Notas Java - 1. Introducción

Cristo Alvarado

7 de julio de 2025

Resumen

En esta primera sesión construiremos sobre los fundamentos de Java, enfocándonos en: Orígenes de Java (historia, fundamentos, `Hola Mundo`, el método `main()`) Tipos de datos (primitivos como `int`, `double`, `boolean` y compuestos como `String`), sus características y aplicaciones; Operaciones para manipular valores mediante operadores aritméticos, relacionales y lógicos. Mediante ejemplos aplicados que ayudarán al entendimiento del lenguaje.

Índice

§1 Java **2**
§1.1 Orígenes de Java 2
§1.2 Hola Mundo 3
§1.3 Método `main()` 3
§1.4 Comentarios 4
§2 Variables **4**
§2.1 Declaración de Variables 4
§2.2 Reglas para Nombrar Variables 6
§3 Tipos de Datos **6**
§3.1 Tipos de Datos Primitivos 7
§3.2 Tipo de dato `String` 7
§4 Operadores **8**
§4.1 Operadores aritméticos 8
§4.2 Operadores de asignación 9
§4.3 Operaciones de comparación 9
§4.4 Operadores lógicos 10
§4.5 Operadores bit a bit 10
Palabras Reservadas de Java **12**

Lista de Códigos

1 Hola Mundo. . . . 3
2 Comentarios. . . . 4
3 Declaración de variables. . . . 5
4 Ejemplo declaracion e impresión de variable. . . . 5
5 Variable final. . . . 5
6 Declaración de variables de varios tipos 5
7 Declaración de múltiples variables en una línea 6
8 Declaración e impresión de variables tipo `char`. . . . 7
9 Declaración e impresión de variable de tipo `String`. . . . 8
10 Suma de enteros. . . . 8
11 Sumas de variables y constantes. . . . 8
12 Operador asignación. . . . 9
13 Operador comparación. . . . 9
14 Operadores lógicos y valores. . . . 10

§1 Java

§1.1 Orígenes de Java

Definición 1.1 (**Lenguaje de Programación Java**)
Java es un *lenguaje de programación y una plataforma informática que fue comercializada por primera vez en 1995 por Sun Microsystems.*

Java se creó como una herramienta de programación para ser usada en un proyecto en una pequeña operación denominada the Green Project en Sun Microsystems en 1991. El equipo (denominado *green team*), compuesto por trece personas y dirigido por James Gosling, trabajó durante 18 meses en Sand Hill Road, Menlo Park (California), para desarrollarlo.

Su sintaxis deriva en gran medida de C y C++, pero tiene menos utilidades de bajo nivel que cualquiera de ellos. Las aplicaciones de Java son compiladas a bytecode (clase Java), que puede ejecutarse en cualquier máquina virtual Java (JVM) sin importar la arquitectura de la computadora subyacente.

Observación 1.1 (**Origen del Nombre**)
Algunas fuentes señalan que el nombre podría tratarse de un acrónimo, como las iniciales de sus diseñadores, **J**ames Gosling, **A**rthur **V**an Hoff y **A**ndy Bechtolsheim, o, sencillamente, ***J**ust **A**nother **V**ague **A**cronym* ('simplemente otro acrónimo ambiguo más').

Actualmente es usado para:

- Aplicaciones móviles.
- Aplicaciones de escritorio.
- Aplicaciones Web.

- Servicios Web y Servidores de Aplicaciones.
- Juegos.
- Conexiones a bases de datos.

y muchos más.

§1.2 Hola Mundo

El clásico *Hola Mundo* en Java se ve de la siguiente manera:

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

Código 1: Hola Mundo.

El nombre del archivo es HolaMundo.java. Antes de irnos a cualquier detalle técnico dentro del lenguaje de Java, vamos por las partes básicas que lo componen al momento de escribirlo.

§1.3 Método `main()`

Definición 1.2 (**Método**)
Un **método** en un lenguaje de programación es *un bloque de código que contiene una serie de instrucciones y que se utiliza para realizar una tarea específica.*

En este caso, todo lo que queramos ejecutar dentro de nuestro código va dentro del método `main()`.

Todo programa en Java debe tener un nombre de clase (o de `class`), que debe coincidir con el nombre del archivo. Además, todo programa debe tener un método main.

Observación 1.2 (**Clase**)
Más adelante se verá qué es una clase y para qué sirve.

Dentro del método `main()` tenemos la línea `System.out.println("Hola Mundo")`.

Definición 1.3 (**Clase System**)
La clase `System` es una clase final que pertenece al paquete `java.lang` y proporciona acceso a funcionalidades y propiedades del sistema.

Esta clase es estática, lo que significa que sus miembros y métodos son accesibles directamente a través de la clase sin necesidad de crear una instancia de la misma.

Esta clase nos permite el acceso a los flujos de entrada, salida y error del programa. En particular, el flujo `out` nos permite imprimir en la consola estándar. En este flujo, tenemos varios métodos:

- `println()`.
- `print()`.
- `printf()`.

¿Cuál es la diferencia entre cada uno de ellos? `println` imprime lo que sea que tenga dentro seguida de un salto de línea, `print` no imprime el salto de línea y `printf` nos permite imprimir con formato. Más adelante veremos lo que eso significa.

Entonces, lo que estamos diciendo al programa cuando colocamos la sentencia `System.out.println("Hola Mundo")` es decirle al programa:

Imprime en la línea de comandos el `String`: "Hola Mundo".

§1.4 Comentarios

Para comentar una línea de texto en Java usamos `//`. En el caso de que queramos comentar secciones enteras de texto, las líneas las encerramos entre `/* ... */`. Esto se vería de la siguiente manera:

```
// Comentario de una linea
/*
    Comentario
    de
    varias
    lineas
*/
```

Código 2: Comentarios.

§2 Variables

Definición 2.1 (**Variables**)
Las **variables** son *contendores de diferentes tipos de información a los que un programa puede acceder.*

En Java tenemos (entre muchos) los siguientes tipos de variables:

Tipo	Descripción
`String`	Almacena texto, como "Hola". Los valores String se rodean con comillas dobles
`int`	Almacena enteros, sin decimales, como **123** o **-123**
`float`	Almacena números de punto flotante, con algunos decimales, como **19.99** o **-19.99**
`double`	Almacena números de punto flotante con muchos decimales, como **3.141592** o **2.718281**
`char`	Almacena caracteres individuales, como 'a' o 'B'. Los valores char se rodean con comillas simples al momento de declararse
`boolean`	Almacena valores con uno de los dos estados: `true` o `false`

§2.1 Declaración de Variables

Para usar estas variables en nuestro programa tenemos que declararlas.

Definición 2.2 (**Declarar una Variable**)
Declarar una variable significa básicamente *crear una variable para poder usarla en el programa.*

Ejemplo 2.1 (**Declaración de Variables en Java**)
Para declarar una variable en Java usamos la siguiente sintaxis:

```
//tipo nombreVariable = valor;
```

Código 3: Declaración de variables.

donde `tipo` es el tipo de variable que queremos usar, `nombreVariable` es el nombre de la variable y el `valor` es el valor que le queremos asignar a la variable.

Observación 2.1 (**Nombres Permitidos**)
En muchos lenguajes de programación (incluído Java) existe una cosa llamada palabras reservadas. Las palabras reservadas son palabras que tiene almacenadas el sistema y que no pueden ser usadas como nombres de variables, métodos, clases o cualquier otro identificador.

En la sección Palabras Reservadas de Java (al final de este documento) viene una lista de todas las palabras reservadas que tiene Java.

Por ejemplo, con el siguiente código hacemos la declaración de la variable `nombre` y luego imprimimos su valor:

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

Código 4: Ejemplo declaracion e impresión de variable.

Observación 2.2 (**Variables Finales**)
Si el valor de una variable no va a cambiar en el código, podemos usar la palabra reservada `final`. Por ejemplo:

```
final double pi = 3.141592;
```

Código 5: Variable final.

si luego en el código intentamos cambiar el valor de esta variable, nos dará error.

Aquí tenemos un ejemplo de declaración de varias variables:

```
int myNum = 5;
float myFloatNum = 5.99f;
char myLetter = 'D';
boolean myBool = true;
String myText = "Hello";
```

Código 6: Declaración de variables de varios tipos

Observación 2.3 (Declaración Simultánea de Variables)
Algo curioso que se pueda hacer en Java es la declaración de varias variables, por ejemplo el código:

```
public class HolaMundo{
    public static void main(String[] args){
        int x, y, z;
        x = y = z = 50;
        System.out.println(x);
        System.out.println(y);
        System.out.println(z);
    }
}
```

Código 7: Declaración de múltiples variables en una línea

imprime 3 veces el valor `50`.

§2.2 Reglas para Nombrar Variables

Al momento de nombrar variables en Java, debemos seguir las siguientes reglas:

- Nombres pueden contener letras, dígitos, _, y signo de dólar $.
- Nombres deben empezar por una letra.
- Nombres deben empezar con una letra minúscula y no deben contener espacios en blanco.
- Nombres pueden empezar con $ y _.
- Nombres son sensibles a las mayúsculas (en este caso, `miVariable` y `mivariable` son dos variables distintas).
- Las palabras reservadas no pueden ser usadas como nombres.

§3 Tipos de Datos

Definición 3.1 (**Tipo de Dato (Data Type)**)
En Java, un **data type (tipo de dato)** *define el tipo de valor que puede almacenar una variable y las operaciones que se pueden realizar con ese valor.*

Esencialmente, le dice al compilador cómo interpretar y manipular la información que se guarda en la memoria.

Observación 3.1
Java es un lenguaje de programación de *tipado estático*, lo que significa que *cada variable debe tener un tipo de dato declarado antes de su uso.*

Un lenguaje de programación que no es de tipado estático es Python.

§3.1 Tipos de Datos Primitivos

Definición 3.2 (**Tipos de Datos Primitivos**)
Los **tipos de datos primitivos** son *tipos de datos predefinidos que almacenan valores simples.*

Hay ocho tipos de datos primitivos en Java: `byte`, `short`, `int`, `long`, `float`, `double`, `boolean` y `char`:

Tipo de Dato	Descripción y Rango
	Números
`byte`	Almacena números enteros en el rango de **-128 a 127** (8 bits con signo)
`short`	Almacena números enteros en el rango de **-32,768 a 32,767** (16 bits con signo)
`int`	Almacena números enteros en el rango de **-2,147,483,648 a 2,147,483,647** (32 bits con signo)
`long`	Almacena números enteros en el rango de **-9,223,372,036,854,775,808 a 9,223,372,036,854,775,807** (64 bits con signo)
`float`	Almacena números de punto flotante con **precisión simple** (6-7 dígitos decimales significativos)
`double`	Almacena números de punto flotante con **precisión doble** (15-16 dígitos decimales significativos)
	Booleano
`boolean`	Almacena valores lógicos: `true` (verdadero) o `false` (falso)
	Caracter
`char`	Almacena un **único carácter** Unicode (16 bits) o valores ASCII (ej: 'A', '7', '$')

Definición 3.3 (**Unicode**)
Unicode es un *estándar de codificación de caracteres que asigna un número único a cada carácter, independientemente del idioma, plataforma o programa, permitiendo que las computadoras representen y procesen texto de diferentes sistemas de escritura.*

Se puede encontrar la codificación de los caracteres Unicode en Tabla de caracteres Unicode.

Definición 3.4 (**ASCII**)
ASCII (Código Estándar Americano para el Intercambio de Información) es un *sistema de codificación que asigna valores numéricos únicos a letras, números, signos de puntuación y otros caracteres utilizados en la comunicación electrónica.*

Al igual que con Unicode, es posible encontrar la codificación de caracteres ASCII en El codigo ASCII.

§3.2 Tipo de dato `String`

El tipo de dato `char` es usado para guardar un sólo caracter. Este caracter debe estar rodeado de comillas simples ". También se pueden usar números para codificar con ASCII estos caracteres:

```
    char myVar1 = 65, myVar2 = 66, myVar3 = 67;
    System.out.println(myVar1);
    System.out.println(myVar2);
    System.out.println(myVar3);
```

Código 8: Declaración e impresión de variables tipo `char`.

Un `String` es un tipo de dato usado para guardar una secuencia de caracteres (texto). Estos deben estar rodeados por comillas dobles:

```
String saludo = "Hola Mundo";
System.out.println(saludo);
```

Código 9: Declaración e impresión de variable de tipo `String`.

§4 Operadores

Definición 4.1 (**Operador**)
Los **operadores** son *funciones usadas para realizar operaciones entre variables y valores (o constantes).*

El operador + es usado para sumar dos números. En el ejemplo:

```
int x = 100 + 50;
```

Código 10: Suma de enteros.

Lo que hacemos en el sentido técnico es, al objeto con valor 100 sumarle 50. El resultado es que el valor de la variable `x` es 150.

El operador + también puede ser usado para sumar una variable y una constante, o una constante y una variable:

```
int sum1 = 100 + 50;          // 150 (100 + 50)
int sum2 = sum1 + 250;        // 400 (150 + 250)
int sum3 = sum2 + sum2;       // 800 (400 + 400)
```

Código 11: Sumas de variables y constantes.

Java hace la división de operadores en las siguientes categorias:

- Operadores aritméticos.
- Operadores de asignación.
- Operaciones de comparación.
- Operadores lógicos.
- Operadores bit a bit.

§4.1 Operadores aritméticos

Los operadores aritméticos son usados para efectuar operaciones matemáticas comúnes:

Operador	Nombre	Descripción	Ejemplo
+	Suma	Agrega dos valores	`x + y`
−	Resta	Sustrae un valor de otro	`x - y`

Operador	Nombre	Descripción	Ejemplo
$*$	Multiplicación	Multiplica dos valores	`x * y`
$/$	División	Divide un valor por otro	`x / y`
$\%$	Módulo	Devuelve el residuo de una división	`x % y`
$++$	Incremento	Aumenta el valor de una variable en 1	`++x`
$--$	Decremento	Disminuye el valor de una variable en 1	`--x`

§4.2 Operadores de asignación

Los operadores de asignación son usados para asignar un valor a una variable. En el siguiente ejemplo se usa el operador asignación para asignar a la variable `var` el valor de `10`.

```
int x = 10;
```

Código 12: Operador asignación.

La siguiente es una lista de todos los operadores de asignación:

Operador	Ejemplo	Equivalente a
$=$	`x = 5`	`x = 5`
$+=$	`x += 3`	`x = x + 3`
$-=$	`x -= 3`	`x = x - 3`
$*=$	`x *= 3`	`x = x * 3`
$/=$	`x /= 3`	`x = x / 3`
$\%=$	`x% = 3`	`x = x% 3`
$\&=$	`x &= 3`	`x = x & 3`
$\|=$	`x |= 3`	`x = x | 3`
$\wedge=$	`x ≙ 3`	`x = x ̂3`
$\gg=$	`x ≫= 3`	`x = x ≫ 3`
$\ll=$	`x ≪= 3`	`x = x ≪ 3`

§4.3 Operaciones de comparación

Los operadores de comparación son usados para comparar los valores de dos variables o valores.

Estos son extremadamente importantes en programación ya que nos ayudan a tomar desiciones. l valor de retorno de un operador de comparación en Java es `true` o `false`. Estos son conocidos como valores `boolean`.

Ejemplo 4.2 (**Uso Operador Comparación**)
En este ejemplo usamos el operador de comparación `>` para determinar si la variable `x` es mayor a la variable `y`.

```
int x = 5;
int y = 3;
System.out.println(x > y); // imprime true, pues 5 es mayor a
    3
```

Código 13: Operador comparación.

Java posee los siguientes operadores de comparación:

Operador	Nombre	Ejemplo
==	Igual a	`x == y`
!=	No igual a	`x != y`
>	Mayor que	`x > y`
<	Menor que	`x < y`
>=	Mayor o igual que	`x >= y`
<=	Menor o igual que	`x <= y`

§4.4 Operadores lógicos

Los operadores lógicos se usan para determinar la lógica entre dos variables o valores.

Operador	Nombre	Descripción	Ejemplo
&&	AND lógico	Retorna `true` si ambas expresiones son verdaderas	`x < 5 && x < 10`
||	OR lógico	Retorna `true` si al menos una expresión es verdadera	`x <5 || x <4`
!	NOT lógico	Invierte el resultado: retorna `false` si es verdadero y viceversa	`!(x < 5 && x < 10)`

Generalmente con los operadores lógicos en Java siempre son usados en combinaciones con operadores de comparación

```
System.out.println((1 > 0) && (0 < 1));
```

Código 14: Operadores lógicos y valores.

Estos son usados regularmente `if`, `if else`, `for` y `while`. Más adelante veremos qué son éstos y para qué nos sirven.

§4.5 Operadores bit a bit

La explicación de estos operadores se verá más adelante, pero de momento colocaremos una tabla donde vienen estos operadores:

Operador	Nombre	Descripción	Ejemplo
&	AND bit a bit	Operación AND en cada bit: 1 solo si ambos bits son 1	`5 & 1`
|	OR bit a bit	Operación OR en cada bit: 1 si al menos un bit es 1	`5 | 2`
∧	XOR bit a bit	Operación XOR: 1 solo si los bits son diferentes	`5 ^3`
∼	Complemento a uno	Invierte todos los bits (unario)	`∼5`
≪	Desplazamiento izquierdo	Desplaza bits hacia izquierda, rellena con 0	`5 << 1`

Operador	Nombre	Descripción	Ejemplo
$\gg$	Desplazamiento derecho con signo	Desplaza bits derecha, conserva signo (rellena con bit de signo)	`-5 >> 1`
$\ggg$	Desplazamiento derecho sin signo	Desplaza bits derecha, siempre rellena con 0	`-5 >>> 1`

Palabras Reservadas de Java

Las siguientes son las palabras reservadas que posee el lenguaje de programación Java:

Palabra clave	Descripción
abstract	Modificador de no acceso. Para clases y métodos: clase abstracta no puede crear objetos (se accede heredando). Método abstracto solo en clase abstracta, sin cuerpo (implementado en subclase)
assert	Para depuración
boolean	Tipo de dato que solo almacena true o false
break	Sale de un bucle o bloque switch
byte	Tipo de dato para números enteros (-128 a 127)
case	Define bloque de código en switch
catch	Captura excepciones de bloques try
char	Tipo de dato para caracteres individuales
class	Define una clase
continue	Continúa a la siguiente iteración de un bucle
const	Define constante. No usado (usar final)
default	Bloque por defecto en switch
do	Crea bucle do-while con while
double	Tipo de dato para números fraccionarios (1.7e-308 a 1.7e+308)
else	En condicionales
enum	Declara tipo enumerado (inmutable)
exports	Exporta paquete con módulo (Nuevo en Java 9)
extends	Extiende clase (hereda de otra clase)
final	Modificador de no acceso: clases, atributos y métodos inmutables (no heredables/no sobrescribibles)
finally	Bloque que siempre se ejecuta (con excepciones)
float	Tipo de dato para números fraccionarios (3.4e-038 a 3.4e+038)
for	Crea bucle for
goto	No usado, sin función
if	Crea condicional
implements	Implementa interfaz
import	Importa paquete, clase o interfaz
instanceof	Comprueba si objeto es instancia de clase/interfaz
int	Tipo de dato para números enteros (-2147483648 a 2147483647)
interface	Clase especial solo con métodos abstractos
long	Tipo de dato para números enteros grandes (-9223372036854775808 a 9223372036854775808)
module	Declara módulo (Nuevo en Java 9)
native	Método no implementado en Java (otro lenguaje)
new	Crea nuevos objetos
package	Declara paquete
private	Modificador de acceso: solo en clase declarada
protected	Modificador de acceso: mismo paquete y subclases
public	Modificador de acceso: accesible por cualquier clase
requires	Especifica librerías requeridas en módulo (Nuevo en Java 9)
return	Finaliza método, puede devolver valor
short	Tipo de dato para números enteros (-32768 a 32767)
static	Modificador de no acceso: métodos/atributos accesibles sin instanciar clase
strictfp	Obsoleto. Precisión restringida en cálculos flotantes

Palabra clave	Descripción
`super`	Referencia a superclase (padre)
`switch`	Selecciona bloque de código a ejecutar
`synchronized`	Modificador de no acceso: solo un hilo accede al método
`this`	Referencia al objeto actual
`throw`	Crea error personalizado
`throws`	Indica excepciones que puede lanzar un método
`transient`	Ignora atributo al serializar objeto
`try`	Crea bloque try...catch
`var`	Declara variable (Nuevo en Java 10)
`void`	Especifica método sin valor de retorno
`volatile`	Atributo no almacenado en caché local (siempre leído de memoria principal)
`while`	Crea bucle while