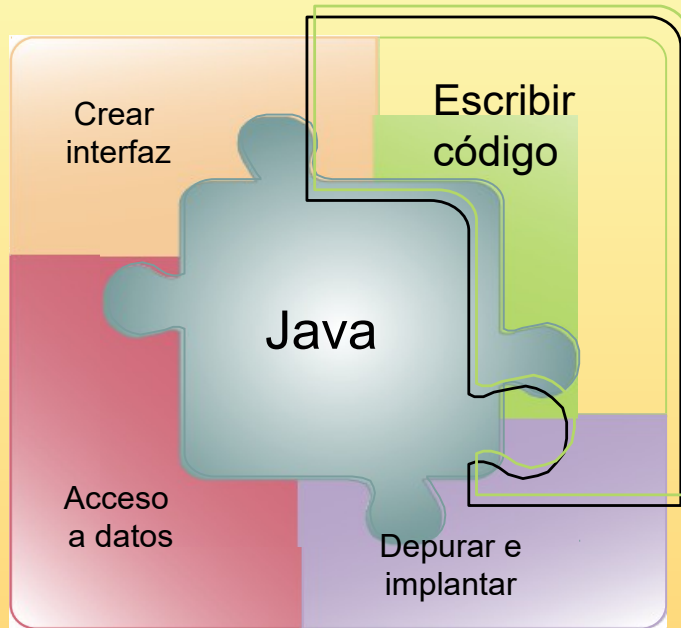


Elementos del lenguaje

■ Unidad 2

Descripción



1. Estructura básica de un programa
2. Variables y tipos de datos
3. Literales
4. Convertir tipos de datos
5. Operadores
6. Entrada /Salida
7. Tipos enumerados
8. Ejercicios
9. Para ampliar...

estructura básica

comentarios y separadores

Estructura básica de un programa

Estructura básica de un programa

```
/*
 * Estructura de una clase de Java
 * Si no es una clase principal el método main no aparece
 */

public class nombreDeLaClase {
    //Declaración de los atributos de la clase

    //Declaración de los métodos de la clase

    //El método main que indica dónde empieza la ejecución
    public static void main (String[] args) {
        //Declaración de las variables del método

        //Sentencias de ejecución del método
    }
}

//Si no es una clase principal el método main no aparece
```

Comentarios y separadores

■ Comentarios:

- `/* */`
- `//`
- `/** */`

■ Separadores:

- `(..)` Listas de parámetros en la definición y llamada a un método
- `{..}` Engloba bloques de código y en valores iniciales de arrays
- `[..]` En la declaración de arrays y en referencias a elementos de éstos
- `;` Separador de instrucciones
- `,` Separador de identificadores y argumentos
- `.` Separador de elementos de un objeto

variables y Tipos

tipos de datos

tipos de datos primitivos

identificadores de variables

declaración de variables

inicialización de variables

asignación

ámbito

constantes

Variables y tipos de datos

Variables y Tipos de datos

- Una variable permite almacenar los datos, resultados y resultados intermedios de un problema en un programa
- Tiene asociado un tipo de datos
- El tipo de datos al que pertenece una variable:
 - define el conjunto de valores que son susceptibles de ser almacenados en dicha variable
 - y las operaciones que se pueden realizar con ella
- Declaración de variables en Java:
Tipo Identificador ;

Tipos de datos

- En Java, los tipos de datos pueden clasificarse en dos grupos:
 - Tipo de datos primitivos
 - Tipo de datos referencia:
Strings, Arrays, Clases e Interfaces
Los estudiaremos más adelante...
- Más información sobre los tipos de datos básicos de Java :
<http://www.codexion.com/tutorialesjava/java/nutsandbolts/datatypes.html>

Tipos de datos primitivos

■ Tipos Numéricos

NOMBRE	TAMAÑO EN BITS	VALOR MÁXIMO
byte	8	127
short	16	32767
int	32	2147483647
long	64	9223372036854775807
float	32	3.4E38
double	64	1.7E+308

Tipos de datos primitivos

■ Tipo carácter

Nombre	Tamaño	Representa
char	2 Bytes	Representa caracteres como letras, números y caracteres especiales Java utiliza la codificación Unicode de 2 bytes

■ Tipo booleano o lógico

Nombre	Tamaño	Representa
boolean	1 Byte	Puede tomar los valores true/false

Identificadores de variables

- Los identificadores son los nombres que se les da a las variables, clases, interfaces, atributos y métodos de un programa
- Reglas para la creación de identificadores:
 - Java hace distinción entre mayúsculas y minúsculas:
 - var1, Var1 y VAR1 son distintos
 - La longitud máxima de los identificadores es prácticamente ilimitada
 - No puede ser una palabra reservada del lenguaje ni los valores lógicos true o false
 - No pueden ser iguales a otro identificador declarado en el mismo ámbito
 - Debe empezar por una letra y seguir con una sucesión de letras y dígitos. Se considera letra caracteres latinos, hebreos, cirílicos, etc... También '\$' (no utilizar) y '_' (subrayado)
 - No pueden utilizarse espacios en blanco ni símbolos coincidentes con operadores

Identificadores de variables II

- Convención:
 - los nombres de las variables y los métodos deberían empezar por una letra minúscula y los de las clases por mayúscula
 - Si el identificador está formado por varias palabras, la primera se escribe en minúsculas (excepto para las clases) y el resto de palabras se empiezan por mayúscula
 - variable: añoDeCreación
 - clase: HolaMundo

Declaración de variables

- **Para declarar una variable:**

```
tipo identificador;
```

```
tipo identificador = valor;
```

```
tipo ident1, ident2, ident3, etc...;
```

```
tipo ident1=valor1, ident2=valor2, etc...;
```

- **Por ejemplo:**

```
int edadPedro=34;
```

```
float precioPatata=1.2F, precioChoco=2.3F;
```

```
char car='A', car2='\u0041';
```

```
// 0041 es el código Unicode en hexadecimal de la A  
mayúscula
```

Inicialización de variables

- Si una variable no ha sido inicializada tiene un valor asignado por defecto:
 - Para las variables de tipo numérico, el valor por defecto es cero (0)
 - Las variables de tipo char, el valor '\u0000'
 - Las variables de tipo boolean, el valor false
 - Para las variables de tipo referencial (objetos), el valor null

Asignación

- La instrucción de asignación permite asignar valores a las variables o modificar los que ya tienen
- Su sintaxis es:
Identificador = Expresión;
- Se puede utilizar en el bloque de declaración de variables para definir valores iniciales:

```
int suma;  
char ch1, ch2='u';  
float f1=2.0F, f2;  
.....  
f2=34.67F;  
suma=suma+2;
```

Ámbito de una variable

- Se llama **ámbito de una variable** a la parte del programa en la que es conocida y se puede utilizar
- Una variable local se declara dentro del cuerpo de un método de una clase y es visible únicamente dentro de dicho método
- Se puede declarar en cualquier lugar del cuerpo, incluso después de instrucciones ejecutables, aunque es una buena costumbre declararlas justo al principio
- También pueden declararse variables dentro de un bloque parentizado por llaves { ... }
 - Sólo serán “visibles” dentro de dicho bloque
 - Las variables definidas en un bloque deben tener nombres diferentes

Constantes

- Para declarar una constante usamos el modificador *final*

final double PI = 3.1415926536;

- El valor de una constante no se puede modificar durante el programa
- Debemos darle un valor a la vez que se declara

literales

Literales I

- **Un literal es un valor que se expresa a sí mismo**
- **Literal entero puede expresarse :**
 - en decimal (base 10)
Ejemplo: 21
 - octal (base 8)
Ejemplo: 025
 - hexadecimal (base 16)
Ejemplo: x03A
 - Puede añadirse al final del mismo la letra L ó l para indicar que el entero es considerado como long
- **Literal real pueden expresarse:**
 - parte entera, el punto decimal (.) y la parte fraccionaria (Ej: 345.678 - 0.00056)
 - notación exponencial o científica (Ej: 3.45678e2 - 5.6e-4)
 - Se puede poner una letra como sufijo:
 - F ó f Trata el literal como de tipo float
 - D ó d Trata el literal como de tipo double
 - Por defecto el literal es double. Si deseamos que se interprete como float debemos añadir el sufijo F

Literales II

■ Literal carácter

- Se representan siempre entre comillas simples ('). Puede ser:
 - Un símbolo (letra)
Ejemplos: 'a' , 'B' , '{' , 'ñ' , 'á'
 - El código Unicode del carácter en octal o en hexadecimal
'\141' código Unicode en **octal** equivalente a 'a'
'\u0061' código Unicode en **hexadecimal** equivalente a 'a'
 - Una “secuencia de escape”, para caracteres especiales

Secuencia Significado

'\" Comilla simple

'\"\" Comillas dobles

'\\' Contrabarra

'\b' Backspace

'\n' Cambio de línea

'\r' Retorno de carro

'\t' Tabulador

Literales III

■ Literal booleano:

- palabras reservadas true y false
Ejemplo: boolean activado = false;

■ Literal Strings o cadena de caracteres

- No forman parte de los tipos de datos elementales en Java
- Encerrado entre comillas dobles (")

Ejemplo:

```
System.out.println("Primera línea\nSegunda línea del string\n");
```

```
System.out.println("Hol\u0061");
```

En java tenemos otro metodo semejante a println que es printf, este último admite caracteres de especialización en el formato impreso de los datos. El % significa aparecerá una expresion de formato Ejemplo:

```
System.out.printf("El area es: % .2f\n", area);
```

4.- Convertir tipos de datos

Conversión de tipos

- Cuando se realiza una instrucción de asignación:

Identificador = expresion;

tanto la variable como la expresión deben de ser del mismo tipo o de tipos compatibles

- Una expresión puede asignarse a una variable siempre que sea de un tipo de tamaño menor que el tipo de la variable. Por lo tanto podemos asignar en este orden:

short → int → long → float → double

- Otras formas de conversión de tipos se pueden realizar explícitamente a través de lo que se llama casting

(tipo) expresión Ejemplo:

num= (int) 34.56

- También utilizando funciones adecuadas de ciertos paquetes.

operadores unarios

operadores aritméticos

operadores de comparación

operadores lógicos

combinar operadores lógicos y de comparación

operadores de asignación

operadores de concatenación

operadores de bit

prioridad de operadores

5.- Operadores

Operadores Unarios

■ Signo

- Poner un signo + o un signo - delante de una expresión
- Ejemplo:
+45
-32

■ Incremento (++) y Decremento (--)

- Aumentar y disminuir en 1 el valor de la variable
- Pueden ir delante(pre) o detrás(post) de la variable
- Ejemplo:
int valor, i=5;
i++; // ahora i vale 6. Es equivalente a i=i+1;
--i; // ahora i vale 5. Es equivalente a i=i-1;
- La diferencia entre pre y post aparece en una instrucción compuesta:
valor=i++; //ahora valor vale 5 y i vale 6
// Es equivalente a { valor=i; i=i+1; }
valor=++i; // ahora valor vale 7 y i vale 7
// Es equivalente a { i=i+1; valor=i; }

Operadores aritméticos I

- Pueden realizar operaciones aritméticas que implican el cálculo de valores numéricos representados por literales, variables, otras expresiones, llamadas de funciones y propiedades, y constantes

- **Sintaxis:**

```
expresion1 operador_aritmético expresion2
```

- **Ejemplo:**

```
int x;  
x = 52 * 17;  
x = 120 / 4;  
x = 67 + 34;  
x = 32 - 12;  
X = 7 % 2;
```

Operadores aritméticos II

+ Suma

- Resta

*** Multiplicación**

/ División

% Resto de la división entera

Operadores de comparación I

- Símbolos que evalúan expresiones condicionales y devuelven un valor boolean
- **Sintaxis:**

`expresion1 operador_de_comparación expresion2`

Operador

< (Menor que)

<= (Menor o igual que)

> (Mayor que)

>= (Mayor o igual que)

= (Igual a)

!= (Distinto de)

Operadores de comparación II

■ Ejemplo:

```
int  cantidad;  
boolean pedidoGrande;  
pedidoGrande = cantidad > 1000
```

```
boolean testResult ;  
testResult = ( 45 < 35 )  
testResult = ( 45 == 45 )  
testResult = ( 4 != 3 )  
testResult = ( 'a' > 'b' )
```

Operadores lógicos I

- Los operadores lógicos realizan una evaluación lógica de expresiones y devuelven un valor boolean

- **Sintaxis:**

```
expresion1 operador_lógico expresion2
```

- **Ejemplo:**

```
edad>18 && sexo=='H'
```

```
edad>18 || sexo=='H'
```

Operadores lógicos II . Tablas de verdad

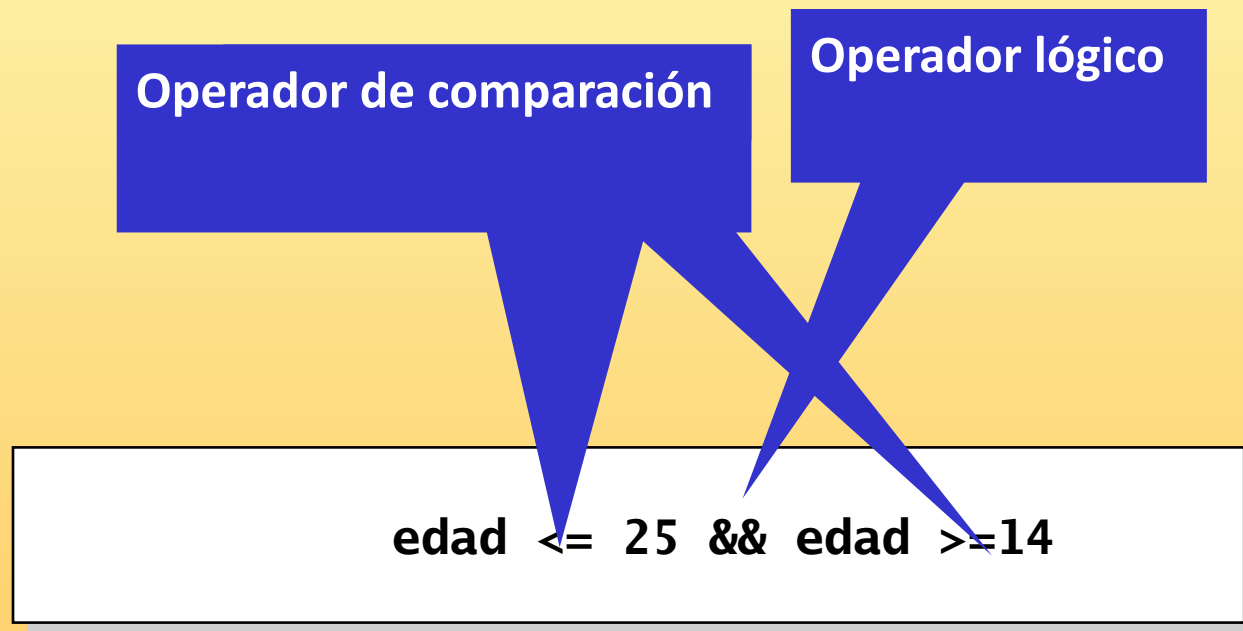
Son los operadores

&& (AND), || (OR), ! (NO)

A	B	A&&B	A B	!A
V	V	V	V	F
V	F	F	V	F
F	V	F	V	V
F	F	F	F	V

Combinar operadores lógicos y de comparación

- Podemos combinar operadores de comparación y operadores lógicos con instrucciones condicionales
- Ejemplo:



Operadores de asignación

Operador	ejemplo
=	edad = 34
+=	edad += 1 edad = edad +1
- =	edad - = 3 edad = edad - 3
* =	edad *= 2 edad = edad * 2
/=	edad /= 2 edad = edad / 2
% =	edad %= 2 edad = edad % 2

Operador de concatenación y Ternario

- Permite generar una cadena de caracteres a partir de otras dos

expresion1 + expresión2

Ejemplo: **“Hola” + “, “ + ”buenos días”**

El operador **? :**


Se interpreta como si la comparacion es cierta entonces ? se ejcecuta la accion tras el interrogante si la comparacion es falsa se ejecuta la accion tras los dos puntos

C ? V1 : V2 Si es cierto C devuelve v1 y si C es falso devuelve v2

Operadores de bits

OPERADOR	USO	SIGNIFICADO
&	A & B	AND lógico. A AND B
	A B	OR logico. A OR B
^	A ^ B	XOR logico. A XOR B
<<	A << B	Desplaza a la izquierda A B bits rellenando con ceros por la derecha
>>	A >> B	Desplaza a la derecha A B bits rellenando con el bit de signo ceros por la izquierda

Prioridad de operadores I

- Cuando aparecen varias operaciones en una expresión se evalúa y se resuelve en un orden predeterminado
- Orden por niveles:
 1. Paréntesis, de dentro a fuera
 2. Unarios
 3. Aritméticos
 - A. Multiplicativos: * / %
 - B. Sumativos: + -
 4. Comparativos o Relacionales
 5. Lógicos o booleanos
 - A. Not
 - B. And
 - C. Or
 6. Asignación
-  Cuando aparecen operadores de la misma prioridad juntos en una expresión el compilador evalúa cada operación de izquierda a derecha

Prioridad de operadores II

■ Ejemplos

`a = -3 + 5 + 2 * 4 - 6 / 4 * 3 - 5 % 2;`

Valor final a igual a 6

`a = 2 + 5 * 3; // * mas precedencia que +.`
`System.out.println ("1. " + a);`

`a = 2 + (5 * 3);`

Valor final a igual a 17

Salida de datos por pantalla

Entrada de datos del teclado

6.- entrada / salida

Salida de datos por pantalla

- Utilizaremos los métodos `print()` / `println()` o `printf()`

- `println()` incluye el retorno de carro al final de la salida, mientras que `print()` no.

```
System.out.print("Se imprime este mensaje sin el  
retorno de carro");
```

```
System.out.println("Se imprime este mensaje con  
un retorno de carro");
```

`Printf(" texto en pantalla con simbolo especial %
que será sustituido por la variable asociada en
orden"). Ejemplo`

```
System.out.println("Variable entera a= %d \n ,  
Variable real con dos decimales b= % .2f € ")
```

Entrada de datos del teclado I

- El método `read()` lee un solo carácter

```
char c = (char) System.in.read();
```

- Esta manera de leer del teclado es muy poco práctica y sería tedioso programar la lectura de un número, de una cadena de caracteres...
- Declarar un objeto de la clase `Scanner` y usar sus métodos
 - Entenderemos los detalles en próximos temas
 - En el ejemplo siguiente vemos como hacerlo

Entrada de datos del teclado II

```
//Importamos el fichero donde están las clases
import java.util.Scanner;

public class Exemple {
    public static void main (String[] args) {
        int primerNum;

        //Se declara el objeto lector de la clase Scanner
        Scanner lector = new Scanner(System.in);

        ...

        System.out.print("Escriu un número i polsa retorn: ");
        //Se lee un valor entero por teclado y se espera el retorno.
        primerNum = lector.nextInt();
        lector.nextLine();

        ...
    }
}
```

Entrada de datos del teclado III

<u>Método</u>	<u>Tipo de dato leído</u>
---------------	---------------------------

<code>lector.nextByte()</code>	<code>byte</code>
--------------------------------	-------------------

<code>lector.nextShort()</code>	<code>short</code>
---------------------------------	--------------------

<code>lector.nextInt()</code>	<code>int</code>
-------------------------------	------------------

<code>lector.nextLong()</code>	<code>long</code>
--------------------------------	-------------------

<code>lector.nextFloat()</code>	<code>float</code>
---------------------------------	--------------------

<code>lector.nextDouble()</code>	<code>double</code>
----------------------------------	---------------------

<code>lector.nextBoolean()</code>	<code>boolean</code>
-----------------------------------	----------------------

<code>lector.next()</code>	<code>String</code>
----------------------------	---------------------

<code>Lector.nextLine()</code>	<code>String</code>
--------------------------------	---------------------

Ejemplo

```
import java.util.*;

class Adivinanza {

    public static void main (String args[]) {
        Scanner tcl = new Scanner(); int valor;
        System.out.println("Piensa un numero"); tcl.nextLine();
        System.out.println("Multiplicalo por 5");tcl.nextLine();
        System.out.println("Sumale 6"); tcl.nextLine();
        System.out.println("Multiplicalo por 4");tcl.nextLine();
        System.out.println("Sumale 9"); tcl.nextLine();
        System.out.println("Multiplicalo por 5");tcl.nextLine();
        System.out.println("Escribe el resultado");
        valor=tcl.nextInt();
        System.out.println("El numero que habias pensado es: ");
        System.out.println((valor-165)/100);

    }

}
```

7.-Tipos Enumerados



Tipos Enumerados

- Son conjuntos de valores constantes para los que no existe un tipo predefinido
 - Por ejemplo: para representar los días de la semana, estaciones del año, meses del año, etc...
- Se implementa de la siguiente manera:

```
public class Semana {  
    public enum DiaSemana{LUNES, MARTES, MIERCOLES, JUEVES,  
        VIERNES, SABADO, DOMINGO}  
    public static void main (String[] args){  
  
        DiaSemana hoy = DiaSemana.JUEVES;  
        DiaSemana ultimo=DiaSemana.DOMINGO;  
  
        System.out.println( "Hoy es " +hoy +"\\n Y el ultimo dia es  
            "+ultimo );  
    }  
}
```

8.- Ejercicios

Ejercicio:

- **Escribe un programa que permita introducir dos números y calcular la suma de los mismos**
- **Provoca errores en el programa:**
 - Errores de compilación:
 - No declarar variables
 - Asignar tipos
 -
 - Errores de ejecución:
 - Introducir un string
 - Introducir un entero demasiado grande
 - ...

la clase Math

el tipo String

sistema de tipos

9.- Para ampliar...

Algunas funciones predefinidas. La clase Math

■ Las constantes E y PI

`Math.E=2.7182818284590452354`

`Math.PI=3.14.159265358979323846`

■ Las funciones de redondeo, con x de tipo double:

- `ceil(x)` : devuelve el número entero más pequeño que es mayor o igual a x
- `floor(x)` : devuelve el número entero más grande que es menor o igual a x
- `round(x)` : convierte el real x al entero más próximo

■ Funciones trigonométricas

■ `sin(x)` : calcula el seno del ángulo (en radianes) x

■ `cos(x)` : calcula el coseno del ángulo (en radianes) x

■ `asin(x)` : calcula el arco seno del ángulo x (x entre -1 y 1)

■ `acos(x)` : calcula el arco coseno del ángulo x (x entre -1 y 1)

■ `atan(x)` : calcula el arco tangente del ángulo x

Algunas funciones predefinidas. La clase Math

■ Otras funciones

- `abs(x)` : calcula el valor absoluto de x (entero o real)
- `exp(x)` : calcula e elevado a x (x es real)
- `log(x)` : calcula el logaritmo natural de x (x real y no negativo)
- `max(x,y)` : compara los números x e y (enteros o reales) y devuelve el mayor
- `min(x,y)` : compara los números x e y (enteros o reales) y devuelve el menor
- `pow(x,y)` : calcula x elevado a y . No está definida si x es negativo o 0 e y no es entero, ni tampoco si $x=0$ e y es negativo o 0
- `random()` : genera un número (pseudo)aleatorio entre 0.0 y 1.0
- `sqrt(x)` : calcula la raíz cuadrada de x (x no negativo)

Tipo String. Uso sencillo

- El tipo String permite representar secuencias de caracteres
- Ejemplo:

```
String frase, palabra, linea;
```

```
frase="En un lugar de la Mancha de cuyo nombre..."
```

```
//.....solicitamos al usuario una frase
```

```
//..... y la Frase de entrada es: Oh! es terrible
```

```
palabra=tcl.next(); //palabra = Oh!
```

```
linea=tcl.nextLine(); // linea = Oh! es terrible
```

Sistema de tipos

- **Dos categorías generales de tipos:**

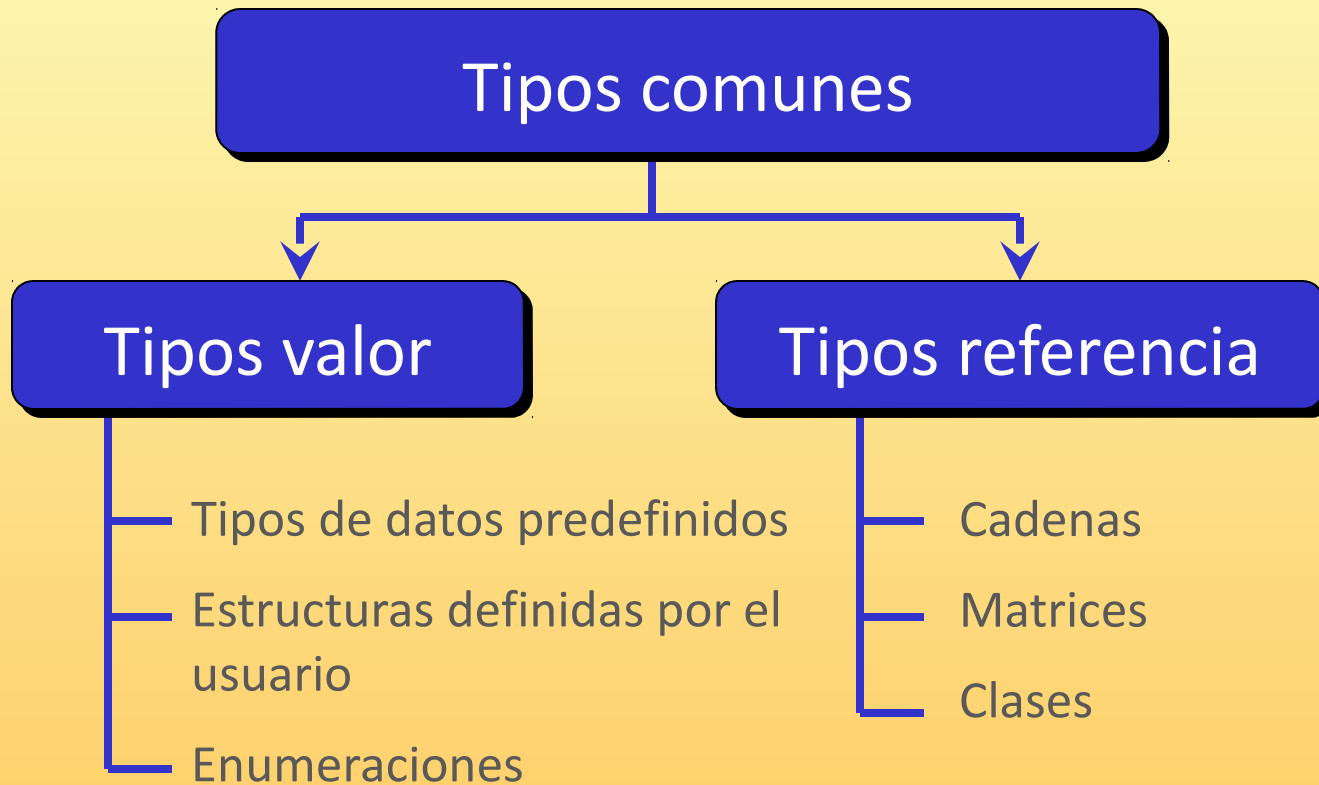
- ***tipo simple o valor***

- Una variable de tipo valor contiene directamente sus datos
- Cada variable de tipo valor tiene su propia copia de datos, de modo que las operaciones en una variable de tipo valor no pueden afectar a otra variable.

- ***tipo referencia***

- Una variable de tipo referencia contiene una referencia o puntero al valor de un objeto
- Dos variables de tipo referencia pueden referirse al mismo objeto, de modo que las operaciones en una variable de tipo referencia pueden afectar al objeto referenciado por otra variable de tipo referencia.

Sistema de tipos



Sistema de tipos

Los tipos de datos simples pueden ser declarados como referenciales (objetos) ya que existen clases que los engloban

Tipos de datos simples Clase equivalente

byte	java.lang.Byte
short	java.lang.Short
int	java.lang.Integer
long	java.lang.Long
float	java.lang.Float
double	java.lang.Double
char	java.lang.Character
boolean	java.lang.Boolean