Java

1. Instalación NetBeans
2. ¿Que es java?
3. Funcionamiento
4. Tipos de datos
5. Variables y constantes en java

¿Qué es Java?

Es una tecnología pensada para desarrollo de aplicaciones de gran envergadura, altamente escalables, de gran integración con otras tecnologías y sumamente robustas.

Multiplataforma: Significa que su código es portable, es decir se puede transportar por distintas plataformas. De esta manera es posible codificar una única vez una aplicación, y luego ejecutarla sobre cualquier plataforma y/o sistema operativo.

Sintaxis basada en C/C++: Aporta gran simplicidad ya que es una de las formas de escribir código más reconocidas y difundidas, y permite incorporar rápidamente a los programadores que conocen este lenguaje.

Orientado a Objetos: Respeta el paradigma de orientación a objetos, permitiendo utilizar los fundamentos del mismo (Herencia Polimorfismo Abstracción Encapsulamiento)

Manejo automático de memoria: No hay que preocuparse por liberar memoria manualmente ya que un proceso propio de la tecnología se encarga de monitorear, y por consiguiente eliminar el espacio ocupado que no esta siendo utilizado. El proceso encargado de realizar este trabajo se denomina Garbage Collector.

Tampoco debemos confundir Java con JavaScript. El primero es el lenguaje de programación que estudiaremos en esta clase. El segundo es un lenguaje de scripting que permite agregar cierta funcionalidad dinámica en las páginas Web, la similitud de los nombres puede aportar confusión por eso, vale la pena aclarar que JavaScript no tiene nada que ver con Java. Son dos cosas totalmente diferentes

Dentro de la plataforma Java se incluyen tres ediciones:

Java SE o Java Standard Edition: Esta área tiene como objetivo el desarrollo de aplicaciones de escritorio, similares a las aplicaciones tipo ventanas creadas con Visual Basic o Delphi. Incluye la funcionalidad básica del lenguaje como manejo de clases, colecciones, entrada/salida, acceso a base de datos, manejo de sockets, hilos de ejecución, etc. JSE significa Java Standard Edition.

Java EE o Java Enterprise Edition: Esta área tiene como objetivo el desarrollo de aplicaciones empresariales, de gran envergadura. Contempla ambientes web, como los ambientes manejados por servidores de aplicación. Las tecnologías principales incluidas en esta área son Servlets, JSP y EJB, entre otras. JEE significa Java Enterprise Edition.

Java ME o Java Micro Edition: Esta área tiene como objetivo el desarrollo de aplicaciones móviles, tales como GPS, Handhelds (por ejemplo la conocida Palm), celulares y otros dispositivos móviles programables. JME significa Java Micro Edition.


Java
Java SE
Development


Java
ENTERPRISE
EDITION


J2ME

DESARROLLO, COMPILACIÓN Y EJECUCIÓN

Organización

El Java Development Kit (JDK)

El Java Development Kit es el kit de desarrollo propuesto por Sun Microsystems para realizar desarrollos en JAVA. Se puede bajar de forma gratuita de la pagina http://www.java.sun.com

El kit incluye herramientas de desarrollo tales como un compilador, un debugger, un documentador para documentar en forma casi automática una aplicación, un empaquetador para crear archivos de distribución, y otras herramientas mas.

El kit no incluye un entorno de desarrollo interactivo (o IDE) como pueden ser Netbeans, Jdeveloper o Eclipse.

El compilador

El compilador viene incluido como una herramienta dentro de la JDK, en el sistema operativo Windows viene presentado como javac.exe

El compilador transforma los archivos de código fuente de java, es decir los archivos de texto con extensión .java, en archivo compilados, también denominados bytecode. Los archivos compilados tienen la extensión .class, y son archivos binarios.

El Java Runtime Environment (JRE)

Java Runtime Environment es el ambiente de ejecución de Java, y también esta incluido en la JDK. Tiene como componentes más importantes a la Java Virtual Machine y a las class libraries, que son las que contienen las clases base del lenguaje de programación JAVA.

El JRE se distribuye también en forma independiente, es decir sin la JDK, ya que cuando es necesario desplegar una aplicacion hecha en JAVA en el cliente, no es necesario instalarle herramientas que son propias del proceso de desarrollo, como ser el compilador, empaquetador, documentador, y otros.

Sin una JRE instalada no es posible ejecutar una aplicacion construida en JAVA.

En Windows, el comando para invocarlo es el java.exe

La Java Virtual Machine (JVM)

¿Qué es?

La Java Virtual Machine viene incluida dentro de la Java Runtime Environment, y tiene como principal objetivo la ejecución de código JAVA compilado, es decir del archivo .class

La JVM se encarga de interpretar el bytecode y convertirlo a código nativo en tiempo de ejecución, lo cual hace que la ejecución sea un poco más lenta pero garantiza la portabilidad, es decir que el lenguaje sea multiplataforma. De esta manera el código compilado JAVA se puede ejecutar en cualquier plataforma (arquitectura + sistema operativo) que tenga instalada el JRE.

"Write once, run anywhere" es la política desde el primer día de JAVA, es decir construir la aplicación una vez y ejecutarla en “cualquier lado”.

La variable de entorno CLASSPATH

La variable de entorno CLASSPATH se utiliza para referenciar el directorio donde estarán ubicadas todas las clases construidas en JAVA, para que el JRE al ejecutar una clase sepa dónde ubicar el resto de las clases o archivos empaquetados que contienen clases.

Tipos de Datos

Los tipos de datos estándares con los que nos cruzaremos comúnmente en el mundo de la programación son los siguientes:

A continuación, explicaremos las características de cada uno de ellos:

Tipo de dato: Entero

El dato de tipo Entero será un **valor numérico dentro** del conjunto de los números enteros. Algunos ejemplos de valores enteros son: **5, 257, 0, -734, -1, 1.**
Recordemos que no nos interesan las cuestiones físicas de los lenguajes o del límite de memoria, por lo tanto, no pondremos límites a los valores, teniendo disponible el conjunto de números enteros desde -∞ hasta ∞.
En un lenguaje de programación los tipos de datos están acotados ya que ocupan una cantidad finita de espacio en la memoria, pero de este tema nos ocuparemos cuando veamos ejemplos de lenguajes.

Tipo de dato: Real

El dato de tipo Real será de un **valor numérico** que forma parte del conjunto de los números reales. Algunos ejemplos de valores reales son: **2,25; 1429,1456; 0,0 -12,0; -27,34.**

Tipo de dato: Caracter

El valor de tipo carácter está formado por **un único elemento** que puede ser un dígito, una letra o un símbolo de puntuación o de los llamados símbolos especiales, como, por ejemplo, los de pregunta, exclamación, etc. Es decir, que un valor de tipo carácter está limitado por el conjunto de símbolos que se pueden escribir en una computadora. Algunos ejemplos de valores de tipo carácter son: **'a', 'B', '1', ':', '^', ' ', '%', 'F', '!'**
Este tipo de dato es muy útil cuando trabajamos con valores o códigos de una sola letra o dígito.

Tipo de dato: Cadena (String)

Este valor de tipo cadena está formado por un **conjunto de caracteres**.
También se suele llamar cadena de caracteres, pero para simplificar, nosotros lo llamamos cadena o String.
Ejemplos de datos de cadena pueden ser nombres, direcciones y cualquier otro valor que esté representado por palabras o textos, como, por ejemplo: **"Juan", "Av. Rivadavia 950", "Matricula", " ".** Este último dato es lo que conocemos

como una cadena vacía (sin caracteres) y la representamos con dos comillas seguidas.

Tipo de dato: Lógico

Un dato lógico está formado por alguno de los siguientes valores: **TRUE** (VERDADERO) **o FALSE** (FALSO). Estos datos los utilizaremos para realizar operaciones lógicas o para trabajar con condiciones. Por ahora solo lo mencionamos, ya que le dedicaremos bastante tiempo y realizaremos muchos ejercicios para afianzar este tema más adelante.
El tipo de dato lógico también se conoce como Boolean, en inglés.

Números, Cadenas de Texto y Booleanos

En Java existen muchos tipos de datos como vimos en el cuadro anterior, en este momento vamos a focalizarnos en tres de ellos: números, cadenas de caracteres y booleanos.

Los números (0, 1, 2, -100, 3000 y todos los que se te ocurran) son valores que nos van a permitir representar información numérica en nuestros programas.

Por ejemplo si queremos imprimir por consola un numero hacemos el siguiente comando:

System.out.println(10);

Los números te van a servir para representar información cuantitativa (años, temperatura, stock, segundos, etc.) y hacer operaciones matemáticas tales como sumas, restas y multiplicaciones (sólo por mencionar algunas).

Las cadenas de caracteres, también conocidas como String, son valores que vamos a representar por medio de un conjunto de caracteres encerrados entre comillas dobles.

La cadena de caracteres:'soy una cadena de caracteres' la podemos imprimir de la siguiente manera:

System.out.println("hola soy una cadena");

Lo importante cuando trabajamos con cadenas de carácter es no mezclar las comillas dobles con comillas simples, fijense que pasa si ingresamos el siguiente código:

System.out.println("hola soy una cadena');

La consola nos va a mostrar un error ya que al principio estamos usando comillas dobles y luego comillas simples.

Los booleanos representan valores de lógica binaria. Esto quiere decir que solo pueden representar 1 de 2 valores posibles: true (verdadero) o false (falso).

Para indicar que queremos usar el valor verdadero escribimos...

System.out.println(true);

Para indicar que algo es falso escribimos...

System.out.println(false);

Tipos de dato primitivos

int

El tipo de dato int es un tipo de dato numerico y entero, se utiliza para almacenar numeros comprendidos entre -2.147.483.648 y 2.147.483.647. Ocupa 4 bytes de memoria.

float

El tipo de dato float es un tipo de dato numérico y de punto flotante, se utiliza para almacenar números comprendidos entre -3.402823E38 a -1.401298E-45 y de 1.401298E-45 a 3.402823E38. Ocupa 4 bytes de memoria, y maneja entre 6 y 7 cifras decimales.

char

El tipo de dato char se utiliza para almacenar un solo caracter, del tipo Unicode. Ocupan 2 bytes en memoria.

boolean

El tipo de dato boolean se utiliza para almacenar las palabras claves true o false, es decir verdadero o falso. Ocupan 1 byte en memoria.

byte

El tipo de dato byte es un tipo de dato numérico y entero, se utiliza para almacenar números comprendidos entre -128 y 127. Ocupa 1 byte de memoria.

short

El tipo de dato short es un tipo de dato numérico y entero, se utiliza para almacenar números comprendidos entre -32768 y 32767. Ocupa 2 bytes de memoria.

long

El tipo de dato long es un tipo de dato numérico y entero, se utiliza para almacenar números comprendidos entre -9.223.372.036.854.775.808 y 9.223.372.036.854.775.807. Ocupa 8 bytes de memoria.

double

El tipo de dato float es un tipo de dato numérico y de punto flotante, se utiliza para almacenar números comprendidos de 1.79769313486232E308 a -4.94065645841247E-324 y de 4.94065645841247E-324 a 1.79769313486232E308. Ocupa 8 bytes de memoria, y maneja unas 15 cifras decimales.

Variables y constantes Java

Variables

Ahora es el momento de desarrollar los conceptos de programación para trabajar con los datos.

Para eso, recordemos que los datos se almacenan en la memoria, por lo tanto, habrá un espacio dedicado a guardar cada dato.

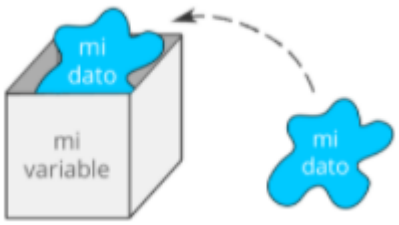
Seguramente estarás pensando que, como todo programa tendrá varios datos, habrá en la memoria varios espacios destinados al almacenamiento de todos los datos. Evidentemente deberá haber alguna forma de no confundir estos espacios y poder trabajar con los datos correctamente. Para evitar esas confusiones utilizaremos un nombre único para ese espacio de memoria a lo largo de todo el programa.

De esta manera podemos definir lo que es una variable:

Una variable es un espacio en memoria, asociado a un nombre lógico y a un valor, con un tipo de dato particular.

Suele ser común relacionar una variable con una caja en donde guardar un valor de un tipo de dato. La caja debe ser lo suficientemente espaciosa para poder contener el valor. Y la memoria, sería un gran fichero en donde guardar elementos con nombre para poder identificarlos.

Gráficamente, podemos imaginarlo así:



Una variable tendrá un nombre y un tipo de dato asociado, es decir, que tendrá un nombre y un valor. Así, tendrás que prestar mucha atención porque es muy fácil confundir el nombre de la variable con su valor.

Como nombramos anteriormente una variable hace referencia a un valor.

Pensemos a una variable como si fuera una caja. A esta caja le vamos a poner un nombre que nos va a permitir acceder a ella cuando lo necesitemos.

Para decirle a Java que vamos a declarar una variable, lo primero que tenemos que hacer es definir el tipo de dato, ejemplo de como definir un tipo de dato entero con el nombre de la variable:

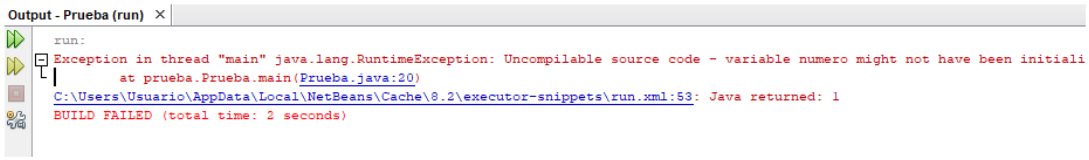
int numero;

A esto se lo conoce como declarar una variable.

Así como los valores, las variables pueden ser mostradas en la consola usando el método

System.out.println(numero);

Pero si lo ejecutamos de esta manera, lo que detectara NetBeans es que esa variable no tiene asignada ningún valor por lo que nos mostrara un error:



Para evitar este erro, el cual nos indica que nuestra variable no tiene un valor asignado, podemos asignarle un valor de tipo entero de la siguiente manera a esa variable:

numero = 10;

De esta manera con el signo igual (=) estamos asignando un valor a la variable llamada número, y ejecutamos el código:

System.out.println(numero);

También podemos hacer ambas cosas, declarar la variable y asignar un valor con el siguiente ejemplo:

int numero2 = 20;

En este caso estamos creando una variable del tipo entero llamada numero2 y asignandole al mismo tiempo el valor 20. Pueden ejecutar el método para imprimir por consola:

System.out.println(numero);

Podemos crear distintos tipos de variales, siempre y cuando definamos cual va a ser y el valor que le pertenece: un número, una cadena de caracteres o un booleano (los únicos tipos de datos que conocemos por el momento).

Entero:

int numero3 = 73;

String nombre = "daniel";

boolean verdad = true;

Con los comandos anteriores estamos declarando variables de distintos tipos y asignando un valor, luego podremos mostrarlas por consola

System.out.println(numero3);

System.out.println(nombre);

System.out.println(verdad);

Lo que podremos hacer con las variables es cambiar su valor (para hacerle honor a su nombre). Esto lo podremos hacer en cualquier momento, pero tiene que ser por el mismo tipo de dato. Una variable puede valer el número 2020 en un momento y luego 128 en otro

int numero4 = 2021

numero4 = 128

System.out.println(numero4);

Como se habrán dado cuenta primero se declaro la variable numero4 con el valor 2021 pero luego se le asigno otro valor 128. Cuando imprimimos por consola la variable nos va a dar el ultimo valor asignado, en este caso 128

Cómo nombrar variables

Usar nombres significativos, que representen el contenido

El nombre puede empezar con cualquier letra, $ ó _

Hay distinción entre mayúsculas y minúsculas (case sensitive)

Por convención se utiliza camelCase

Concepto de Constante

Una Constante, al igual que una Variable, tiene un nombre asociado a un valor de un tipo de dato.

Ahora bien, ¿cuál es la diferencia?

Como su nombre lo indica, una Constante tiene el mismo valor durante toda la ejecución del programa, no se puede cambiar.

Para declarar una constante en Java lo haremos de la siguiente manera, agregando la palabra final al inicio de elegir el tipo de dato:

final int numero3 = 4;

Como se ve en el ejemplo anterior podemos apreciar que se utiliza la palabra reservada final lo cual indica que estamos creando una constante, también es importante asignar un valor al momento de crear la misma.

Ejemplos de constantes son las constantes matemáticas, por ejemplo, PI, e, o algún valor que sabemos que no va a cambiar durante la ejecución del programa.

Tips

Siempre que sea posible, declara tus variables al principio de tus programas.

Siempre que sea posible, asigna un valor a tus variables al momento de declararlas, para evitar que tengan como valor no initializable cuando quieras usarlas.

Usá nombres significativos. Para Java es lo mismo si tu variable se llama n, numero o numerofavorito. Tiene que tener sentido para vos que vas a escribir el código una vez pero leerlo 100 veces.

Respeta los tipos de variables a usar. Si la vas a usar para guardar un número, intenta que siempre tenga un número.

Crea tantas variables como necesites. No hay un límite.