INTRODUCCIÓN A JAVA

Qué es Java?

JAVA es una tecnología pensada para desarrollo de aplicaciones de gran envergadura, altamente escalables, de gran integración con otras tecnologías y muy robustas.

Sus principales características son:

- Lenguaje orientado a objetos: respeta el paradigma de orientación a objetos, permitiendo utilizar los fundamentos del mismo: herencia, polimorfismo, abstracción, encapsulamiento, etc.
- 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.
- Es 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.
- 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 está siendo utilizado. El proceso encargado de realizar este trabajo se denomina Garbage Collector.
- Evolución permanente: la tecnología está en constante evolución debido a la gran cantidad de “consumidores” que poseen, JAVA es uno de los lenguajes más utilizados en el mundo, y SUN pretende estar a la altura de la situación ofreciendo constantemente nuevas entregas.

Organización

La tecnología está organizada en tres grandes áreas bien definidas:

JME (Mobile / Wireless): 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.

JSE (Core / Desktop): 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.

JEE (Enterprise / Server): 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.

La historia

En el año 1990 nace Java, bajo el diseño y la implementación de la empresa Sun Microsystems. El padre-fundador de la tecnología es el James Gosling, a través de una filial dentro de Sun llamada First Person Inc.

Gosling tuvo la visión inicial de construir una lenguaje de programación capaz de ejecutar su código sobre cualquier set de instrucciones, de distintos procesadores. Inicialmente el proyecto apuntó a la programación unificada de distintos electrodomésticos, es decir programar una sola vez y que el programa generado fuera útil para cualquier dispositivo.

El proyecto inicial de Java fue técnicamente un éxito, aunque comercialmente no tuvo el rendimiento esperado, y debió ser relegado unos años.

Aparición de Internet

En el año 1993, Internet da el gran salto, y se convierte de una interfaz textual a una interfaz gráfica.

Java ve una oportunidad y entra fuertemente a Internet con los Applets, pequeños programas construidos en Java – con todos sus beneficios – capaces de ejecutarse dentro de un navegador. Es aquí donde Java comienza a dar sus primeros pasos firmes como lenguaje a difundir masivamente. En el año 1995, el navegador Netscape Navigator comienza formalmente a soportar los Applets Java.

Adicionalmente, el lenguaje podía adaptarse fácilmente a las múltiples plataformas, con lo cual surge una de las primeras aplicaciones multiplataformas más conocidas: WebRunner (hoy HotJava), un navegador multiplataforma construido en Java.

Por qué el nombre JAVA

Inicialmente la intención fue nombrar al lenguaje de programación con el nombre de Oak, pero este ya estaba registrado. La leyenda cuenta que una visita a la cafetería le dio rápida solución al problema.
En las confiterías norteamericanas hay un café denominado Java, en el cual está inspirado el nombre del lenguaje de programación. El logotipo de Java es justamente una taza café.

Siglas

J2ME = Java2 Micro Edition

J2SE = Java2 Standard Edition

J2EE = Java2 Enterprise Edition

JRE = Java Runtime Environment

JVM = Java Virtual Machine

SDK = Software Development Kit

JDK = Java Development Kit

El compilador

Incluido en el JDK

Comando javac.exe

Transforma archivos .java en .class

La Java Virtual Machine (JVM)

"Write once, run anywhere"

No es un compilador, es un intérprete de Java

Archivos .class se denominan bytecodes

Bytecodes: instrucciones de máquina para la JVM

Interpreta el bytecode y lo convierte a código propio del CPU

JRE solo para ejecutar aplicaciones Java

Comando java.exe

IDE a utilizar

IDE = Integrated Development Environment

Netbeans (ultima version)

Links -> Downloads

http://java.sun.com/

http://www.netbeans.org/

SINTAXIS Y SEMÁNTICA DE JAVA

Variables

Introducción

Una variable es un nombre que se asocia con una porción de la memoria del ordenador, en la que se guarda el valor asignado a dicha

variable. Consiste en un elemento al cual le damos un nombre y le atribuimos determinado tipo de información. Las variables pueden ser consideradas como la base de la programación.

Los datos que se manejan en nuestro programa se almacenan en variables. El concepto de variable debe verse como un contenedor de información

Qué es una variable?

Una variable es un nombre que se asocia con una porción de la memoria del ordenador, en la que se guarda el valor asignado a dicha variable.

Consiste en un elemento al cual le damos un nombre y le atribuimos determinado tipo de información.

Las variables pueden ser consideradas como la base de la programación.

Los datos que se manejan en nuestro programa se almacenan en variables. El concepto de variable debe verse como un contenedor de información

De este modo podríamos escribir en un lenguaje ficticio:

a="perro"

b="labra"

La variable que nosotros llamamos "a" posee un elemento de información de tipo texto que es "perro". Asimismo, la variable "b" contiene el valor "ladra".

Podríamos definir una tercera variable que fuese la suma de estas dos:

c=a+b

Si introdujeramos una petición de impresión de esta variable en nuestro lenguaje ficticio:

Imprimir (c)

El resultado podría ser:

Perro ladra

Podríamos de la misma forma trabajar con variables que contuviesen números y construir nuestro programa:

a=3

b=4

c=a+b

Imprimir (c)

El resultado de nuestro programa sería: 7

Hay varios tipos de variables que requieren distintas cantidades de memoria para guardar datos.

Todas las variables han de declararse antes de usarlas, la declaración consiste en una sentencia en la que figura el tipo de dato y el nombre que asignamos a la variable.

Una vez declarada se le podrá asignar valores.

Identificador

Un identificador es un nombre que identifica a una variable, a un método o función miembro, a una clase. Todos los lenguajes tienen ciertas reglas para componer los identificadores.

Definición de variables en Java

Identificador (nombre)

No puede comenzar con un número

puede comenzar con "_" o "$"

no puede utilizar caracteres "%" o "*" o "@" por que están reservados para otras operaciones

Puede incluir, pero no comenzar por un número.

No puede incluir el carácter espacio en blanco.

Distingue entre letras mayúsculas y minúsculas.

No se pueden utilizar las palabras reservadas como identificadores.

Tipos de variables

Utilización de los tipos de datos. Una variable es algo que cambia, o varía. En Java, una variable almacena datos. Los tipos de datos definen el tipo de dato que puede ser almacenado en una variable y los límites de los datos

A toda variable que se use en un programa, se le debe asociar (generalmente al principio del programa) un tipo de dato específico.

Un tipo de dato define todo el posible rango de valores que una variable puede tomar al momento de ejecución del programa y a lo largo de toda su vida útil.

tipo de dato primitivo int, long, float, double, etc.

referencias Objetos o arrays

Ejemplo de declaración y definición

de una variable simple

int var = 5;

de un array

int[] vec = new int[10]; vec[0] = 150; vec[1] = 500;

int vec[] = new int[10]; vec[0] = 150; vec[1] = 500;

int[] vec = {150,500,3,4,5,6};

es un objeto, y tiene una variable miembro denominada length

Tipos de dato primitivos

boolean 1 byte. Valores true y false

char 2 bytes. Unicode. Comprende el código ASCII

byte 1 byte. Valor entero entre -128 y 127

short 2 bytes. Valor entero entre -32768 y 32767

int 4 bytes. Valor entero entre -2.147.483.648 y 2.147.483.647

long 8 bytes. Valor entre -9.223.372.036.854.775.808 y 9.223.372.036.854.775.807

float 4 bytes (entre 6 y 7 cifras decimales equivalentes). De -3.402823E38 a -1.401298E-45 y de 1.401298E-45 a 3.402823E38

double 8 bytes (unas 15 cifras decimales equivalentes). De -1.79769313486232E308 a -4.94065645841247E-324 y de 4.94065645841247E-324 a 1.79769313486232E308

Carácter

En Java los caracteres no están restringidos a los ASCII sino son Unicode.

Un carácter está siempre rodeado de comillas simples como 'A', '9', 'ñ', etc.

El tipo de dato char sirve para guardar estos caracteres.

Boolean

Una variable booleana solamente puede guardar uno de los dos posibles valores: true (verdadero) y false (falso).

Enteros

Una variable entera consiste en cualquier combinación de cifras precedidos por el signo más (opcional), para los positivos, o el signo menos, para los negativos.

Ejemplo: 10, -12, 432, -128

Como ejemplos de declaración de variable enteras tenemos:

Int. Numero = 8451;

Int. x,y;

Int. es la palabra reservada para declarar una variable entera.

En el primer caso, el compilador reserva una porción de 32 bits de memoria en el que guarda el número 8451.

Se accede a dicha porción de memoria mediante el nombre de la variable, número.

Enteros Largos

Son valores enteros que van desde -9.223.372.036.854.775.808 y 9.223.372.036.854.775.807

long x;

long y=50L;

En la primer línea definimos la variable x como long.

En la segunda línea asignamos a la variable “y” el valor 50, que es un número de tipo int. por defecto, le ponemos el sufijo L en mayúsculas o minúsculas para indicar que es de tipo long.

Flotante

Las variables del tipo float o double (coma flotante) se usan para guardar números en memoria que tienen parte entera y parte decimal.

double PI=3.14159;

double g=9.7805, c=2.9979e8;

El primero es una aproximación del número real p, el segundo es la aceleración de la gravedad a nivel del mar, el tercero es la velocidad de la luz en m/s, que es la forma de escribir 2.9979 108. El carácter punto '.', separa la parte entera de la parte decimal, en vez del carácter coma ',' que usamos habitualmente en nuestro idioma.

float x=16.2f;

float y=9f;

double z=21.0;

double g=7d;

Conceptualmente, hay infinitos números de valores entre dos números reales. Ya que los valores de las variables se guardan en un número prefijado de bits, algunos valores no se pueden representar de forma precisa en memoria. Por tanto, los valores de las variables en coma flotante en un ordenador solamente se aproximan a los verdaderos números reales en matemáticas. La aproximación es tanto mejor, cuanto mayor sea el tamaño de la memoria que reservamos para guardarlo. De este hecho, surgen las variables del tipo float y double.

Palabras reservadas

Las palabras reservadas no pueden ser usadas como nombre de variable.

El lenguaje JAVA tiene la siguiente lista de palabras reservadas.

abstract	double	int	strictfp	boolean	else
interface	super	break	extends	long	switch
byte	final	native	synchronized	case	finally
new	this	catch	float	package	throw
char	for	private	throws	class	goto
protected	transient	const	if	public	try
continue	implements	return	void	default	import
short	volatile	do	instanceof	static	while

OPERADORES

Definición

Un operador es una expresión que produce otro valor (así como las funciones o construcciones que devuelven un valor).

Existen operadores de comparación, de negación o de incremento y decremento.

Las operaciones matemáticas se comportan de igual manera en PHP. Las operaciones * y / tienen precedencia sobre la suma y la resta y se pueden utilizar paréntesis para expresiones más complejas.

Un operador es un símbolo especial que indica al compilador que debe efectuar una operación matemática o lógica.

En todo lenguaje de programación existen un conjunto de operadores que permiten realizar operaciones con los datos. Nos referimos a operaciones aritméticas, comparaciones, operaciones lógicas y otras operaciones (por ejemplo, concatenación de cadenas, operaciones algebraicas entre valores números, etc.).

Qué es un Operador?

Un operador es un símbolo especial que indica al compilador que debe efectuar una operación matemática o lógica.

En todo lenguaje de programación existen un conjunto de operadores que permiten realizar operaciones con los datos. Nos referimos a operaciones aritméticas, comparaciones, operaciones lógicas y otras operaciones (por ejemplo, concatenación de cadenas, operaciones algebraicas entre valores números, etc.).

Operadores Aritméticos

suma (+), resta (-), multiplicación (*), división (/)

resto de la división (%)

Operadores lógicos

Los operadores lógicos o también llamados de comparación devuelven siempre verdadero (true) o falso (false). Este tipo de operadores se utilizan con gran frecuencia en todos los lenguajes de programación, para realizar una acción u otra en función de que cierta condición sea verdadera o falsa.

Se utilizan para construir expresiones lógicas, combinando valores lógicos (true y/o false)

En ciertos casos el segundo operando no se evalúa porque ya no es necesario

Operador Utilización Resultado

&&	op1 && op2	true si op1 y op2 son true. Si op1 es false ya no se evalúa op2
||	op1 || op2	true si op1 u op2 son true. Si op1 es true ya no se evalúa op2
!	! op	true si op es false y false si op es true
&	op1 & op2	true si op1 y op2 son true. Siempre se evalúa op2
|	op1 | op2	true si op1 u op2 son true. Siempre se evalúa op2

Operadores de Asignación

Operador	Utilización	Expresión equivalente
=	op1 = op2	op1 = op2
+=	op1 += op2	op1 = op1 + op2
-=	op1 -= op2	op1 = op1 - op2
*=	op1 *= op2	op1 = op1 * op2
/=	op1 /= op2	op1 = op1 / op2
%=	op1 %= op2	op1 = op1 % op2

Operador concatenación de caracteres

Se utiliza para concatenar cadenas de caracteres

Ejemplo "Se han comprado " + variableCantidad + " unidades";

Operador condicional?:

También denominado inline-if

Uso expresionBooleana ? res1 : res2

Evalúa expresionBooleana y devuelve res1 si el resultado es true o res2 si el resultado es false

Ejemplo a=100; b=50; String z = (a<b) ? “a es menor” : “a es mayor”;

Operadores incrementales y decrementales.

incremento (++)

decremento (--)

Operadores relacionales

Sirven para realizar comparaciones de igualdad, desigualdad y relación de menor o mayor.

El resultado de estos operadores es siempre un valor booleano (true or false)

Operador	Utilización	El resultado es true
>	op1 > op2	si op1 es mayor que op2
>=	op1 >= op2	si op1 es mayor o igual que op2
<	op1 < op2	si op1 es menor que op2
<=	op1 <= op2	si op1 es menor o igual que op2
==	op1 == op2	si op1 y op2 son iguales
!=	op1 != op2	si op1 y op2 son diferentes

Operadores aplicables a bits

Operador	Utilización	Resultado
>>	op1 >> op2	Desplaza los bits de op1 a la derecha una distancia op2
<<	op1 << op2	Desplaza los bits de op1 a la izquierda una distancia op2
&	op1 & op2	Operador AND a nivel de bits
|	op1 | op2	Operador OR a nivel de bits
^	op1 ^ op2	Operador XOR a nivel de bits (1 si sólo uno de los operandos es 1)
~	~op2	Operador complemento (invierte el valor de cada bit)

Clasificación

Operadores Unarios

Aquellos que necesitan un único operando

Por ejemplo, el operador incremento (++) o el operador negación (!)

Operadores Binarios

Aquellos que necesitan dos operandos

Por ejemplo, el operador suma (+) o el operador AND (&&)

Operadores Ternarios

Aquellos que necesitan tres operandos

En Java, el único operador ternario es el operador condicional?:

Sentencias

Una sentencia es una orden que se le da al programa para realizar una tarea específica, esta puede ser: mostrar un mensaje en la pantalla, declarar una variable (para reservar espacio en memoria), inicializarla, llamar a una función, etc. Las sentencias acaban con ;

El carácter “;” separa una sentencia de la siguiente. Normalmente, las sentencias se ponen unas debajo de otras, aunque sentencias cortas pueden colocarse en una misma línea. He aquí algunos ejemplos de sentencias

```
int x=10;
import java.util.*;
System.out.println("Hola Mundo");
```

En el lenguaje Java, los caracteres espacio en blanco se pueden emplear libremente.

Es muy importante para la legibilidad de un programa la colocación de unas líneas debajo de otras empleando tabuladores.

El editor del IDE nos ayudará plenamente en esta tarea sin apenas percibirlo.

ASPECTOS ADICIONALES

Bloques de Código

Un bloque de código es un grupo de sentencias que se comportan como una unidad.

Un bloque de código está limitado por las llaves de apertura { y cierre }.

Expresiones

Una expresión es todo aquello que se puede poner a la derecha del operador asignación =. Por ejemplo:

x=123;

y=(x+100)/4;

La primera expresión asigna un valor a la variable x.

La segunda, realiza una operación.

Caracteres especiales

La barra representa el carácter de escape

Utilizados en chars y Strings, por ejemplo:

n Nueva línea

t Tabulador

’ Comilla simple

” Comilla doble

Valores externos

Pasaje de valores desde la línea de comando

Argumentos que puede recibir el método main() de una clase

Se utiliza el arreglo String[] args dentro de método main

Desde Netbeans Properties del proyecto >> Running Project >> Arguments

Valores Constantes

Definición

Es un identificador que expresa un valor fijo.

Su valor no puede ser modificado ni eliminado una vez que se ha definido.

Las constantes son como las variables con la diferencia que no pueden ser redefinidas.

Las constantes no pueden almacenar arreglos de caracteres. Solo valores escalares.

Uso de constantes en Java

Cuando se declara una variable de tipo final, se ha de inicializar y cualquier intento de modificarla en el curso de la ejecución del programa da lugar a un error en tiempo de compilación.

Normalmente, las constantes de un programa se suelen poner en letras mayúsculas, para distinguirlas de las que no son constantes. He aquí ejemplos de declaración de constantes.

final double PI=3.141592653589;

final int IVA MAX_DATOS=21;

Comentarios

Introducción

Un comentario es un texto adicional que se añade al código para explicar su funcionalidad, bien a otras personas que lean el programa, o al propio autor como recordatorio.

Los comentarios son una parte importante de la documentación de un programa.

Los comentarios son ignorados por el compilador, por lo que no incrementan el tamaño del archivo ejecutable; se pueden por tanto, añadir libremente al código para que pueda entenderse mejor.

Es un texto adicional que se añade al código para explicar su funcionalidad, bien a otras personas que lean el programa, o al propio autor como recordatorio. Los comentarios son una parte importante de la documentación de un programa. Los comentarios son ignorados por el

compilador, por lo que no incrementan el tamaño del archivo ejecutable; se pueden por tanto, añadir libremente al código para que pueda entenderse mejor.

Tipos de Comentarios

En línea, por ejemplo

```
int num = 5; // este es el comentario
```

Habitualmente, usaremos comentarios en una sola línea //, ya que no tiene el inconveniente de aprendernos los símbolos de comienzo y terminación del bloque, u olvidarnos de poner este último, dando lugar a un error en el momento de la compilación. En la ventana de edición del Entorno Integrado de Desarrollo (IDE) los comentarios se distinguen del resto del código por el color del texto.

En bloque, ejemplo:

```
/*
Esto es un comentario
 Este es otro comentario
*/
```

Como podemos observar un comentario en varias líneas es un bloque de texto situado entre el símbolo de comienzo del bloque /*, y otro de terminación del mismo */

Para el javadoc se utilizan:

```
/** Acá van los comentarios */
```

Los comentarios de documentación es un bloque de texto situado entre el símbolo de comienzo del bloque /**, y otro de terminación del mismo */. El programa javadoc utiliza estos comentarios para generar la documentación del código.