Java

Tema 1

El lenguaje Java

Contenidos

1. Variables
2. Tipos de datos primitivos
3. Operadores
4. Entrada y salida de datos por consola

Java

Variables

3

Variables

- Una variable es un contenedor de datos, almacena información y se identifica mediante un nombre.
- **Declaración** de variables.

```
tipo_de_dato  nombreVariable;
```

- tipo_de_dato es el tipo de información que almacenará la variable,
- nombreVariable es el identificador de la variable

```
Ejemplo:   int   edad;
```

- Declaración e **inicialización** de variables.

```
tipo_de_dato  nombreVariable = valor_inicial;
```

- valor_inicial primer valor según el tipo de dato. La inicialización es opcional al declarar.

4

¡Para usar una variable en una expresión debe contener algún valor!

Variables

Ejemplo de declaración de variables

Se pueden declarar **varias variables a la vez** del mismo tipo de dato, separándolas por comas

```
public class EjemploVariables {
  public static void main(String[] args
    boolean unBoolean = true;
    int unInt = 10;
    double unDouble = 3.14, otroDouble;

    System.out.println("El boolean vale " + unBoolean);
    System.out.println("El Int vale " + unInt);
    System.out.println("El double vale " + unDouble);
  }
}
```

Todas las sentencias deben terminar con punto y coma ;

5

Variables: indentificador

- El nombre de una variable es único dentro de su contexto.
- Deben cumplir las siguientes reglas (ver en oracle.com):
  - Pueden tener uno o más caracteres de longitud
  - Deben empezar por una letra, subrayado (_) o el dólar ($).
  - Después del primer carácter se pueden usar números
  - Se distingue entre mayúsculas y minúsculas (case-sensitive)
  - No se permiten espacios, debe ser una sola palabra
  - No deben ser palabras reservadas por java como: new, for, if, etc…
  - **IMPORTANTE**: Deben reflejar el significado o uso de lo que almacena: longitud, numPersonas, grupo, etc...
  - La primera letra debe ser minúscula.

6

Palabras reservadas de java

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

Asignación de valores a variables

- El operador de asignación "=" asigna/almacena un valor (ó resultado de una expresión) a una variable. Tiene esta forma general:

nombre_variable = expresión/valor;

Ejemplos:

```
int x  = 5;     // x almacena  un 5
int y = 100;  // y almacena un 100
x = x + y;    // x almacena la suma de x y y (100+5)
x = 6;        // x almacena 6
```

El valor de 105 se ha perdido (sobreescrito)

Variables constantes

- Se utiliza el modificador **final** antes del tipo de dato.
- Una variable constante no puede modificar su valor una vez asignado:

final tipo_dato nombre_variable = expresión/valor;

Ejemplos:

```
final int IVA = 5;
IVA = 10;
```

Error: variable final ya inicializada

- Se definen con mayúsculas y representan valores fijos de nuestro programa. Ejemplos: IVA, CAMBIOEURODOLAR, PI …..

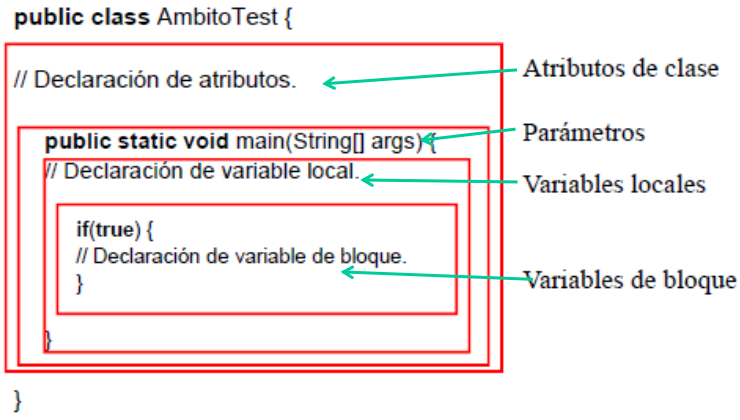
9

Ámbito de las variables

- Es la zona del código donde se puede referenciar/usar esa variable a través de su identificador.
- El lugar donde se define una variable establece su ámbito
- Ámbitos:
  - Atributos (o variables miembro)
  - Parámetros de método
  - Variables locales: siempre hay que inicializarlas
  - Variables de bloque: siempre hay que inicializarlas

10

Ámbito de las variables



11

Ámbito de las variables

```
public class AmbitoTest2 {

  public static void main(String[] args) {

    if(true) {
    int i = 12;
    }
  System.out.println("El valor de i es: " + i);
  }
}
```

Error: la variable i esta declarada dentro del bloque de sentencia if y no se puede usar una vez cerrado este bloque

12

Ámbito de las variables

```
public class AmbitoTest3 {

static int i = 5;
   public static void main(String[] args) {
   int i = 10;
   System.out.println("El valor de i es: " + i);
   }
}
```

Advertencia: la variable i esta definida dos veces. Prevalece la que esta más cerca de la función println

13

Java

Tipos de datos primitivos

14

Tipos de datos

- Java define dos categorías de tipos de datos: orientados a objetos y tipos primitivos.
  - Los tipos orientados a objetos están definidos por clases
  - Los tipos primitivos (elementales o simples) representan a números, letras, verdadero/falso … y **NO son objetos**.

15

Tipos primitivos.

Tipo	Descripción	Tamaño	Rango de valores
short	Número entero corto	2 bytes	-32768 a 32767
int	Número Entero con signo	4 bytes	-2^{31} a $2^{31}-1$
long	Número Entero largo con signo	8 bytes	-2^{63} a $2^{63}-1$
float	Número coma flotante simple precisión	4 bytes	$3.4x10^{-38}$ a $3.4x10^{+45}$
double	Número coma flotante doble precisión	8 bytes	$1.8x10^{-308}$ a $1.8x10^{+324}$
char	Carácter Unicode entre comillas simples	2 bytes	
boolean	Verdadero o Falso	1 byte	true o false

Ejemplos:

```
int a = 4;  long b = 10000L;   float c = 4.6;  double d = 10000.678;
char ch = 'A', otro='b';     boolean resultado =  true
```

16

Valores literales

- **Valor constante de tipo Entero.**
  - En decimal: 21, -45. Es la forma habitual, serán int
  - Para el tipo long se debe añadir un sufijo L (o l): 2L, 45l.

- **Valor constante de tipo Coma Flotante**
  - Por defecto, son de tipo double: 3.54, 0.7
  - Para el tipo float se debe añadir un sufijo F (o f): 10.5F, 5.5f
  - Sufijo D (o d) o nada para tipo double.

17

Literales

- **Valor constante de tipo Carácter.**
  - Carácter simple Unicode entre comillas simples: 'a', 'Z', '?', '!', '@', ...
  - Internamente se almacena el código numérico Unicode.
  - Caracteres especiales: entre comillas simples con **secuencia de escape**

Secu.	Descripción	Secu.	Descripción
\'	Comilla simple	\"	Comillas dobles
\\	Contrabarra	\r	Retorno de carro
\n	Salto de línea	\t	Tabulador
\b	Retroceso	\xxx	Constante octal
\0	Carácter nulo (null)		

Para almacenar una comilla simpler: `char comilla = '\'' ;`

Para almacenar una contrabarra: `char contrabarra = '\\' ;`

18

Literales

- **Valor constante de tipo cadena de texto.** Entre comillas dobles.

  Ejemplo:

```
String s = "Esto es una cadena";
```

- Las cadenas no son un tipo primitivo, sino objetos de la clase String.

19

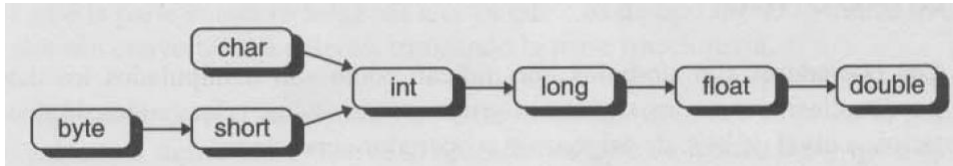
Conversión de tipos

- Conversión implícita ó automática de tipos. En una asignación, el valor del lado derecho es automáticamente convertido al tipo de dato del lado izquierdo si:
  - Los dos tipos son compatibles.
  - El tipo destino es más grande que el tipo de origen.

  Ejemplo:

```
float numero1;
numero1 = 2; //el entero 2 se convierte en float
```

- El sentido de las flechas indican las conversiones implícitas



Conversión de tipos

- Un *casting* (conversión explícita entre tipos incompatibles) es una sentencia para convertir un tipo en otro.

  Sigue la siguiente sintaxis: (tipo destino) expresión;

  Ejemplo:

```
int x = 5, y = 3, n;
double a = 12.5, b = 5.7;
a = (double)(x/y);  // correcto
n =(int)(a/b); //No se debe hacer, se pierde información
```


Java

Operadores

Tipos de Operadores

- Los operadores Java son:

- **Aritméticos:** Realizan operaciones aritméticas.
- **Relacionales:** Comparan números o caracteres
- **Lógicos:** evalúan expresiones lógicas (tipo boolean)
- **Asignación:** Realizan operación y asignación de una sola vez

23

Operadores Aritméticos

Operador	Significado
+	Suma
-	Resta (o signo menos)
*	Multiplicación
/	División entera si ambos operandos son enteros o decimal si alguno de los dos es nº decimal
%	Módulo (resto de la división entera)
++	Incremento en una unidad (x++ ó ++x ➔ x=x+1)
--	Decremento en una unidad (x– ó --x ➔ x=x-1)

24

Operadores aritméticos

// Ejemplo que demuestra el uso de los operadores aritméticos división y resto

```
class Main {
   public static void main ( string args[ ] ) {
      int iresul, iresto;
      double dresul, dresto;
      iresul = 10 / 3;
      iresto = 10 % 3;
      dresul = 10.0 / 3.0;
      dresto = 10.0 % 3.0;
     // cociente y resto enteros
      System.out.println("Resultado de 10/3: " + iresul);
      System.out.println("Resto de 10/3: " iresto);
     // cociente y resto en coma flotante
      System.out.println("Resultado de 10.0 /3.0: " + dresul);
      System.out.println("Resto de 10.0 /3.0: " + dresto);
   }
}
```


Operadores aritméticos

- Operadores incremento ++ y decremento --.

++x , --x : Primero se incrementa / decrementa y después se obtiene el valor

x++, x-- : Sufijo. Primero se obtiene el valor y después se incrementa / decrementa

```
int x = 0, y = 0;
y = ++x; //primero incrementa x = x+1
         // y luego asigna el valor  y = x
System.out.println("y vale: " + y + " ,x vale " + x);
y = x++; //primero asigna el valor y = x
         // y luego incrementa x = x+1
System.out.println("y vale: " + y + " ,x vale " + x);
```

Resultado

y vale 1, x vale 1
y vale 1, x vale 2

Operadores relacionales

Operador	Significado	Ejemplo
<	menor que	a < 10
<=	menor o igual que	a <= 10
>	mayor que	a > 10
>=	mayor o igual que	a >= 10
==	igual que	a == 10
!=	distinto	a != 10

El resultado de la comparación es un boolean (true o false)
Si "a" vale 5 ¿Qué resultado devolvería cada comparación?

27

Operadores lógicos

Operador	Operandos	Significado	Ejemplo
!	unario/uno	NO (NOT)	!bisiesto
||	binario/dos	O (OR)	c == 1 || d == 2
&&	Binario/dos	Y (AND)	(d != 0) && (d > 10)

- Tabla de verdad

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

Evalúa las condiciones ejemplo sabiendo c=1, d=3 y bisiesto=false

28

Operadores de asignación

- Java proporciona operadores especiales de asignación que sirven para **simplificar ciertas operaciones**.
- Los operadores de asignación de método abreviado son:

Operador	Ejemplo		Significado
+=	a += 5;	equivale	a = a + 5;
-=	b -=3;	equivale	b = b - 3;
*=	c *= 7;	equivale	c = c * 7;
/=	d /= 2;	equivale	d = d / 2;
%=	f %=6;	equivale	f = f % 6;

29

Precedencia de operadores.

Mayor precedencia

( ) [ ] .
++ - - ~ !
new (tipo) conversion
* / %
+ -
> >= < <=
== !=
&&
||
?: (condicional)
= +=, -=, *=,...

Menor precedencia

Ver en oracle.com

30

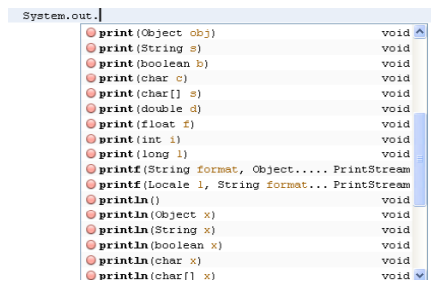
Java

Entrada y salida de datos por consola

Salida de datos por consola (out)

- El objeto System.out representa la consola del sistema, y cuenta, entre otros, con los métodos println y print que envían cadenas de caracteres a la consola.

```
System.out.print("Hola");
System.out.println("Hola");
```



Salida de datos por consola (out)

```
/* Un sencillo programa que escribe varias líneas de texto en la
pantalla, mediante println y print*/
```
comentario varias líneas

```
public class Main {
// método main donde se inicia el programa
```
comentario de una línea

```
   public static void main() {
```
Inicio de bloque de código

```
      System.out.print("En un lugar de la mancha ");
      System.out.println("de cuyo nombre no quiero acordarme.");

      System.out.print ("No ha mucho tiempo que ");
      System.out.println("vivía Don Quijote.");
```
final de sentencia

```
   }
```
final de bloque de código

```
}
```


Salida de datos por consola (out)

- El método **printf** nos permite crear una **cadena de texto con formato** intercalando valores de variables.
- El símbolo **%** representa a una variable de la lista final en orden. La letra que le sigue representa al tipo de datos de esa variable.

```
int edad=50;
String nombre="Pepe";
System.out.printf("%s tiene %d años", nombre, edad);
```

símbolo	Tipo de datos
%d	Entero (short, int, long)
%s	String
%c	char
%f	Decimal (double, float)
%b	boolean

Salida de error por consola (err)

- El objeto System.err representa la salida de error, que se realiza por la consola
- Cuenta, entre otros, con los métodos println y print que envían cadenas de caracteres a la consola y aparecen en color rojo.

```
System.err.println("Error");
```

```
compile-single:
run-single:
Error
BUILD SUCCESSFUL (total time: 1 second)
```


Entrada de datos por consola.

- La clase Scanner permite leer datos desde la entrada tal como el teclado, un fichero, o incluso un texto en una cadena de caracteres en memoria.

Representa al teclado

```
Scanner sc = new Scanner(System.in);
```

- Los métodos principales de la clase Scanner son:

String **next**()	Lee un solo valor de tipo **String**
String **nextLine**()	Lee una **cadena de texto** hasta un salto de linea

```
System.out.println("Introduce tu nombre y apellidos");
Strins nombre = sc.nextLine();
```

Entrada de datos por consola.

La clase Scanner se encuentra en el paquete java.util que hay que importar para poderla usar en el programa.

```
Ejemplo de uso de la clase Scanner
import java.util.Scanner;

public class Main{
   public static void main(){

      Scanner datos = new Scanner(System.in);

      System.out.print("Introduce tu nombre y apellidos: ");
      String nombre = datos.nextLine();

      System.out.println("Tus datos son: ");
      System.out.println(nombre);
   }
}
```

37