

UD1: Análisis de las fases en el desarrollo de un programa. Elementos del lenguaje.

**IES LOS SAUCES – BENAVENTE
CFGs DESARROLLO DE APLICACIONES WEB
PROGRAMACIÓN**

¿Qué es la programación?



Es el proceso de diseñar, codificar, depurar y mantener el código fuente de los PROGRAMAS.

El código fuente es escrito en un LENGUAJE DE PROGRAMACIÓN.

```
int main(void) {  
    printf ("Hello World");  
    printf ("\n");  
}
```

¿Qué es un programa?



Es una secuencia de instrucciones, escritas para realizar una tarea específica en el ordenador.

Un programa traduce a un lenguaje de programación concreto un ALGORITMO.



¿Qué es un algoritmo?



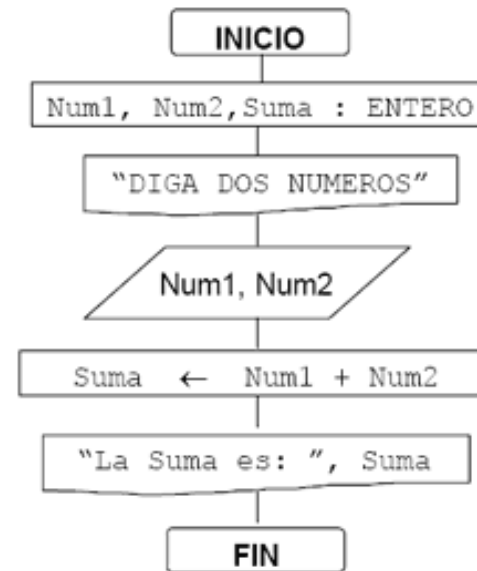
Es un conjunto ordenado y finito de operaciones que permite hallar la solución de un problema.

Un algoritmo debe ser preciso, finito y correcto.

Pseudocódigo:

```
INICIO
  Num1, Num2, Suma : ENTERO
  ESCRIBA "Diga dos números: "
  LEA Num1, Num2
  Suma ← Num1 + Num2
  ESCRIBA "La Suma es:", Suma
FIN
```

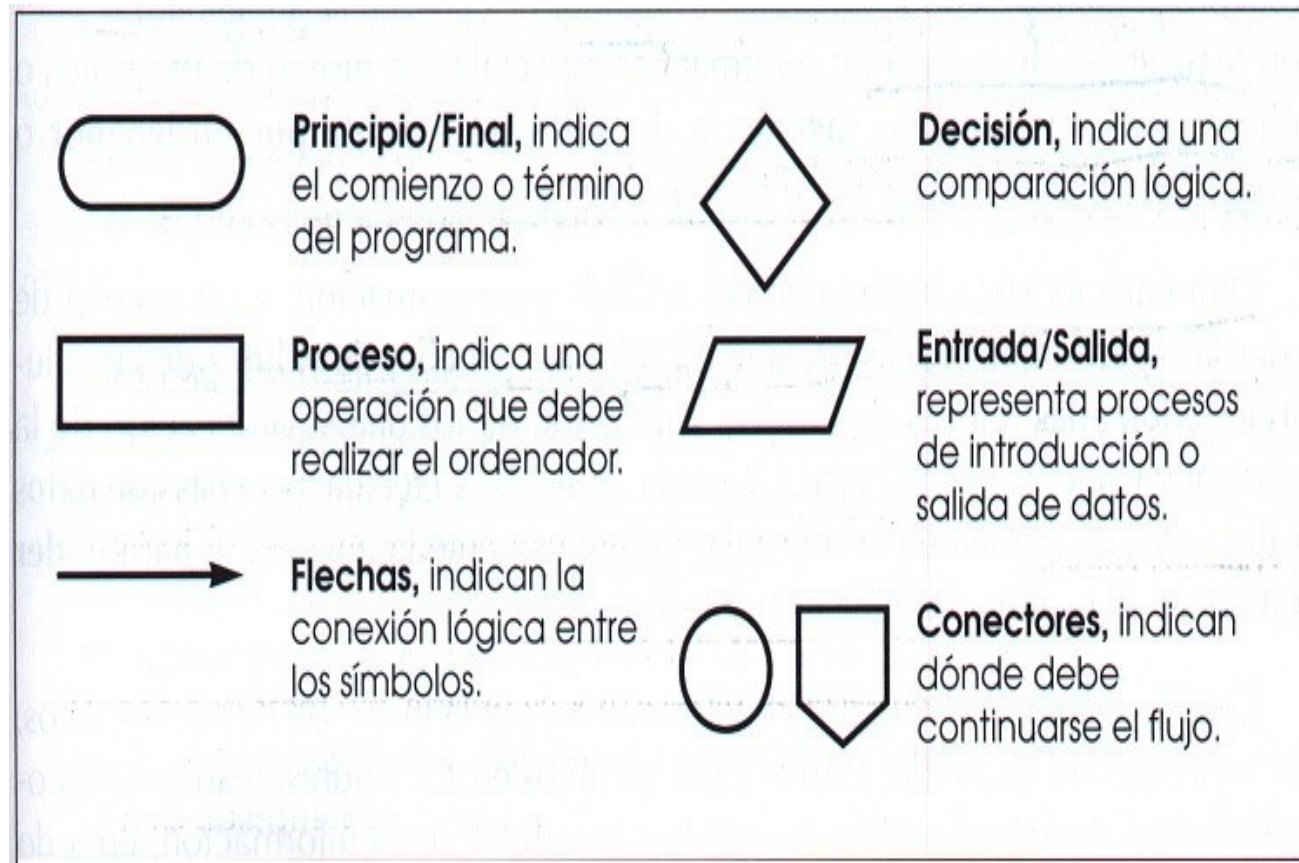
Diagrama de flujo:



¿Qué es un diagrama de flujo?



Es la representación gráfica de un algoritmo.

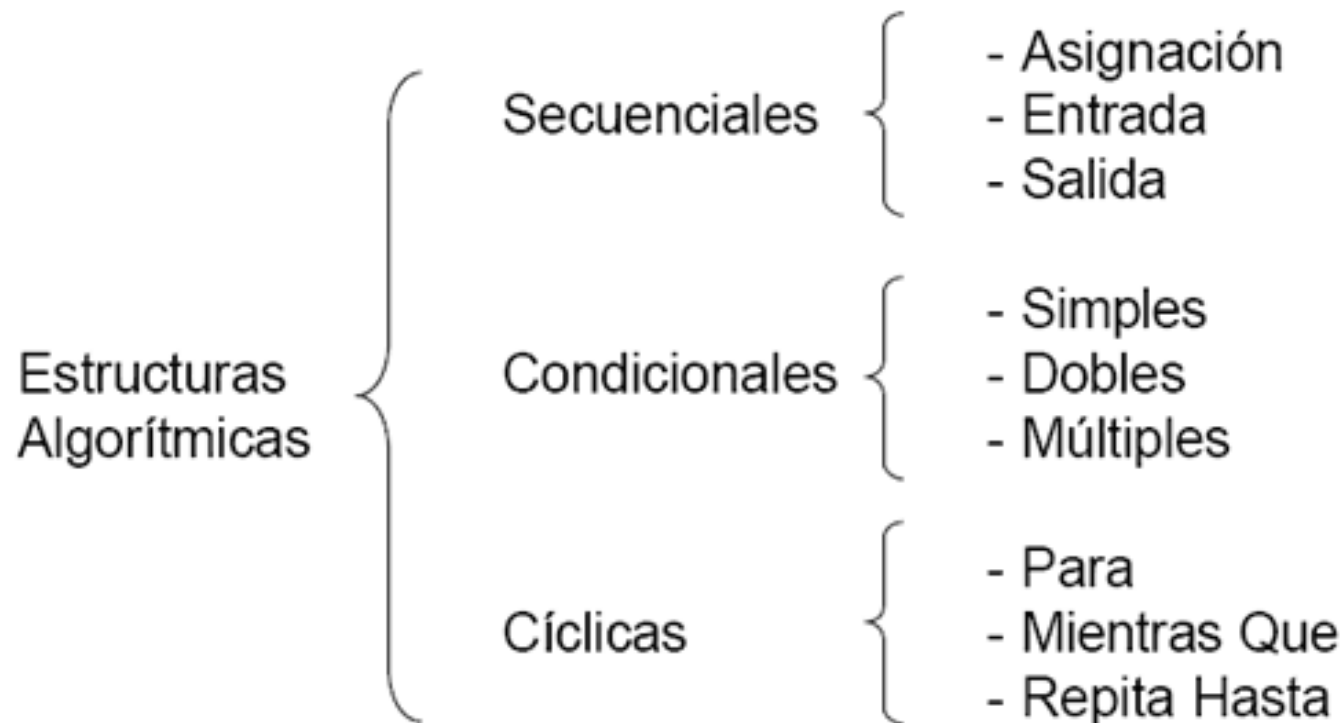


¿Qué es el pseudocódigo?



Es la escritura de un algoritmo en un lenguaje cercano al natural.

Es independiente del lenguaje de programación que se vaya a utilizar.



¿Qué es un paradigma de programación?



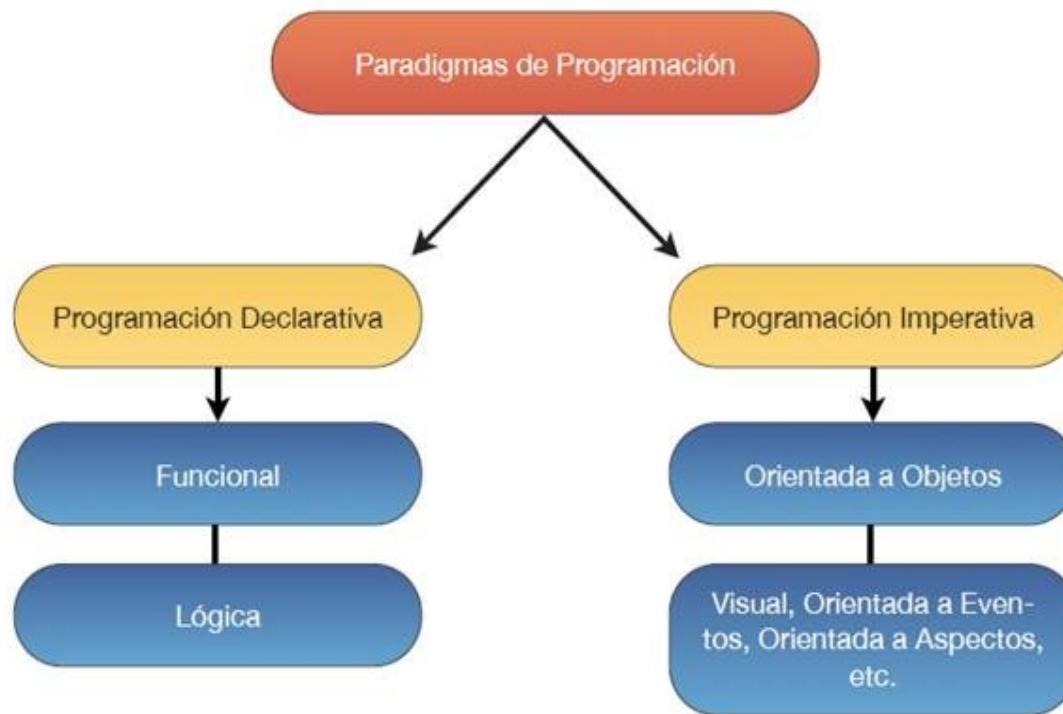
Es un modelo básico para el diseño e implementación de programas.

Este modelo determinará como será el proceso de diseño y la estructura final del programa.



Paradigmas de programación

Un lenguaje de programación puede estar basado en uno o más paradigmas.

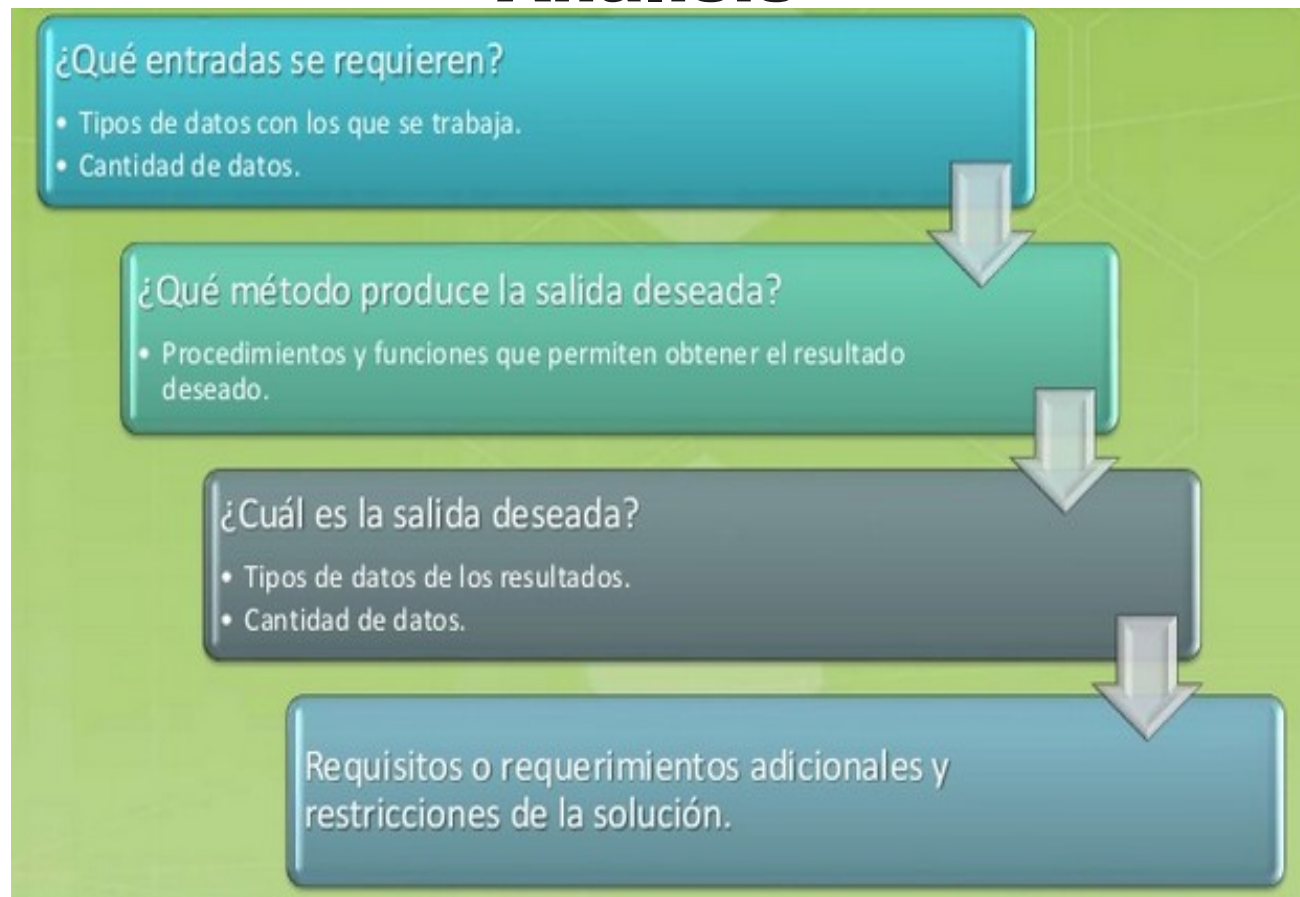


Etapas en el desarrollo de un programa



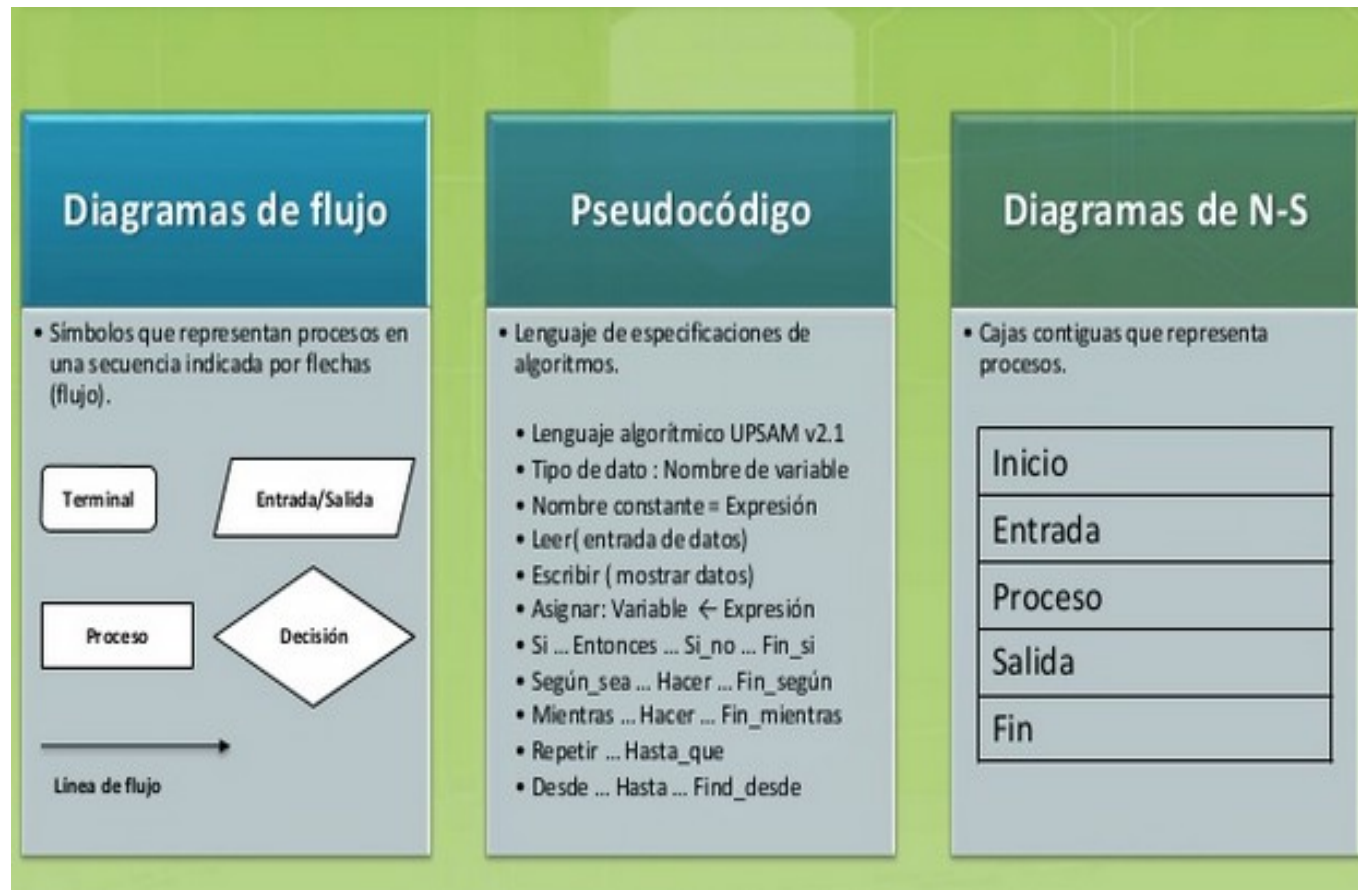
Etapas en el desarrollo de un programa

Análisis



Etapas en el desarrollo de un programa

Diseño



Etapas en el desarrollo de un programa

Codificación

Algoritmo Calcular_Edad

Var

Entero : AN, AA, E

Inicio

Escribir("Año de nacimiento: ")

Leer(AN)

Escribir("Año actual: ")

Leer(AA)

Si AA>AN Entonces

E ← AA – AN

Escribir("Edad: ",E)

Si_no

Escribir("Año actual debe ser mayor al Año de nacimiento")

Fin_si

Fin



```
public class Calcular_Edad
```

```
{
```

```
    public static void Main()
```

```
    {
```

```
        int AN, AA, E;
```

```
        System.Console.Write("Año de nacimiento: ");
```

```
        AN = System.Convert.ToInt32( System.Console.ReadLine() );
```

```
        System.Console.Write("Año actual: ");
```

```
        AA = System.Convert.ToInt32( System.Console.ReadLine() );
```

```
        if ( AA > AN )
```

```
        {
```

```
            E = AA – AN;
```

```
            System.Console.Write("Edad: {0}", E);
```

```
        }
```

```
        else
```

```
        {
```

```
            System.Console.Write("Año actual debe ser mayor que Año de nacimiento");
```

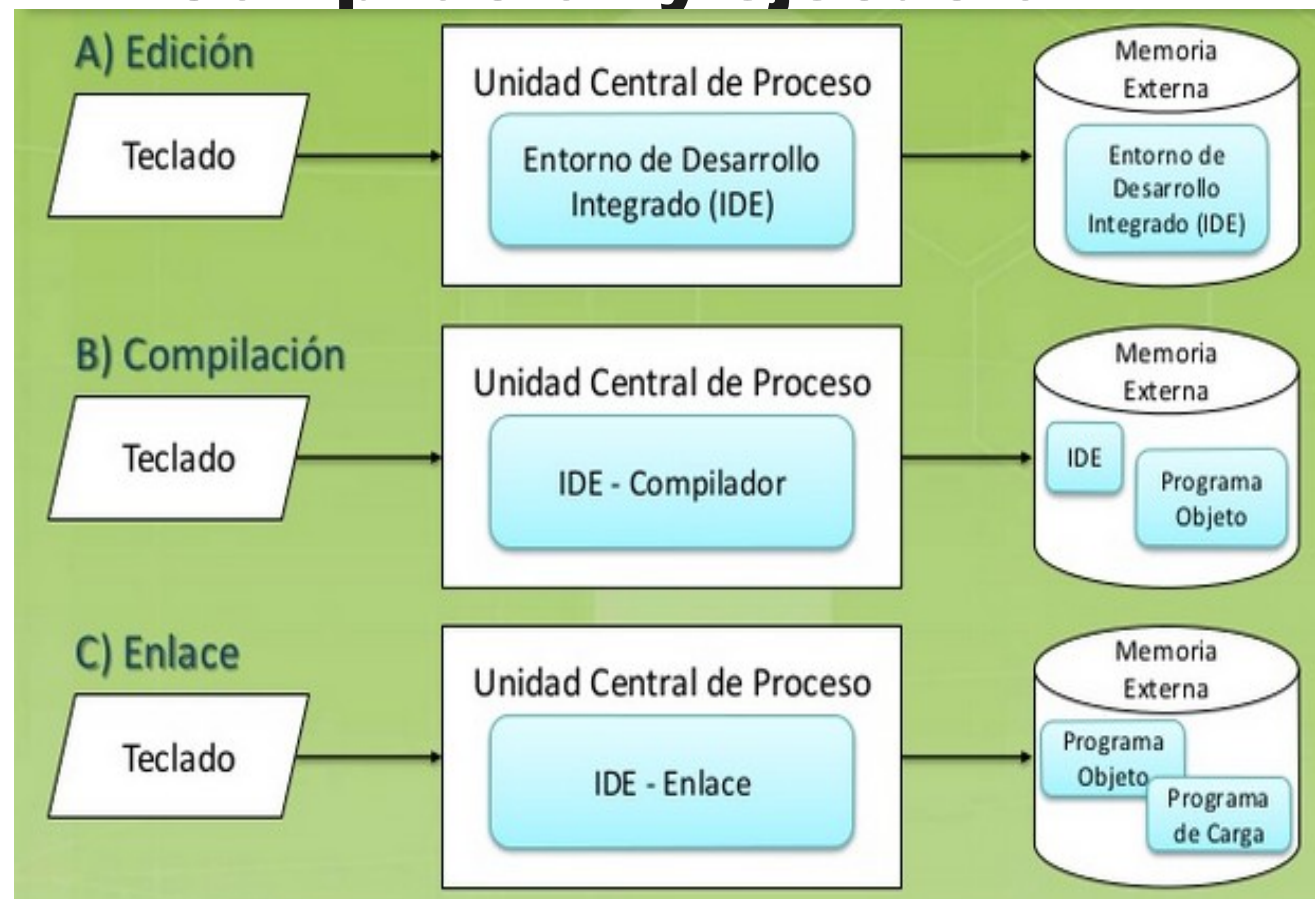
```
        }
```

```
    }
```

```
}
```

Etapas en el desarrollo de un programa

Compilación y ejecución



Etapas en el desarrollo de un programa

Verificación y depuración

- ❶ **Errores de Compilación.** *Se producen normalmente por un uso incorrecto de las reglas del lenguaje de programación y suelen ser errores de sintaxis.*
- ❷ **Errores de Ejecución.** *Estos errores se producen por instrucciones que la computadora puede comprender pero no ejecutar.*
- ❸ **Errores Lógicos.** *La fuente de estos errores suele ser el diseño del algoritmo. Estos errores son los más difíciles de detectar, en estos casos se debe volver a la fase de diseño.*

Etapas en el desarrollo de un programa

Documentación y mantenimiento

- ❶ **Documentación Interna.** *Es la contenida en las líneas del código fuente a manera de comentarios.*
- ❷ **Documentación Externa.** *Incluye análisis, diagramas de flujo y/o pseudocódigos, y manuales.*
- ❸ **Mantenimiento.** *La documentación es vital cuando se desean corregir posibles errores futuros o introducir cambios en el programa.*
- ❹ *Después de cada cambio debe ser actualizada la documentación.*
- ❺ *Es practica frecuente numerar las sucesivas versiones de los programas.*



¿Qué es un lenguaje de programación?

Es un lenguaje diseñado para hacer llegar al ordenador las diferentes tareas que tiene que realizar, y así poder llevar a cabo una determinada actividad.



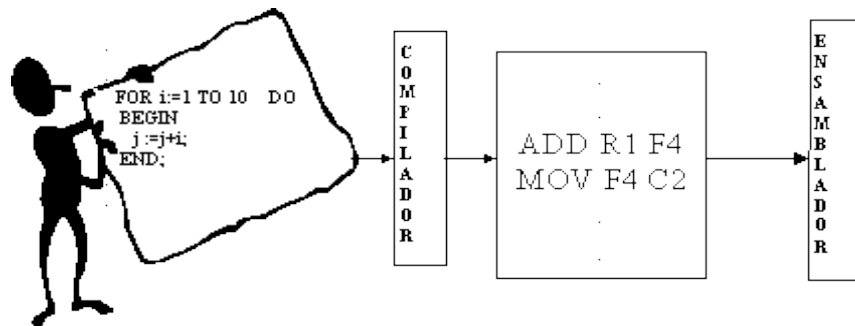
Clasificación de los lenguajes de programación

Según su nivel de abstracción:

✓ Lenguajes de bajo nivel

- ✓ máquina
- ✓ ensamblador

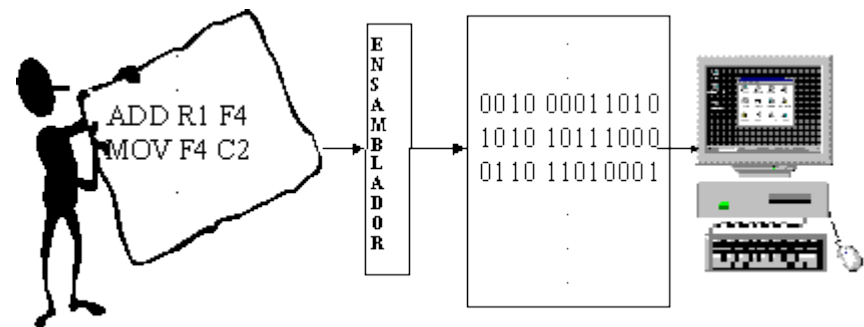
✓ Lenguajes de alto nivel



Según su forma de ejecución:

✓ Lenguajes compilados

✓ Lenguajes interpretados



Hola mundo



Ensamblador

```
.model small
.stack
.data
Cadenal DB 'Hola Mundo.$'
.code
programa:
    mov ax, @data
    mov ds, ax
    mov dx, offset Cadenal
    mov ah, 9
    int 21h
end programa
```

C

```
#include <stdio.h>
int main()
{
    printf("¡Hola, mundo!");
    return 0;
}
```

C++

```
#include <iostream>

int main()
{
    std::cout << "¡Hola, mundo!" << std::endl;
    return 0;
}
```

C#

```
using System;

class MainClass
{
    public static void Main()
    {
        Console.WriteLine("¡Hola mundo!");
    }
}
```

Kotlin

```
fun main(args: Array<String>) {
    println("Hello, world!")
}
```

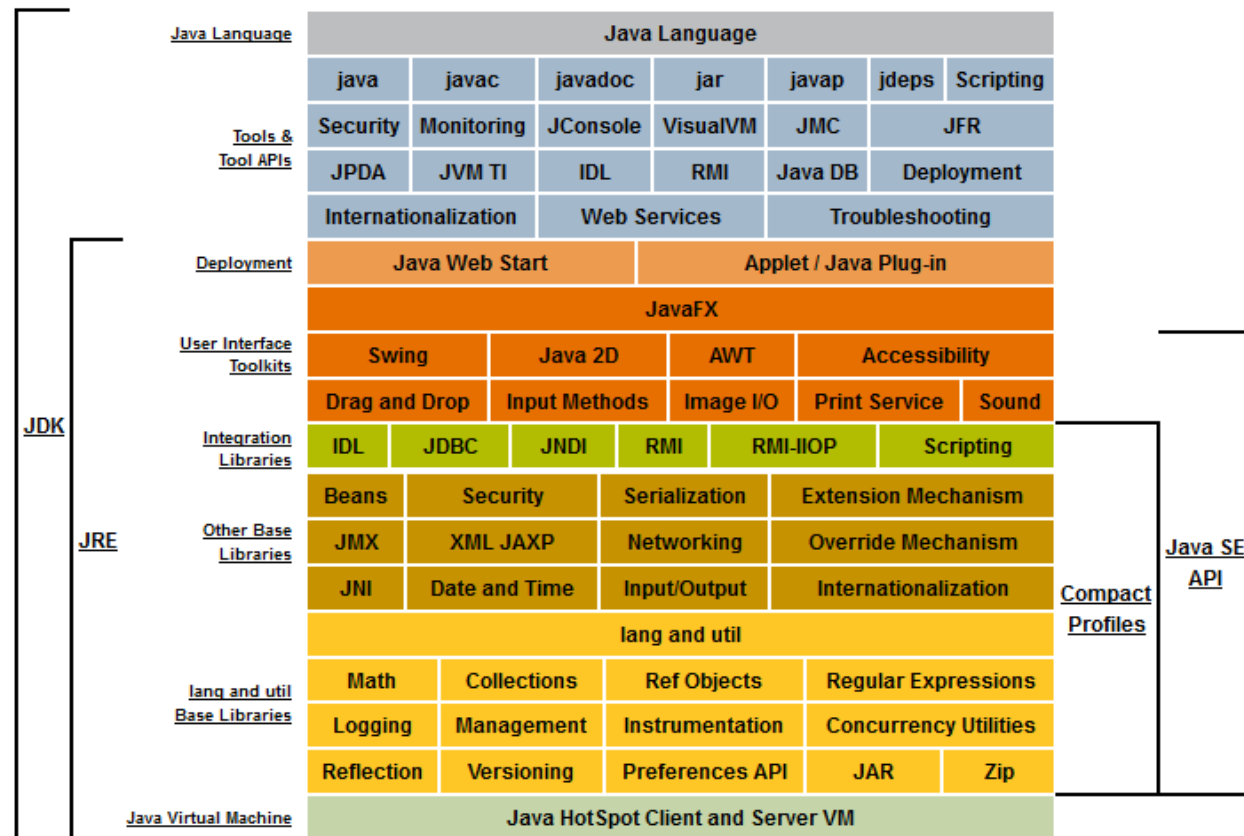
Python 3

```
print("¡Hola, mundo!")
```

¿Qué es Java?



Java es un lenguaje de programación y una plataforma informática.



¿Qué es Java?

Lenguaje de programación



Máquina virtual

- ✓ JVM (Java Virtual Machine).
- ✓ Encargada de ejecutar los programas.
- ✓ Proporciona un entorno de ejecución homogéneo e independiente de la plataforma.

Archivo de clase

- ✓ Encargado de almacenar los bytecodes, resultado de compilar los programas.
- ✓ Contiene instrucciones que no son específicas de ninguna máquina.
- ✓ Permite independencia de plataforma y movilidad en redes.

Interfaz de programación de aplicaciones

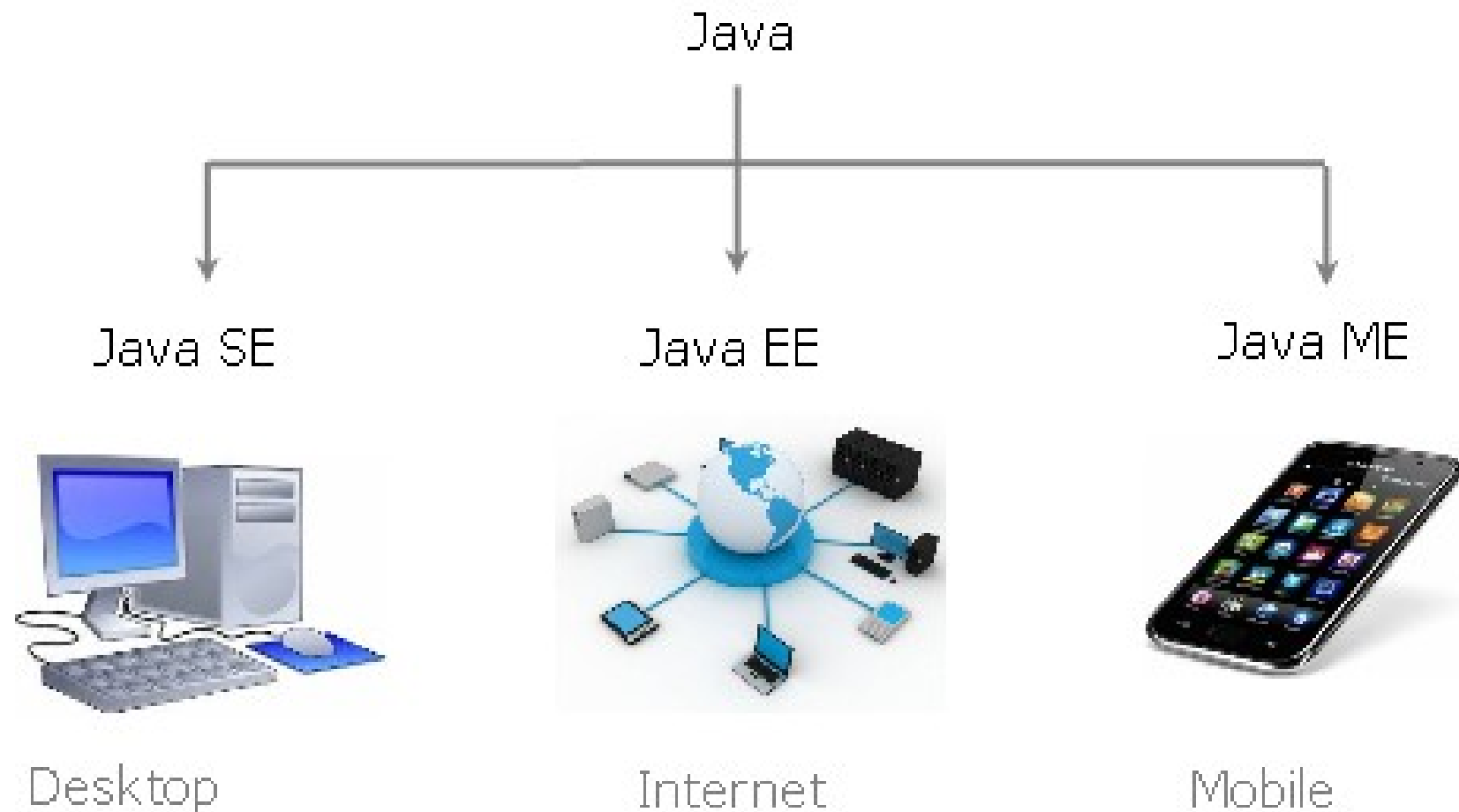
- ✓ API (Application Programming Interface).
- ✓ Conjunto de bibliotecas que ofrecen una vía estándar de acceso a los recursos del sistema.

Características de java



Simple	Orientado a objetos
Distribuido	Interpretado y compilado
Seguro	Independiente de la arquitectura
Portable	Multihilo
Dinámico	Rápido (JIT)

Ediciones de Java



Versiones de Java



Version	Type	Class file format version ^[7]	Release date	End of public updates (free)	End of extended support (paid)
JDK 1.0		44	23rd January 1996	May 1996	—
JDK 1.1		45	18th February 1997	October 2002	—
J2SE 1.2		46	4th December 1998	November 2003	—
J2SE 1.3		47	8th May 2000	March 2006	—
J2SE 1.4		48	13th February 2002	October 2008	—
J2SE 5.0 (1.5)		49	30th September 2004	October 2009	—
Java SE 6 (1.6)		50	11th December 2006	April 2013	December 2016 for Red Hat ^[8] October 2018 for Oracle ^[9] March 2026 for BellSoft Liberica ^[10] December 2027 for Azul ^[11]
Java SE 7 (1.7)		51	28th July 2011	July 2015	June 2020 for Red Hat ^[8] July 2022 for Oracle ^[12] March 2026 for BellSoft Liberica ^[10] December 2027 for Azul ^[11]
Java SE 8 (1.8)	LTS	52	18th March 2014	April 2019 for Oracle November 2026 for Eclipse Temurin ^[13] November 2026 for Red Hat ^[8] July 2026 for Amazon Corretto ^[14] December 2030 for Azul ^[11] March 2031 for BellSoft Liberica ^[10]	December 2030 for Oracle ^[4]
Java SE 9 (1.9)		53	21st September 2017	March 2018	—
Java SE 10 (1.10)		54	20th March 2018	September 2018	—
Java SE 11	LTS	55	25th September 2018	April 2019 for Oracle October 2024 for Red Hat ^[8] March 2027 for BellSoft Liberica ^[10] September 2027 for Microsoft Build of OpenJDK ^[15] October 2027 for Eclipse Temurin ^[13] October 2027 for Amazon Corretto ^[14] January 2032 for Azul ^[11]	January 2032 for Oracle ^[4]
Java SE 12		56	19th March 2019	September 2019	—
Java SE 13		57	17th September 2019	March 2020	—

Java SE 14		58	17th March 2020	September 2020	—
Java SE 15		59	16th September 2020	March 2021	—
Java SE 16		60	16th March 2021	September 2021	—
Java SE 17	LTS	61	14th September 2021	September 2024 for Oracle ^[4] September 2027 for Microsoft Build of OpenJDK ^[15] October 2027 for Eclipse Temurin ^[13] October 2027 for Red Hat ^[8] October 2029 for Amazon Corretto ^[14] September 2029 for Azul ^[11] March 2030 for BellSoft Liberica ^[10]	September 2029 for Oracle ^[4]
Java SE 18		62	22nd March 2022	September 2022	—
Java SE 19		63	20th September 2022	March 2023	—
Java SE 20		64	21st March 2023	September 2023	—
Java SE 21	LTS	65	19th September 2023	September 2028 for Oracle ^[4] September 2028 for Microsoft Build of OpenJDK ^[15] December 2029 for Red Hat ^[8] December 2029 for Eclipse Temurin ^[13] October 2030 for Amazon Corretto ^[14] September 2031 for Azul ^[11] March 2032 for BellSoft Liberica ^[10]	September 2031 for Oracle ^[4]
Java SE 22		66	19th March 2024	September 2024	—
Java SE 23		67	September 2024	March 2025	—
Java SE 24		68	March 2025	September 2025	—
Java SE 25	LTS	69	September 2025	September 2030 for Oracle ^[4]	September 2033 for Oracle ^[4]

Legend: Old version Older version, still maintained Latest version Future release

JDK – Java Development Kit



Software que provee herramientas de desarrollo para la creación de programas Java.

- ✓ **Compilador:** **javac**
- ✓ **Intérprete:** **java**
- ✓ **Generador de documentación:** **javadoc**
- ✓ **Depurador:** **jdb**
- ✓ **Utilidad para crear archivos comprimidos:** **jar**
- ✓ **Visor de applets:** **appletviewer**
- ✓ **Desensamblador de clases:** **javap**

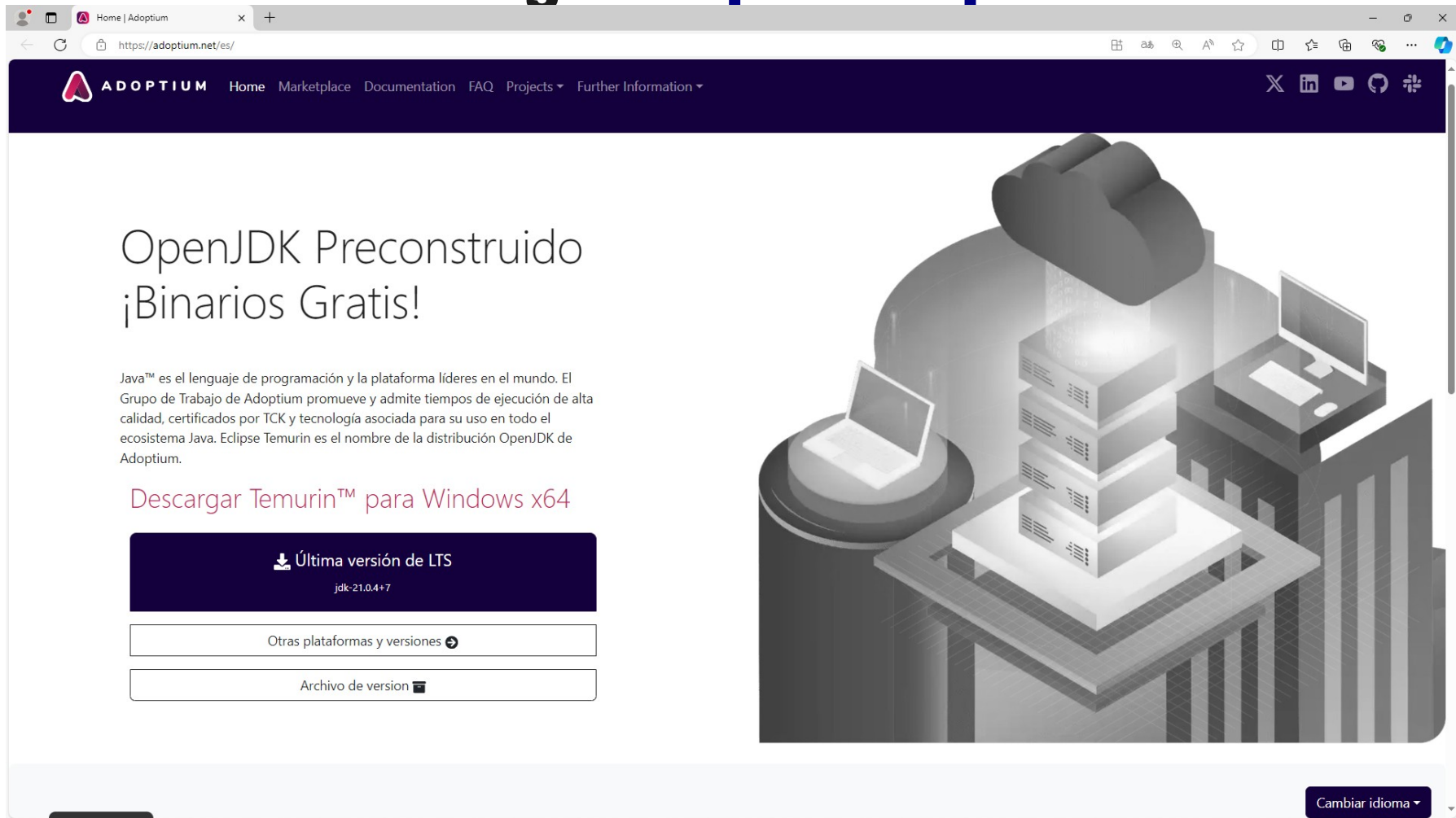
JDK – Java Development Kit

Distribuciones:

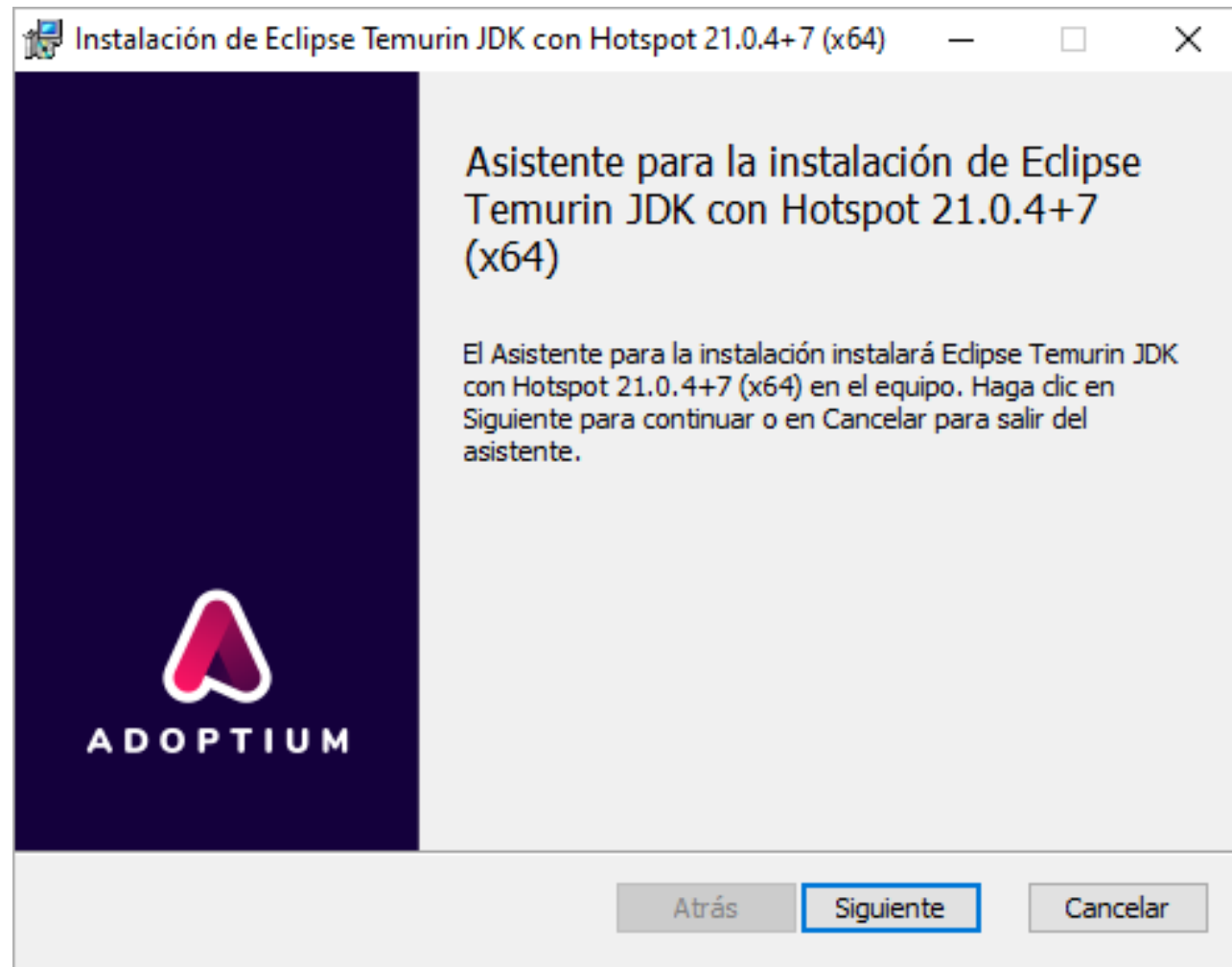
- ✓ **Oracle JDK**
- ✓ **Oracle OpenJDK**
- ✓ **Distribuciones OpenJDK de otros proveedores (<https://javaalmanac.io/>)**
 - ✓ **Adoptium** (sucesor de AdoptOpenJDK)
 - ✓ **Amazon – Corretto**
 - ✓ **Azul Zulu**
 - ✓ **IBM**
 - ✓ **RedHat**

Descarga del JDK

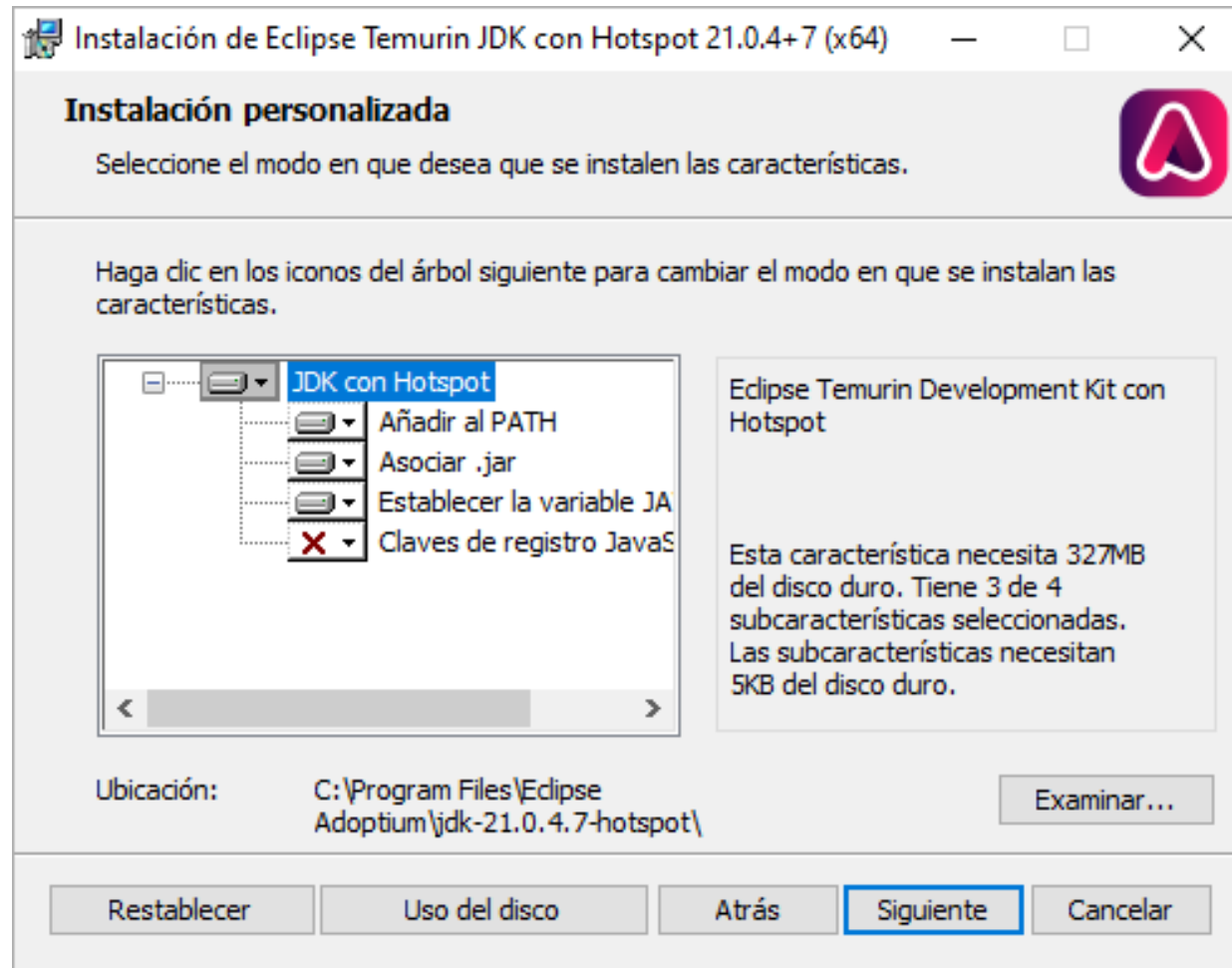
Enlace de descarga: <https://adoptium.net/>



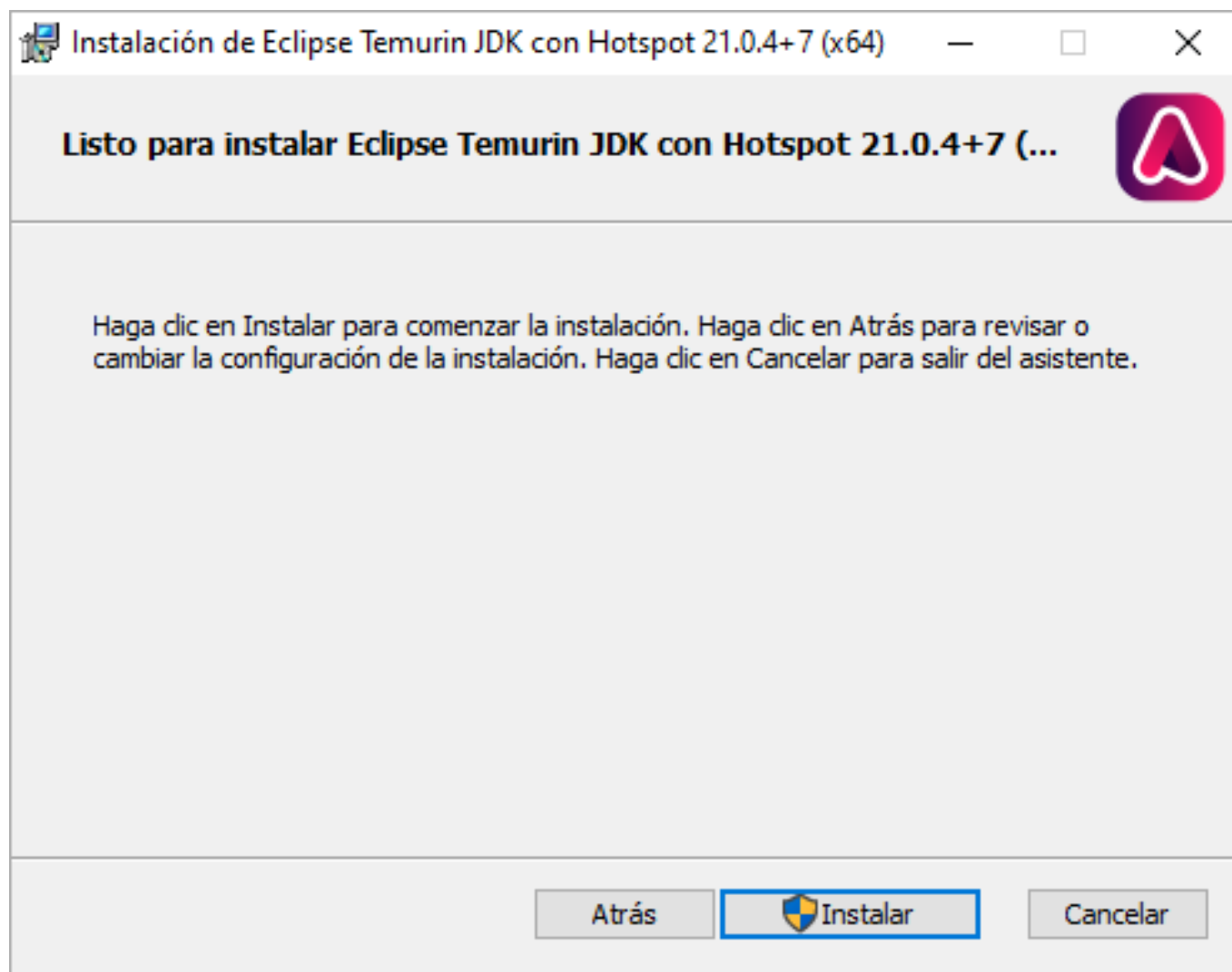
Instalación del JDK



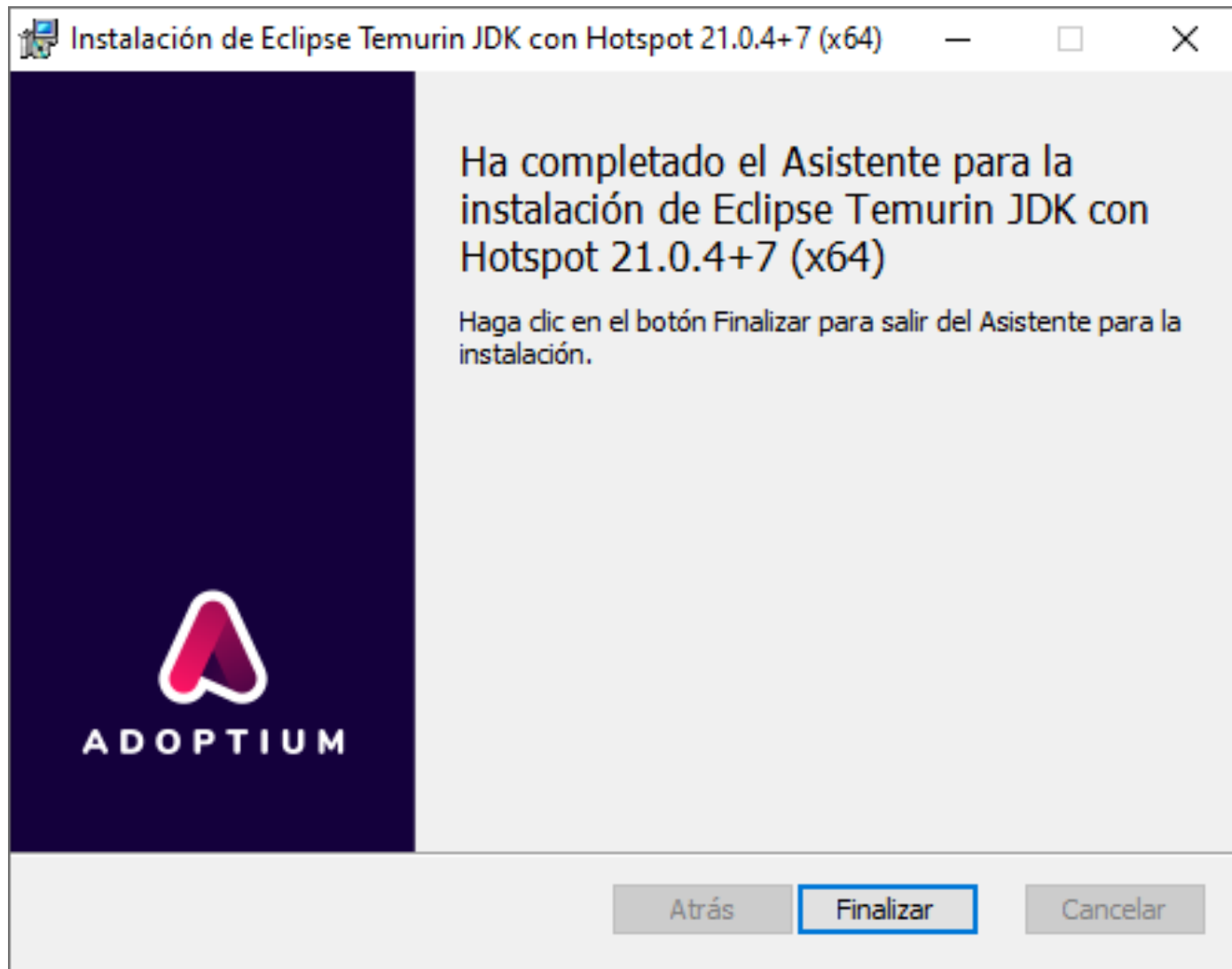
Instalación del JDK



Instalación del JDK

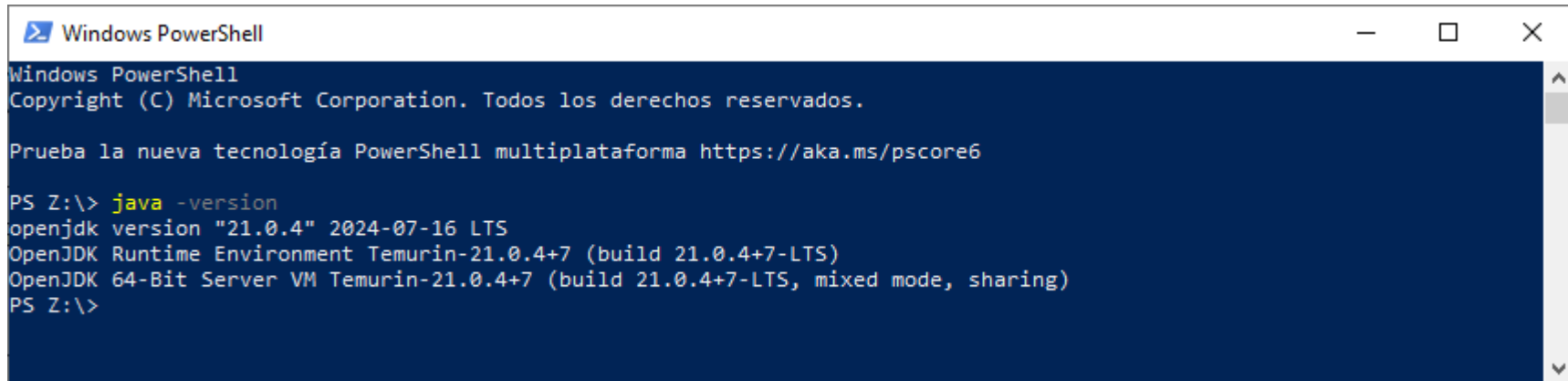


Instalación del JDK



Instalación del JDK

Comprobamos la versión del JKD instalada:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

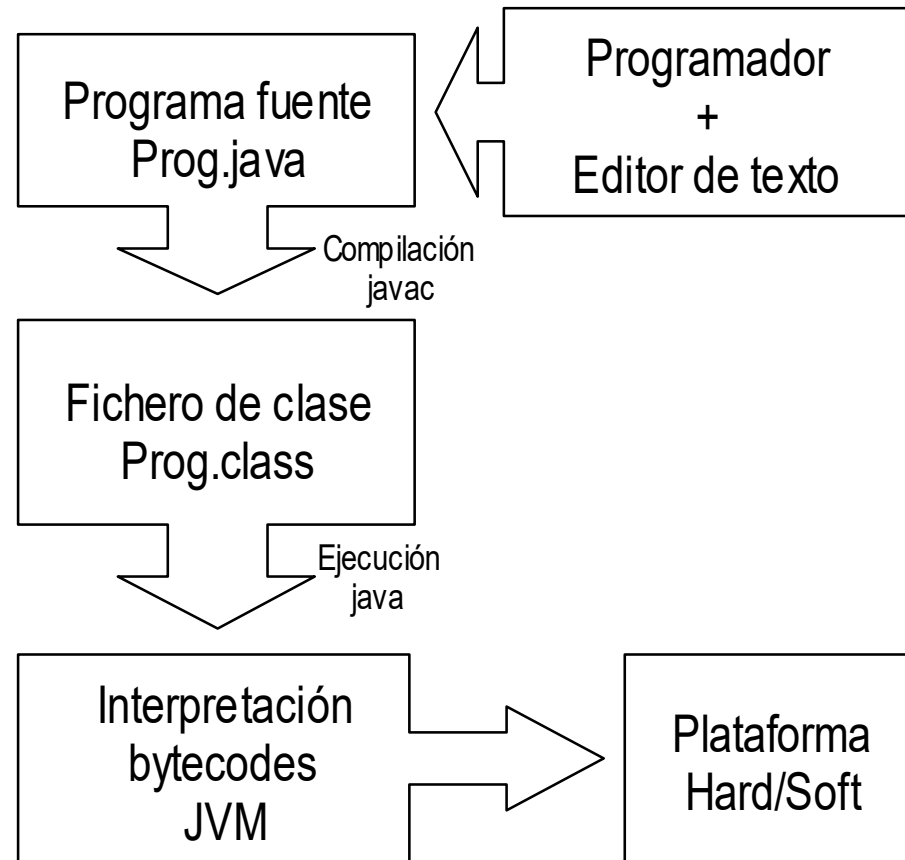
PS Z:\> java -version
openjdk version "21.0.4" 2024-07-16 LTS
OpenJDK Runtime Environment Temurin-21.0.4+7 (build 21.0.4+7-LTS)
OpenJDK 64-Bit Server VM Temurin-21.0.4+7 (build 21.0.4+7-LTS, mixed mode, sharing)
PS Z:\>
```

Primer programa



- ✓ **Escribir en un fichero de texto un programa respetando la sintaxis del lenguaje.**
- ✓ **Grabarlo con la extensión java.**
- ✓ **Compilar (javac). Se traduce a bytecode.**
- ✓ **El fichero generado tiene extensión class y el mismo nombre que el fichero compilado.**
- ✓ **Ejecutar (java). La máquina virtual es la encargada de cargar los ficheros de clase y ejecutarlos.**

Primer programa



Primer programa

Edita el texto siguiente:

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

Grábalo con el nombre: **HolaMundo.java**

Compila el fichero fuente: **javac HolaMundo.java**

Generará un nuevo fichero: **HolaMundo.class**

Ejecuta el programa: **java HolaMundo**

Primer programa

Desde Java 11 se añade la posibilidad de ejecutar un archivo fuente directamente sin necesidad de compilarlo previamente con javac.

Para ejecutar el programa: `java HolaMundo.java`

Entornos de desarrollo integrado

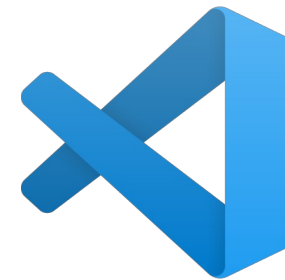


Aplicación informática que proporciona servicios integrales para facilitarle al programador el desarrollo de software.

Normalmente, un IDE consiste en un editor de código fuente, herramientas de construcción automáticas y un depurador.



Apache NetBeans



eclipse

Entornos de desarrollo integrado



CODEN**ENVY**



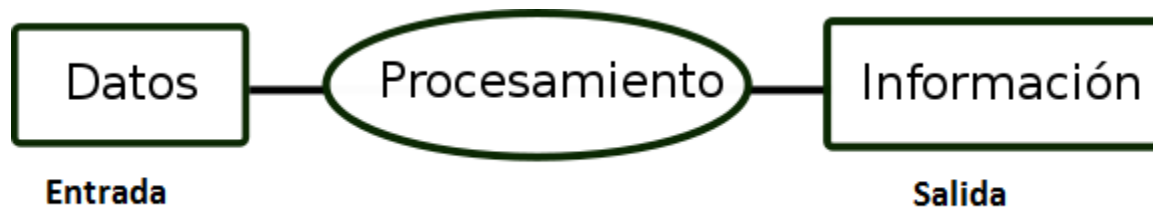
jGRASP

¿Qué es un dato?



Es información dispuesta de manera adecuada para su tratamiento por un ordenador.

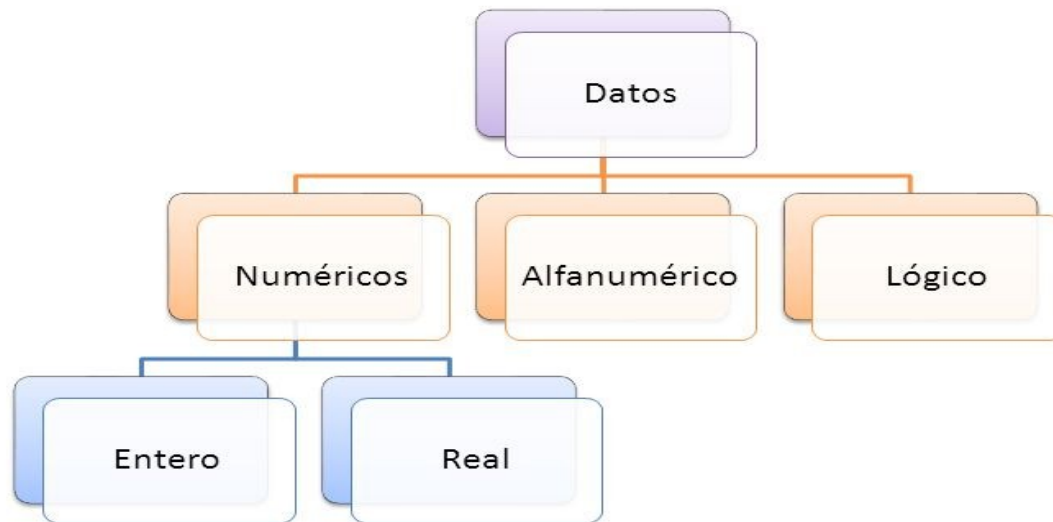
Es la unidad mínima de almacenamiento con la que puede trabajar un programa.



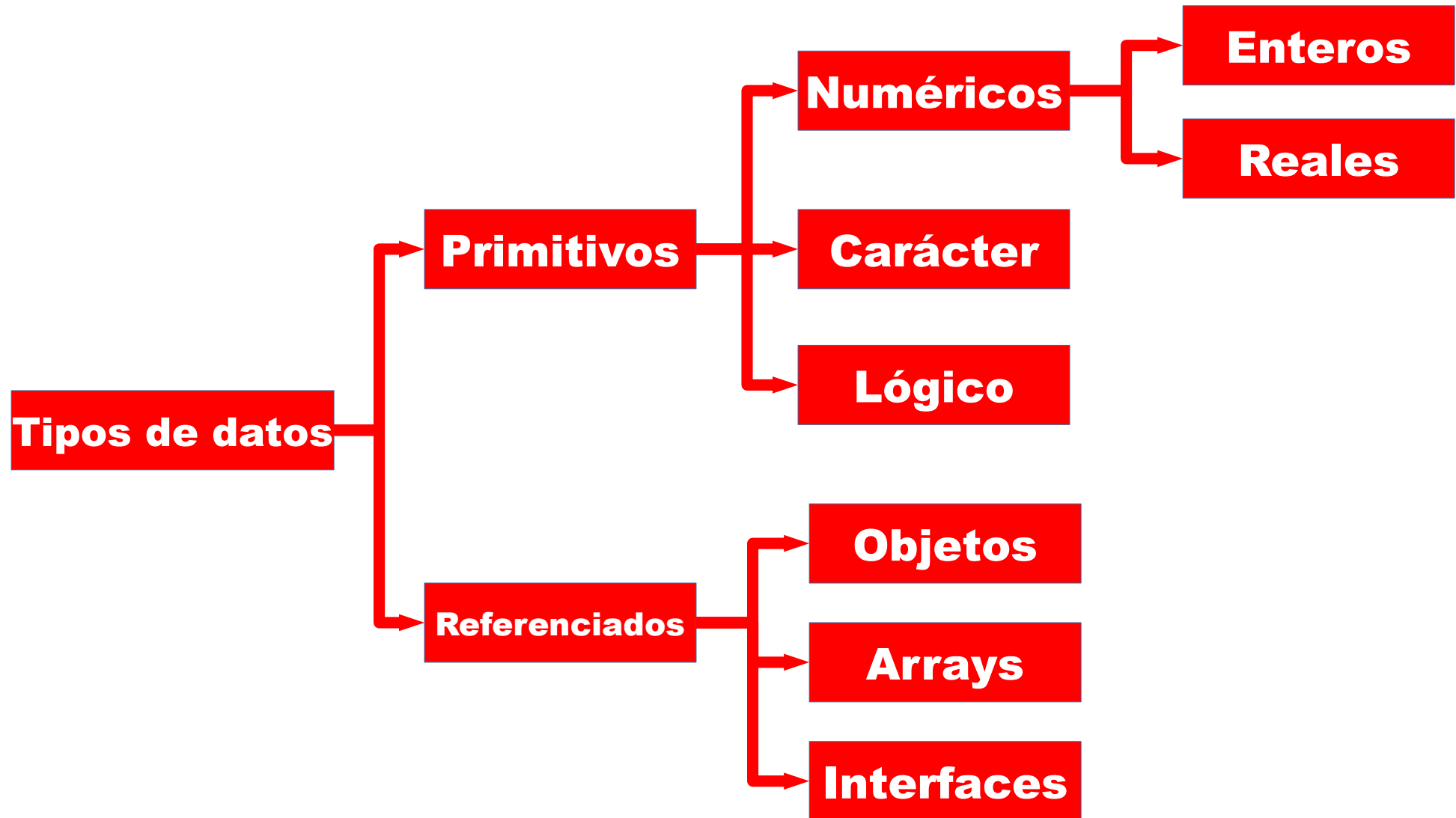


Conjunto de valores distintos que puede tomar un dato

TIPOS DE DATOS PRIMITIVOS



Tipos de datos en Java



Tipos de datos en Java

TIPO	BYTES	RANGO
boolean	-	true, false
byte	1	-128 a 127
short	2	-32768 a 32767
int	4	-2147483648 a 2147483647
long	8	-9223372036854775808 a 9223372036854775807
float	4	+/-1.4E-45 a +/-3.4028235E+38
double	8	+/-4.9E-324 to +/-1.7976931348623157E+308
char	2	UNICODE

Literales en Java

Un literal es una notación para representar un determinado valor en el código fuente.

TIPO	DESCRIPCIÓN	EJEMPLOS
boolean		true false
char	Carácter entre comillas simples	'a' 'A' '5' '\n'
byte, short, int	Entero en decimal Entero en octal Entero en hexadecimal Entero en binario	267 0413 0x10B 0b100001011
long	Entero con sufijo L o l	764398609800L
double	Número real con punto decimal Con notación científica	120.1762 1.201762e2
float	Misma notación que double pero con sufijo F o f	20.129F
String	Cadena entre comillas dobles	"Hola mundo!"

Identificadores en Java

Los identificadores son símbolos que nombran entidades del lenguaje (constantes, variables, métodos, clases, ...).

Sintaxis de un identificador en Java:

- ‘Debe estar formado por uno o más caracteres.**
- ‘No se pueden utilizar espacios en blanco.**
- ‘El primer carácter debe ser una letra, el carácter de subrayado (_) o el carácter dólar (\$).**
- ‘No puede coincidir con ninguna de las palabras reservadas.**

Identificadores en Java

Convención de nombres para identificadores Java:

- ‘Las constantes se escriben en mayúsculas. Si el identificador consta de más de una palabra se separarán mediante un guion bajo (_).
- ‘Las variables se escriben en **lowerCamelCase**. No deben comenzar con guion bajo (_) o el carácter dólar (\$). La elección de nombre debe ser mnemónico.
- ‘Los métodos deben ser verbos o comenzar con un verbo seguido de otras palabras. Se escriben en **lowerCamelCase**.
- ‘Las clases deben ser sustantivos en **UpperCamelCase**.

Palabras reservadas en Java



List of Java Keywords

Primitive Types and void

- 1.boolean
- 2.byte
- 3.char
- 4.short
- 5.int
- 6.long
- 7.float
- 8.double
- 9.void

Modifiers

- 1.public
- 2.protected
- 3.private
- 4.abstract
- 5.static
- 6.final
- 7.transient
- 8.volatile
- 9.synchronized
- 10.native

Declarations

- 1.class
- 2.interface
- 3.enum
- 4.extends
- 5.implements
- 6.package
- 7.throws

Control Flow

- 1.if
- 2.else
- 3.try
- 4.catch
- 5.finally
- 6.do
- 7.while
- 8.for
- 9.continue
- 10.break
- 11.switch
- 12.case
- 13.default
- 14.throw
- 15.return

Miscellaneous

- 1.this
- 2.new
- 3.super
- 4.import
- 5 instanceof
- 6.null
- 7.true
- 8.false
- 9.strictfp
- 10.assert
- 11._ (underscore)
- 12.goto
- 13.const

Variables en Java

Una variable está formada por un espacio de memoria y un identificador que está asociado a dicho espacio.

Ese espacio de memoria contiene un valor.

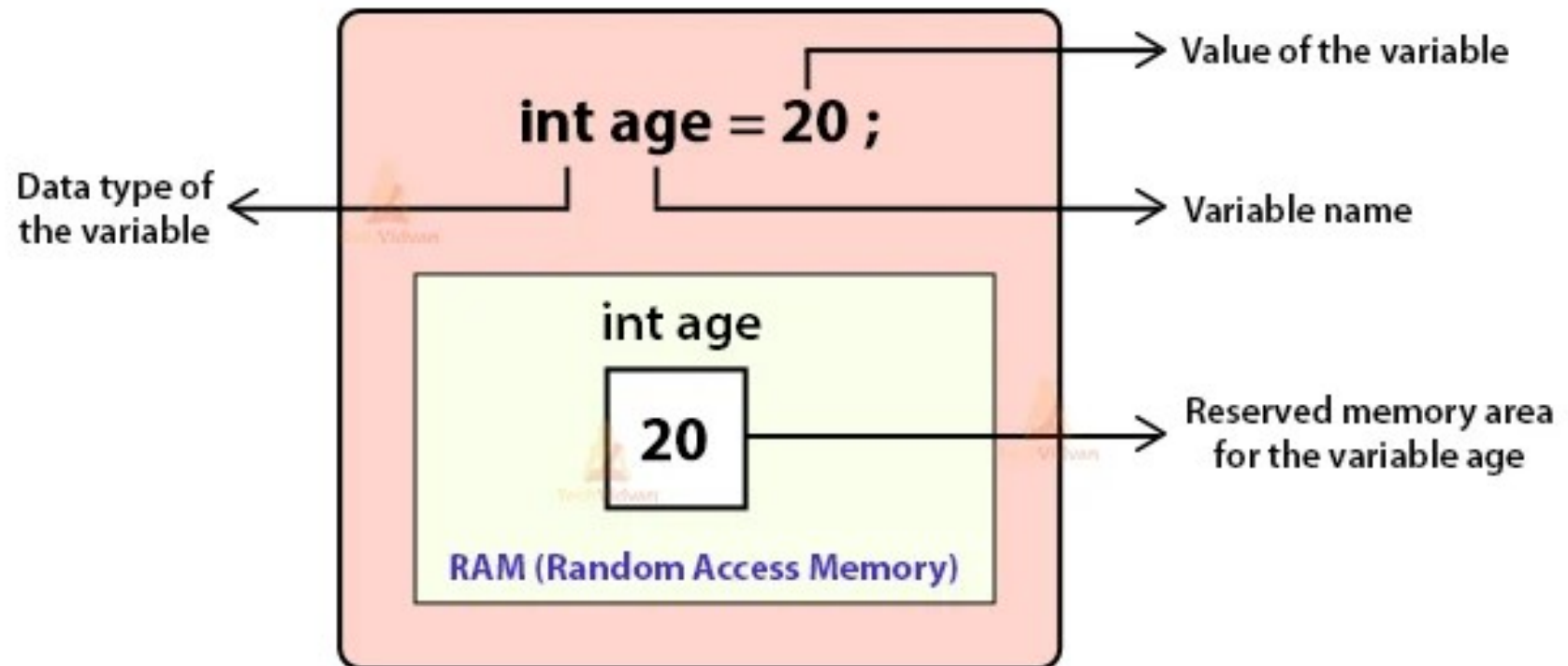
El nombre de la variable nos permitirá referirnos al valor almacenado.

El valor de la variable puede cambiar durante la ejecución del programa.

Una variable tiene un ámbito que define su visibilidad y una duración.

Variables en Java

Java Variable Declaration & its Memory Allocation



Variables en Java

Una variable tiene un ámbito que define su visibilidad y una duración.

Tipo de variable	Ámbito	Duración
Variable de instancia	Toda la clase excepto los métodos estáticos	Hasta que el objeto esté disponible en memoria
Variable de clase	Toda la clase	Hasta el final del programa
Variable local	Dentro del bloque en el que se declara	Hasta que el control sale del bloque en el que se declara

Variables en Java

Antes de utilizar una variable tenemos que declararla.

Sintaxis de la declaración de una variable:

[modificador] tipo_de_dato identificador [= valor];

Modificadores

static, final, public, protected, private

Ejemplos

```
boolean respuesta = true;  
int age = 20;  
char letra = 'a';
```

Variables en Java

```
public class Variables{  
    public static void main(String[] args){  
        String cadena="Cadena de caracteres";  
        int entero=10;  
        float real=1.75f;  
        boolean booleano=true;  
        System.out.print("La variable cadena contiene: ");  
        System.out.println(cadena);  
        System.out.print("La variable entero contiene: ");  
        System.out.println(entero);  
        System.out.print("La variable real contiene: ");  
        System.out.println(real);  
        System.out.print("La variable booleano contiene: ");  
        System.out.println(booleano);  
    }  
}
```

Variables en Java

Inferencia de tipos para variable locales.

Desde la versión 10 de Java disponemos de la palabra reservada **var** que permite no indicar el tipo de la variable al declararlo. El compilador inferirá el tipo en función de la inicialización de la variable.

No podrá utilizarse para variables que no se inicialicen en tiempo de declaración.

```
var identificador = valor;
```

Ejemplos

```
var cadena = "Cadena de caracteres";  
var character = 'a';  
var entero = 10;  
var logico = true;  
var flotante = 3.14f;
```

Constantes en Java

Una constante es un espacio de memoria identificado por un nombre cuyo valor permanece invariable durante la ejecución de un programa.

Para declarar una constante en Java añadiremos el modificador *final* a la declaración.

Ejemplo:

```
final float PI = 3.1415f;
```

Constantes en Java

```
public class Constantes{  
    public static void main(String[] args){  
        final float PI=3.1415f;  
  
        System.out.print("La constante PI contiene: ");  
        System.out.println(PI);  
    }  
}
```

Expresiones

Una expresión es una combinación de uno o más literales, variables, constantes y operadores cumpliendo unas determinadas reglas de construcción.

Las expresiones serán evaluadas de acuerdo a unas normas de precedencia y asociación.

Las expresiones, según el resultado que produzca la evaluación, se clasifican en:

- ✓ **Numéricas**
- ✓ **Alfanuméricas**
- ✓ **Booleanas**

Operadores en Java

Un operador lleva a cabo operaciones sobre un operando (operador unario), dos operandos (operador binario) o tres operandos (operador ternario).

Tipos de operadores en Java:

- ✓ **Aritméticos**
- ✓ **Asignación**
- ✓ **Relacionales**
- ✓ **Lógicos**
- ✓ **Concatenación**
- ✓ **Condicional**
- ✓ **A nivel de bit**

Operadores en Java

OPERADORES ARITMÉTICOS		
OPERADOR	SIGNIFICADO	USO
+	Suma	12 + 10
-	Resta	12 - 10
*	Multiplicación	12 * 10
/	División	12 / 10
%	Resto de la división	12 % 10
++	Incrementa el valor del operando en 1.	++12 12++
--	Decrementa el valor del operando en 1.	--12 12--

Operadores en Java

```
public class OperadoresAritmeticos{  
    public static void main(String[] args){  
        int entero1=10,entero2=5;  
        int suma, resta, multiplicacion, division, resto;  
  
        suma = entero1 + entero2;  
        resta = entero1 - entero2;  
        multiplicacion = entero1 * entero2;  
        division = entero1 / entero2;  
        resto = entero1 % entero2;  
        System.out.println("Suma= "+suma);  
        System.out.println("Resta= "+resta);  
        System.out.println("Multiplicación= "+multiplicacion);  
        System.out.println("División= "+division);  
        System.out.println("Resto= "+resto);  
    }  
}
```

Operadores en Java

```
public class Incremento{  
    public static void main(String[] args){  
        int entero=10;  
        int preInc, postInc;  
  
        System.out.println("Valor de entero= "+entero);  
  
        preInc=++entero;  
        System.out.println("Valor de entero= "+entero);  
        System.out.println("Valor de preInc= "+preInc);  
  
        postInc=entero++;  
        System.out.println("Valor de entero= "+entero);  
        System.out.println("Valor de postInc= "+postInc);  
    }  
}
```

Operadores en Java

```
public class Decremento{  
    public static void main(String[] args){  
        int entero=10;  
        int preDec, postDec;  
  
        System.out.println("Valor de entero= "+entero);  
  
        preDec=--entero;  
        System.out.println("Valor de entero= "+entero);  
        System.out.println("Valor de preDec= "+preDec);  
  
        postDec=entero--;  
        System.out.println("Valor de entero= "+entero);  
        System.out.println("Valor de postDec= "+postDec);  
    }  
}
```

Operadores en Java

OPERADORES ASIGNACIÓN		
OPERADOR	SIGNIFICADO	USO
=	Asignación	variable = expresión
+=	Suma	a+=b (equivalente a=a+b)
-=	Resta	a-=b (equivalente a=a-b)
=	Multiplicación	a=b (equivalente a=a*b)
/=	División	a/=b (equivalente a=a/b)
%=	Resto	a%=b (equivalente a=a%b)

Operadores en Java

```
public class OperadoresAsignacion{  
    public static void main(String[] args){  
        int entero=10;  
  
        System.out.println("entero= "+entero);  
        entero += 5;  
        System.out.println("entero= "+entero);  
        entero -= 5;  
        System.out.println("entero= "+entero);  
        entero *= 5;  
        System.out.println("entero= "+entero);  
        entero /=5;  
        System.out.println("entero= "+entero);  
    }  
}
```

Operadores en Java

OPERADORES RELACIONALES		
OPERADOR	SIGNIFICADO	USO
==	Igual que	$a == b$
!=	Distinto	$a != b$
<	Menor que	$a < b$
<=	Menor o igual que	$a <= b$
>	Mayor que	$a > b$
>=	Mayor o igual que	$a >= b$

Operadores en Java

```
public class OperadoresRelacionales{  
    public static void main(String[] args){  
        int entero1=10, entero2=5;  
  
        System.out.println(entero1 == entero2);  
        System.out.println(entero1 != entero2);  
        System.out.println(entero1 < entero2);  
        System.out.println(entero1 <= entero2);  
        System.out.println(entero1 > entero2);  
        System.out.println(entero1 >= entero2);  
    }  
}
```

Operadores en Java

OPERADORES LÓGICOS

OPERADOR	SIGNIFICADO	USO
&&	AND	a && b
 	OR	a b
!	NOT	!a

OTROS OPERADORES

OPERADOR	SIGNIFICADO	USO
+	Concatenación	"Hola " + "mundo"
?:	Operador condicional	exp_lógica ? exp1 : exp2

Operadores en Java

A	B	A && B	A B	!A
true	true	true	true	false
true	false	false	true	false
false	true	false	true	true
false	false	false	false	true

Operadores en Java

```
public class OperadorCondicional{  
    public static void main(String[] args){  
        int entero1=10, entero2=5;  
  
        System.out.print(entero1 > entero2 ? entero1 : entero2);  
    }  
}
```

Separadores en Java

SEPARADOR	DESCRIPCIÓN
()	Permite modificar la prioridad de una expresión, contener expresiones para el control de flujo y realizar conversiones de tipo. Pueden contener la lista de parámetros o argumentos, tanto en la definición de un método como en la llamada al mismo.
{ }	Permite definir bloques de código y ámbitos y contener los valores iniciales de las variable array.
[]	Permite declarar variables array y referenciar sus elementos.
;	Permite separar sentencias.
,	Permite separar identificadores consecutivos en la declaración de variables y en las listas de parámetros. También se emplea para encadenar sentencias dentro de un bucle for.
.	Permite separar el nombre del atributo o método de su instancia de referencia. También separa el identificador de un paquete de los de los subpaquetes y clases.

Precedencia de operadores

OPERADOR	OPERACIÓN
++ - - !	Incremento (unario), decremento (unario) y negación (unario)
* / %	Multiplicación, división y resto.
+ - +	Suma, resta y concatenación.
< <= > >=	Menor que, menor o igual que, mayor que, mayor o igual que.
== !=	Igual que, distinto.
&&	AND lógico.
 	OR lógico.
= += -= *= /=	Asignación.

Conversión de tipos primitivos en Java

En Java es posible transformar el tipo de una variable en otro diferente al original con el que fue declarado.

Este proceso se denomina conversión de tipos.

Las conversiones pueden ser implícitas o explícitas.

Conversión implícita

Se tiene que cumplir lo siguiente:

Los dos tipos son compatibles.

El tipo de destino tenga un rango mayor que el de origen.

Conversión explícita (casting)

```
variable_destino = (tipo_variable_destino) expresión_origen
```

Conversión de tipos primitivos en Java

```
public class ConversionesTipos{  
    public static void main(String[] args){  
        int entero=10;  
        float real=4.5f;  
  
        //conversión implícita  
  
        real=entero;  
        System.out.println("real= "+real);  
  
        // conversión explícita  
  
        entero=(int)real;  
        System.out.println("entero= "+entero);  
    }  
}
```

Comentarios en Java

Los comentarios son mensajes incluidos en el código fuente destinados a facilitar el mantenimiento o la reutilización del código.

Comentarios de línea

// comentario de línea

Comentarios de bloque

**/* comentarios que pueden
ocupar varias líneas */**

Comentarios de documentación

/ comentarios utilizados
por javadoc */**

Salida por pantalla en Java

Hay distintas formas de visualizar mensajes por pantalla.

Para mostrar por pantalla el resultado de evaluar una expresión.

```
System.out.print(expresión);
```

Para mostrar por pantalla el resultado de evaluar una expresión y añadir un salto de línea.

```
System.out.println(expresión);
```


Entrada por teclado en Java

Clase Scanner

```
import java.util.Scanner;
public class EntradaTeclado{
    public static void main(String[] args){
        Scanner teclado=new Scanner(System.in);
        String nombre;
        System.out.print("Introduce nombre: ");
        nombre=teclado.nextLine();
        System.out.println("Hola "+nombre);
    }
}
```

Entrada por teclado en Java

Clase Scanner

- ✓ **Para leer un número entero:** `teclado.nextInt()`
- ✓ **Para leer un número real:** `teclado.nextFloat()`
- ✓ **Para leer una línea:** `teclado.nextLine()`

Entrada por teclado en Java

Clase Scanner

```
import java.util.Scanner;
public class EntradaTeclado2{
    public static void main(String[] args){
        Scanner teclado=new Scanner(System.in);
        int entero;
        System.out.print("Introduce entero: ");
        entero=teclado.nextInt();
        System.out.println("Has introducido: "+entero);
    }
}
```

Entrada por teclado en Java

Clase BufferedReader

```
import java.io.*;
public class EntradaTeclado3{
    public static void main(String[] args) throws IOException{
        InputStreamReader flujo=new InputStreamReader(System.in);
        BufferedReader teclado=new BufferedReader(flujo);
        String nombre;
        System.out.print("Introduce nombre: ");
        nombre=teclado.readLine();
        System.out.println("Hola "+nombre);
    }
}
```

Entrada por teclado en Java

Clase BufferedReader

La lectura realizada siempre será de una cadena.

Si queremos leer un número deberemos convertir la cadena.

Conversión de cadenas en números:

Byte.parseByte(cadena)

Short.parseShort(cadena)

Integer.parseInt(cadena)

Long.parseLong(cadena)

Float.parseFloat(cadena)

Double.parseDouble(cadena)

Entrada por teclado en Java

Clase BufferedReader

```
import java.io.*;
public class EntradaTeclado4{
    public static void main(String[] args) throws IOException{
        InputStreamReader flujo=new InputStreamReader(System.in);
        BufferedReader teclado=new BufferedReader(flujo);
        String cadena;
        int entero;
        System.out.print("Introduce entero: ");
        cadena=teclado.readLine();
        entero=Integer.parseInt(cadena);
        System.out.println(entero);
    }
}
```