educación IT
CODE YOUR FUTURE

Bootcamp Java Developer

Fase 1 - Java Analyst

Módulo 7

Introducción

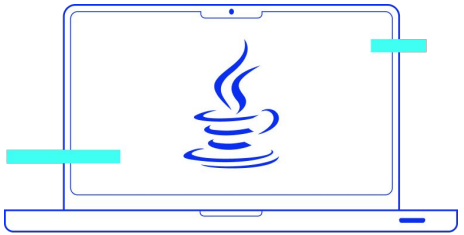
¿Qué es la programación?

Programar no es otra cosa que indicarle a una computadora qué actividades debe realizar, de qué forma y en qué orden.

Una definición más formal sería: **programar es escribir instrucciones para realizar una actividad en algún lenguaje de programación con la finalidad de que sean ejecutadas por la computadora para solucionar un problema**.

Dicho de una manera más simple, si el programador le "*dice*" a la computadora *"Apágate"*, la computadora lo hará.

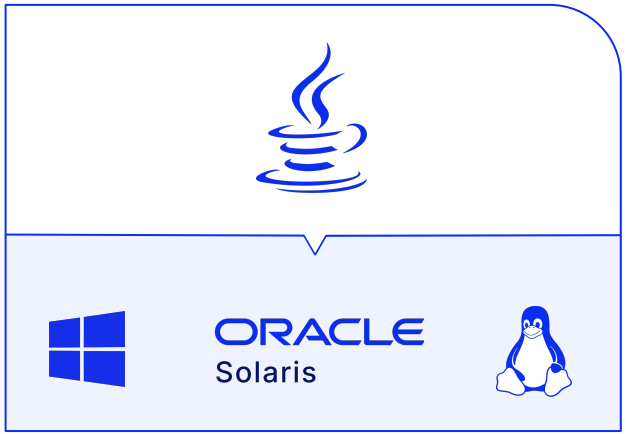
Es decir, únicamente hará lo que explícitamente le digamos que haga.



¿Qué es Java?

Java es un **lenguaje de programación y una plataforma informática** comercializada, por primera vez en 1995, por Sun Microsystems, así de fácil resume la propia web de Java qué es esta tecnología.

Nació con el objetivo de ser **un lenguaje de programación de estructura sencilla que pudiera ser ejecutado en diversos sistemas operativos** (Multiplataforma) por lo que puede ser utilizado en proyectos de gran envergadura.



Organización de Java

		
JSE *Java Standard Edition*	**JME** *Java Platform, Micro Edition*	**JEE** *Java Enterprise Edition*

JSE *(Java Standard Edition)*

Ofrece un entorno de 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.

JME *(Java Platform, Micro Edition)*

Ofrece un entorno flexible y sólido para aplicaciones que se ejecutan en dispositivos móviles e integrados: teléfonos móviles, TDT, reproductores Blu-ray, dispositivos multimedia digitales, módulos M2M, impresoras y más.

JEE *(Java Enterprise Edition)*

Es un conjunto de tecnologías y especificaciones relacionadas que están orientadas hacia el desarrollo de aplicaciones empresariales a gran escala, extiende de JSE y lo complementa para lo antes indicado.

¿Cómo trabaja?

Java es un lenguaje de programación que **necesita ser compilado e interpretado**, pero ¿qué significa esto?

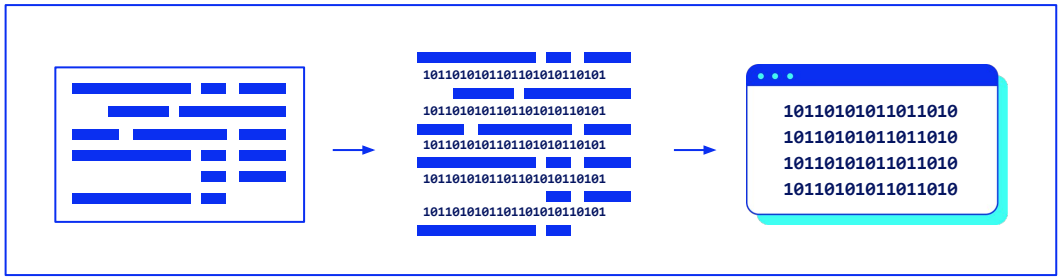
La diferencia entre un lenguaje compilado (*C, C++, GO*) y otro interpretado (*JavaScript, Ruby*) es que el compilado requiere un paso adicional antes de ser ejecutado: la **compilación**, que convierte al código que escribes en lenguaje de máquina. En cambio, un lenguaje interpretado se convierte a lenguaje de máquina a medida que es ejecutado.

Java necesita los dos pasos: compilar a un nivel intermedio llamado *bytecode*, que después es interpretado.

Un archivo Java posee la extensión *NombreArchivo.java* y después de compilado se crea otro: *NombreArchivo.class*, que es el archivo en bytecode.

Para poder **interpretar** este bytecode debes tener instalado **JRE *(Java Runtime Environment).***

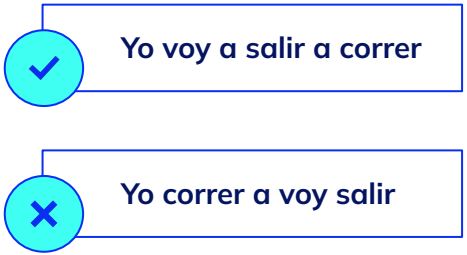
Si deseas **compilar** código Java no es suficiente con instalar el JRE, necesitas el **JDK *(Java Development Kit)*** que incluye el compilador, entre otras herramientas de desarrollo.



Sintaxis

Como cualquier lenguaje de programación, Java tiene su propia **estructura y reglas de sintaxis**. Es un derivado del lenguaje C, por lo que sus reglas de sintaxis son similares.

Poco a poco, veremos que hay varias formas de escribir palabras en Java por lo que se tocarán estos conceptos de forma genérica.



¡Resulta fundamental mantener una **sintaxis correcta**!

Bloques de código y sentencias

- **Bloque de código**: es un conjunto de sentencias delimitadas por llaves "{ }".

  En algunos casos, lo que se necesita ejecutar es tan simple que solo se requiere ejecutar una sentencia en el bloque. Para estos casos, es posible omitir las llaves. Esto, al principio, suele confundir pero, con el tiempo, se logrará más seguridad para hacerlo de esa manera.

- **Sentencia**: es la unidad mínima de ejecución de un lenguaje de programación para resolver un problema en particular y siempre debe terminar con un punto y coma ";".

```
bloque { // inicio de bloque

        sentencia1;
        sentencia2;

} // fin de bloque
```

Comentarios

Sabemos que hay ciertas reglas para escribir en este lenguaje, pero ¿qué pasa si nos toca revisar un código realizado por otro desarrollador, o si simplemente examinamos algo hecho por nosotros que no hemos visto hace tiempo?

Los lenguajes de programación contemplan esta situación. Brindan la posibilidad de crear **bloques de comentarios que ayuden a escribir y a explicar el código.**

Cuando el compilador de Java ve estas estructuras, **omite la evaluación de la sintaxis**.

Comentarios de documentación

Comienzan con barra y doble asterisco **"/**"**. Cada línea intermedia debe poseer un asterisco **"*"** al principio. Para finalizar el comentario se utiliza un asterisco con barra **"*/"**.

Comentarios de Múltiples Líneas

Comienzan con barra y asterisco **"/*"**. Finalizan con asterisco y barra: **"*/"**.

Comentarios de una sola Línea

Se realizan con doble barra **"//"**.

Ejemplo

```
/**Inicio del comentario de documentación
*Para este comentario se pueden aprovechar las etiquetas HTML
*Por ejemplo <h1> Inicio del Software </h1>
*fin del comentario */

/*Inicio del comentario
Comentario de
Varias Lineas
fin del comentario */

//Comentario de una sola linea
```

Nomenclatura

Así como al escribir una carta o un correo debemos colocar los nombres de personas en mayúsculas, o comenzar una oración en mayúsculas; **en Java también hay ciertos estándares** que aunque no dan error es bueno seguirlos.

Estos estándares nos ayudan a escribir **palabras compuestas** en general, además, nos ayudan a **leer más cómodamente el código** y que sea más fácilmente **entendible.**

Upper Case

Todas las letras del nombre se encuentran en mayúsculas.

Ejemplo: **EJEMPLODENOMENCLATURA**
Es usado por lo general para las constantes.

Snake Case

Cuando cada una de las palabras, se separa por un guión bajo "_".

Ejemplo: **ejemplo_de_nomenclatura**
Es usado también para variables.

Camel Case

El nombre viene porque se asemeja a las dos jorobas de un camello, y se puede **dividir en dos tipos:**

- **Upper Camel Case**: Es cuando la primera letra de cada una de las palabras es mayúscula.

  Ejemplo: EjemploDeNomenclatura
  Es usado para nombrar las clases.

- **Lower Camel Case**: Igual que la anterior con la excepción de que la primera letra es minúscula.

  Ejemplo: ejemploDeNomenclatura
  Es usado para paquetes, variables y métodos.

IDE (Integrated Development Environment)

Aunque podemos usar el editor de texto básico que trae el sistema operativo para construir programas en Java, en ese caso, debemos realizar comandos de consola como **Javax** (***Compilar***) y **Java *(Interpretar***).

Existen aplicaciones que hacen el trabajo más fácil, eficaz, y evitan que se comentan errores de sintaxis. **Eclipse es uno de esos programas.** Consiste en un **editor de código, un compilador, depurador y un constructor de interfaz gráfica,** en algunos casos.

Maven

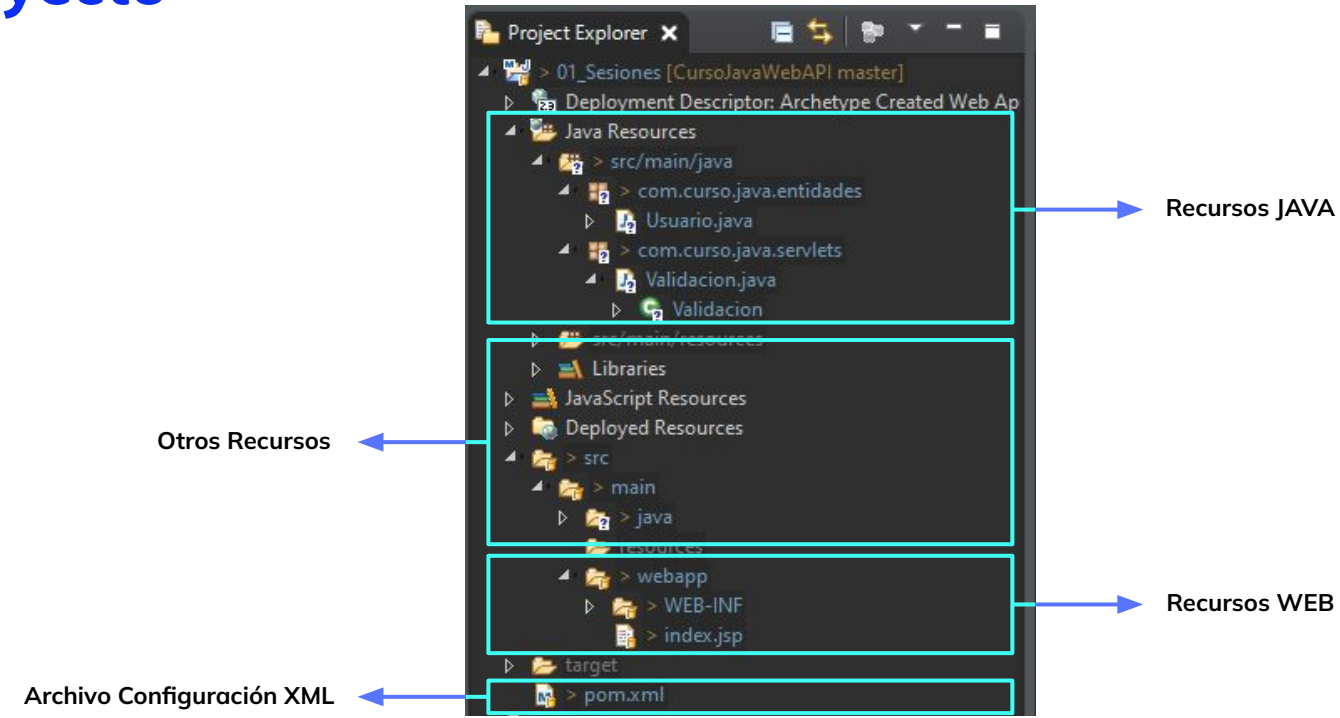
¿Cómo se comienza un **proyecto** y qué estructura tendrá?

Los *IDE* proporcionan diferentes formas de crear un proyecto, pero en el mundo IT es bueno **regirse por estándares y buenas prácticas.** Maven solventa esto y brinda muchos beneficios.

Beneficios

- Gestión de dependencias (librerías) y sus versiones.
- Construcción de proyectos (de código a entregable .war, .ear, .zip, y otros).
- Estructura única de proyecto compatible con todos los entornos de desarrollo y sistemas de integración continua.

Proyecto


Project Explorer
> 01_Sesiones [CursoJavaWebAPI master]
Deployment Descriptor: Archetype Created Web Ap
Java Resources
> src/main/java
> com.curso.java.entidades
Usuario.java
> com.curso.java.servlets
Validacion.java
Validacion
Libraries
JavaScript Resources
Deployed Resources
> src
> main
> java
> webapp
> WEB-INF
> index.jsp
target
> pom.xml
Recursos JAVA
Otros Recursos
Recursos WEB
Archivo Configuración XML

Primer Programa

Este es el primer programa que todo desarrollador realiza cuando toma contacto con un nuevo lenguaje de programación.

Primero se debe crear el método **main**, que en Java es un estándar utilizado por la **JVM** para iniciar la ejecución de cualquier programa, es como la llave de encendido de un automóvil, por allí empieza todo.

```java
public class Principal {

    public static void main(String[] args) {

        //Salida de datos por Pantalla
        System.out.println("Hola Mundo");

    }

}
```

Mostrar información

Como se observa en el slide anterior, se mostró un mensaje en la consola del IDE con la utilidad **`System.out.println()`**. También es posible hacerlo **`System.out.print()`**.

Los dos métodos son útiles para mostrar texto por pantalla y la única diferencia es que **`println()` muestra el texto y luego deja un salto de línea, mientras que `print()` no lo hace**.

```
System.out.println("Hola Mundo con un salto de linea al final");
System.out.print("Hola Mundo");
```

¡Sigamos trabajando!