INTRODUCCIÓN A JAVA

SEGUNDO CICLO

CARRERA DE SOFTWARE

¿QUÉ ES EL LENGUAJE DE PROGRAMACIÓN JAVA?

- Java es un lenguaje de programación de propósito general, tipado, orientado a objetos,… que permite el desarrollo desde aplicaciones básicas, pasando por aplicaciones empresariales hasta aplicaciones móviles.
- Java nacía como un lenguaje de programación que pudiese ser multiplataforma y multidispositivo, bajo el paradigma ***"Write Once Run Anywhere" (WORA)***
- De esta forma un programa Java escrito una vez podemos ejecutarle sobre diferentes plataformas, siendo soportados los sistemas operativos **Windows, MacOs y UNIX**. Y a su vez en diferentes tipos de dispositivos.

CARACTERÍSTICAS DEL LENGUAJE JAVA

- Dentro de las características principales del lenguaje Java encontramos:

CARACTERÍSTICAS	
Independiente de Plataforma	Arquitectura Neutral
Orientado a Objetos	Portabilidad
Sencillez	Robustez
Seguridad	Multi-Hilo
Alto Rendimiento	Interpretado

INDEPENDIENTE DE PLATAFORMA

- Cuando compilamos código fuente en Java, no se genera un código máquina específico para una plataforma concreta. En cambio, lo que se produce son **bytecodes**. Estos bytecodes son fragmentos de código que son interpretados por la **Java Virtual Machine (JVM)**.
- La JVM es una máquina virtual que se encarga de ejecutar los bytecodes, convirtiéndolos en código máquina específico para la plataforma en la que se está ejecutando. Gracias a la JVM, el mismo código fuente Java puede ser ejecutado en múltiples plataformas, ya sean Windows, Linux, macOS u otras, sin necesidad de hacer ninguna modificación. Esta es una característica que le da a Java una gran versatilidad y portabilidad

ORIENTADO A OBJETOS

En el lenguaje de programación Java, **cada elemento es considerado un objeto.** Esta es una característica distintiva de los lenguajes orientados a objetos. En estos objetos se encapsulan los datos, lo que significa que se agrupan y se protegen para asegurar su integridad. Para interactuar con estos datos, se utilizan métodos, que son funciones específicas que permiten acceder a los datos, modificarlos o realizar operaciones con ellos. Este enfoque ofrece numerosas ventajas, como la modularidad y la reutilización de código, y es fundamental para la forma en que Java estructura y organiza el código.

SENCILLEZ

Java, como lenguaje de programación, ha sido diseñado con el objetivo principal de ser un lenguaje fácil de aprender y usar para los programadores. Esta es una de las muchas razones por las que ha ganado tanta popularidad a lo largo de los años

Se deberán de entender los conceptos básicos de la programación orientada a objetos (POO). Esta es una metodología de programación que utiliza objetos y sus interacciones para diseñar aplicaciones y programas de computadora. Es una filosofía de diseño que permite a los programadores pensar en problemas de programación en términos de objetos del mundo real y sus interacciones.

SEGURIDAD

Los programas de Java se ejecutan dentro de un entorno controlado conocido como la **Java Virtual Machine (JVM)**. Este entorno funciona como una "*caja de arena*", proporcionando un espacio aislado donde los programas pueden operar. Esta característica es esencial para prevenir que los programas accedan a partes del sistema que están fuera de su alcance, lo que garantiza la seguridad del sistema en general.

ARQUITECTURA NEUTRAL

Esto significa que Java es independiente de la arquitectura hardware en la que se ejecuta. Ya sea que esté trabajando en una arquitectura de 32 bits o de 64 bits, Java garantiza la consistencia en términos de cómo maneja los tipos de datos. En otras palabras, independientemente del sistema en el que se ejecute, los tipos de datos en Java siempre ocuparán el mismo espacio. Esto es un testimonio de la portabilidad y la versatilidad de Java como lenguaje de programación, capaz de mantener su funcionalidad y eficiencia en una variedad de entornos de hardware.

PORTABILIDAD

Java es un lenguaje de programación que se destaca por su portabilidad. Esta característica esencial de Java proviene de su diseño independiente de la plataforma. En otras palabras, no hay elementos en Java que estén vinculados a una plataforma o sistema operativo específico. Esto significa que los programas escritos en Java pueden ejecutarse en una variedad de plataformas diferentes, ya sean Windows, Mac, Linux u otras, sin la necesidad de ser reescritos para cada una

ROBUSTEZ

El lenguaje Java ha sido diseñado con un fuerte enfoque en la robustez, lo que significa que se ha hecho un esfuerzo considerable para gestionar y controlar las situaciones que pueden conducir a errores. Este control se manifiesta tanto en los procesos de compilación como de ejecución. Los compiladores de Java son capaces de detectar muchas situaciones problemáticas en la fase de compilación, antes de que se ejecute el código. Esta capacidad de anticipación reduce significativamente la probabilidad de que el software falle durante su ejecución.

PROGRAMACIÓN MULTI-HILO EN JAVA

Esta característica esencial significa que un único programa Java puede abrir y gestionar múltiples hilos de ejecución al mismo tiempo. Esto permite que los procesos se ejecuten de manera más eficiente y rápida, ya que se pueden realizar varias tareas de forma simultánea en lugar de secuencial. Esta capacidad de manejar múltiples hilos de ejecución es una de las razones por las que Java es una opción popular para el desarrollo de aplicaciones y software complejos.

INTERPRETADO

En el concepto de **bytecodes**, estos son interpretados en tiempo real a código máquina. Esto significa que el código fuente es convertido a una serie de instrucciones, conocidas como bytecodes, que luego son ejecutadas por la máquina virtual durante su ejecución. Esta característica permite una portabilidad y flexibilidad significativas, ya que los bytecodes pueden ser ejecutados en cualquier dispositivo que tenga una máquina virtual compatible.

ALTO RENDIMIENTO

Una de las principales razones de la alta eficiencia y rendimiento de Java es el uso de los **compiladores Just-In-Time (JIT)**. Los **compiladores Just-In-Time** son una característica esencial en la arquitectura de Java que permite traducir el código fuente en tiempo real. Este proceso de traducción en tiempo real significa que el código se compila y se ejecuta simultáneamente, lo que reduce el tiempo general de ejecución. Esta funcionalidad permite que las aplicaciones de Java funcionen de manera eficiente y rápida, proporcionando un rendimiento optimizado.

LOS ELEMENTOS DE UN PROGRAMA EN JAVA

- Se describe la definición léxica y sintáctica de los elementos

de un programa Java:

- comentarios
- identificadores,
- variables y valores,
- tipos primitivos, literales
- operadores, y expresiones aritmético-lógicas

LOS ELEMENTOS DE UN PROGRAMA EN JAVA

COMENTARIOS

- 1.Comentario de bloque. Empieza por /* y termina por */. El compilador
- ignora todo el texto contenido dentro del comentario.

```
/*
* El programa HolaMundo se utiliza para aplicar los
* métodos System.out.print() y System.out.println()
*/
```

- 2. Comentario de documentación. Empieza por /** y termina por */. Java
- dispone de la herramienta javadoc para documentar automáticamente los
- programas. En un comentario de documentación normalmente se indica el
- autor y la versión del software

```
/**
* Programa HolaMundo
* @author Fundamentos de Informática
* @version 1.0
* @see Referencias
*/
```

LOS ELEMENTOS DE UN PROGRAMA EN JAVA

COMENTARIOS

- 3 Comentario de línea. Empieza con //. El comentario comienza con estos
- caracteres y termina al final de la línea.

- // El método System.out.println() salta la línea

- El uso de comentarios hace más claro y legible un programa. En los comentarios se debe decir qué se hace, para qué y cuál es el fin de nuestro programa. Conviene utilizar comentarios siempre que merezca la pena hacer una aclaración sobre el programa

IDENTIFICADORES

COMENTARIOS

- El programador tiene libertad para elegir el nombre de las variables, los métodos y de otros elementos de un programa. Existen reglas muy estrictas sobre los nombres que se utilizan como identificadores de clases, de variables o de métodos.
- Debe empezar con una letra y seguida de varios caracteres o dígitos
- Se puede utilizar el guion bajo
- EJEMPLOS :
- numero_impar, nombrePadre,apellido1,suma
- No se puede utilizar las palabras reservadas del lenguaje para identificar objetos, atributos

NORMAS BÁSICAS PARA IDENTIFICADORES

- Los nombres de variables y métodos empiezan con minúsculas. Si se trata de un nombre compuesto cada palabra debe empezar con mayúscula y no se debe utilizar el guión bajo para separar las palabras.
- Por ejemplo, los identificadores calcularSueldo, setNombre o getNombre son válidos.
- Los nombres de clases empiezan siempre con mayúsculas. En los nombres compuestos, cada palabra comienza con mayúscula y no se debe utilizar el guión bajo para separar las palabras. Por ejemplo,
- HolaMundo, PerimetroCircunferencia, Alumno o Profesor son nombres válidos.

NORMAS BÁSICAS PARA IDENTIFICADORES

- Los nombres de constantes se escriben en mayúsculas. Para nombres compuestos se utiliza el guión bajo para separar las palabras. Por ejemplo, PI, MINIMO, MAXIMO o TOTAL_ELEMENTOS son nombres válidos.

TIPOS DE DATOS

Tipos primitivos

- Enteros (con signo)

		MIN_VALOR	*MAX_VALOR*
byte	8 bits	-128	+127
short	16 bits	-32768	+32767
int	32 bits	-2147483648	+2147483647
long	64 bits	-2^{63}	$2^{63}-1$

- Reales (coma flotante, según estándar IEEE 754-1985):

float	32 bits
double	64 bits

- Lógico o booleano (1 bit)

boolean	1 bit

- Caracteres (según estándar ISO Unicode 1.1 de 16 bits)

char	16 bits

Por ejemplo `'a'` `'A'` `'\n'` `'\t'` `'\u0058'` → `'X'`

CARACTERES

Caracteres. Los valores de tipo carácter representan un carácter Unicode. Se escriben siempre entre comillas simples, por ejemplo 'a', 'A', '0', '9'. En Java un carácter se puede expresar por su código de la tabla Unicode en octal o en hexadecimal. Los caracteres que tienen una representación especial se indican en la siguiente tabla.

Carácter	Significado
\b	Retroceso
\t	Tabulador
\n	Salto de línea
\r	Cambio de línea
\"	Carácter comilla doble
\'	Carácter comilla simple
\\	Carácter barra hacia atrás

OPERADORES

- Una operación aditiva se refiere a la suma y la resta: 2+3, 5-2.
- Una operación multiplicativa multiplica o divide dos valores: 5*2, 5/2. El operador % calcula el resto de la división entera 5%2.
- Un incremento o decremento aumenta o decrementa en 1 el valor de una variable: ++numero, numero++, --numero, numero--. Si el operador va antes de la variable, primero se realiza la operación y se modifica el valor de la variable. Si el operador va después de la variable, su valor se modifica al final.
- Un operador relacional permiten comparar dos valores: >, <, >= y <=. El resultado de la comparación es un valor booleano que indica si la relación es verdadera o falsa.
- Un operador de igualdad compara si dos valores son iguales o no. El operador == devuelve verdadero si los dos valores son iguales, el operador != devuelve verdadero si son diferentes. El resultado de la comparación es un valor booleano que indica si la igualdad o desigualdad es verdadera o falsa.
- Un operador de asignación permite asignar un valor o el resultado de una operación a una variable: =, +=, -=, *=, /=, %=.

BOOLEANOS

Booleanos. Los operadores que se aplican a los valores lógicos son: negación, Y lógico, O lógico.

- La negación (!) devuelve `true` si el operando es `false`.
- El Y lógico (&&) devuelve `false` si uno de los operandos es `false`.
- El O lógico (||) devuelve `true` si uno de los operandos es `true`.

Las palabras reservadas de Java

En todos los lenguajes de programación existen palabras con un significado especial. Estas palabras son reservadas y no se pueden utilizar como nombres de variables.

abstract	final	public
assert	finally	return
boolean	float	short
break	for	static
byte	if	strictfp
case	implements	super
catch	import	switch
char	instanceof	synchronized
class	int	this
continue	interface	throw
default	long	throws
do	native	transient
double	new	true
else	null	try
enum	package	void
extends	private	volatile
false	protected	while

ELEMENTOS P.O.O

método	Objeto	Clase	biblioteca
Main			
println	out	System	Java.lang
		HolaMundo	

System.out.println(“Hola mundo”);
Una clase contiene métodos, además de otras definiciones y un método a su vez contiene sentencias

EJERCICIOS

Realizar la suma, resta, multiplicación y división

```
class Caritmetica
{
Public static void main (String[ ] args)
{  int dato1, dato2,resultado;
dato1=20;
Dato2=10;
// suma
Resultado=dato1+dato2;
System.out.println(dato1+"+")+dato2+"=" +resultado);
....
}
}
```

```
//Resta
Resultado= dato1+dato2;
System.out.println(dato1+"+")+dato2+"=" +resultado);
//Multiplicación
```