



Qu'est ce qu'un algorithme?

L'algorithme de programmation est une liste qui nous permet d'écrire des instructions pour résoudre un problème.

Qu'est ce qu'un langage de programmation?

Un langage de programmation est une langue qui nous permet d'écrire des instructions pour résoudre un problème.

Qu'est ce qu'un programmeur?

Un programmeur est une personne qui sait écrire des algorithmes et des programmes.

Qu'est ce qu'un logiciel?

Un logiciel est un ensemble d'algorithmes et de programmes qui sont exécutés par un ordinateur.

Mais alors que faire avec tous ces algorithmes et programmes? Il faut savoir comment utiliser les algorithmes et programmes pour résoudre un problème.

Il existe plusieurs langages de programmation.

¿ Que es un algoritmo ?

¿ Que es un lenguaje de programación ?

¿ Hay diferencias entre uno y otro ?



**Lenguaje de programación es
aquel que me permite darle
instrucciones precisas a un
procesador o computadora para que
haga algo de valor**

**Nuevo teclado
profesional para
programadores**



Mientras que algoritmo es la versión abstracta y matemática de los lenguaje de programación

Algoritmo es una secuencia de pasos que me solucionan un problema.

Programa es una secuencia de comandos que le dicen a una computadora como resolver un problema.

Lenguajes de programación

Se clasifican en

De máquina

De bajo nivel
(ensamblador)

De alto nivel

Ejemplo

Pascal

C

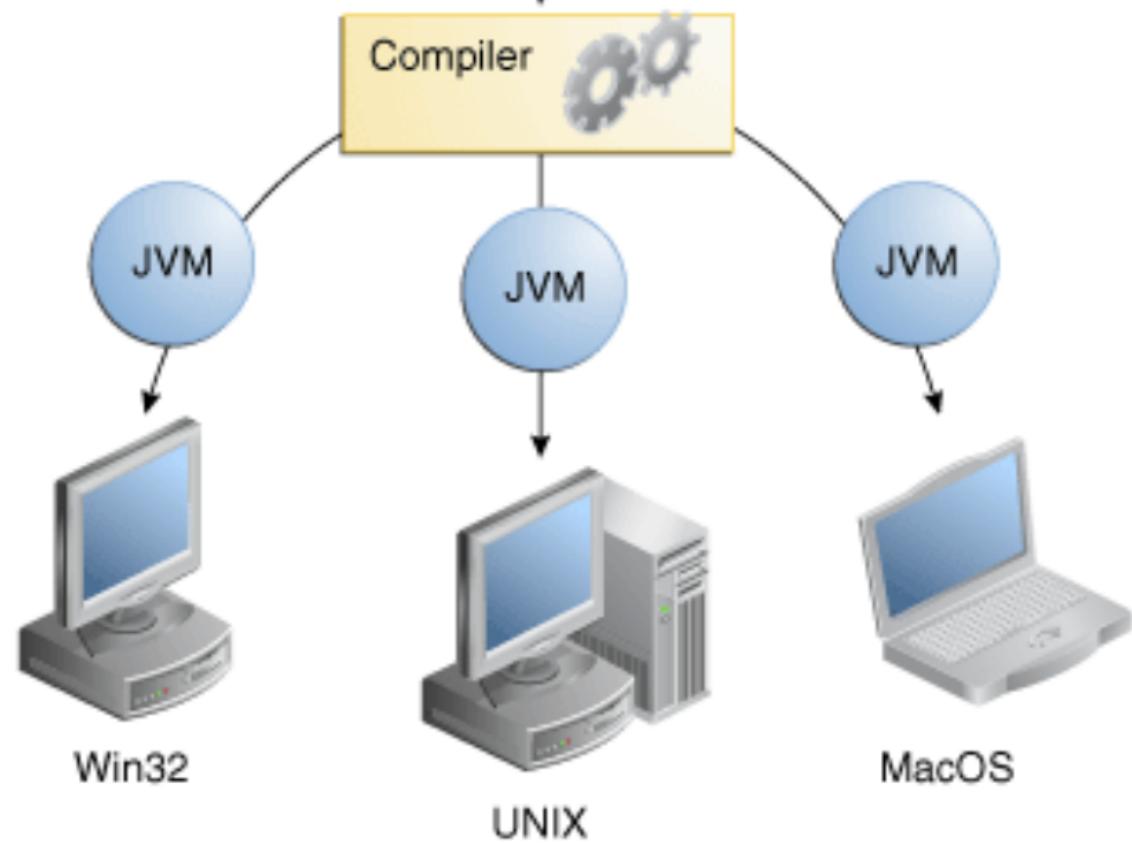
Java

Cobol

Java sería un lenguaje para decirle a la computadora que haga algo de valor.

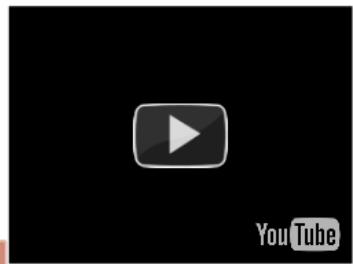
```
class HelloWorldApp {  
    public static void main(string [] args) {  
        System.out.println("Hello World!");  
    }  
}
```

HelloWorldApp.java



Comandos

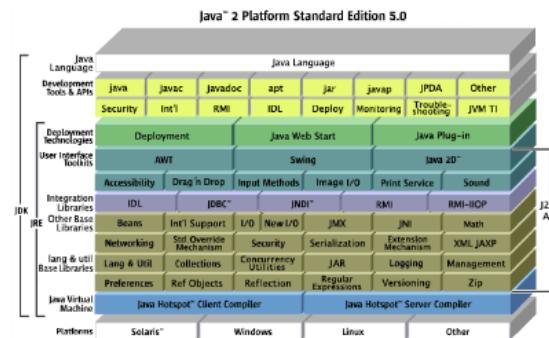
javac HelloWorldApp.java
java HelloWorldApp



Componentes de la plataforma java

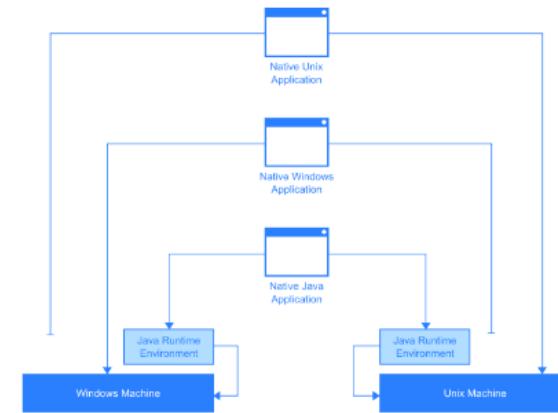
componentes

JDK (java development kit)



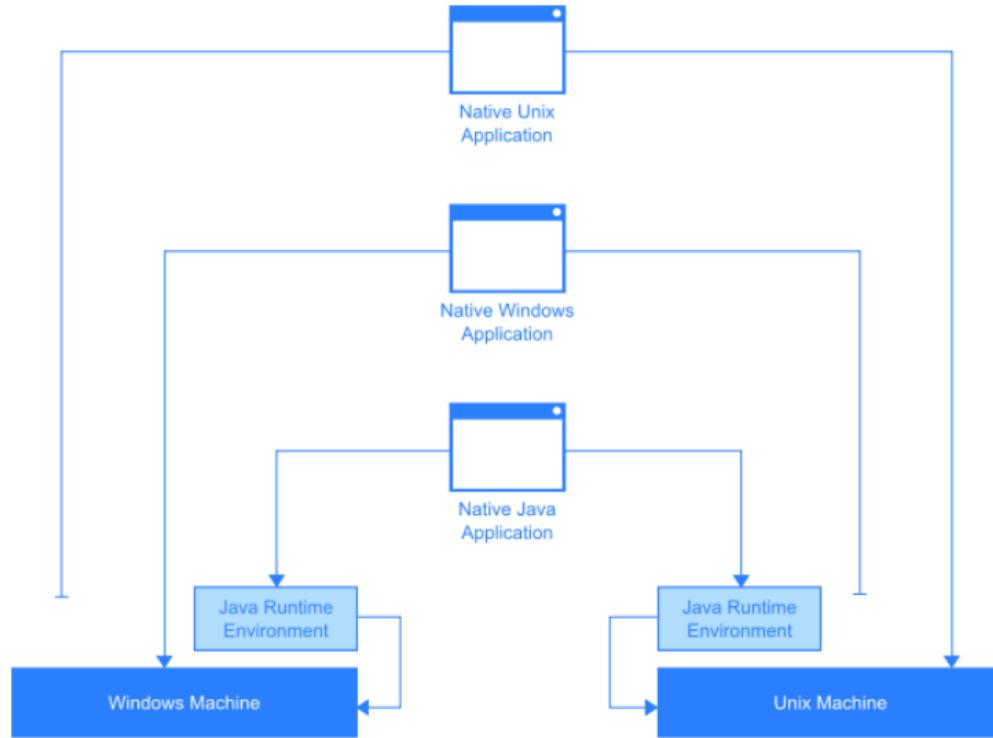
Trae todas las herramientas y librerías para el desarrollo para el desarrollo en java incluye a la JRE

JRE (java runtime environment)



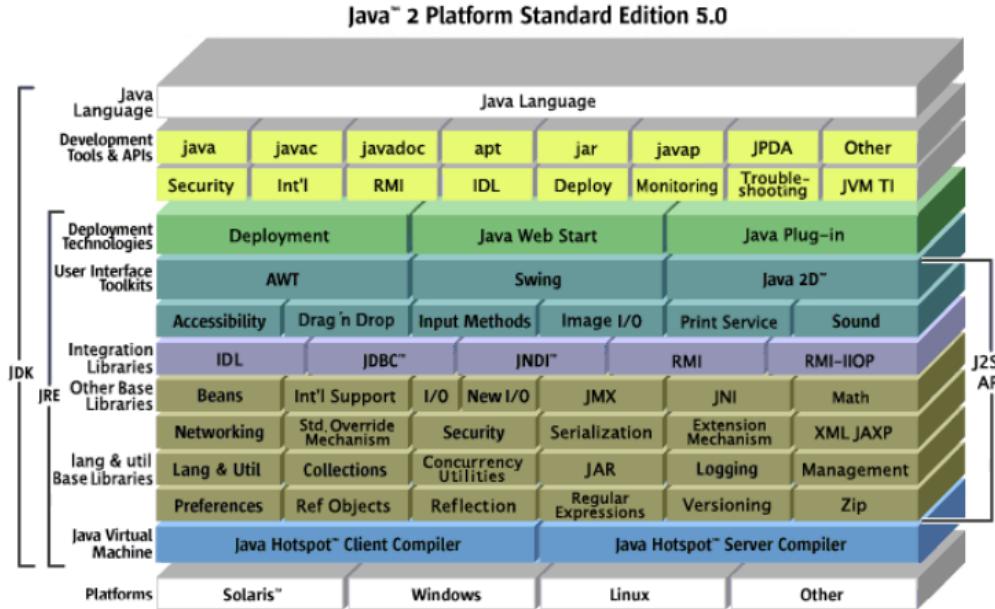
trae solo lo necesario para ejecutar aplicaciones java instalación de usuario final

JRE (java runtime environment)



**trae solo lo necesario para
ejecutar aplicaciones java
instalación de usuario
final**

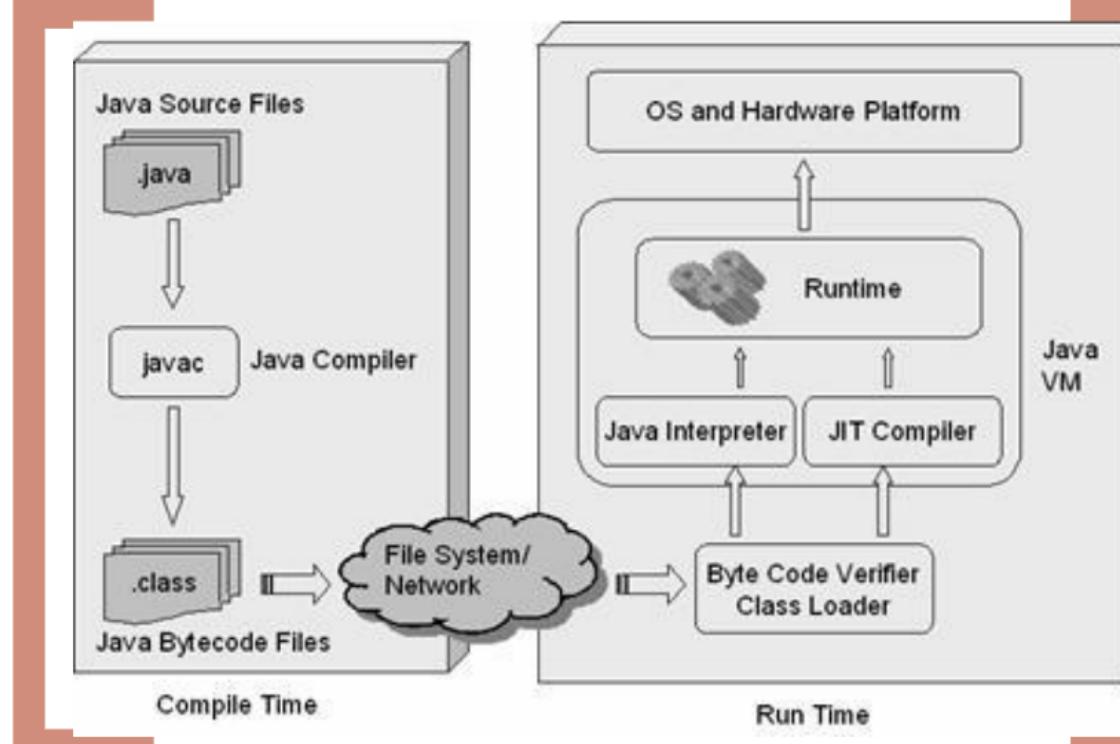
JDK (java development kit)



Trae todas las herramientas y librerías para el desarrollo para el desarrollo en java incluye a la JRE

Proceso de compilación

El proceso de compilación en java consiste en transformar un archivo de texto como lo es un archivo .java en un conjunto de "codebytes" o archivos .class que pueden ser ejecutados por una maquina virtual o jvm



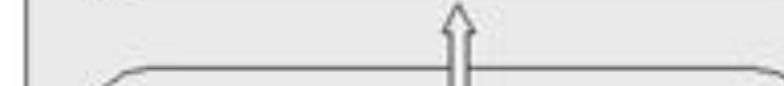
Proceso de compilacion

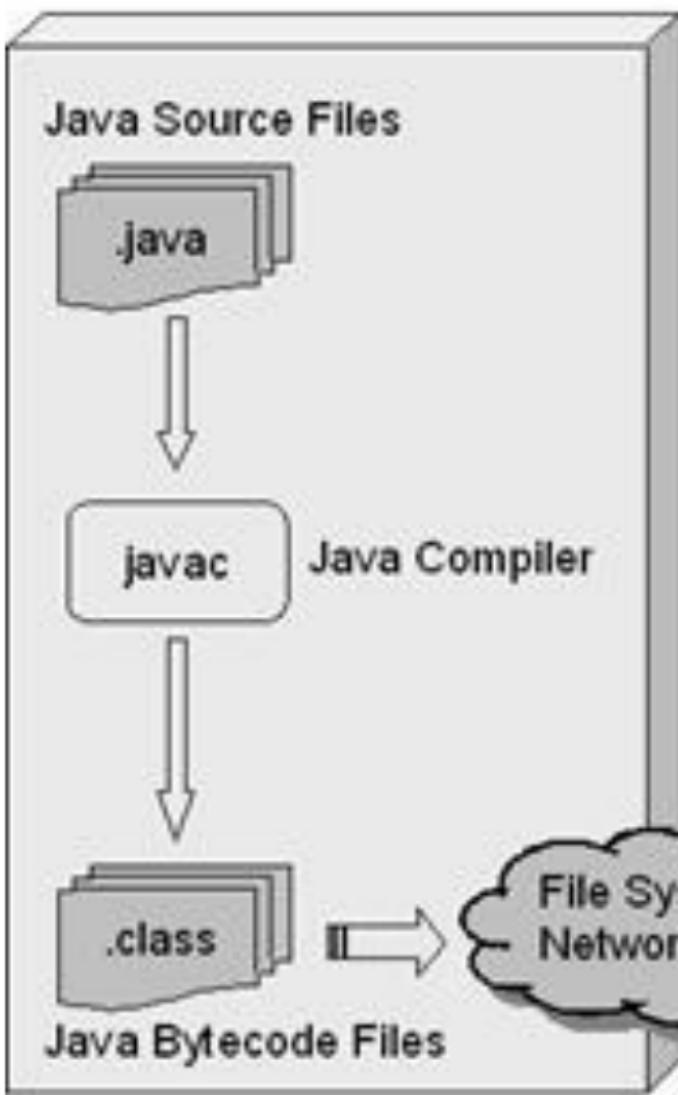
El proceso de compilación en java consiste en transformar un archivo de texto como lo es un archivo .java en un conjunto de "codebytes" o archivos .class que pueden ser ejecutados por una maquina virtual o jvm

Java Source Files

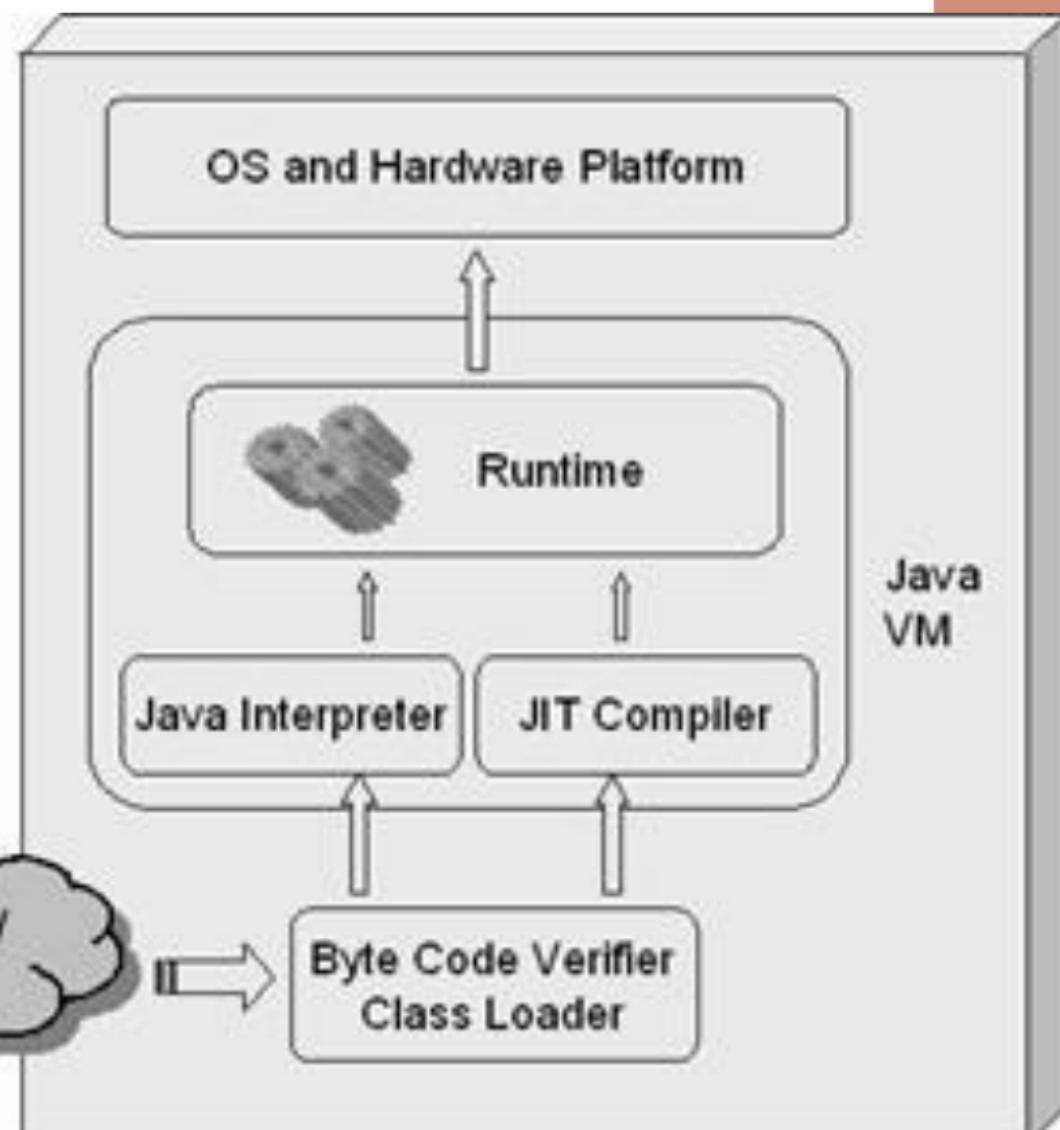
Preziva

OS and Hardware Platform





Compile Time



Run Time

Estuctura basica de un programa

modificador de acceso
de clase

nombre de la clase

inicio cuerpo clase

```
public class HolaMundo {
```

parametros por consola

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

Bloque de
ejecución
principal,
llamado
método
main . inicio
y fin

```
        System.out.println("Primera Sentencia");  
        System.out.println("Segunda Sentencia");  
        System.out.println("Fin del programa ");
```

```
}
```

```
}
```

fin cuerpo clase



	NOMBRE	TIPO	OCUPA	RANGO APROXIMADO		
TIPOS PRIMITIVOS (sin métodos; no son objetos; no necesitan una invocación para ser creados)	byte	Entero	1 byte	-128 a 127		
	short	Entero	2 bytes	-32768 a 32767		
	int	Entero	4 bytes	$2 \cdot 10^9$		
	long	Entero	8 bytes	Muy grande		
	float	Decimal simple	4 bytes	Muy grande		
	double	Decimal doble	8 bytes	Muy grande		
	char	Carácter simple	2 bytes	---		
TIPOS DE DATOS EN JAVA	boolean	Valor true o false	1 byte	---		
	Tipos de la biblioteca estándar de Java		String (cadenas de texto) Muchos otros (p.ej. Scanner, TreeSet, ArrayList...)			
	Tipos definidos por el programador/usuario		Cualquiera que se nos ocurra, por ejemplo Taxi, Autobus, Tranvia			
	arrays	Serie de elementos o formación tipo vector o matriz. Lo consideraremos un objeto especial que carece de métodos.				
		Byte Short Integer Long Float Double Character Boolean				
	Tipos envoltorio o wrapper		<small>(Equivalentes a los tipos primitivos pero como objetos.)</small>			

CATEGORÍAS DE TIPOS DE DATOS

Los tipos de datos utilizados en Java se pueden clasificar según diferentes categorías:

De acuerdo con el tipo de información que representan. Esta correspondencia determina los valores que un dato puede tomar y las operaciones que se pueden realizar con él. Según este punto de vista, pueden clasificarse en:

- **Datos de tipo primitivo:** representan un único dato simple que puede ser de tipo char, byte, short, int, long, float, double, boolean. Por ejemplo: 'a', 12345, 750.68, False,... Cada tipo de dato presenta un conjunto de valores o constantes literales.
- **Variables referencia** (variables arrays, de una clase/instancias, interfaces . . .). Se implementan mediante un nombre o referencia (puntero) que contiene la dirección en memoria de un valor o conjunto de valores (objeto creado con new).

Según cambie su valor o no durante la ejecución del programa. En este caso, se tienen:

- **VARIABLES:** sirven para almacenar datos durante la ejecución del programa; el valor asociado puede cambiar varias veces durante la ejecución del programa.
- **Constantes o variables finales:** también sirven para almacenar datos pero una vez asignado el valor, éste no puede modificarse posteriormente.

Según su papel en el programa. Pueden ser:

- **VARIABLES MIEMBRO DE UNA CLASE.** Se definen dentro de una clase, fuera de los métodos. Pueden ser de tipos primitivos o referencias y también variables o constantes.
- **VARIABLES LOCALES.** Se definen dentro de un método o, en general, dentro de cualquier bloque de sentencias entre llaves {}. La variable desaparece una vez finalizada la ejecución del método o del bloque de sentencias. También pueden ser de tipos primitivos o referencias.

Declaraciones de variables

¿que es una variable?

→ un recipiente para un valor de un tipo determinado que puede variar en el tiempo

Ejemplos

```
int a ;  
double c , b;  
String str = "Hola Mundo"
```

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

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

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



Declaraciones de variables

¿que es una variable?

→ un recipiente para un valor de un tipo determinado que puede variar en el tiempo

Ejemplos

```
int a ;  
double c , b;  
String str = "Hola Mundo"
```

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

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

abstract	continue	finally	int	public	throw
assert	default	float	interface	return	throws
boolean	do	for	long	short	transient
break	double	goto	native	static	true
case	else	if	new	strictfp	try
catch	enum	implements	null	super	void



Casting de variables

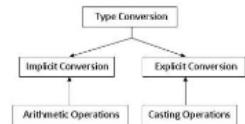
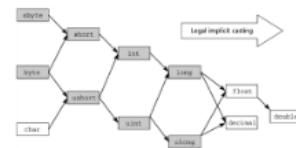
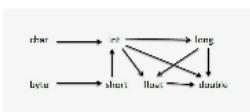
El casting de variables no es mas que transformar el dato de un tipo en otro

Explícito: lo hace el programador con() ejemplo

```
int a = (int)32.23
```

Implícito: lo hace la maquina virtual si intervención del programador ejemplo

```
double a = valueInt
```



Operadores Aritméticos

Operador	Uso	Descripción
+	op1 + op2	Suma op1 y op2 (*)
-	op1 - op2	Resta op2 de op1
*	op1 * op2	Multiplica op1 y op2
/	op1 / op2	Divide op1 por op2
%	op1 % op2	Obtiene el resto de dividir op1 por op2
++	op ++	Incrementa op en 1; evalúa el valor antes de incrementar
++	++ op	Incrementa op en 1; evalúa el valor después de incrementar
--	op --	Decrementa op en 1; evalúa el valor antes de decrementar
--	-- op	Decrementa op en 1; evalúa el valor después de decrementar

la precedencia se da de izquierda a derecha y se puede modificar mediante paréntesis

```
programa ejemplo con operadores
public class Aritmetica{
    public static void main(String[] args){
        int i = 12;
        int j = 10;
        int suma = i + j;
        int resta = i - j;
        int mult = i * j;
        int div = i / j;
        int modulo = i % j;

        System.out.print("Suma :");
        System.out.println(suma);
        System.out.print("Resta :");
        System.out.println(resta);
        System.out.print("Multiplicacion :");
        System.out.println(mult);
        System.out.print("Division :");
        System.out.println(div);
        System.out.print("Modulo :");
        System.out.println(modulo);

    }
}
```

Operadores booleanos

permiten evaluar expresiones booleanas

Operator	Result
&	Logical AND
	Logical OR
^	Logical XOR(exclusive or)
	Short-circuit OR
&&	Short-circuit AND
!	Logical unary NOT
&=	AND assignment
=	OR assignment
^=	XOR assignment
==	Equal to
!=	NOT equal to
?:	Ternary if-then-else

```
public class TablaAnd {  
    public static void main(String args[]){  
        boolean x = true;  
        boolean y = false;  
        boolean a1 = x && x;  
        boolean a2 = x && y;  
        boolean a3 = y && x;  
        boolean a4 = y && y;  
  
        System.out.println("Tabla de verdad de la conjunción");  
        System.out.println(x + " AND " + x + " = " + a1);  
        System.out.println(x + " AND " + y + " = " + a2);  
        System.out.println(y + " AND " + x + " = " + a3);  
        System.out.println(y + " AND " + y + " = " + a4);  
    }  
}  
  
public class Condicional {  
    public static void main(String args[]){  
        int x = 0;  
        int y = 2;  
        boolean b = (x != 0) && ((y / x) != 0);  
        System.out.println(b);  
    }  
}
```

(exp boolean) operador (exp boolean)

boolean a = true | false

boolean a = true & false

boolean a = true | false

boolean a = true == false

boolean a = true || false

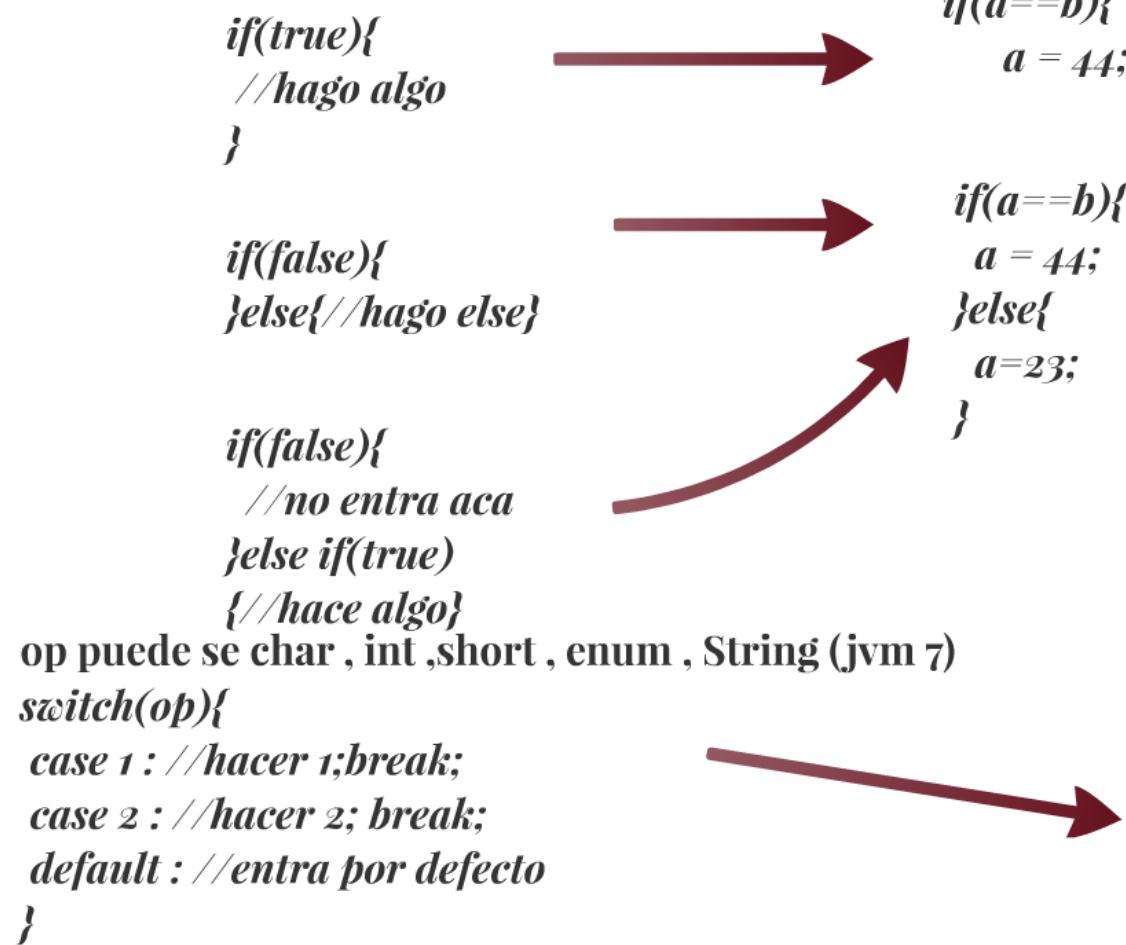
boolean a = false && true

int a = true ? 23 : 0;



Estructuras de control

Son estructuras para modificar el flujo de ejecución de un programa



ejemplos

```
public class Main{  
    public static void main(String args[]){  
        int a = 5;  
        int b = 5;  
        char op = '=';  
        String resultado = "El resultado es : ";  
        if (op == '+')  
            System.out.println(a + b);  
        else if (op == '-')  
            System.out.println(a - b);  
        else if (op == '*')  
            System.out.println(a * b);  
        else if (op == '/')  
            System.out.println(a / b);  
    }  
}
```

```
public class Main{  
    public static void main(String args[]){  
        int a = 5;  
        int b = 5;  
        char op = '=';  
        String resultado = "El resultado es : ";  
        if (op == '+')  
            System.out.println(a + b);  
        else if (op == '-')  
            System.out.println(a - b);  
        else if (op == '*')  
            System.out.println(a * b);  
        else if (op == '/')  
            System.out.println(a / b);  
    }  
}
```

Bucles de iteración

Son las estructuras necesaria para recorrer colecciones o hacer acciones repetidamente

bucles indefinidos : no se cuantas iteraciones hay

ejemplos

```
public class Decimal{  
    public static void main(String[] args){  
        int a = 1;  
        System.out.println("a = " + a);  
        while(a <= 10){  
            a++;  
            System.out.println("a = " + a);  
        }  
    }  
}
```

```
boolean c = true;  
while(c){  
    if(a == 2){  
        c = false;  
    }  
}
```

```
boolean a = true;  
do{  
    if(b == 3)  
        a = false;  
}while(a)
```

ejemplos

```
public class Circulo{  
    public static void main(String[] args){  
        int radio = 4; //radio  
        int diametro = 2 * radio;  
        int area = 3.14159 * radio * radio;  
        int perimetro = 2 * 3.14159 * radio;  
        System.out.println("radio = " + radio);  
        System.out.println("diametro = " + diametro);  
        System.out.println("area = " + area);  
        System.out.println("perimetro = " + perimetro);  
    }  
}
```

bucles definidos : sabemos de antemano cuantas iteraciones hay

```
for(int a = 0;a < 20; a++){  
    //hacer algo  
}
```

for (;;){}
for (; ;)

```
for ( int k=1, j=10 ;k < j ;k++ ,j-- ){  
    System.out.println(k + " " + j);  
}
```

ejemplos

```
for ( int factor = 1; factor <= 9; factor++ ) {  
    System.out.println("3 x " + factor + " = " + 3*factor);  
}
```

```
int[] laiEnteros = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};  
for (int liElemento : laiEnteros)  
    System.out.println (liElemento);
```

```
int[] laiEnteros = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};  
for (int i = 0; i < 10; i++)  
    System.out.println (laiEnteros[i]);
```



Arreglos

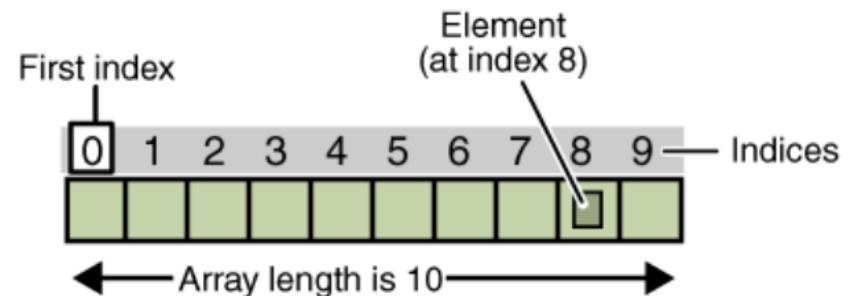
Arreglos (Array)

- Un arreglo es una secuencia de datos primitivos o de objetos, todos del mismo tipo, unidos bajo un identificador común.
- Todos los elementos de un arreglo tienen el mismo nombre pero se diferencian por la posición que ocupan en él.
- Son de tipo referencia
- Sinónimos:
 - Vector
 - Tabla
 - Matriz

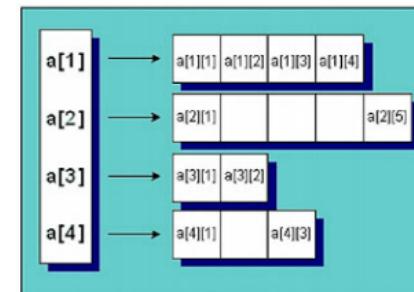
```
public class Arreglo001 {  
    public static void main(String args[]){  
        int arregloInt[] = new int[4];  
        for (int i = 0; i < arregloInt.length; i++){  
            arregloInt[i] = i * 2;  
            System.out.println(arregloInt[i]);  
        }  
    }  
}
```

```
int [] a = new int[10]
```

```
int [] a = {1,2,3,4,5,6,7,8,9,10}
```



Para obtener el tamaño de un array de manera dinámica se utiliza la propiedad `.length`



```
int[][] multi = new int[5][10];  
multi[0] = new int[10];  
multi[1] = new int[7];  
multi[2] = new int[10];  
multi[3] = new int[10];  
multi[4] = new int[10];
```