



# Fundamentos de Java

## Introducción a Java



**Transformación Digital**  
Agencia de Transformación Digital y Telecomunicaciones



TECNOLÓGICO  
NACIONAL DE MÉXICO®

El estudio de los fundamentos del lenguaje de programación Java es importante para adentrarse en el mundo del desarrollo de *software*.

Java se ha consolidado como uno de los lenguajes más utilizados a nivel global gracias a su versatilidad, estabilidad y amplia adopción en diversas industrias.

Con él podemos realizar aplicaciones que se ejecuten en celulares, computadoras o en el ambiente *Web*, incluyendo diferentes sistemas operativos.

Por ello iniciar con los fundamentos de este lenguaje nos proporciona una base robusta que permite comprender cómo funcionan las aplicaciones y todos los términos comunes en el uso de esta tecnología.

## **La tecnología Java:**

- Es un lenguaje de programación (es un conjunto de símbolos y reglas que permiten a una persona escribir algoritmos).
- Con un entorno de desarrollo (es el conjunto de herramientas, programas y configuraciones que permiten al programador escribir, editar, compilar, ejecutar y depurar código).
- Entorno de aplicación (un entorno de aplicación se refiere al conjunto de condiciones, recursos y plataformas en las que un programa puede ejecutarse correctamente).
- Entorno de implementación (es el conjunto de herramientas, configuraciones, y recursos necesarios para instalar y ejecutar una aplicación en su forma final).
- Sintaxis similar a C++.

Todo este conjunto es utilizado para el desarrollo de aplicaciones

Las características que permiten cumplir estos objetivos son:

- Una Máquina Virtual Java “(*JVM Java-Virtual-Machine*)”.
- El Entorno de Ejecución Java (*JRE Java-Runtime-Environment*).
- Herramientas de Desarrollo Java (*JDK Java-Developer-Kit*).

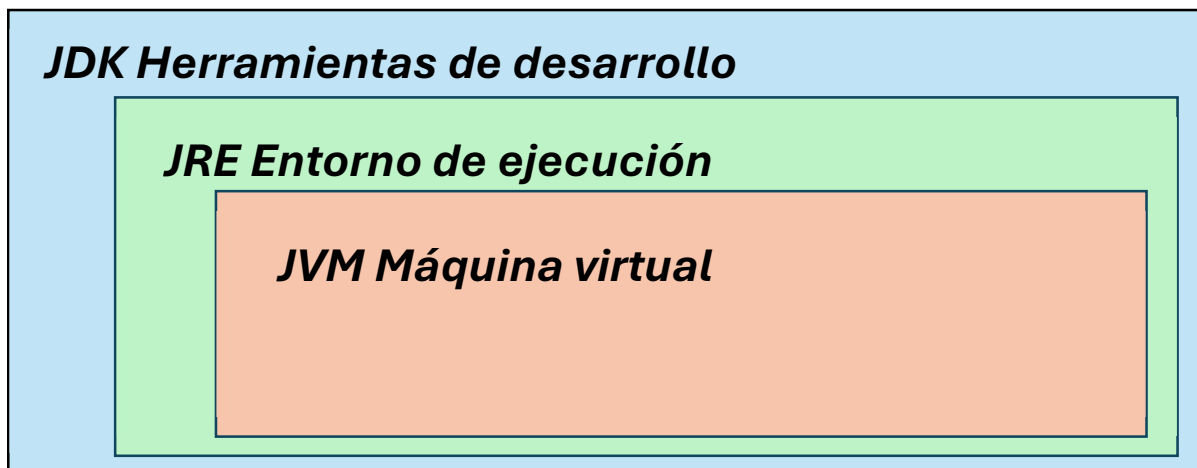


Fig. 1 Alcance de las características de las herramientas Java

La JVM, o Máquina Virtual de Java, es un entorno de ejecución que permite que el código de Java se ejecute en cualquier plataforma. Actúa como una capa intermedia entre el código Java y el *hardware*. (Lindholm, 2025)

La JRE es el *software* necesario para ejecutar programas Java. Contiene la Máquina Virtual de Java (JVM), las bibliotecas y otros archivos de soporte. Actúa como intermediario entre el programa Java y el sistema operativo, traduciendo el código Java para que funcione en diferentes sistemas operativos sin modificaciones “(Schildt, 2022).”

El JDK es un conjunto completo de herramientas que permite desarrollar, compilar, depurar y ejecutar aplicaciones Java. Incluye el compilador de Java (*javac*), la *Java Virtual Machine* (JVM), las bibliotecas estándar y utilidades como el *debugger* y el *JavaDoc*. Es necesario para programadores Java,

mientras que el JRE (*Java Runtime Environment*) solo permite ejecutar aplicaciones Java ya compiladas “(Schildt, 2022).”

Con esto se logran los objetivos de: lenguaje de fácil uso, orientado a objetos y al crear código es simplificado para ser interpretado y fácilmente portable. El compilador y el ejecutor Java tienen roles diferentes, estos roles se muestran a continuación:

Característica	Rol del compilador	Rol del ejecutor
Rol principal	Traducir el lenguaje fuente Java a lenguaje intermedio ByteCode	Carga, verifica, interpreta y ejecuta el Bytecode en una plataforma específica
Entrada	Archivo en código fuente (.java)	Archivo en Bytecode (.class)
Salida	Archivo en Bytecode (.class)	El programa en ejecución y sus resultados

Tabla. 1 comparador de los roles entre el compilador y el ejecutor

Estructura y ciclo de un programa Java:

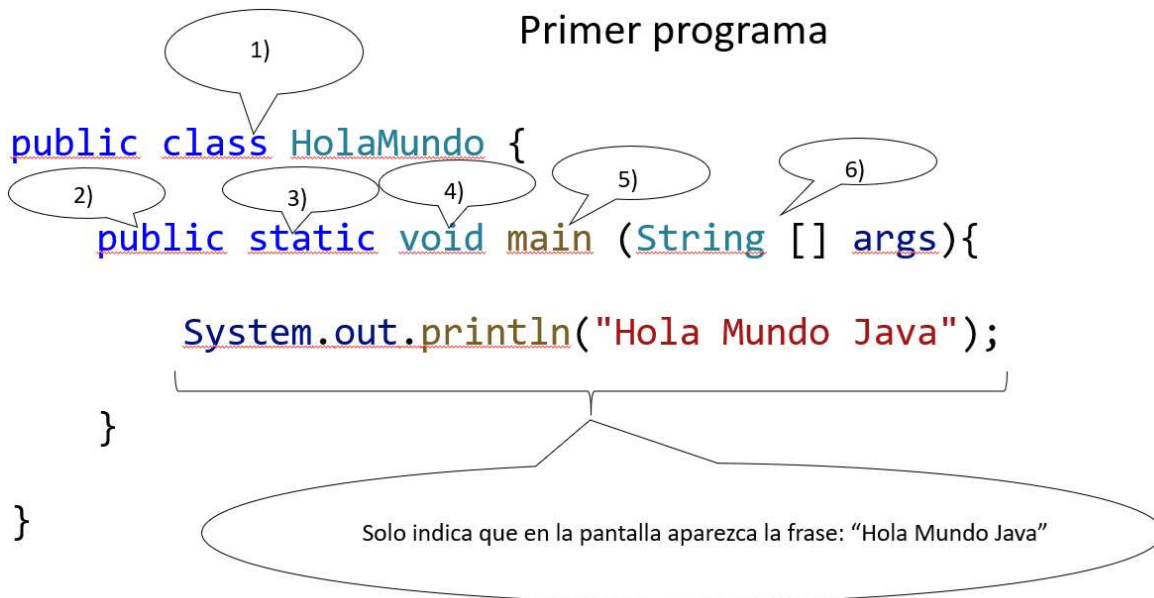
Un programa Java tiene una estructura modular basada en clases y métodos. La clase principal es fundamental porque contiene el punto de entrada del programa (*main*).

Los métodos (equivalente a las funciones), nos permiten manejar la información contenida dentro de las clases.

El método más importante es el método *main*. Este es el punto de entrada obligatorio de cualquier aplicación Java que se ejecute de manera independiente. Para que la JVM lo reconozca, debe cumplir ciertas reglas estrictas:

- 1) La clase debe ser pública y llamarse como se llama el archivo.
- 2) El método de entrada debe ser *public*.  
Permite que la JVM pueda acceder al método desde fuera de la clase.
- 3) Debe ser *static*.  
Permite que la JVM lo llame sin crear una instancia de la clase.  
Esto es necesario porque no hay objetos creados cuando comienza la ejecución.
- 4) Debe tener tipo de retorno *void*.  
No retorna ningún valor a la JVM.
- 5) Debe recibir un arreglo de *String* como parámetro  
*String[] args* permite recibir argumentos desde la línea de comandos.  
También se puede usar *String args[]*, ambas formas son válidas.
- 6) Nombre exacto: *main*.  
Si se cambia el nombre, la JVM no reconocerá el método como punto de entrada.

Estas reglas se muestran en la siguiente interpretación del programa *HolaMundo*:



Ejemplo de ejecución del programa *HolaMundo*:

```
Símbolo del sistema
Microsoft Windows [Versión 10.0.22000.2538]
(c) Microsoft Corporation. Todos los derechos reservados.

d:\Java>javac HolaMundo.java

d:\Java>java HolaMundo
Hola Mundo Java

d:\Java>
```

## Glosario

<b>Algoritmo</b>	<p>Un algoritmo puede definirse como una secuencia finita, ordenada y no ambigua de pasos o instrucciones, cuya ejecución permite transformar una entrada (datos, condiciones) en una salida, resolviendo un problema o realizando una tarea determinada (Briceño, 2023).</p>
<b>Clases</b>	<p>En programación orientada a objetos, una clase es un modelo o plantilla que define las propiedades (atributos) y comportamientos (métodos) de los objetos que se crean a partir de ella.</p> <p>Las clases permiten organizar y reutilizar código, promoviendo encapsulamiento, modularidad y mantenimiento eficiente del <i>software</i>.</p> <p>Cada objeto instanciado de una clase posee sus propios valores de atributos, pero comparte la estructura y comportamiento definidos por la clase (Schildt, 2022).</p>
<b>C++</b>	<p>C++ es un lenguaje de programación multiparadigma — soporta programación procedimental, orientada a objetos y genérica — que ha evolucionado constantemente. Gracias a sus características modernas (como plantillas genéricas, manejo de memoria con punteros inteligentes, expresiones <i>lambda</i>, bibliotecas estándar</p>

	<p>ampliadas, concurrencia, entre otras) permite escribir código eficiente, flexible y reutilizable.</p> <p>C++ sigue siendo muy usado en desarrollos que requieren alto rendimiento, control bajo nivel, y flexibilidad a la hora de diseñar software (Horton, 2023).</p>
<b>Hardware (HW)</b>	<p>El <i>hardware</i> es el conjunto de componentes físicos y tangibles que conforman un sistema informático. Incluye dispositivos internos como la <i>CPU</i>, memoria, <i>buses</i>, placas de expansión y unidades de almacenamiento; además dispositivos externos como monitores, teclados, impresoras y periféricos. Su función es proporcionar la infraestructura física sobre la cual el <i>software</i> puede ejecutarse y controlar el procesamiento de información (Bryant, 2023).</p>
<b>Objetos</b>	<p>En programación orientada a objetos, un objeto es una instancia de una clase, que combina atributos (datos) y métodos (funcionalidades) definidos en la clase. Cada objeto tiene su propio estado y puede interactuar con otros objetos mediante el envío de mensajes (llamadas a métodos).</p> <p>Los objetos permiten representar conceptos del mundo real dentro del <i>software</i>, facilitando la modularidad, reutilización y mantenimiento del código (Schildt, 2022).</p>

<b>POO</b>	<p>La Programación Orientada a Objetos (POO) es un paradigma de programación que organiza el <i>software</i> en objetos, los cuales combinan datos (atributos) y comportamientos (métodos).</p> <p>Este enfoque permite modelar conceptos del mundo real dentro del programa, promoviendo reutilización, modularidad y mantenimiento del código.</p> <p>Los principios fundamentales de la POO incluyen encapsulamiento, herencia, polimorfismo y abstracción, lo que facilita la creación de sistemas más robustos y escalables (Schildt, 2022).</p>
<b>Sistema Operativo (SO)</b>	<p>Un sistema operativo (SO) es el software fundamental que administra y coordina los recursos de <i>hardware</i> de una computadora, ofreciendo servicios esenciales para la ejecución de aplicaciones.</p> <p>Actúa como intermediario entre el <i>hardware</i> y el usuario, gestionando procesos, memoria, almacenamiento, entrada/salida y mecanismos de seguridad.</p> <p>Gracias al sistema operativo, múltiples programas pueden ejecutarse de forma eficiente, segura y concurrente (Tanenbaum, 2024).</p>
<b>Software (Sw)</b>	<p>El <i>software</i> es el conjunto de programas, datos e instrucciones que permiten que una computadora ejecute tareas específicas.</p> <p>A diferencia del <i>hardware</i>, el <i>software</i> es intangible y comprende tanto los programas de sistema (como</p>

sistemas operativos y controladores) que gestionan los recursos de *hardware*, como los programas de aplicación que permiten al usuario realizar actividades particulares, tales como editar documentos, navegar por internet o procesar información.

Su función esencial es indicar al *hardware* qué hacer y cómo hacerlo siguiendo un conjunto lógico de instrucciones. (Bryant, 2023)

## Ejemplo de códigos:

### Ejemplo 1:

Programa que pide el nombre del programador y lo saluda

```
import java.util.Scanner;

public class SaludoSimple {
    public static void main(String[] args) {

        //Permite leer datos escritos por el usuario en la consola.
        Scanner sc = new Scanner(System.in);

        //Captura el texto que el usuario introduce.
        System.out.print("Escribe tu nombre: ");
        String nombre = sc.nextLine();

        //Imprime el saludo personalizado
        System.out.println("¡Hola, " + nombre + "! Bienvenido a Java.");

        //Dejamos de leer desde el teclado
        sc.close();
    }
}
```

## Ejemplo 2:

Programa que pide el nombre, apellido paterno y apellido materno de una persona y la saluda uniendo todos los datos

```
import java.util.Scanner;

public class SaludoCompleto {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        // Leer el nombre y apellidos
        System.out.print("Ingresa tu nombre: ");
        String nombre = sc.nextLine();

        System.out.print("Ingresa tu apellido paterno: ");
        String apellidoPaterno = sc.nextLine();

        System.out.print("Ingresa tu apellido materno: ");
        String apellidoMaterno = sc.nextLine();

        // Concatenar para formar el nombre completo
        String nombreCompleto = nombre + " " + apellidoPaterno + " " + apellidoMaterno;

        // Mostrar el saludo
        System.out.println("¡Hola, " + nombreCompleto + "! Bienvenido/a a Java.");

        sc.close();
    }
}
```

## Elaboró contenido:

Dr. Ulises Juárez Martínez

M.C.C. Manuel Panzi Utrera

## Referencias:

- Briceño, I. (2023). *Fundamentos de la algoritmia*. Universidad Andrés Bello.
- (Bryant, 2023) Bryant, R. E., & O'Hallaron, D. R. (2023). *Computer Systems: A Programmer's Perspective* (Cuarta edición). Pearson.
- Horton, I., & Van Weert, P. (2023). *Beginning C++23: From Beginner to Pro*. (Séptima edición). Apress.
- Lindholm, T., Yellin, F., Bracha, G., Buckley, A., & Smith, D. (2025). *The Java® Virtual Machine Specification, Java SE 25 Edition*. Oracle / Pearson.
- Schildt, H. (2022). *Java: The Complete Reference* (Duodécima edición). McGraw-Hill Education.
- Tanenbaum, A. S., & Bos, H. (2024). *Modern Operating Systems*. (Quinta edición). Pearson.