

Notas Java - 1. Introducción

Cristo Alvarado

6 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; Casting (conversión implícita y explícita entre tipos) considerando sus riesgos; y Construcción de relaciones lógicas con operadores comparativos (`>`, `==`, `!=`, etc.). Mediante ejemplos aplicados que ayudarán al entendimiento del lenguaje.

Índice

§1	Java	2
§1.1	Orígenes de Java	2
§1.2	Hola Mundo	2
§1.3	Método <code>main()</code>	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	6
§4	Casting	7
§5	Operadores	7
	Palabras Reservadas de Java	8

Lista de Códigos

1	Hola Mundo.	3
2	Comentarios.	4

3	Declaración de variables.	4
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	5

§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 **G**osling, **A**rthur **V**an Hoff y **A**ndy **B**echtolsheim, 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 línea
/*
    Comentario
    de
    varias
    líneas
*/
```

Código 2: Comentarios.

§2 Variables

Definición 2.1 (Variables)

Las **variables** son *contenedores de diferentes tipos de información a los que un programa puede acceder*.

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

Tipo	Descripción
<code>String</code>	Almacena texto, como "Hola". Los valores String se rodean con comillas dobles
<code>int</code>	Almacena enteros, sin decimales, como 123 o -123
<code>float</code>	Almacena números de punto flotante, con algunos decimales, como 19.99 o -19.99
<code>double</code>	Almacena números de punto flotante con muchos decimales, como 3.141592 o 2.718281
<code>char</code>	Almacena caracteres individuales, como 'a' o 'B'. Los valores char se rodean con comillas simples al momento de declararse
<code>boolean</code>	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 (incluido 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
<i>N ú m e r o s</i>	

Tipo de Dato	Descripción y Rango
<code>byte</code>	Almacena números enteros en el rango de -128 a 127 (8 bits con signo)
<code>short</code>	Almacena números enteros en el rango de -32,768 a 32,767 (16 bits con signo)
<code>int</code>	Almacena números enteros en el rango de -2,147,483,648 a 2,147,483,647 (32 bits con signo)
<code>long</code>	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)
<code>float</code>	Almacena números de punto flotante con precisión simple (6-7 dígitos decimales significativos)
<code>double</code>	Almacena números de punto flotante con precisión doble (15-16 dígitos decimales significativos)
<i>B o o l e a n o</i>	
<code>boolean</code>	Almacena valores lógicos: true (verdadero) o false (falso)
<i>C a r a c t e r</i>	
<code>char</code>	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 código ASCII](#).

§4 Casting

§5 Operadores

Palabras Reservadas de Java

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

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

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