

Java Inicial

(20 horas)





Temario

1. Programación Orientada a Objetos
2. **Introducción y Sintaxis Java**
3. Sentencias Control Flujo
4. POO en Java
5. Relaciones entre Objetos
6. Polimorfismo, abstracción e interfaces
7. Excepciones
8. Conceptos avanzados



Tema 2

Introducción y Sintaxis Java



Objetivos

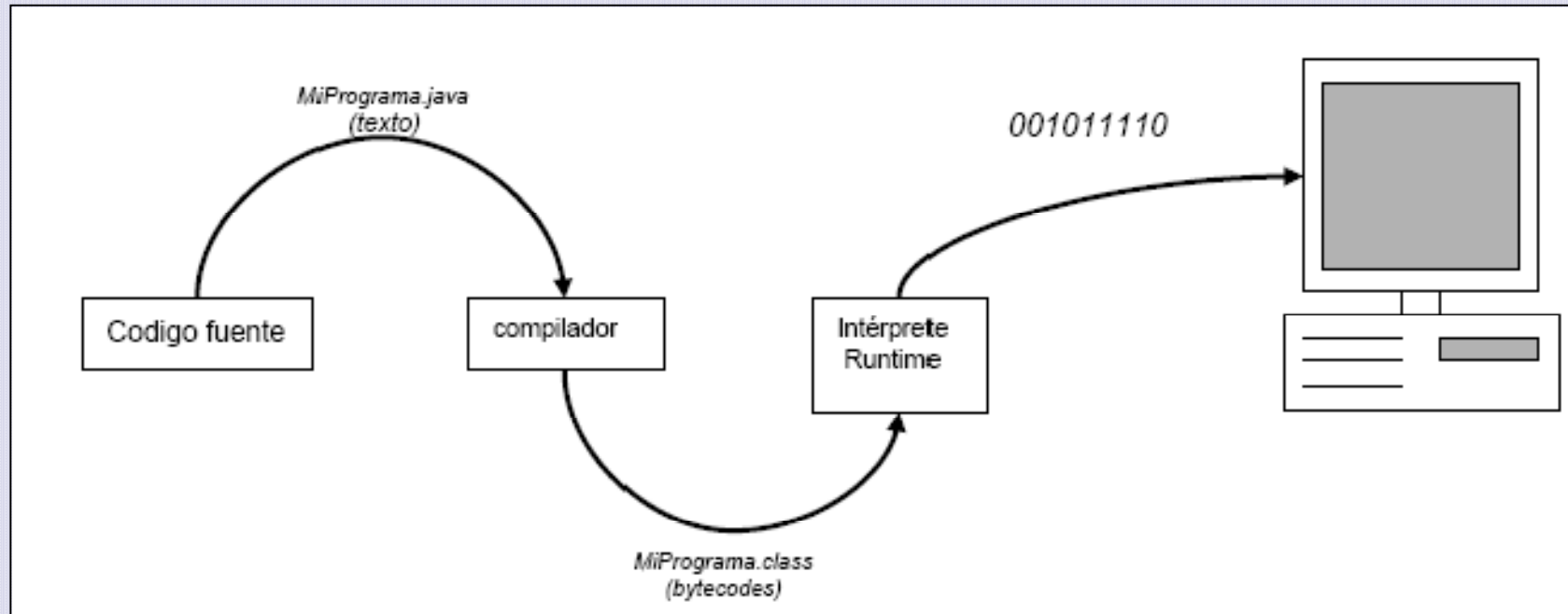
1. Programación Orientada a Objetos
2. **Introducción y Sintaxis Java**
3. Sentencias Control Flujo
4. POO en Java
5. Relaciones entre Objetos
6. Polimorfismo, abstracción e interfaces
7. Excepciones
8. Conceptos avanzados

- **Introducción**
 - Historia
 - Características
 - Entorno y Programas
- **Sintaxis**
 - Comentarios
 - Palabras reservadas
 - Tipo de datos
 - Constantes y Variables
 - Expresiones
 - Sentencias
 - Operadores
 - Conversión de datos
 - Arrays

■ Historia

- (1990) James Gosling y Mike Sheridan,
- Empleados de Sun Microsystems desarrollaron OAK, primera versión de Java
- Basado en C++
 - Quería eliminar su complejidad
 - Alto coste en pruebas y depuración
- Las primeras aplicaciones eran para electrodomésticos (neveras, TV, tostadoras)

■ Características (I)



intermedio (bytecodes)

- Se ejecutará e interpretará en la JVM (Java Virtual Machine)

■ Características (II)

□ Independiente de plataforma

- Los bytecodes son interpretados por la máquina virtual del SO (Linux, Mac, ...) a código máquina.

□ Lenguaje **orientado a Objetos**

- Encapsulación, Modularización, Abstracción y Herencia.

□ Lenguaje **dinámico**

- Podemos ir escalando nuestra aplicación con las librerías.

■ Características (III)

□ Lenguaje **seguro** (4 niveles de seguridad)

- *de Lenguaje*: sintaxis más fácil (ausencia de punteros)
- *de Verificación de bytecodes*: busca irregularidades según la gramática de los compiladores.
- *de Cargador de Clases*: reconocimiento de las clases de una aplicación, ya estén en el equipo local o en un equipo remoto.
- *de API de Java*: para evitar errores a la hora de acceder a recursos del sistema y provocar inconsistencias.

□ Lenguaje **Concurrente**:

- Crear Procesos que se ejecuten simultáneamente (Thread)

■ Entorno y Programas

- javac.exe
Realiza la compilación del código del fichero “.java”, obteniendo los *bytecodes*:

```
javac Fichero.java
```
- java.exe
Es el intérprete de Java. Ejecuta nuestra aplicación

```
java Fichero
```
- javadoc.exe
Generador de documentación de nuestra aplicación
- appletviewer.exe
Permite la ejecución de Applets sin necesidad de tener un Web Browser
- jar.exe
Permite la generación de ficheros JAR
- jdb.exe
Es el *debugger* de Java que nos permite depurar la ejecución de nuestros programas
- javap.exe
Nos permite obtener información de un fichero previamente compilado

```
javap Fichero
```

■ JDK: Java Development Kit

□ Variables de entorno

- PATH: Permite buscar las aplicaciones (javac, java, ...) con indiferencia del path.
- CLASSPATH: Indica la ruta donde se encuentran nuestros ficheros (clases) para la correcta ejecución de nuestra aplicación.

□ Para el desarrollo de nuestro código podemos usar:

- Editor texto: NotePad, TextEdit, ...
- IDE: Integrate Development Enviroment.

■ IDE (I)

- Integrated Development Enviroment
- Básicamente un programa para escribir programas.

```
#!/usr/bin/tclsh

itcl::class Persona {
    private variable apellidos
    private variable nombres
    private variable edad 18
    public variable nacimiento 1984
    constructor {vapeellidos vnombres} {
        set apellidos $vapeellidos
        set nombres $vnombres
        puts "¡Bienvenido, $nombres $apellidos!"
    }
    method setEdad {vedad} {
        set edad $vedad
    }
    method getEdad {} {
        return "$edad"
    }
    method saludo {lugar} {
        puts "¡Buen día, $nombres!"
        if {$lugar != ""} {
            puts "Nosotros estamos en $lugar"
        }
    }
    method esMayor {} {
        expr {$edad>=18}
    }
    method nombrecompleto {} {
        return "$nombres $apellidos"
    }
}

#Se crea una instancia de la clase Persona
```

Problemas Tareas Consola TCl Documentation

```
<terminado> pkgindex.tcl [Tcl Script] /usr/bin/tclsh (09/07/2009 20:12:04)
¡Bienvenido, José Hernández!
José Hernández es menor, ya que tiene 16 años.
José Hernández nació en el año 1993.
¡Buen día, José!
Nosotros estamos en la Costa Azul
```

IDE (II)

■ Ventajas

- Menor esfuerzo y tiempo de desarrollo
- Estandares de desarrollo
- Presentación visual de componente

■ Desventajas

- Curva de aprendizaje (proyectos pequeños)
- No adecuado para principiantes.
- Inhibe cualidades desarrollo y diseño.
 - Cómo todo lo hace y corrige el IDE

■ Eclipse Java

- Desarrollado por IBM
- Competencia netbeans Sun (Sol)
- Libre



■ IntelliJ IDEA

- Es de pago
- Desarrollado por JBrains
- Mejor refactoring



■ Aplicación Orientada a Objetos

- En una aplicación orientada a objetos debe existir una clase que represente la **propia aplicación**. Este sería el punto donde comenzaría la ejecución de la misma.
- En lenguajes no totalmente orientados como C++ en la función ***main*** se crea una instancia de esta clase y se llama a alguna operación como *ejecutar* para arrancar la aplicación.

No orientado a objetos

- **Aplicación Orientada a Objetos (II)**
 - En un lenguaje POO “puro” como Java esta clase de aplicación es obligatoria.
 - La máquina virtual Java se encarga de instanciar esta clase y llamar a una operación especial con nombre *main*.
 - La existencia de una operación estática pública con este nombre es lo que caracteriza la clase de aplicación

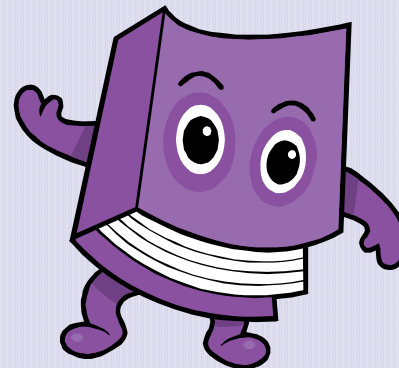
- `public static void main (String[] args)`
 - La clase aplicación debe ser pública y no tener ningún constructor o un constructor por defecto.
 - Al menos debe implementar la operación *main*, con la siguiente declaración:
 - *`public static main (String[] args)`*

```
public class BancoApp {  
    public static void main (String[] args) {  
        Cuenta c1 = new Cuenta (18400200, "Pedro Jiménez", 0.1f);  
  
        c1.ingreso (1000);  
  
        system.out.println ("Ingreso realizado");  
    }  
}
```

■ Ejercicio práctico: Hola Mundo

- Crear una clase Java que contenga el método *main* e imprimir por pantalla el típico “Hola Mundo”

■ Suerte!!!



■ Sintaxis Java

□ A continuación mostramos los diferentes elementos del lenguaje:

- Comentarios
- Palabras reservadas
- Tipos de datos
- Variables y Constantes
- Sentencias y Expresiones
- Operadores
- Arrays

■ Comentarios

- Toda aplicación debe estar bien documentada.
- Un comentario no se tiene en cuenta en compilación.

■ Comentarios de Línea

```
// Este es un comentario de línea
//Y este es otro comentario de línea
```

■ Comentarios de Párrafo

```
/* Este comentario ocupa
varias líneas */
```

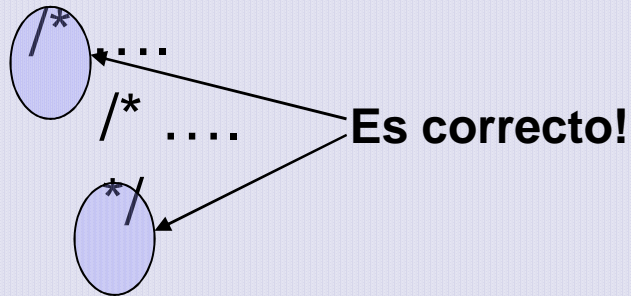
■ Comentarios Especiales

- Javadoc: Generar comentarios para documentación
- Ver javadoc en la lista de ejemplos

cuentacorriente Classes Cliente CuentaCorriente Main	void setDni (java.lang.String dni) Set the value of dni
	void setDomicilio (java.lang.String domicilio) Set the value of domicilio
	void setNombre (java.lang.String nombre) Set the value of nombre

■ Comentarios: OJO

- Los comentarios no se enlazan



- `/*` `*/` no tiene significado en un `// Comment`
`// /* ...` *[Todo la línea es un comentario]*

- Idem para `//` con los `/* Comment */`

```
/* ...  
    // ...  
*/
```

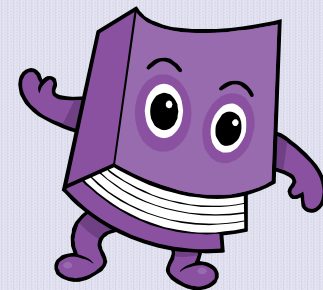
■ Identificador

- Es un secuencia de '*letrasJava*' y *dígitos* donde la primera letra debe ser una *letraJava*:
 - letraJava = [a..z] o [A..Z] o '_'
 - dígitos: = [0..9]

- No puede ser igual a:
 - Literal boolean = [*true* | *false*]
 - Null Literal = [*NULL* | *null*]
 - Keyword = Palabra reservada

■ Ejercicio práctico:

- Indicar cuales de los siguientes nombres son validos
- _CosFI
- de2
- \$alguna
- 3terna
- númeroDeCliente
- porc_de_casos
- esCierto?
- número positivo
- lost+found
- añoBisiesto



■ Palabras reservadas

- Identificadores que no pueden ser utilizados para otro caso que para el que fueron diseñados.

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

Palabras reservadas sin utilización actual

cast	future	generic	inner	operator
outer	rest	var		

■ Tipo de Datos

- Java es un lenguaje de tipado fuerte de datos
 - Ayuda a detectar errores en tiempo de compilación
- Cada variable y expresión tiene un **tipo de dato** conocido en tiempo de compilación.
- El tipo limita los valores que una variable puede almacenar.
- Determinan también el significado de las operaciones (*ver operadores*)
- Dos tipos
 - Primitivos = *booleanos y numéricos*
 - Referenciados = *clases, interfaces y arrays*

■ Tipo de Datos Primitivos

□ Tipos Primitivos

Tipo	Tamaño	Descripción
byte	8 bits complemento a 2	Entero de un byte
short	16 bits complemento a 2	Entero corto
int	32 bits complemento a 2	Entero
long	64 bits complemento a 2	Entero largo
float	32 bits IEEE 754	Coma flotante de precisión simple
double	64 bits IEEE 754	Coma flotante de precisión doble
char	16 bits	Un solo carácter
boolean	true o false	Valor booleano

□ Clases contenedoras

Tipo primitivo	Clase contenedora	Método de acceso al valor
byte	Byte	byteValue()
short	Short	shortValue()
int	Integer	intValue()
long	Long	longValue()
float	Float	floatValue()
double	Double	doubleValue()
char	Character	charValue()
boolean	Boolean	booleanValue()

No confundir

■ Literales: *Integers*

□ *Decimal*

- 0
- 2
- 1965

□ *Hexadecimal*

- *0x00FF00FF*
- *0xDadaCafe*

□ *Octal*

- 0372
- 017777777777

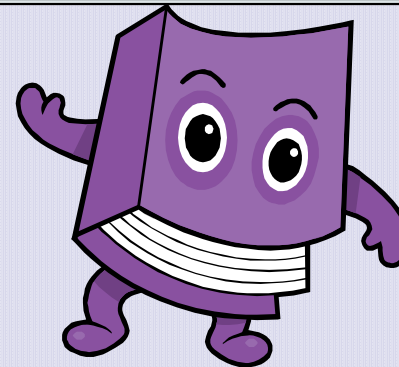
■ Ejercicio práctico:

- Ver la diferente representación de los literales integer en los

- Decimal
- Octal
- Hexadecimal

```
Output
Java DB Database Process x GlassFish V2 x HelloWorldApp (run) x
run:
Numero decimal:124
Numero octal:84
Numero hexadecimal:292
BUILD SUCCESSFUL (total time: 8 seconds)
```

- Suerte!!!



■ Literales: *floating*

□ float literals:

- 1e1f
- 2.f
- .3f
- 6.022137e+23f

□ double literals:

- 1e1
- 2.
- .3
- 0.0
- 3.14
- 1e-9d
- 1e137

■ Literales

□ ***boolean*** : [true | false]

- Representa la lógica

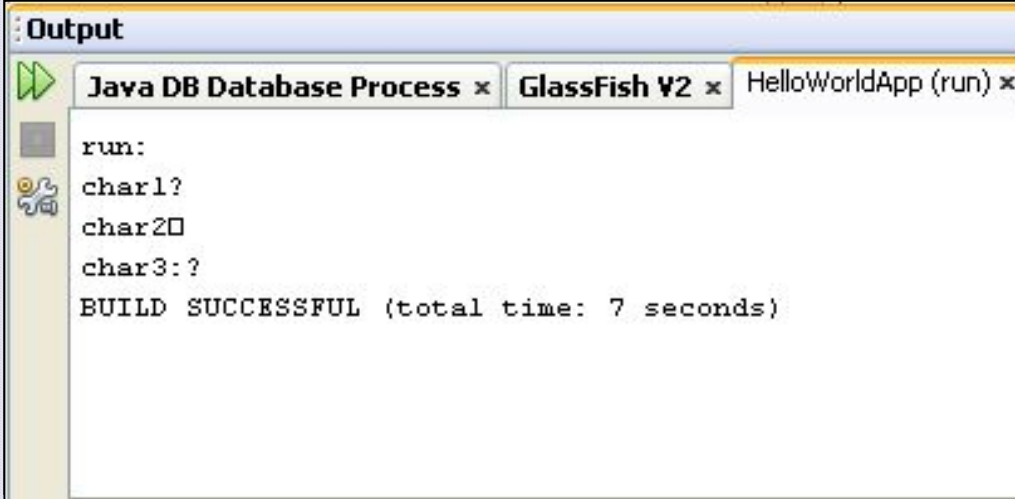
□ ***chars***

- Ejemplos Character:

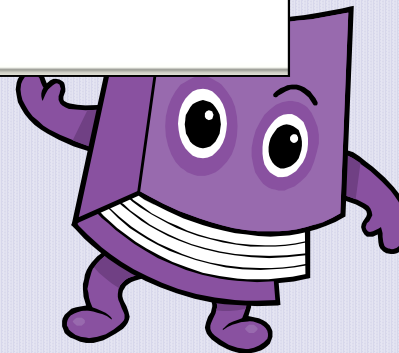
'a' '%' '\t' '\\' '\"' '\u03a9' '\r' '\177'

■ Ejercicio práctico:

- Ver la diferente representación de los literales character

- 

■ Suerte!!!



■ Literales

□ **Strings:** Consiste en cero o más characters encerrados entre comillas (“, ‘)

■ Ejemplos

- "" // *string vacio*
- "\"" // *Un string conteniendo solo "*
- "Esto es un String" // *un String de 16 characters*
- "Esto es " + // *cadena de caracteres*
- "un string en 2lineas" // *formada por dos literales*

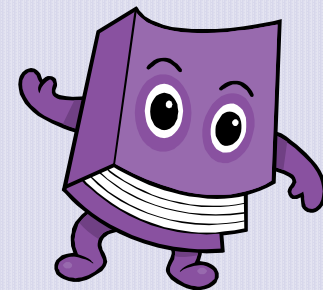
□ **Null:** *null*

- No se puede declarar ninguna variable de tipo null
- Representa el ‘vacío’

■ Ejercicio práctico:

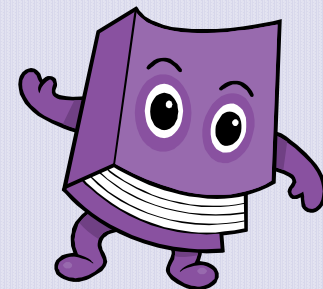
□ Declarando variables: ¿Cuáles son válidas?

- nombre String = "Claudio";
- int suma = 23;
- long t = 93;
- boolean cerrado = false;
- char = 'E';
- String boolean = "False";
- char e = "\u00ff";
- int Int = 32;
- double velocidad = 300.000;



■ Ejercicio práctico:

- ¿Qué tipos son los adecuados?
- El número de alumnos es 456.
- La empresa tiene el nombre de Curro S.A.
- El límite de velocidad es de 80 km/h
- La tarifa de importación es del 5.25 %
- Para finalizar presione el signo numeral
- Información que no se sabe
- Son 11 los jugadores
- Nuestra galaxia tiene alrededor de 100 000 millones de estrellas
- La capacidad maxima de carga es de 50.000 Kg.
- Se ha disminuido un 18% el número de accidentes.
- La distancia entre la Tierra y la Luna es de 384.400 km
- La aceleración de gravedad de 9.8 m/s²



■ Ejercicio workspace:

EjemploTiposNumericos.java

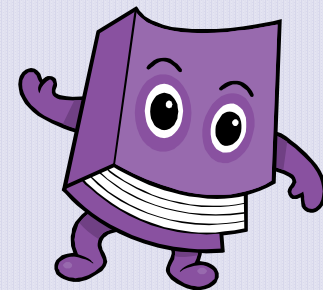
- Analizar como en este ejemplo se utilizan los tipos numéricos de distinta modo.

EjemploRandom.java

- En este ejemplo se puede ver cómo utilizar un método que devuelve un double aleatorio del 0 al 1.

EjemploOverflow.java

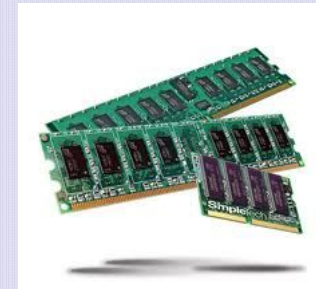
- En este ejemplo se puede ver lo que sucede si excedemos el valor máximo permitido a una variable.



■ Tipo de Datos referenciados

□ El valor no es el literal sino una posición de memoria a:

- Una clase
- Un interfaz
- Un array



□ Hasta ahora, nuestras variables las podíamos ver como “cajitas” que almacenan información.

□ Ahora, esas “cajitas” contienen la dirección en donde está almacenada la información.

■ Constantes

- Posición de memoria fija.
- No puede ser modificado durante la ejecución del programa.
- Su valor se define en el momento de la declaración.

```
//Constante que define el valor del EURO en pesetas  
final double EURO = 166.386;
```

□ Reglas de estilo

- El nombre del identificador debe estar en MAYÚSCULAS
- Una constante debe ser final

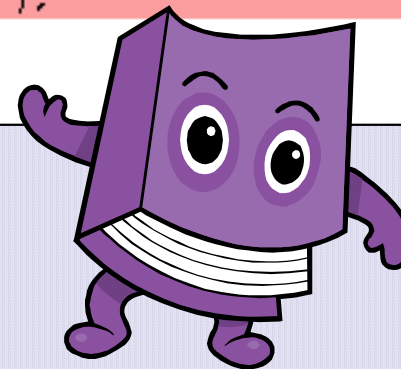
■ Ejercicio práctico:

- Ver que es el compilador quien detecta si el valor de una **CONSTANTE** cambia

```
/**
 * @param args the command line arguments
 */
public static void main(String[] args) {
    //Constante que define el valor del EURO en pesetas
    final double EURO = 166.386;

    EURO = 165;
    cannot assign a value to final variable EURO
    System.out.println("Hola clase:");
}
}
```

■ Suerte!!!



■ Variables

- Es una localización en memoria que tiene un tipo asociado
 - Puede ser Tipo primitivo
 - Tipo Referenciado
- Contiene siempre un valor que es **compatiblemente asignable** a su tipo.

```
tipoPrimitivo identificador;  
    int edad;  
    char letra;
```

```
tipoPrimitivo identificador = valorInicial;  
    int edad = 10;  
    char letra = 'A';
```

■ Variables (II)

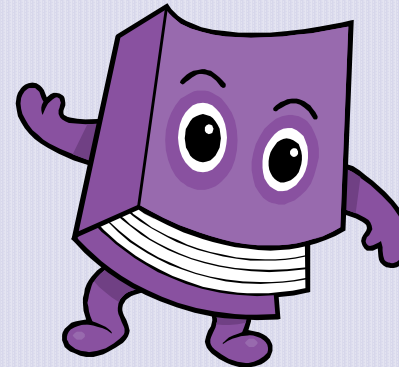
- El valor puede ser modificado durante la ejecución.
 - A través de una asignación
 - Operadores
- Reglas de estilo
 - El identificador de una variable debe comenzar por un carácter (a..z,A..Z) o el carácter “_” (guión bajo)
 - El identificador de una variable no puede ser una palabra reservada

■ Ejercicio práctico:

- Ver Como cambia el valor de una variable

```
public class UsoVariables{  
    public static void main(String args[]){  
        String saludo;  
        saludo = "Hola Mundo!";  
        System.out.println( saludo );  
        saludo = ("Estoy utilizando variables");  
        System.out.println( saludo );  
    }  
}
```

- Suerte!!!



■ Expresiones

- Es un conjunto de operandos unidos por un operador (expresión simple) o varios operadores (expresión compuesta) que devuelve un valor.

```
7 + 5  
i < 5
```

- Operando:
 - Numero: 7
 - Variable: *a*
 - Resultado método: *this.sumar()*
 - Otra expresión: *3 + a*

■ Sentencias

- Es un conjunto de expresiones cuyo resultado se tratará en ese mismo instante. Se podría decir que una sentencia es una expresión final.

```
System.out.println(7+5);    // Escribimos en pantalla
                          // el resultado de la suma
suma = 7 + 5 ; // Asignamos a la variable suma
                          // el resultado de la suma
```

□ Sentencia

- Operando = Expresión
 - Expresión = operando *operador* operando

■ Operadores

- Realizan operaciones sobre uno o varios operandos
- Todos los operadores devuelven un valor que deberá ser tratado
 - Almacenado en una variable.
 - Ser el argumento de llamada a una función.
 - Imprimirlo o almacenarlo en un dispositivo de E/S.
- Ejercicio: ¿qué devuelven estas operaciones?
 - `int + int`
 - `int + float`
 - `(int * int) * float * int`
 - `int / int`
 - `String + float`
 - `Boolean + int`
 - `String + boolean`

■ Operadores Aritméticos

- Retornarán un valor del tipo determinado por los operandos.

Operador	Uso	Descripción
+	op1 + op2	Suma de operandos
-	op1 - op2	Resta de operandos
*	op1 * op2	Multiplicación de operandos
/	op1 / op2	División de operandos
%	op1 % op2	Módulo (resto de la división)

■ Operadores Unarios

De cambio de signo

```
- operando
```

De Incremento o Decremento

```
++ operando; --operando; //notación prefija  
operando++; operando--; //notación postfija
```

Ejemplo

```
int x = 10;  
int y;  
y = ++x;
```

x e y será 11.

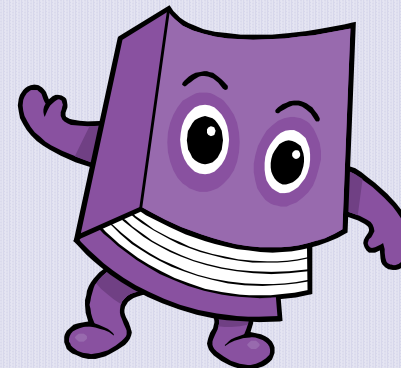
```
int x = 10;  
int y;  
y = x++;
```

x=11 y la variable y=10

■ Ejemplo workspace:

□ EjemploIncrementales.java

- En este ejemplo se muestra cómo utilizar los operadores incrementales ++ y – en distintos ordenes.



■ Operadores Relacionales

- Retornarán un valor booleano.
 - *True*: Si la expresión es verdadera
 - *False*: Si la expresión es evaluada como falsa.

Operador	Uso	Descripción
<	op1 < op2	Menor que
<=	op1 <= op2	Menor o Igual que
>	op1 > op2	Mayor que
>=	op1 >= op2	Mayor o Igual que
==	op1 == op2	Igual que
!=	op1 != op2	Distinto que

■ Operadores Lógicos

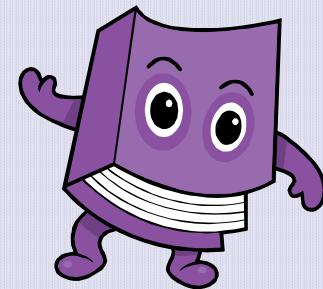
- Permiten realizar expresiones lógicas compuestas.
- Estos operadores devolverán un valor **true** o **false**.

Operador	Uso	Descripción
&&	op1 && op2	Y lógico
	op1 op2	O lógico
!	!op	Negación

- Ejercicio: ¿qué devuelven estas expresiones?:
 - *!true*
 - *false || true || (false && true)*
 - *false && false && 12 > 11 && (true || 1 == 2)*
 - *(((1+2) <= 3) || (5<6)) && (2<4) && (!false)*

■ Ejercicio workspace:

- EjemploComparaciones.java
 - Este ejemplo genera dos números aleatorios. Al ejecutarlo se puede ver lo que devuelve cada una de las comparaciones.
- EjemploComparaciones2.java
 - Ejemplo de cómo realizar comparaciones numéricas.
- EjemploComparacionString.java
 - En este ejemplo se puede ver cómo se comparan Strings y se crean substrings.



■ Operadores de Bits

- Realizan operaciones sobre operandos pero **a nivel de bits**

Operador	Uso	Descripción
&	op1 & op2	Y de bits (multiplicación)
	op1 op2	O de bits (suma)
^	op1 ^ op2	O exclusivo
~	~ op2	Complemento a 1
>>	op1 >> n	Desplaza n bits a la dcha.
<<	op1 << n	Desplaza n bits a la izda.
>>>	op1 >>> n	Desplaza n bits a la dcha. de op1 (sin signo)

- **Ejercicio: ¿Cuál es el resultado?:**

7 & 7	10 & 7	10 & 0xFFFF
0xFF 0	10 5	10^10
0xFF >>4	10 >> 2 5	0xF >>(0x10 >> 3)

■ Operadores de Asignación

- Son una forma de realizar una operación y una asignación al mismo tiempo.
- Se puede utilizar cualquier operador binario junto con el operador de asignación

Operador	Uso	Equivale a	Descripción
=	v = 5		Asignación
+=	v += 5	v = v + 5	Asign. con suma
-=	v -= 5	v = v - 5	Asign. con resta
*=	v *= 5	v = v * 5	Asign. con multiplicación
/=	v /= 5	v = v / 5	Asign. con división
%=	v %= 5	v = v % 5	Asign. con módulo
&&=	op1 &&= op2	op1 = op1 && op2	Asign. con Y
=	op1 = op2	op1 = op1 op2	Asign. con O
<<=	op1 <<= n	op1 = op1 << n	Asign. con desplazamiento

■ Operadores: Precedencia

- Las expresiones se evalúan siguiendo la precedencia de los operadores que las forman.
- Cuanto más arriba esté el operador, mayor prioridad tendrá.
- Dentro del mismo grupo, tendrá mayor prioridad el que esté más a la izquierda

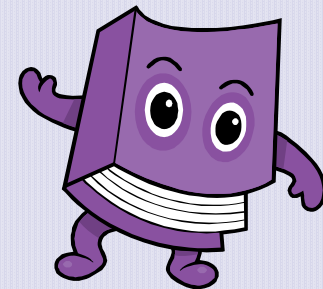
Tipos de operadores	Operadores
Unarios Sufijos	[] . op++ op--
Unarios Prefijos	++op --op -op ~ !
Creación o cast	new (tipo)expresion
Multiplicativos	* / %
Adición	+ -
Desplazamiento de bits	<< >> >>>
Relacionales Desigualdad	< > <= >= instanceof
Relacionales Igualdad	== !=
AND de bits	&
OR exclusivo de bits	^
OR de bits	
AND Lógico	&&
OR Lógico	
Operador ternario	?:
Asignación y operación	= += -= *= /= etc.

- false == false || true ?

■ Ejercicio workspace:

EjemploMath.java

- En este ejemplo se muestra cómo utilizar algunos de los métodos de la librería Math.



■ Conversión de tipos

- A veces necesitamos el valor en otro tipo de datos.

- ***conversión de tipos.***

- **NUNCA** modifica el tipo sino que la evalúa.
- La conversión puede acarrear la pérdida.
 - *XJ: double → int*
- No se puede convertir cualquier cosa a cualquier cosa.

■ Conversión de tipos (II)

□ Conversión implícita

- La propia expresión realiza la conversión.

```
int n = 10;  
double d = i;  
System.out.println(d);
```

□ Conversión explícita

- También conocido como **CAST**.
- Consiste en indicar el tipo entre paréntesis.

```
double d2 = 10.9;  
int i2 = (int)d2;  
System.out.println (i2);
```

■ Arrays

- En java a diferencia del lenguaje C, existe un tipo de variable “especial”, el Array
- Este tipo de variables no es más que un conjunto secuencial de memoria a las que se accede a través de un índice de posición.
- Los arrays en Java son objetos, por lo que cuentan con propiedades y métodos para manipularlos.
- Se pueden declarar arrays de tipos
 - De datos primitivos
 - De objetos (referencias)

■ Arrays: Tipos

- Arrays Unidimensionales o Vectores

```
tipo identificador[ ];    ó bien  
tipo [ ] identificador;  
int numeros[ ];
```

- Arrays Bidimensionales o Matrices

```
identificador = new tipo[numFilas][numColumnas];  
int tablero[ ] [ ];  
tablero = new int[10][20];
```

- Arrays n-Dimensionales

```
int matrizNDimensional[ ] [ ] [ ];
```

```
matrizNDimensional = new int[2][3][4];
```

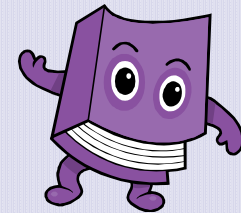
■ Arrays: Consideraciones

- Los elementos se inicializan *valor por defecto*
- **identificador.length** = Tamaño del array
- Para acceder se utiliza *[índice]*
 - *Donde el índice = 0 ... length - 1*
- Acceder a una posición que no existe
 - Provoca un **ArrayIndexOutOfBoundsException**.
- Ejemplo:
 - Definir un array de 10 elementos, (del 0 al 9) sumar los 5 primeros números y guardarlos en la posición 6
 - Comprobar que pasa cuando queremos guardar el elemento de la posición 0 en la posición 10

■ Ejercicio workspace:

EjemploArray.java

- En este ejemplo se muestra cómo utilizar de una manera básica los arrays y algunos de sus atributos (.length)



EjemploMatriz.java

- En este ejemplo se muestra cómo utilizar los arrays de arrays (matrices)

Ejercicio

- Crear dos matrices de tamaño 5x5. Asignarle números (por ejemplo, del 1 al 25). Crear una tercera matriz que contenga la suma de las dos primeras.

Ejercicio

- Crear una matriz de tamaño 5x5. Asignarle números (por ejemplo, del 1 al 25). Hacer que la aplicación invierta los números (los que están en la posición x,y ahora deben estar en la posición y, x)



Conclusiones

1. Programación Orientada a Objetos
2. **Introducción y Sintaxis Java**
3. Sentencias Control Flujo
4. POO en Java
5. Relaciones entre Objetos
6. Polimorfismo, abstracción e interfaces
7. Excepciones
8. Conceptos avanzados

- Introducción
 - Historia
 - Características
 - Entorno y Programas
- Sintaxis
 - Comentarios
 - Palabras reservadas
 - Tipo de datos
 - Constantes y Variables
 - Expresiones
 - Sentencias
 - Operadores
 - Conversión de datos
 - Arrays



Referencias

- Introducción al Lenguaje Java:
<http://java.sun.com/new2java/gettingstarted.jsp>
- Sintaxis Java
 - http://java.sun.com/docs/books/jls/second_edition/html/jTOC.doc.html