Práctica #1 – Java: Variables, tipos y concatenación

RESULTADOS DE APRENDIZAJE:

Al finalizar esta sesión de práctica el estudiante deberá:

- ✓ Ejecutar código básico del lenguaje de programación Java.
- ✓ Declarar variables y comprender los tipos de variables.
- ✓ Concatenar y formatear cadenas de texto.

Nuestro primer programa será sencillo y mostrará el saludo "Hola Mundo" en la pantalla. Para crear el programa necesitaremos realizar los siguientes pasos:

1. Crear el código fuente. Un archivo de código fuente contiene texto escrito en el lenguaje de programación Java. Se puede utilizar un simple editor de texto para crear y editar el código.
2. Compilar el código fuente. El compilador translada el código fuente en instrucciones que la máquina virtual de Java pueda entender. El compilador crea esas instrucciones en un archivo bytecode.
3. Ejecutar el programa. El intérprete Java, instalado en el sistema operativo, implementa la máquina virtual de Java. Este intérprete transforma el bytecode en instrucciones que pueda entender el sistema operativo.

```java
package test;

public class Test {
    public static void main(String[] args) {
    System.out.println("Hello World!");
    }
}
```

ACTIVIDAD 1

1. Escribe una nueva línea que imprima el texto: Mi primer programa en Java!
2. Busca en internet la forma de comentar código en java (línea única y multi-linea)

VARIABLES

Las variables son una de las características fundamentales de los lenguajes de programación, permiten acceder a la memoria para almacenar y recuperar los datos con los que nuestros programas van a trabajar. Son por tanto el mecanismo que los lenguajes de programación ponen a nuestra disposición para acceder a la memoria.

Se trata de un mecanismo de lo más sencillo, sólo tenemos que dar un nombre a nuestras variables, a partir de ese momento el compilador traducirá de forma automática ese nombre en un acceso a memoria. Por ejemplo:

Java es un lenguaje tipado y nos obliga a declarar nuestras variables antes de poder hacer uso de ellas, con esta declaración le indicamos al compilador el espacio en memoria que debe de reservar para almacenar la información. Por ejemplo:

```
//Almacenamos un dato en memoria referenciado por el nombre edad
edad = 5;
//Recuperamos el dato almacenado y lo modificamos
edad = edad + 1;
```

Aquí estamos reservando memoria para una variable de tipo String y la identificamos con el nombre "cliente". De ahora en adelante si en el programa hablamos de cliente, estamos haciendo referencia a esa porción de memoria y al valor que contiene.

```
String cliente;
```

Podemos asignarle algún valor en el momento de declarar una variable. Por ejemplo:

```
String cliente = "Isaac Newton";
```

Aquí reservamos memoria para una cadena de caracteres y le asignamos el valor "Isaac Newton". También podemos declararla y en otro lugar del programa fijarle un valor:

```
String cliente; // declaración

 ... // El programa sigue

cliente = "Isaac Newton"; // le damos un valor
```

El nombre debe ser único en el contexto del programa. Además debe seguir las siguientes reglas:

1. No puede ser una palabra reservada del lenguaje o un literal booleano (true o false)
2. Puede contener cualquier carácter Unicode, pero no puede comenzar con un número
3. No debe contener los símbolos que se utilicen como operadores (+ , - , ?, etc)

Por convención, los nombres de variables comienzan con una letra en minúscula. Si un nombre consiste en más de una palabra, se escribirá sin espacios entre ellas y cada palabra (salvo la primera) comenzará con una letra mayúscula (por ejemplo : estaBienEsteNombre)

TIPOS DE VARIABLES

Cada variable debe tener un tipo de dato predefinido. Esto determina el rango de valores que puede almacenar y qué operaciones se pueden realizar así como el resultado que te dará. Por ejemplo, una variable de tipo entero puede almacenar números sin decimales y puede realizar operaciones aritméticas, pero no puede contener palabras.

Existen dos categorías de variables: las de tipo primitivo y las referenciadas. Una variable de tipo primitivo accede al valor asignado directamente. Las referenciadas acceden a través de un puntero, es decir, no almacenan un valor sino una dirección de memoria. Estas últimas son utilizadas por las matrices, las clases y las interfaces.

Tipo	Tamaño y formato
	ENTEROS
byte	8 bits - complemento a 2
short	16 bits - complemento a 2
int	32 bits - complemento a 2
long	64 bits - complemento a 2
	NÚMEROS REALES
float	32 bits - IEEE 754
double	64 bits - IEEE 754
	OTROS
char	16 bits - caracteres UNICODE
boolean	1 bit

En otros lenguajes de programación, el formato o el tamaño de los tipos primitivos dependen del microprocesador o del sistema operativo en el que se están ejecutando. En cambio, Java pretende ser independiente de la plataforma y mantiene los formatos sin cambios. Para los caracteres alfanuméricos utiliza la codificación UNICODE de 16 bits, para permitir la inclusión de varios alfabetos.

ACTIVIDAD 2

1. Escribe y ejecuta el siguiente código:

```
String user = "Jaime Guevara";
System.out.println(user);
System.out.println();

int salarioSemanal = 32;

int salarioMensual = salarioSemanal * 4;
System.out.println(salarioMensual);
System.out.println();
```

2. Utiliza Renombramiento por Refactor para cambiar nombre a las siguientes variables.

   `salarioSemanal -> userSalarioSemanal`

   `salarioMensual -> userSalarioMensual`

3. Añade y ejecuta el siguiente código:

```
    int semanas  = 130;
    double anios = semanas / 52;
    System.out.println(anios);
    System.out.println();
```

4. Interpreta y comenta con tus compañeros el error en la operacion y soluciónalo.
5. Añade y ejecuta el siguiente código:

```
    int myHexadecimal = 0x0F;
    int myBinary = 0b010101;
    System.out.println(myHexadecimal);
    System.out.println(myBinary);
    System.out.println();
```

CONCATENACIÓN Y FORMATOS DE CADENA

La **concatenación** es el proceso de anexar una cadena al final de otra cadena. Al **concatenar** literales o constantes de cadena mediante el operador +, el compilador crea una única cadena.

```
    String saludo = "hola mi nombre es ";
    String nombre = "Goku";
    int ssj = 3;
    System.out.println("My name is " + "Juan");
    System.out.println(saludo + nombre);
    System.out.println("Puedo convertirme en un SSJ nivel: " + ssj);
    System.out.println();
```

En java podemos parsear un conjunto de valores a una cadena mediante el API de la clase 'String'. Para ello se debe hacer una llamada al método 'format' pasándole como parámetros:

1. Patrón de formateo, donde indicaremos la estructura de la salida del método.
2. El resto de parámetros separados por comas

```
    int accountBalance = 200;
    String bankMessage = String.format("El balance de su cuenta es de %d",
accountBalance);
    System.out.println(bankMessage);
```

Existen muchos especificadores de conversión para el tipo de parámetro recibido, a continuación mostramos una tabla de los soportados por el método 'String.format()':

Conversor	Valor
%b	Booleano
%h	Hashcode
%s	Cadena
%c	Caracter unicode
%d	Entero decimal
%o	Entero octal
%x	Entero hexadecimal
%f	Real decimal
%e	Real notación científica
%g	Real notación científica o decimal
%a	Real hexadecimal con mantisa y exponente
%t	Fecha u hora

ACTIVIDAD 3

1. Modifique los mensajes de impresión del código que se ha escrito en esta guía. Utiliza concatenación y/o formato de cadena para anteponer texto que proporcionen un mejor significado para el usuario.

2. Escriba y ejecute el siguiente código:

```
    int age = 34;
    String name = "Jaime";
    String output = String.format("%s tiene %d años de edad.", name, age);
    System.out.println(output);

    String intro = "El resultado de";
    String plus = "más";
    double primero = 2.5;
    double segundo = 5.8;
    double resultado = primero + segundo;
    String cadena = String.format("%s %f %s %f es: %f", intro, primero, plus,
segundo, resultado);
    System.out.println(cadena);
```

3. Escribe conclusiones sobre lo que se ha estudiado el día de hoy.