Universidad Andrés Bello®
Conectar · Innovar · Liderar

FUNDAMENTOS DE PROGRAMACIÓN EN JAVA

El entorno Java para la Programación

El entorno Java: Instalación y primeros pasos

Java es un lenguaje de programación de propósito general y capaz de adaptarse a la mayorgama de proyectos y aplicaciones. Comercializado por primera vez en 1995 por Sun Microsystems y actualmente propiedad de Oracle, Java se ha convertido en uno de los lenguajes más populares, ganando mucha importancia tanto en el ámbito de Internet como en el ámbito de la informática en general. Hay muchas aplicaciones y sitios web queno funcionarán a menos que tengan Java instalado y cada día se crean más. Java es rápido,seguro y fiable. Desde computadores portátiles hasta centros de datos, desde consolas para juegos hasta súper computadoras, desde teléfonos móviles hasta Internet, Java estáen todas partes.

Una de las principales características por la que Java es uno de los lenguajes de programación más famosos, es que es un lenguaje independiente de la plataforma, lo quequiere decir que, si se realiza un programa en Java, este podría funcionar en cualquier computador del mercado. Esto lo consigue porque se ha creado una máquina virtual de Java para cada plataforma, que funciona a modo de puente entre el sistema operativo y el programa de Java, y posibilita que estos componentes se comuniquen perfectamente.

Java sirve para crear aplicaciones y procesos en una gran diversidad de dispositivos. Se basa en los conceptos de la programación orientada a objetivos, permitiendo ejecutar un mismo programa en diversos sistemas operativos, e inclusive en sistemas remotos,
siempre de manera segura.

Su ámbito de aplicación es tan amplio que Java se utiliza tanto en móviles como en electrodomésticos. Algunos programadores también utilizan este lenguaje para crear pequeñas aplicaciones que se insertan en el código HTML de una página para que puedaser ejecutada desde un navegador.

Instalación

Comenzar a usar el lenguaje Java es sencillo y gratuito, además el sitio oficial dispone de una instalación en línea y otra offline.
Para la instalación en línea, se debe descargar un archivo de programa ejecutable para instalar desde la red, requiriendo apenas una intervención mínima de parte del usuario. Al ejecutarlo, el programa obtiene desde Internet todos los archivos necesarios, por lo que es imprescindible permanecer conectado a esta última durante la instalación. Es importante considerar que se necesitan permisos de administrador para instalar Java en un sistema operativo Windows.

Los pasos para la instalación fuera de línea son los siguientes:

1. Vaya a la página de descarga manual.
2. Haga clic en la opción “Windows en línea”.
3. Aparecerá el cuadro de diálogo *Descarga de archivos* y solicitará que ejecute o guarde el archivo descargado.
    a. Para ejecutar el instalador, haga clic en *Ejecutar*.
    b. Para guardar el archivo y ejecutarlo más tarde, haga clic en Guardar.
4. Seleccione la ubicación de la carpeta y guarde el archivo en el sistema local.
5. Sugerencia: guarde el archivo en una ubicación conocida de su equipo; por ejemplo, en el escritorio.
6. Haga doble clic en el archivo guardado para iniciar el proceso de instalación.
7. Se iniciará el proceso de instalación. Haga clic en el botón *Instalar* para aceptar lostérminos de la licencia y continuar con la instalación.
8. Se abrirán varios cuadros de diálogo con información para completar las últimas etapas del proceso de instalación; haga clic en la opción *Cerrar* en el último cuadrode diálogo. Con esta acción se completará el proceso de instalación de Java.

Por otro lado, para la instalación fuera de línea es necesario descargar un archivoejecutable disponible en la página de descarga manual de Java y que incluye todos los archivos necesarios para que el usuario realice la instalación completa. No es necesario permanecer conectado a Internet durante la instalación. El archivo puede copiarse también e instalarse en otro equipo que no tenga conexión a Internet. Se necesitan permisos de administrador para instalar Java en Windows, igual que para la versión en
línea.

Primeros pasos

Antes de comenzar a desarrollar en un lenguaje de programación como Java, existen un par de conceptos que se debe conocer. El primero es el de compilación; compilar significatraducir el código escrito en “Lenguaje entendible por humanos” (por ejemplo, Java, C, Pascal, Fortran), a un código en “Lenguaje Máquina”, que entienden las máquinas, pero no entendible por los seres humanos. Se hace esto porque a los humanos les resultaría casi imposible trabajar directamente con el lenguaje de los computadores. Es por eso quese usa un lenguaje más asequible para las personas (en este caso Java), y luego se empleará un traductor (compilador). La creación de programas en muchos lenguajes se basa en el mismo proceso: escribir código fuente, compilar y obtener un programa ejecutable. El compilador se encarga de evitar que se pueda traducir un programa con código fuente mal escrito y de hacer otras verificaciones previas, de modo tal que el código máquina tiene ciertas garantías de cumplimiento de estándares de sintaxis obligatorios de un lenguaje.

Bajo la lógica anterior, el archivo ejecutable no es válido en cualquier computador. Por ejemplo, si se ha generado el ejecutable para Windows, no podrá utilizarse en un sistemaoperativo Mac.

La novedad introducida fue que Java se hizo independiente del hardwarey del sistema operativo en que se ejecutaba: los programas Java no se ejecutan en la máquina real (en nuestro computador o servidor), sino que Java simula una “máquina virtual” con su propio hardware y sistema operativo. El proceso, entonces, se amplía en un paso: del código fuente se pasa a un código intermedio denominado habitualmente “bytecode”, entendible por la máquina virtual Java, y es esta máquina virtual simulada, denominada Java Virtual Machine o JVM, la encargada de interpretar el bytecode dando lugar a la ejecución del programa.

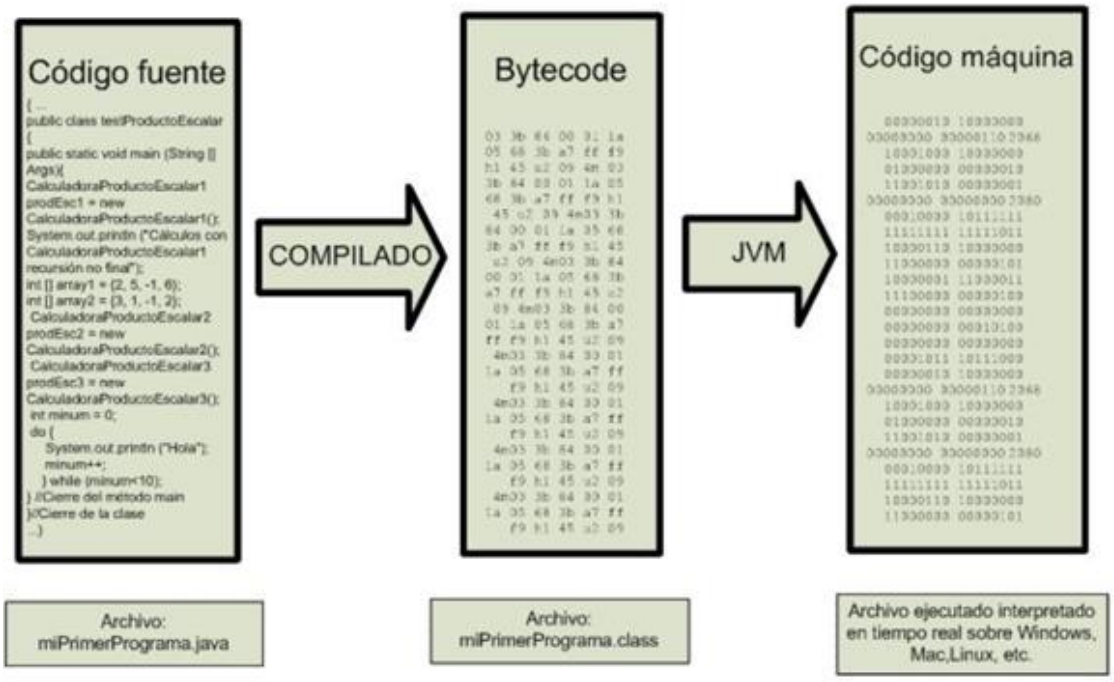

Ilustración 1: Compilación y ejecución de código Java

Lo anterior permite que Java pueda ejecutarse en una máquina con el Sistema OperativoUnix, Windows, MacOS o cualquier otro, dado que en realidad no va a ejecutarse en ninguno de los sistemas operativos, sino en su propia máquina virtual que se instala junto con Java. La desventaja de esta arquitectura, es que todo equipo que quiera correr una aplicación Java ha de tener instalado Java con su máquina virtual.
El compilador Java javac.exe: se encarga de compilar el código fuente

Es necesario considerar que, al ser Java un programa que se interpreta en una máquina virtual, el archivo resultante de la compilación es un archivo con la extensión .class, interpretable por la máquina virtual. Este archivo .class está escrito en un lenguaje de máquina virtual (bytecode).
Para que la “Máquina Real” (nuestro computador) ejecute el programa, hay que “interpretar” (traducir) el archivo .class a un código en “Lenguaje de Máquina Real”.

. .

Estaes la labor de lo que llamamos “intérprete” o traductor del lenguaje de la máquina virtual a la máquina real.

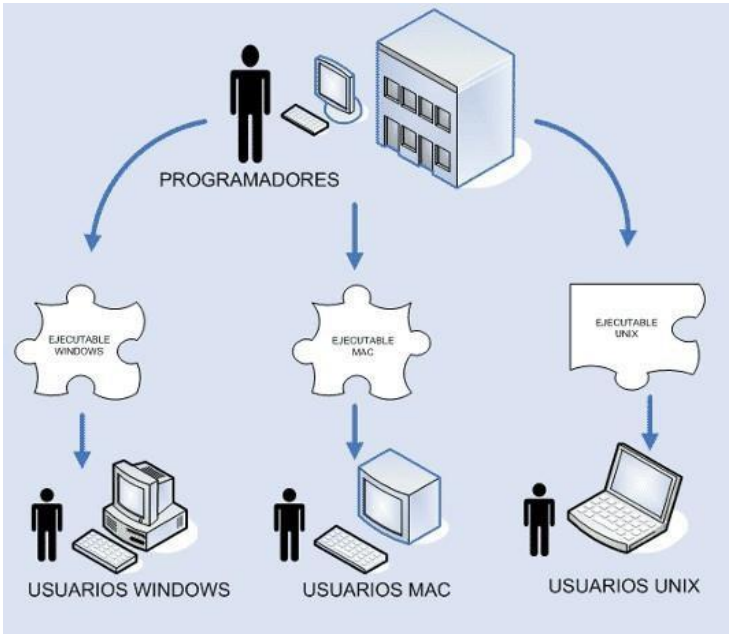

Ilustración 2: Lenguajes compilados

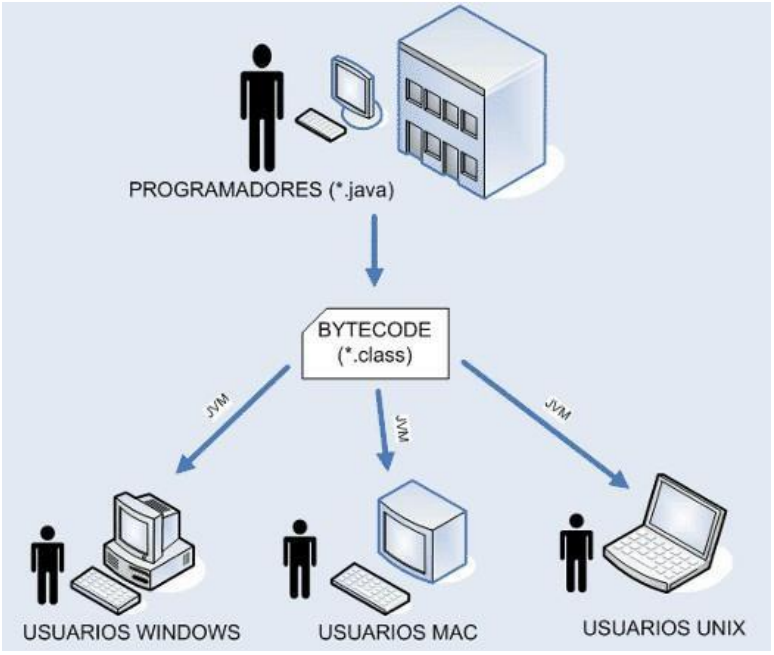

Ilustración 3: Arquitectura propuesta en Java

Los archivos respectivos que se encargan de estas tareas son:

- El intérprete Java
    - o java.exe: se encarga de interpretar los archivos .class (bytecode).

Primer programa en Java

Se comenzará con el ejercicio más básico y popular utilizado en cualquier lenguaje de programación: crear un programa del tipo “Hola Mundo”. Este ejercicio consiste simplemente en mostrar por pantalla la frase "Hola Mundo" por la salida estándar, en este caso el monitor.

Lo primero por hacer para resolver el ejercicio será crear la clase HolaMundo.

```
public class HolaMundo {

}
```

Esta clase se debe guardar en un fichero con extensión .java. Es importante que el ficherose llame tal cual se llama la clase, haciendo coincidir tanto mayúsculas como minúsculas.El fichero se llamará, entonces, HolaMundo.java.

Al compilar y ejecutar la clase, el código que se ejecuta en primer lugar es aquel que estádentro del método main. La forma de hacer lo recién mencionado es la siguiente:

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

    }
}
```

A continuación, se debe mostrar por la consola el texto "Hola Mundo". Para ello se debe utilizar la clase estática System.out, la cual permite acceder a la salida de la consola, paraluego utilizar el método println().

Finalmente, el código quedará de la siguiente forma:

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

. .

Para compilar y ejecutar este código, ejecutar las siguientes sentencias:

```
javac HolaMundo.java
java HolaMundo
```

En caso que la consola no reconozca el comando “javac”, probablemente se deba a que la variable de entorno “PATH” no está configurada. En las referencias de este documentose indica cómo realizar este proceso en diferentes versiones del sistema operativo.

Entorno integrado de desarrollo

Un entorno de desarrollo integrado o *IDE* (por sus siglas en inglés), es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI).

Los IDE proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación conocidos, entre ellos C++, PHP, Python, Java, C#, Delphi, Visual Basic, etc.En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto.

Algunos ejemplos de entornos integrados de desarrollo (IDE) son los siguientes:

- Eclipse
- NetBeans
- MS Visual Studio
- Visual C++

Un IDE debe tener las siguientes características:

- **Multiplataforma:** en caso de estar disponible para diferentes sistemas operativos, los proyectos generados deben ser compatibles en todos ellos sin mayor inconveniente.
- **Soporte para diversos lenguajes de programación:** en la mayoría de los IDE se puede desarrollar proyectos de diversa naturaleza, en diferentes lenguajes de programación. Se debe considerar también que hoy en día el desarrollo de aplicaciones es diverso, abarcando desarrollo web, de escritorio, móvil, servicios, entre otros.

- **Integración con Sistemas de Control de Versiones:** estos sistemas permiten mantener una historia de los cambios, tanto a nivel local como en la nube. En el desarrollo actual son aplicaciones fundamentales, y la mayoría de los programadores las usan en sus labores cotidianas.
- **Reconocimiento de Sintaxis:** todo lenguaje de programación tiene una forma particular de expresar términos e instrucciones; es necesario que un IDE pueda reconocer cualquier error generado por el programador, de lo contrario las ejecuciones posteriores arrojarán un error.
- **Extensiones y Componentes para el IDE:** toda instalación inicial de un IDE contiene una serie de aplicaciones pre instaladas de acuerdo a la versión adquirida. Los IDE conocidos tienen, además, un conjunto de aplicaciones complementarias que ayudan en la tarea. Se debe considerar que cada complemento instalado, es espacio en memoria almacenada y utilizada.
- **Integración con Framework populares:** un framework es un entorno de desarrollopara diversos lenguajes de programación; este tipo de plataformas proponen unanueva forma de programar, sobre la sintaxis regular de los lenguajes comunes, incorporando acciones que son transversales a muchos sistemas.
- **Depurador:** un depurador (o *debugger*) es una aplicación usada para probar, depurar y eliminar los errores de algún programa. Son fundamentales ya en ocasiones es complejo encontrar la respuesta a un error de codificación debido a que los mensajes de error que entregan los diferentes lenguajes de programaciónno son suficientemente claros.
- **Importar y Exportar proyectos:** todo IDE debe tener la capacidad de crear, editary almacenar un proyecto en memoria. Sumado a ello, debe permitir cargar un proyecto existente, haya sido o no construido en el mismo equipo de apertura, y debe dar la opción asimismo de exportar un proyecto en diferentes formatos.
- **Múltiples idiomas:** los IDE más conocidos están desarrollados, por defecto, en idioma inglés. En la actualidad esto no es un impedimento, ya que en la mayoría de los casos incluyen traducciones a una cantidad importante de idiomas.
- **Manual de Usuarios y Ayuda:** toda plataforma de desarrollo que se considere enun estado de madurez superior, debe tener documentación disponible en línea, en diferentes idiomas. Además, existen comunidades de colaboración que ayudanal buen desarrollo, y que dan respuesta a inconvenientes que puedan ocurrir.

En la actualidad el desarrollar en un IDE se considera prácticamente una necesidad. Las ventajas que existen en este tipo de plataformas se destacan:

- La curva de aprendizaje es muy baja.
- Es más ágil y óptimo para los usuarios que no son expertos en manejo de consola.
- Formateo de código.
- Funciones para renombrar variables, funciones.
- Adventencias y errores de sintaxis en pantalla de algo que no va a funcionar al interpretar o compilar.
- Herramientas de refactoring; este concepto corresponde a un proceso a través delcual se consigue mejorar el código fuente sin crear nuevas funcionalidades.

Tipos de datos en Java

Los primeros lenguajes de programación no usaban objetos, solo variables. Una variable se definió anteriormente como un espacio de la memoria del computador, a la que se asigna un contenido que puede ser un valor numérico (sólo números, con su valor de cálculo) o de tipo carácter o cadena de caracteres (valor alfanumérico que constará sólo de texto o de texto mezclado con números).

A modo de ejemplo se puede definir una variable **a** que contenga un valor de 32, lo cual se escribe como **a = 32**. Posteriormente, se puede cambiar el valor de **a** y hacer **a = 78**, o bien hacer que la variable **a** sea equivalente al valor de otra variable **b**, expresado a travésdel comando **a = b**.

En los programas escritos en lenguaje Java, puede ser necesario tanto el uso de datos elementales como de datos complejos. Por lo mismo, Java usa el término "Tipos de datos" para englobar a cualquier cosa que ocupa un espacio de memoria y que puede ir tomando distintos valores o características durante la ejecución de un programa. Es decir, en vez de hablar de tipos de variables o de tipos de objetos, simplemente se usará el concepto de tipos de datos. Sin embargo, a veces "coloquialmente" no se utiliza la terminología deforma estricta: puede existir textos o páginas web donde se habla de una variable en alusión a un objeto.

En Java coexisten dos tipos de datos: por un lado, los tipos primitivos, que correspondena los tipos de variables en lenguajes como C y que son los datos elementales, y por otro lado están los tipos objeto, los que normalmente incluyen atributos, métodos, constructores, entre otros aspectos.

Los tipos de datos primitivos no tienen métodos, no son objetos y no necesitan una invocación para ser creados. A continuación se indica el listado de tipos de datos primitivos:

Nombre	Tipo	Tamaño	Rango aproximado	Valor por defecto
Byte	Entero	1 byte	-128 a 127	0
Short	Entero	2 bytes	-32768 a 32767	0
Int	Entero	4 bytes	2*10^9	0
Long	Entero	8 bytes	Muy grande	0
Float	Decimal simple	4 bytes	Muy grande	0.0f
Double	Decimal doble	8 bytes	Muy grande	0.0d
Char	Carácter simple	2 bytes	---	'u0000'
Boolean	Valor de verdad	1 byte	True o False	False

En cuanto a los tipos objeto es necesario considerar que, a diferencia de la clasificación anterior, sí poseen métodos y necesitan una invocación para ser creados. Los tipos de datos objeto en Java son los siguientes:

- **Tipos de la biblioteca estándar de Java:** String (cadenas de texto), Scanner,TreeSet,

ArrayList, entre otros.

- **Tipos definidos por el programador / usuario:** Son tipos de objetos creados por el desarrollador de acuerdo a la necesidad de la aplicación desarrollada.
- **Arrays:** son series de elementos en un formato de vector o matriz. Se consideran como un objeto especial que carece de métodos.
- **Tipos envoltorio o wrapper:** son equivalentes a los tipos primitivos, pero como objetos. Entre ellos se pueden encontrar el tipo Byte, Short, Integer, Long, Float, Double, Character y Boolean.

Operadores en Java

Un operador realiza una función, toma uno o más argumentos y devuelve un resultado. Cuando se analizó el concepto de "variable", se definió un tipo de dato con un conjunto de operaciones asociadas.

Al igual que en el caso de los algoritmos en pseudo código, en Java existen diferentes tipos de operadores. En este sentido es importante recordar que existen operadores unarios y binarios, en los cuales interviene un solo operando y dos operandos, respectivamente.

A continuación, se hará alusión a los distintos tipos de operadores que existen en el lenguaje Java.

Operadores aritméticos

Se utilizan para realizar operaciones aritméticas simples en tipos de datos primitivos.

- ✓ * = Multiplicación
- ✓ / = División
- ✓ % = Módulo
- + = Suma
- ✓ - = Resta

Operadores de asignación

El operador de asignación se usa para entregar un valor a cualquier variable. Tiene una asociación de derecha a izquierda, es decir, el valor dado en el lado derecho del operadorse asigna a la variable de la izquierda y, por lo tanto, el valor del lado derecho debe declararse antes de usarlo, o bien debe ser una constante.

El formato general del operador de asignación es "*variable = valor;*"

En muchos casos, el operador de asignación se puede combinar con otros operadores para construir una versión más corta de la declaración llamada "*Declaración Compuesta*".Un ejemplo de su uso sería: en lugar de a = a + 5, podemos escribir a + = 5.

Operadores relacionales

Estos operadores se utilizan para verificar relaciones tales como igualdad, mayor que, menor que, entre otras, que devuelven un resultado booleano después de la comparación y se usan ampliamente en las instrucciones de un ciclo, así como en las sentencias condicionales if/else. El formato general es variable operador_relacion valor

Algunos de los operadores relacionales son:

- **==** (Igual a): devuelve verdadero si el valor del lado izquierdo es igual al del ladoderecho.
- ✓ **!=** (No igual a): devuelve verdadero si el valor del lado izquierdo no es igual al dellado derecho.
- ✓ **<** (Menor que): el resultado será verdadero si el valor del lado izquierdo es inferioral del lado derecho.
- ✓ **<=** (Menor o igual que): devuelve un valor verdadero si el valor del lado izquierdoes menor o igual que el lado derecho.
- ✓ **>** (Mayor que): devuelve verdadero si el valor del lado izquierdo es mayor queaque del lado derecho.
- ✓ **>=** (Mayor que o igual a): regresa verdadero si el valor del lado izquierdo es mayoro igual que el lado derecho.

Operadores lógicos

Estos operadores se utilizan para realizar operaciones “lógicas AND” y “lógicas OR”, es decir, la función similar a la puerta AND y la puerta OR en electrónica digital. Una cosa a tener en cuenta es que la segunda condición no se evalúa si la primera es falsa, es decir, tiene un efecto de cortocircuito. Se usa ampliamente para probar varias condiciones paratomar una decisión.

Los operadores condicionales son:

- ✓ **&&** (AND lógico): devuelve verdadero cuando ambas condiciones son verdaderas.
- ✓ **||** (O lógico): devuelve verdadero si al menos una condición es verdadera.

Sentencias de control

Las sentencias de control de flujo determinan el orden en que se ejecutarán las otras sentencias dentro del programa. El lenguaje Java soporta varias sentencias de control deflujo, incluyendo las siguientes categorías:

- ✓ **Toma de decisiones:** if-else, switch-case
- ✓ **Bucles o ciclos:** for, while, do-while
- ✓ **Excepciones:** try-catch-finally, throw
- ✓ **Misceláneas:** break, continue, return

En este capítulo sólo se tratarán las sentencias de tomas de decisiones y las estructuras misceláneas, abordando los bucles y excepciones en otro capítulo.

Sentencias de toma de decisiones: La sentencia if-else

La sentencia **if-else** de java proporciona a los programas la posibilidad de ejecutarselectivamente otras sentencias, basándose en algún criterio. Esta es la versión más sencilla de la sentencia **if**: la sentencia gobernada por **if** se ejecuta si alguna condición es verdadera. Generalmente, la forma sencilla de **if** se puede escribir de la siguiente manera:

```
if (expresión)
    sentencia;
```

Sin embargo, ¿qué pasaría si se desea ejecutar un conjunto diferente de sentencias si la expresión es falsa? Para ello, se puede utilizar la sentencia **else**, que ejecuta la condiciónopuesta a la definida en la opción **if**.

```
if (expresión)
    sentencia;
else
    otrasentencia;
```

Este uso particular de la sentencia **else** es la forma de capturarlo todo.

Existe otra forma de la sentencia **else**. La sentencia **else if** ejecuta una sentencia basada en otra expresión.

Ejemplo 1: Desarrolle un programa que asigne notas basadas en la puntuación de un examen. Para obtener una calificación “Sobresaliente” debe obtener una puntuación del90% o superior, un “Notable” se obtiene con el 80% o superior, “Bien” se obtiene con unanota superior a 70%, y un “Suficiente” con un puntaje de 60% o superior. En cualquier otro caso, se obtiene una calificación “Insuficiente”.

```
int puntuacion;
String nota;

if (puntuacion >= 90) {
    nota = "Sobresaliente";
} else if (puntuacion >= 80) {
    nota = "Notable";
} else if (puntuacion >= 70) {
    nota = "Bien";
} else if (puntuacion >= 60) {
    nota = "Suficiente";
} else {
    nota = "Insuficiente";
}
```

Respecto de este tipo de estructuras, es necesario hacer los siguientes alcances:

- Una sentencia **if** puede tener cualquier número de sentencias de acompañamiento **else if**.
- En caso que dentro de una sentencia **if**, **else if** o **else** exista solo una línea osentencia, el uso del paréntesis de llave “{ }” es optativo. Si existe más de una sentencia se debe usar el paréntesis de manera obligatoria, o probablemente ocurrirá un error de sintáxis o de negocio.
- No es obligatorio usar una sentencia **else** junto con un **if**.

Sentencias de toma de decisiones: La sentencia switch-case

La sentencia **switch** se utiliza para realizar sentencias condicionalmente basadas en alguna expresión.

Ejemplo 2: desarrolle un programa que contenga un entero llamado **mes** cuyo valor indicael mes en alguna fecha, y que despliegue el nombre del mes basándose en su número entero equivalente. Use la sentencia **switch-case** para resolver este problema.

```
int mes;

switch (mes) {
    case 1: System.out.println("Enero"); break;
    case 2:  System.out.println("Febrero"); break;
    case 3: System.out.println("Marzo"); break;
    case 4: System.out.println("Abril"); break;
    case 5: System.out.println("Mayo"); break;
    case 6: System.out.println("Junio"); break;
    case 7: System.out.println("Julio"); break;
    case 8:  System.out.println("Agosto"); break;
    case 9: System.out.println("Septiembre"); break;
    case 10:  System.out.println("Octubre");  break;
    case 11: System.out.println("Noviembre"); break;
    case 12: System.out.println("Diciembre"); break;
    default: System.out.println("No es un mes válido");
    break;
}
```

La sentencia **switch** evalúa la expresión, en este caso el valor de mes, y ejecuta la sentencia **case** apropiada.

Decidir cuándo utilizar las sentencias **if** o **switch** dependen del juicio personal. Se puede decidir cuál utilizar basándose en la buena lectura del código o en otros factores. Cada sentencia **case** debe ser única y el valor proporcionado a cada sentencia debe ser del mismo tipo que el tipo de dato devuelto por la expresión proporcionada a la sentencia **switch**.

Otro punto de interés en la sentencia switch son las sentencias **break** después de cada case. La sentencia **break** hace que el control salga de la sentencia **switch** y continúe con la siguiente línea. La sentencia **break** es necesaria porque las sentencias case se siguen ejecutando hacia abajo. Esto es, sin un **break** explícito, el flujo de control seguiría secuencialmente a través de las sentencias **case** siguientes.

Finalmente, se puede utilizar la sentencia **default** al final de la sentencia **switch** para manejar los valores que no se han manejado explícitamente por una de las sentencias **case**.

Sentencias de ramificación: break y continue

Las sentencias de ramificación son aquellas que permiten romper con la ejecución lineal de un programa.
Un programa creado en Java siempre se va ejecutando de forma lineal, sentencia a sentencia. Si se desea romper esta linealidad, se las sentencias de ramificación.
Las sentencias de ramificación en Java son: break y continue. En el caso de **break**, sirve para salir de bloque de sentencias, mientras que **continue** sirve para ir directamente al siguiente bloque.

Anexo: Referencias

1.- Instalar Java en línea
Referencia: https://www.java.com/es/download/help/ie_online_install.xml

2.- Instalar Java offline
Referencia: https://www.java.com/es/download/help/windows_offline_download.xml

3.- ¿Cómo puedo establecer o cambiar la variable del sistema PATH?
Referencia: https://www.java.com/es/download/help/path.xml

4.- Documentación oficial de Java (en inglés)
Referencia: https://docs.oracle.com/en/java/

5.- Tutorial de Java
Referencia: https://docs.oracle.com/javase/tutorial/