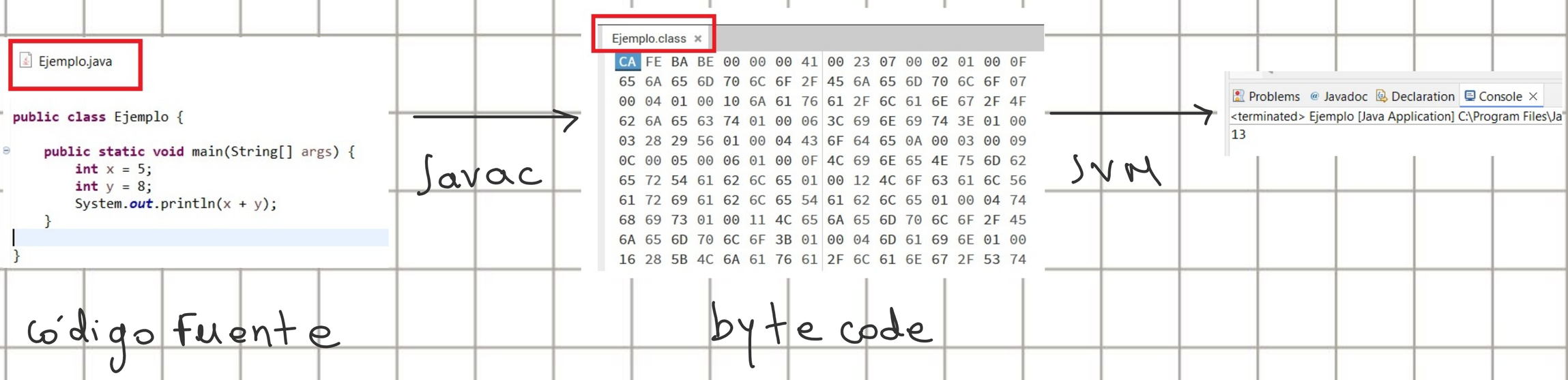
Introducción a Java

Características de Java:

- lenguaje de alto nivel
- versátil, multipropósito.
- portable
- orientado a objetos
- interpretado y compilado
- Robusto
- soporta multihilos, programación funcional, etc.

JDK está compuesto por:

- Compilador (Java c)
- Java virtual machine (JVM)
- Application Programming Interface (API)



Instalación en linux:

```
sudo apt-get update

sudo apt-get upgrade

sudo apt-get install default-jre
```

Primer Programa por consola

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

```
C:\Users\andre\Documents\1 - clases\2 - cursos\curso java>javac Saludo.java
C:\Users\andre\Documents\1 - clases\2 - cursos\curso java>java Saludo
Hola mundo!!
```

→ compilación

→ ejecución

Tipos de datos - variables

Tipos de datos

Almacena	Tipo	Ocupa (bytes)	Rango / Almacena
Número	byte	1	-127..128
	short	2	-32768..32767
	int	4	$-2^{31}..2^{31}-1$
	long	8	$-2^{63}..2^{63}-1$
	float	4	Decimales precisión simple
	double	8	Decimales precisión doble
Carácter	char	2	Un carácter tipo Unicode
Booleana	boolean	1	true / false

Palabras reservadas

abstract	const *	Finally	int	public	this
boolean	continue	Float	interface	return	throw
break	default	For	long	short	throws
byte	do	goto *	native	static	transient
case	double	If	new	strictfp	try
catch	else	implements	package	super	void
char	extends	Import	private	switch	volatile
class	final	instanceof	protected	syncrohnized	while

* ya no se usan

Variables

int edad ; → declaración de la variable

Nombre de la variable

Tipo de dato entero

edad = 25 ; → Asignación de la variable

25

edad

Ejemplos

int x = 0x1a; ⟶ 26 en base hexa

double y = 18e-3; ⟶ 0.018 (18×10^{-3})

float z = 10.3f; ⟶ error porque 10.3 lo toma como double.

float z = 10.3f; ⟶ ✓ toma z = 10.3 como de presición simple

Además, los tipos flotantes pueden albergar otras constantes.

double x = 10.0/0.0; ⟶ x = Infinity

double y = 10.0/(-0.0); ⟶ y = -Infinity

Ingreso de datos por teclado

```
package prueba;
import java.util.Scanner;

public class Ppal {

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

        System.out.print("Ingrese su nombre: ");
        String nombre = entrada.nextLine();

        System.out.print("Ingrese su edad: ");
        int edad = entrada.nextInt();

        System.out.println("Hola " + nombre + " tienes " + edad + " anios");

        entrada.close();
    }
}
```

Librería para trabajar con Scanner para el ingreso de datos por teclado

Crea un objeto Scanner para el ingreso de datos por teclado

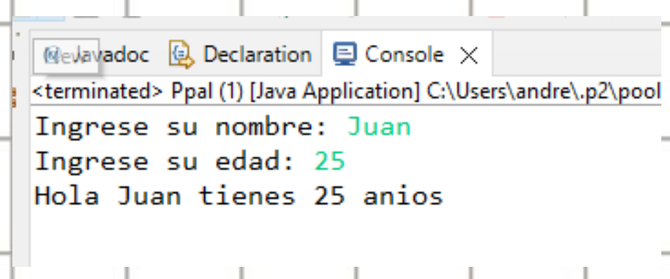
Lee un entero y lo guarda en la variable edad

Lee una línea y la guarda en la variable nombre

Imprime las variables leídas

Cierra el escaner

```
Ingrese su nombre: Juan
Ingrese su edad: 25
Hola Juan tienes 25 anios
```



Operadores

Operadores de asignación " = "

Asigna la expresión del lado derecho a la del lado izquierdo

```
acumulador = acumulador + cuenta;  // Forma expandida
acumulador += cuenta;              // Forma abreviada
```

Operadores aritméticos

+ suma
- resta
* multiplicación
/ división
% resto

/ realiza la división entera si los dos valores son enteros. Si hubiera un valor de tipo flotante, realiza la división flotante.

Ejemplos

double x = 20 / 6 ⟶ x = 3.0

double x = 20.0 / 6 ; ⟶ x = 3.333

int x = 20 % 6 ; ⟶ x = 2

Incremento - decremento

++ incrementa 1

-- decrementa 1

int x = 5 ; x ++ ; ⟶ x = 6

int x = 5 ; x -- ; ⟶ x = 4

Operadores relacionales

> , >= , < , <= , == , !=

Tablas de verdad

Negación

A	! A
V	F
F	V

Conjunción y disyunción

A	B	A && B	A || B
V	V	V	V
V	F	F	V
F	V	F	V
F	F	F	F

Operadores lógicos

Comparan expresiones booleanas y devuelven otra expresión booleana

- && and (conjunción)
- || or (disyunción)
- ! not (negación)

Strings La clase String no es un tipo de dato básico, como lo es un int o un double.

la utilizaremos para almacenar datos en forma de texto.

```
String nombre = “Juan”;
String apellido = “Pérez”;
```

Concatenación Con el operador ++ podemos unir dos cadenas.

```
String nombre = “Juan”;
String apellido = “Pérez”;
String nombreYApellido = nombre + “ ” + apellido;

// Resultado: Juan Pérez
```

Longitud de una cadena

Con el método (función) length obtenemos la cantidad de caracteres que tiene una cadena.

```
String nombre = "Juan";
System.out.println(nombre.length());  // imprime 4
```

Comparar dos cadenas

Para saber si dos cadenas son iguales, no nos sirve el operador ==, debemos utilizar el método equals.

```
String nombre = "Juan";
nombre.equals("Juan");     // devuelve verdadero (true)
nombre.equals("Pedro");    // devuelve falso (false)
nombre.equals("JUAN");     // devuelve falso (false)
```

Para saber si dos cadenas son iguales sin importarnos si difieren en mayúsculas o minúsculas, debemos utilizar el método equalsIgnoreCase.

```
String nombre = "Juan";
nombre.equals("Juan");          // devuelve verdadero (true)
nombre.equals("JUAN");          // devuelve falso (false)
nombre.equalsIgnoreCase("JUAN"); // devuelve true
```