



El futuro digital  
es de todos

MinTIC



Universidad  
Pontificia  
Bolivariana

Vigilada Mineducación

Misión  
TIC 2022

# Wrappers y modificadores de acceso



## Wrappers

Java es un lenguaje orientado a objetos *casi* puro. Aparte de estos, existen las entidades que no son objetos. Son valores de tipo simple, es decir, números (enteros y reales), caracteres y booleanos. En el cuadro II-1 se ilustran los tipos de datos primitivos en Java. Para cada tipo de dato primitivo, en Java existe una clase correspondiente que representa el mismo tipo de dato en términos de objetos. A estas clases se les llama *wrappers*.

Tipo de dato	Representación	Tamaño (Bytes)	Rango de Valores	Valor por defecto	Clase Asociada
byte	Numérico Entero con signo	1	-128 a 127	0	Byte
short	Numérico Entero con signo	2	-32768 a 32767	0	Short
int	Numérico Entero con signo	4	-2147483648 a 2147483647	0	Integer
long	Numérico Entero con signo	8	-9223372036854775808 a 9223372036854775807	0	Long
float	Numérico en Coma flotante de precisión simple Norma IEEE 754	4	$\pm 3.4 \times 10^{-38}$ a $\pm 3.4 \times 10^{38}$	0.0	Float
double	Numérico en Coma flotante de precisión doble Norma IEEE 754	8	$\pm 1.8 \times 10^{-308}$ a $\pm 1.8 \times 10^{308}$	0.0	Double
char	Carácter Unicode	2	\u0000 a \uFFFF	\u0000	Character
boolean	Dato lógico	-	true ó false	false	Boolean
void	-	-	-	-	Void



El futuro digital  
es de todos

MinTIC



La clase wrapper de un dato primitivo representa el mismo tipo de dato pero en un estilo orientado a objetos y proporciona funcionalidad clásica para operar con el tipo de dato que representa. Enseguida presentamos la lista de clases wrapper para cada tipo primitivo. Nótese que los nombres de las clases wrappers comienzan con mayúscula.

```
bool → Boolean
char → Character
int → Integer
long → Long
float → Float
double → Double
void → Void
```

Así, por ejemplo, si tenemos:

```
Double precio;
```

*precio* se comporta como una variable de tipo *double*, pero también es un objeto y, por ejemplo, se obtiene su equivalente en cadena de caracteres, invocando uno de sus métodos de la siguiente forma: *precio.toString()*.

Mision  
TIC2022



El futuro digital  
es de todos

MinTIC



Universidad  
Pontificia  
Bolivariana  
Vigilada Mineducación

Mision  
TIC2022

## Definición de constantes

En Java, las constantes se definen como atributos con el descriptor *final* y, por ser constantes, se declaran *static*. La sintaxis es la siguiente:

```
static final <tipoConstante> <nombreConstante> = <valor>;
```

Por convención, los nombres de las constantes deben estar en mayúsculas, con las palabras separadas por subrayado “\_”. Por ejemplo, declaramos la constante *PI* en la clase *Test2*, de la siguiente forma:

```
public class Test2 {  
    public static final double PI = 3.1415;  
    ...  
}
```

Para asignar a la variable *v* el valor de la constante *PI*:

```
double v = Test2.PI;
```



# Operadores de Java

Operadores	Sintaxis	Ejemplo / Declaración
Aritméticos	+ [adición] - [sustracción] * [multiplicación] / [división] % [resto] ++ [incremento] -- [decremento]	suma=a + b; resta=c - d; x=a/b; i++; --c;
Comparación o relación	> [mayor que] >= [mayor o igual que] < [menor que] <= [menor o igual que] == [igual a] != [distinto de]	boolean ok=a < b;
Lógicos	&& [ambos ciertos]    [cierto al menos uno] ![negación]	(a && b) es cierto si a y b son ciertos
Sobre bits	>> [desplazamiento a la derecha] << [desplazamiento a la izquierda] >>> [desplazamiento sin signo] & [operador Y –and-]   [operador O –or-] ^ [operador O exclusivo –xor-] ~ [complemento a]	c=a   b;

Asignación	=, +=, -=, /=, *=, %=	a += b;
Conversión de tipos	(tipo) variable o expresión	a=(int)b/c;
Instanceof	objectName instanceof className	Permite saber si un objeto pertenece o no a una determinada clase. Es un operador binario que devuelve true o false
Condicional ? o ternario	Expresion_Booleana ? res1 : res2	Se evalúa Expresion_Booleana y si devuelve res1 es true, por el contrario, res2 si es false x=1; y=10; z= (x<y) ? x+3 : y+8; se asigna 4 a z, es decir x+3



El futuro digital  
es de todos

MinTIC

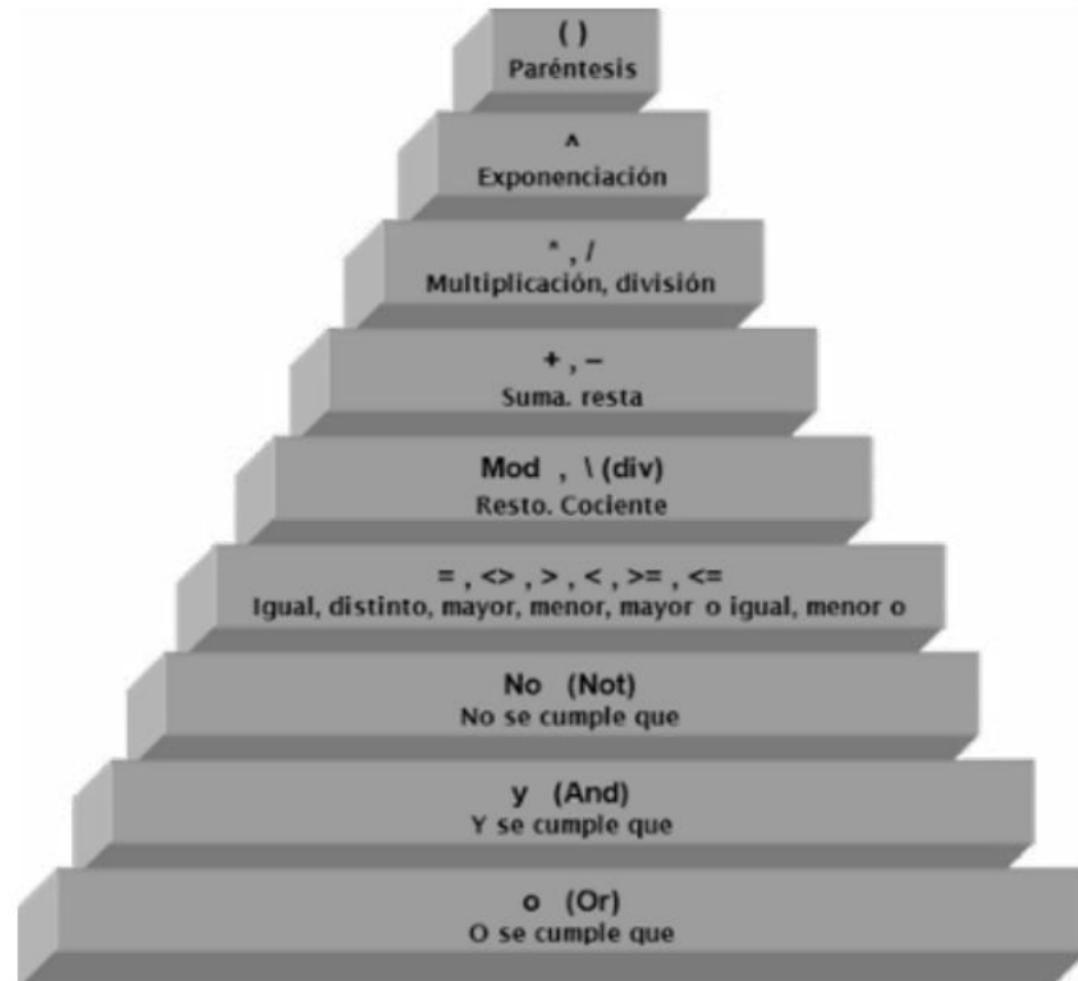


Universidad  
Pontificia  
Bolivariana

Vigilada Mineducación

## Prioridad de los operadores en Java

Los operadores en java tienen un orden de prioridad o procedencia para evaluar las operaciones, así como se observa en la siguiente figura.





El futuro digital  
es de todos

MinTIC

# Estructuras de control en Java

Son estructuras de control de flujo que permiten modificar el orden de la ejecución de las instrucciones de un programa.

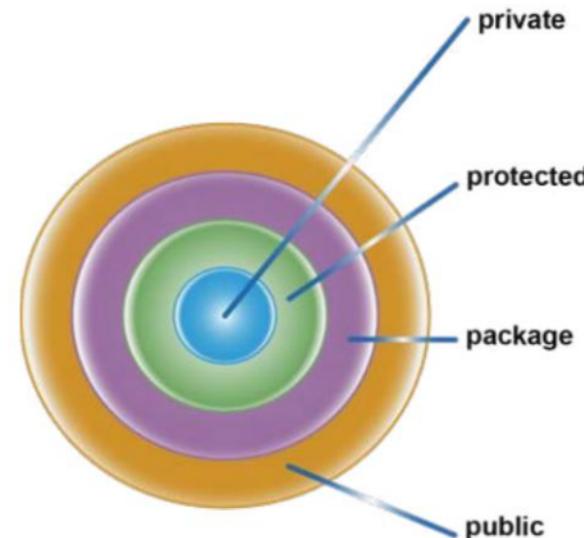
Operadores	Sintaxis	Ejemplo / Declaración
Bucles for	for( var=min; var < max; inc ) { // acciones }	for( i=0; i < 8; i++ ) { System.out.println(i); }
Bucles while	while( condición ) { // acciones }	while( i < 8 ) { System.out.println(i); i++ }
Bucles do..while	do { // acciones } while( condición );	do { System.out.println(i); i++; } while( i < 8 );
Bifurcaciones if..else	if( condición ) { // acciones } else { // acciones }	if( i== 0 ) { System.out.println("cero"); } else if( i==(i/2)*2) { System.out.println("par"); } else{ System.out.println("impar"); }
Bifurcaciones switch	switch( variable ) { case n1: // acciones break; case n2: // acciones break; // otros case default: // acciones alternativas }	switch( i ) { case 0: System.out.println("cero"); break; case 1: System.out.println("uno"); break; default: System.out.println("> uno"); }



## Visibilidad de una clase (Modificadores de acceso)

Los modificadores de acceso permiten establecer el nivel de ocultamiento de una clase en un paquete o de los miembros (datos y métodos) en una clase. Las protecciones pueden ser: *public*, *package*, *protected* y *private*. El nivel *public* es la más baja protección y *private* las más alta.

Los datos (variables) miembro de una clase especificadas como *private* o *protected* necesitan ser accedidos de alguna forma, por lo que se acostumbra implementar dos métodos públicos *get* / *set* para lectura y escritura respectivamente. A una clase sólo se pueden aplicar dos tipos de niveles de visibilidad: *public* o *package*. Si la clase no se define como *public* entonces, por defecto, es *package*.



Los miembros **private** de una clase sólo son accesibles por métodos de la propia clase y clases internas. Es el nivel más alto de ocultamiento.

Los miembros **protected** de una clase son accesibles por métodos de la propia clase, clases internas y clases derivadas.

Un paquete puede agrupar clases. Los miembros **package** de una clase son accesibles por métodos de las clases que conforman el paquete. Este es el especificador por defecto; se aplica cuando no se indica ninguno otro especificador. También se conocen como miembros amigables.

Los miembros **public** de una clase son visibles para los métodos de todas las clases. Es el nivel más bajo de protección, se puede referenciar (leer o modificar) desde cualquier parte.



El futuro digital  
es de todos

MinTIC



Misión  
TIC 2022

## Atributos estáticos

Para definirlos se antepone al tipo la palabra *static*; se deben inicializar independientemente a los constructores.

Estos atributos están compartidos por todos los objetos de la clase y son accesibles desde cada uno de ellos. Cualquier modificación por parte de un objeto afectará al resto.

Para el acceso a un atributo, en lugar de utilizar *this*, se utiliza el nombre de la clase: *Clase.atributo*.

Normalmente, las constantes asociadas a la clase son buenas candidatas para realizarlas estáticas. Un ejemplo sería:

```
public class MiClase{
    private static final int MAX = 100;
    //...
}
```



El futuro digital  
es de todos

MinTIC



Universidad  
Pontificia  
Bolivariana

Vigilada Mineducación

Mision  
TIC2022

## Métodos estáticos

Al igual que los atributos, se antepone la palabra *static* al tipo devuelto.

No puede utilizar en el código la palabra *this*, ya que está asociado exclusivamente a las instancias. Se pueden acceder a los atributos estáticos del mismo modo que pueden las instancias, anteponiendo el nombre de la clase: *Clase.atributo*.

Se permite definir un código estático para inicializar los atributos estáticos. Su ejecución se dispara una sola vez al principio. Tiene el siguiente formato:

```
static {  
    //...  
}
```

Vamos a anadir algunos aspectos estaticos a la clase Punto:

```
public class Punto {  
    public static final double PI = 3.1415;  
    private static int numeroInstancias;  
    private int x, y;  
    static {  
        Punto.numeroInstancias = 0;  
    }  
    //Constructores. Deben incrementar el atributo numeroInstancias  
    public static int instanciasCreadas() {  
        return Punto.numeroInstancias;  
    }  
    //...  
}
```



El futuro digital  
es de todos

MinTIC



Mision  
TIC2022

## Cadenas de caracteres - Clase String

En Java, las cadenas de caracteres son objetos de la clase *String*. Se tratan como objetos, pero sin cambios de estado, como valores constantes, es decir, el objeto no es modificable (deberían haberse llamado *ConstString*, pero quizás sea un nombre demasiado largo). Por lo tanto, las cadenas no deben ser vistas como contenedores de caracteres. En Java, la clase que permite manipular cadenas como contenedores de caracteres se llama *StringBuffer*. Los siguientes son ejemplos de cadenas de caracteres (*String*).

```
String nombre = "Sandra Alvarez";
String carrera = "Informatica";
String frase = nombre + " es un estudiante de la carrera de "
              + carrera;
```

Las constantes del tipo *String* se encierran entre comillas:

```
"hasta"
"luego"
```



El futuro digital  
es de todos

MinTIC



Universidad  
Pontificia  
Bolivariana

Vigilada Mineducación

Misión  
TIC 2022

El lenguaje Java proporciona un soporte particular para los objetos *String*, proporcionando el operador de concatenación + y también formas automáticas de conversión de tipos de datos primitivos en cadenas. Por ejemplo:

```
double valor = 17.5;  
String st = "La temperatura es " + valor + " grados";
```

En el ejemplo anterior, *valor* se convierte automáticamente en *String* y se concatena a las otras dos cadenas. Así obtenemos la cadena final:

"La temperatura es 17.5 grados"

que se asigna a la variable *st*



En Java, todos los objetos tienen una representación textual, que se obtiene invocando automáticamente al método `toString()` cuando es necesario. El método `toString()` pertenece a la clase `Object`, de la cual derivan todas las clases. En el siguiente ejemplo se invoca a `c.toString()` automáticamente cuando se usa `c` como parámetro de `println`:

```
Counter c = new Counter( 1 );
c.incrementar();
System.out.println( c );
```

Es posible definir una representación textual específica para los objetos de una clase *redefiniendo* el método `toString()`, por ejemplo:

```
class Persona {
    private String nombre;
    private Fecha fechaNacimiento;

    public Persona( String n, Fecha fn ) {
        nombre = n;
        fechaNacimiento = new Fecha( fn );
    }

    public String toString() {
        return nombre + " nació el: " + fechaNacimiento;
    }
}
```



Un error frecuente consiste en usar el operador == para probar la igualdad (o != para la desigualdad) entre dos cadenas. Es un error porque == compara las referencias a los objetos y no los propios objetos. La siguiente clase ilustra lo que sucede con diferentes tipos de comparaciones entre cadenas:

```
public class TestDeIgualdad {  
    public static void main( String[] args ) {  
        String s1 = "prueba";  
        String s2 = "prueba"; // se le asigna el mismo objeto  
                           // que a s1  
        String s3 = new String( "prueba" ); // otro objeto igual  
        boolean test1 = ( s1 == s2 ); // son el mismo objeto  
        boolean test2 = ( s1 == s3 ); // no son el mismo  
        boolean test3 = ( s1.equals( s2 ) ); // contienen lo mismo  
        boolean test4 = ( s1.equals( s3 ) ); // contienen lo mismo  
        System.out.println( test1 );  
        System.out.println( test2 );  
        System.out.println( test3 );  
        System.out.println( test4 );  
    }  
}
```

La salida del programa seria la siguiente:

```
true  
false  
true  
true
```



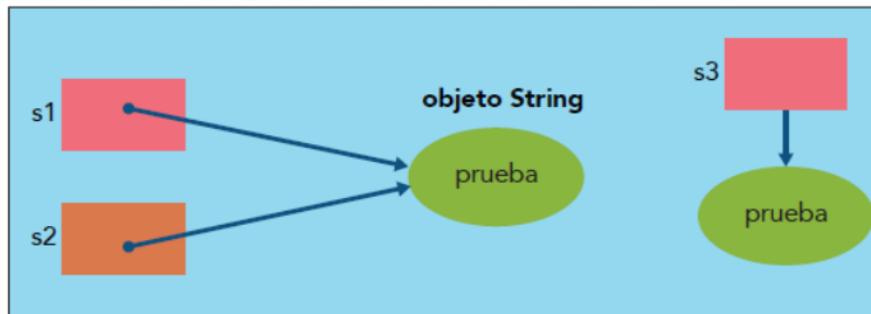
El futuro digital  
es de todos

MinTIC



Mision  
TIC2022

La siguiente figura ilustra la ubicación en la memoria de las cadenas de caracteres del código anterior:



Las instrucciones

```
String s1 = "prueba";  
String s2 = "prueba";
```

Indican que tanto s1 como s2 apuntan al mismo objeto: "prueba". Sin embargo con

```
String s3 = new String( "prueba" );
```

se aloja una nueva ubicación de memoria con el mismo contenido. Es por eso que con  $test1 = (s1 == s2)$  el resultado es verdadero y con  $test2 = (s1 == s3)$  el resultado es falso, ya que verificamos si s1, s2 y s3 están apuntando a la misma localidad y no si el contenido de lo que están apuntando es igual.



El futuro digital  
es de todos

MinTIC



Universidad  
Pontificia  
Bolivariana

Vigilada Mineducación

Mision  
TIC2022

Las instrucciones

```
test3 = ( s1.equals( s2 ) );
test4 = ( s1.equals( s3 ) );
```

sí comparan el contenido de las cadenas, por lo que en los dos casos la comparación resulta verdadera.



## Clase StringBuffer

Representa una cadena de caracteres con una estructura interna que facilita su modificación, ocupa más que la clase *String*, pero las operaciones de modificación son muchos más rápidas.

Para la modificación del contenido, cuenta con los métodos *insert* y *append*.

En el siguiente código, el bucle realizado con *StringBuffer* es aproximadamente 1000 veces más rápido que el bucle realizado con *String*. Es importante tener en cuenta, que cuando se requiera modificar una cadena de forma dinámica, es mucho mejor la clase *StringBuffer*.

```
String s = "";
StringBuffer sb = new StringBuffer();

for (int i = 0; i < 100000; i++) {
    s += ".";
}

for (int i = 0; i < 100000; i++) {
    sb.append(".");
}
```



El futuro digital  
es de todos

MinTIC



Misión  
TIC 2022

## Tipos de datos compuestos o agregados

Son tipos de datos cuyos valores representan una colección de elementos de datos; un tipo de dato compuesto o agregado se compone de tipos de datos definidos, entre estos están: arreglos, clases y estructuras de datos dinámicas.

Un array o arreglo es un conjunto de datos del mismo tipo de un tamaño de longitud finita, los datos son accesible en tiempo de ejecución, para acceder a los elementos individuales se requiere de un número entero denominado índice. El tipo de dato de los elementos de un arreglo pueden ser: primitivos o compuestos.

```
public static void main(String[] args) {  
    int[] numeros = new int[3];  
    numeros[0] = 5;  
    numeros[1] = 10;  
    numeros[2] = 15;  
    numeros[3] = 20;  
}
```

Los tipos de datos clase pueden crearse a partir de clases predefinidas en Java o de clases personalizadas.