hacerlo desde el intérprete de comandos. Cuando en el intérprete de comandos tecleamos la palabra javac, él irá a buscar los directorios especificados en la variable PATH y buscará si en alguno de esos directorios existe algún programa llamado "javac", cuando lo encuentra, entonces ejecuta el comando dado.

La variable PATH

Para ver el contenido de la variable PATH en windows 10, debemos dar los siguientes pasos:

1. Abre un explorador de windows cualquier.
2. Localiza el icono Mi PC o Este equipo y da clic derecho sobre él.
3. Elige la opción propiedades.
4. En la ventana que abre, elige la opción "Configuración Avanzada del Sistema" que se localiza en el lateral izquierdo.
5. Se abrirá la ventana "Propiedades del sistema". En esta ventana, localiza la pestaña "Opciones Avanzadas".
6. Dentro de la pestaña "Opciones Avanzadas", localiza el botón "Variables de Entorno" que se encuentra hasta abajo y da clic sobre él.
7. Se abrirá la ventana "Variables de entorno". En la primera sección de la ventana, localiza la variable Path y da clic sobre ella.
8. Elige la opción "Editar". Se abrirá una ventana de edición de la variable de entorno.
9. Da clic en el botón "Nuevo" para agregar una nueva ruta a la variable PATH.
10. Localiza la ruta de la carpeta bin del JDK instalado (Usualmente se instala en C:\Program Files\Java\jdkx.x.x\bin. Copia la ruta y ponla en la nueva ruta creada anteriormente.
11. Da clic en el botón "Aceptar" del editor de variables de entorno, luego en la ventana de variables de entorno y finalmente en la ventana de propiedades del sistema.
12. Con esto queda lista la variable de entorno PATH para utilizar los comandos del JDK dentro del intérprete de comandos.

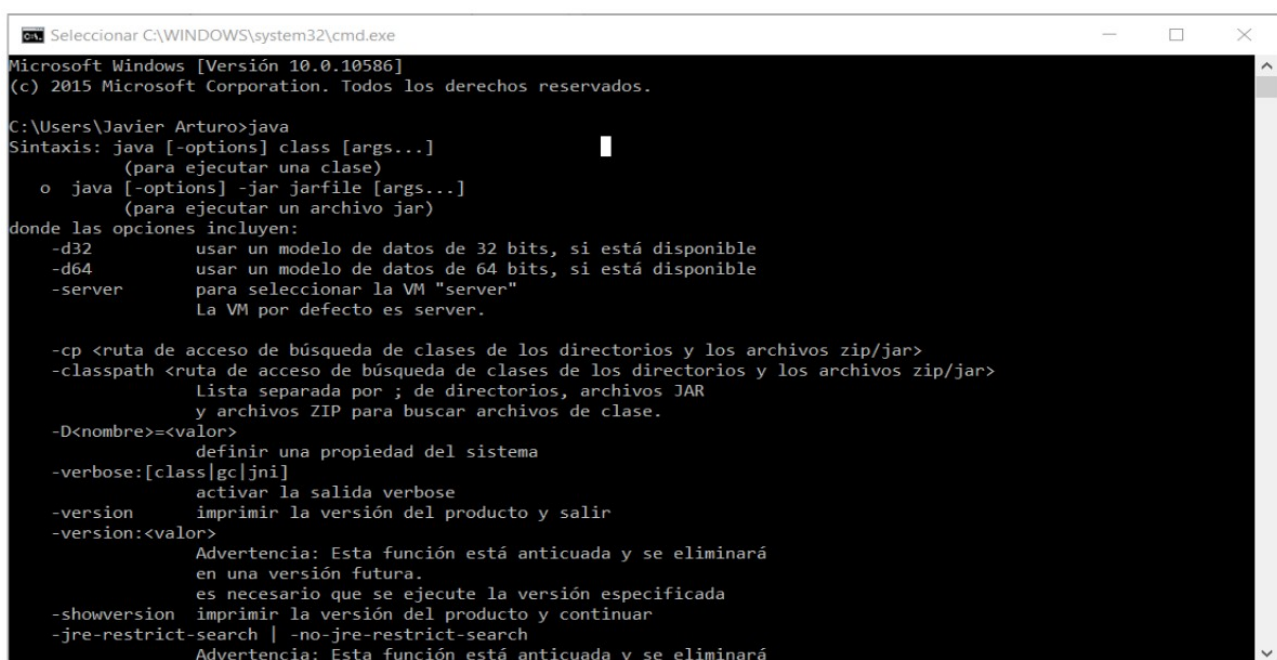
13. Puede que requieras reiniciar tu equipo.

En versiones diferentes de windows 10 puede haber algunas diferencias, por ejemplo, en lugar de crear un nuevo directorio dentro de la variable de entorno mediante el botón nuevo, se debe pegar la ruta de la carpeta bin en el campo de valor de la variable PATH. Recordar que estos valores deben ir siempre separados por punto y coma.

Prueba en el Interprete de comandos.

Para confirmar que la configuración se ha realizado correctamente, abra un intérprete de comandos presionando la tecla windows y la tecla "r" simultáneamente. En el cuadro de diálogo que abre, teclee la palabra "cmd" y de enter. Esto hará que se abra el intérprete de comandos.

Ya en el intérprete de comandos, teclee la palabra "java". El terminal debe dar una apariencia similar a esta.



```
Selecc... C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Versión 10.0.10586]
(c) 2015 Microsoft Corporation. Todos los derechos reservados.

C:\Users\Javier Arturo>java
Sintaxis: java [-options] class [args...]
           (para ejecutar una clase)
    o  java [-options] -jar jarfile [args...]
       (para ejecutar un archivo jar)
donde las opciones incluyen:
    -d32      usar un modelo de datos de 32 bits, si está disponible
    -d64      usar un modelo de datos de 64 bits, si está disponible
    -server   para seleccionar la VM "server"
              La VM por defecto es server.

    -cp <ruta de acceso de búsqueda de clases de los directorios y los archivos zip/jar>
    -classpath <ruta de acceso de búsqueda de clases de los directorios y los archivos zip/jar>
              Lista separada por ; de directorios, archivos JAR
              y archivos ZIP para buscar archivos de clase.
    -D<nombre>=<valor>
              definir una propiedad del sistema
    -verbose:[class|gc|jni]
              activar la salida verbose
    -version   imprimir la versión del producto y salir
    -version:<valor>
              Advertencia: Esta función está anticuada y se eliminará
              en una versión futura.
              es necesario que se ejecute la versión especificada
    -showversion  imprimir la versión del producto y continuar
    -jre-restrict-search | -no-jre-restrict-search
              Advertencia: Esta función está anticuada y se eliminará
```

TIPOS DE DATOS EN JAVA.-

Datos Primitivos y Objetos:

Tipo de dato	Representación	Tamaño (bytes)	Rango de valores	Valor por defecto.	Clase Asociada.
byte	Número entero con signo	1	-128 a 127	0	Byte
short	Número entero con signo	2	-32,768 a 32,767	0	Short
int	Número entero con signo	4	-2147483648 a 2147483647	0	Integer
long	Número entero con signo	8	-9223372036854775808 a 9223372036854775807	0	Long
float	Número en coma flotante de precisión simple	4	$\pm 3.4 \times 10^{-38}$ a $\pm 3.4 \times 10^{38}$	0.0	Float
double	Numérico en coma flotante de precisión doble	8	$\pm 1.8 \times 10^{-308}$ a $\pm 1.8 \times 10^{308}$	0.0	Double
char	Carácter Unicode	2	\u0000 a \uFFFF	\u0000	Character
Boolean	Dato lógico	-	true ó false	false	Boolean

En Java existen dos tipos de variables: Los objetos y las variables primitivas.

Las variables primitivas son entidades elementales: un número, un carácter, un valor verdadero o falso. En los programas en java puede ser necesario tanto el uso de datos elementales como de datos complejos (objetos). Por eso en Java se usa el término "Tipo de datos" para englobar a cualquier cosa que ocupa un espacio de memoria y que puede ir tomando distintos valores o características durante la ejecución del programa.

De manera que en una declaración de variable como la siguiente:

```
double velocidad = 100.0;
```

Debemos entenderla como una variable llamada *velocidad* "**de tipo**" *double*.

Un ejemplo más para que quede claro:

```
int edad = 34;
```

Debemos entenderla como una variable llamada *edad* "**de tipo**" *int*.

Hay ocho tipos de datos primitivos soportados por Java. Los tipos de datos primitivos están predefinidos por el lenguaje (Java) y nombrados por una palabra clave. Veamos ahora en detalle acerca de los ocho tipos de datos primitivos.

byte

- Almacena números enteros.
- Ocupa 1 byte de memoria RAM
- Puede almacenar números enteros de entre -128 hasta 127 (incluyendo el 127).
- Cuando no se asigna un valor, su valor por defecto es 0.
- Ejemplo: byte b = 100

short

- Almacena números enteros.
- Ocupa 2 bytes de memoria RAM
- Puede almacenar números enteros entre -32,768 y 32,767 (incluyéndolo).
- Cuando no se asigna un valor, su valor por defecto es 0.
- Ejemplo: short s = 32,000

int

- Almacena números enteros
- ocupa 4 bytes de ram.
- Puede almacenar enteros entre -2,147,483,648 y 2,147,483,648 (incluyéndolo).
- int suele ocuparse como el tipo predeterminado para almacenar enteros.
- Cuando no se asigna un valor, su valor por defecto es 0.
- Ejemplo: int entero = 5

long

- Almacena números enteros de gran tamaño.
- ocupa 8 bytes de memoria RAM.
- Puede almacenar enteros entre -9,223,372,036,854,775,808 y 9223,372,036,854,775,807.
- No se debe abusar del uso de este tipo de dato primitivo, ya que ocupa gran cantidad de espacio en memoria.
- Cuando no se asigna un valor, su valor por defecto es 0.
- Ejemplo: long p = 235456542213154654.

Float

- Es un dato de coma flotante de precisión simple.
- ocupa 4 bytes de memoria RAM.
- El valor por defecto es de 0.0f
- Ejemplo: float f1 = 234.5f

double

- Es un dato de coma flotante también, pero de mayor precisión.
- Utiliza 8 bytes de memoria RAM.
- Este tipo de datos se utiliza generalmente como el tipo de datos predeterminado para valores decimales.
- El valor por defecto es 0.0d
- Ejemplo: double d1 = 123.4

char

- Char es un caracter unicode de 16 bits.
- El valor mínimo es \u0000)o 0).
- El valor máximo es \uffff.
- Tipo de datos char se utilizar para almacenar cualquier caracter.
- Ejemplo char letra = 'A'

boolean

- boolean representa sólo un bit de información.
- sólo puede tomar dos posibles valores; true y false.
- El valor predeterminado es false.
- Ejemplo: boolean a = true;

Vale la pena hacer una aclaración: Cuando se declara una variable de cualquier tipo, java reserva el espacio en memoria que va a utilizar dependiendo del tipo de dato. No se fija si lo que quieres almacenar es un número muy pequeño o muy grande. Por citar un ejemplo; supongamos que deseas almacenar en una variable, números que no van a pasar de los tres dígitos (989, 543, 443, etc); luego entonces decides almacenar estos números en una variable de tipo long. Además de ser absurdo, también es poco efectivo; ya que long separa 8 bytes por cada variable declarada no

importando si dichas variables van a ocupar sólo una muy pequeña porción del espacio o si van a aprovechar bien toda su capacidad. Por tanto, es conveniente reflexionar sobre el uso que se le dará a tal o cual variable para elegir correctamente su tipo de datos.

Iniciación de variables

Por último, resta hacer una mención respecto de las variables: Una variable puede iniciarse desde su creación o puedo no hacerlo. Ejemplo:

```
int a = 32;  
int a;
```

Ambas declaraciones de variables son permitidas y ambas son correctas dependiendo el propósito para el cual fueron creadas. Por ejemplo, puede crearse una variable destinada a almacenar el dato de edad y no iniciarse al momento de su creación, sino esperar hasta que el usuario de nuestro programa le dé un valor desde su teclado.

Recordemos también que una variable puede "variar" su valor una o muchas veces durante la ejecución de un programa.