Programación. 1º DAW

Unidad 2:
Identificación de elementos de un programa informático

Identificación de elementos de un programa informático

- Estructura y bloques fundamentales.
- Variables
  - Tipos de datos básicos
  - Literales
  - Nombres de las variables
  - Variables estáticas o constantes
- Operadores en java
  - Aritméticos
  - De concatenación de cadenas.
  - Relacionales o de comparación
  - Lógicos
- Separadores
- Comentarios
- Conversiones de tipos

Estructura y bloques fundamentales

- ***Recuerda:*** Un programa es una serie de órdenes o instrucciones ordenadas con una finalidad concreta que realizan una función determinada.

- Los programas mas simples escritos en lenguajes imperativos suelen realizar tres tareas de forma secuencial:
  - Entrada de datos
  - Procesamiento de los datos
  - Salida de resultados

- *Al ser Java un lenguaje de POO, todo debe estar dentro de una clase, quedando así:*

```
public class miclase
{
    Sentencias escritas en java
}
```

Estructura y bloques fundamentales

- El punto de entrada en un programa en Java es la función o método *main*:

```
public static void main (Strings args[])
{
    ..........
}
```

- Cuando java va a ejecutar el código de una clase, lo primero que hace es buscar el método ***main*** de dicha clase para ejecutarlo
- El método ***main*** tiene las siguientes particularidades:
  - Es público (**public**). Esto es para poder llamarlo desde cualquier lado.
  - Es estático (**static**). Se le puede llamar sintener que instanciar la clase.
  - No devuelve ningún valor (modificador **void**).
  - Admite una serie de parámetros (**String[] args**), donde el parámetro de *args* es una tabla de referencias a objetos de tipo String que se usa para recibir parámetros de la línea de comandos del sistema, aunque también podemos ignorarlo si no hace falta.
- Para mostrar un texto por pantalla utilizamos la siguiente linea:

```
System.out.println("Nuestro texto.....");
```

Variables: Tipos de datos básicos

- En Java hay ocho tipos básicos de variables o tipos primitivos:

Tipo	Ocupa	Valor por defecto	Valores
`boolean`	1 bit	false	false, true
`byte`	8 bits	0	-128...127
`short`	16 bits	0	-32768 ... 32767
`char`	16 bits	`\u0000`	`\u0000 ... \uffff`
`int`	32 bits	0	-2147483648 ... 2147483647
`long`	64 bits	0	-9,22E18 ... 9,22E18 (aprox)
`float`	32 bits	0.0	reales de 32 bits IEEE 754
`double`	64 bits	0.0	reales de 64 bits IEEE 754

- Todos las demás variables se construyen a partir de éstas usando vectores (arrays) o creando clases.

Variables: Tipos de datos básicos

- Para declarar variables se escribe el tipo de la variable que se quiere declarar y luego los nombres de las variables declaradas separados por comas y terminando la lista por punto y coma. Ejemplos:
  - int i,j,k,n;
  - char a;
  - double x,y,z,pi;
  - String s;

- Se pueden declarar variables sin asignarles valores, es decir, sin inicializarlas.

- **No** se pueden declarar variables sin especificar su tipo.

Variables: Tipos de datos básicos

- Para inicializar las variables en el momento de declararlas se utiliza el **operador de asignación =**.
  - int i=0,j=-1,k=0177,n=0xff0000ff;
  - char a='z', b=\0032, c='\n';
  - boolean A=false, B=true;
  - float x=(float)0.123;
  - double pi=Math.PI;
- Observe que para asignar un float es necesario hacer una conversión de tipo (class casting) al literal pues los literales con punto decimal siempre son de tipo double.
- ***Recuerda:***
  - Una variable es una zona de memoria donde se puede almacenar información del tipo.
  - La variables miembro de una clase se inicializan por defecto (las numéricas con 0, los caracteres con '\0' y las referencias a objetos y cadenas con null) mientras que las variables locales no se inicializan por defecto.

Literales

- Hay **cuatro** tipos de literales: booleanas, numéricas, de caracteres y de cadenas.
  - Los **literales numéricas** se utilizan tanto para las variables *enteras* (byte, short, int y long) como para las variables de *coma flotante* (float y double).
    - Todas ellas se escriben utilizando *dígitos* y el signo - para indicar cantidades negativas.
    - El separador de decimales es el punto y no la coma.

Literales

- **Literales numéricos**
  - Para las variables de tipo *entero* se pueden usar literales *decimales* (en base 10), *hexadecimales* (en base 16) y *octales* (en base 8).
    - Los literales *decimales* utilizan como dígitos: 0,1,2,3,4,5,6,7,8 y 9. Ejemplos:
      - 25, 2678, -12567
    - Los literales *hexadecimales* utilizan como dígitos: 0,1,2,3, 4,5,6, 7,8,9, A,B,C,D,E y F. Las letras A,B,C,D,E y F pueden ser mayúsculas o minúsculas. Para indicar que un literal es *hexadecimal* se le escribe como prefijo: 0x. Ejemplos:
      - 0x1F, 0x2678, -0xa2e0cd
    - Los literales *octales* utilizan como dígitos: 0,1,2,3,4,5,6 y 7. Para indicar que un literal es *octal* se le escribe como prefijo un 0 (cero). Ejemplos:
      - 075, 02676, -0563725

Literales

- Los **literales de caracteres**, son los que pueden asignar a variables de tipo char, se escriben entre comillas simples. Por ejemplo:
  - 'a', 'X', '_', ' ', 'ñ', '\n'
  - Hay algunos caracteres *especiales* que no se pueden escribir con el teclado o que juegan un papel especial, y que se representan con una diagonal invertida seguida de una letra u otro símbolo. Estos caracteres especiales son los siguientes:

retroceso	'\b'
alimentación de forma	'\f'
paso de línea	'\n'
retorno de carro	'\r'
tabulador	'\t'
diagonal invertida	'\\'
doble comilla	'\"'
comilla sencilla	'\''

  - La diagonal invertida también se puede utilizar para representar los caracteres usando números. Por ejemplo:
    - '040', '\u0020'
    - Son dos representaciones del caracter *espacio* ' ' utilizando números. La primera es octal y la segunda es la representación *unicode*

Literales

- Los **literales booleanas** son solamente true y false.
  - En Java, a diferencia del C, a las variables booleanas no se les pueden asignar valores numéricos.
- Los **literales de cadenas** se escriben entre comillas dobles.
  - Las cadenas son los objetos de la clase String y noconstituyen un tipo básico, aunque en muchos sentidos se tratan como si lo fueran.
  - Ejemplos de literales de cadenas son:
    - "Hello World", "primera\040línea\nsegunda\u0020línea"
    - Como puede verse en el segundo ejemplo, dentro de la expresión para una cadena se pueden insertar las representaciones de los carateres especiales usando diagonales invertidas y de todos los caracteres usando representaciones octales o unicode.

Nombres de las variables

- ***Identificadores***
  - Los identificadores nombran variables, funciones, clases y objetos; cualquier cosa que el programador necesite identificar o usar.
  - En Java, un identificador
    - Comienza con una letra, un subrayado (_) o un símbolo de dólar ($).
    - Los siguientes caracteres pueden ser letras o dígitos.
    - Se distinguen las mayúsculas de las minúsculas
    - No hay longitud máxima.

Nombres de las variables

- ***Identificadores***
  - Serían identificadores válidos:
    - Identificador
    - nombre_usuario
    - Nombre_Usuario
    - _variable_del_sistema
    - $transaccion
  - Su uso sería, por ejemplo:
    - int contador_principal;
    - char _lista_de_ficheros;
    - float $cantidad_en_Ptas;

Nombres de las variables

- ***Palabras Clave***
  - Hay algunas palabras reservadas al lenguaje Java que no pueden utilizarse como nombres de variables, clases o métodos.
  - Las palabras reservadas en Java son:

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

* palabras reservadas que no se usan en java: const y goto.

Variables estáticas o constantes

- ***Constantes en Java***
  - Una constante hace referencia a un valor que no puede modificarse. Se define así:

    final <tipo> identificador = valor;
    - Por Ejemplo:

      final double PI = 3.141592
    - Los identificadores asociados a las constantes se suelen poner en mayúsculas y separados con guión bajo

- ***Recuerda:*** Las constantes se declaran en Mayúsculas mientras que las variables se hacen en minúsculas (por estilo)

Operadores en Java

Aritméticos

- Hay cinco **operadores aritméticos** que actúan sobre variables numéricas:

operador	significado
+	suma
-	resta
*	multiplicación
/	división
%	módulo

```
int  n1=2, n2;
n2=n1 * n1;      // n2=4
n2=n2-n1;        // n2=2
n2=n2+n1+15; // n2=19
n2=n2/n1;        // n2=9
n2=n2%n1;        // n2=1
```

Operadores en Java

- ***Aritméticos***
  - Los operadores aritméticos pueden combinarse con el de asignación para crear unos **operadores mixtos** que tienen estos significados:

operador	significado
+=	x += y significa x = x + y
-=	x -= y significa x = x - y
*=	x *= y significa x = x * y
/=	x /= y significa x = x / y
%=	x %= y significa x = x % y

```
int m=2, n=5;
m++; // m=3
n--; //  n=4
```

```
int num=5;
num += 5; // num = 10, equivale a
          // num = num + 5
```

Operadores en Java

- ***De Concatenación de Cadenas***
  - El operador **suma** actúa sobre cadenas (Strings) realizando una concatenación. Por ejemplo:

```
String s1="primera cadena ";
String s2="segunda cadena ";
System.out.println(s1+s2);
```

    - escribirá:

```
primera cadena segunda cadena
```

  - si se "suma" un número a una cadena, se obtiene una cadena. Por ejemplo:

```
int a=1, b=2;
String s1=" cadena "+a;
String s2=" cadena "+b;
System.out.println(s1+s2);
```

    - escribirá:

```
cadena 1 cadena 2
```

Operadores en Java

- ***Relacionales o de comparación***
  - Hay seis **operadores de comparación** que actúan sobre variables primitivas:

operador	significado
==	igual
!=	distinto
<	menor que
>	mayor que
<=	menor o igual a
>=	mayor o igual a

```
int  m=2, n=5;
boolean res;
res =m > n;//res=false
res =m < n;//res=true
res =m >= n;//res=false
res =m <= n;//res=true
res =m == n;//res=false
res =m != n;//res=true
```

  - El resultado de estos operadores es un valor booleano, por lo que, si se desea, pueden asignarse a variables booleanas. Por ejemplo:

```
boolean b=(x!=y);
```

Operadores en Java

- ***Lógicos***
  - Hay seis **operadores lógicos** que actúan sobre expresiones booleanas:

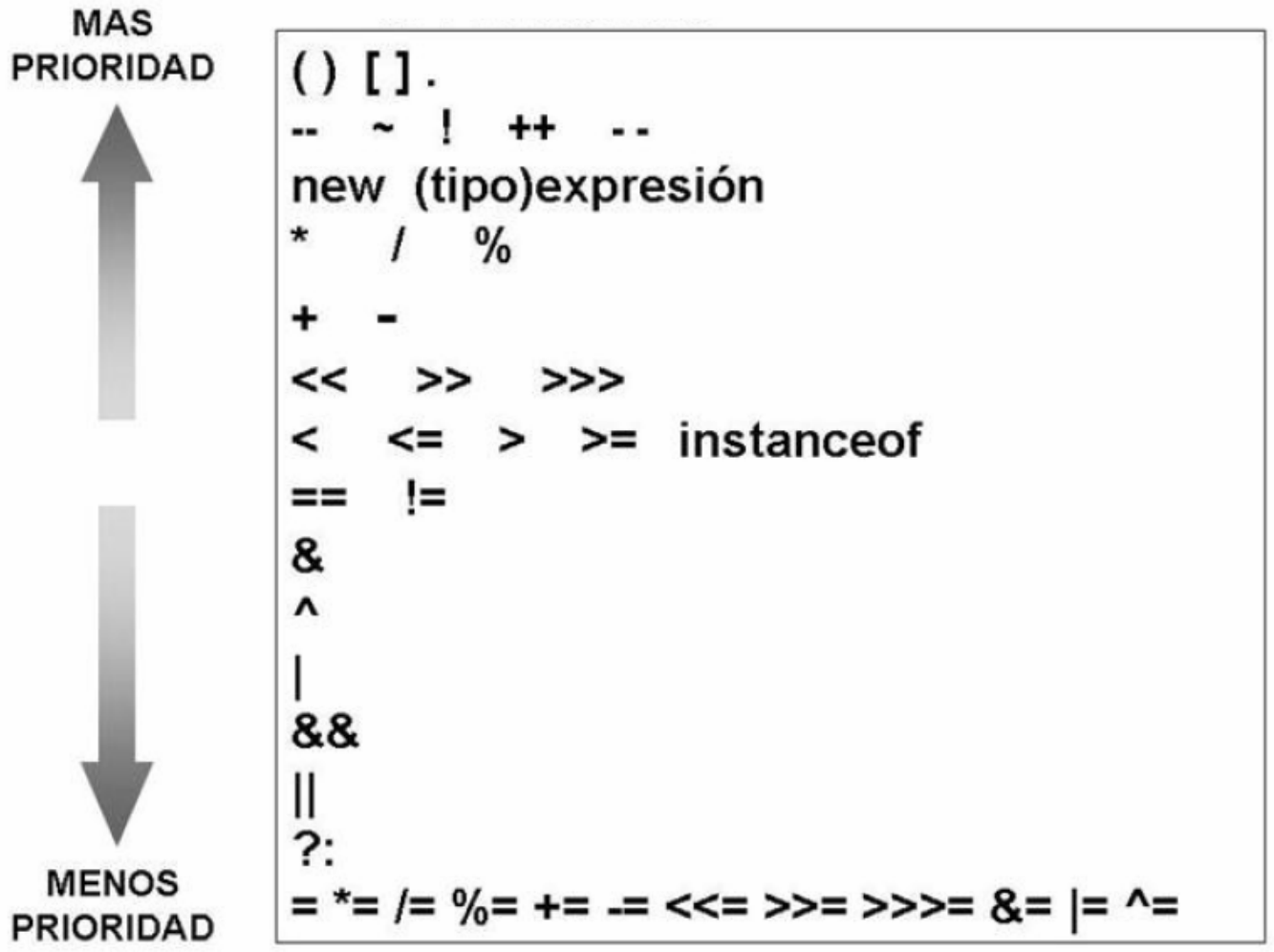
operador	significado	valor
&&	AND de corto circuito	true sólo si ambos operandos son true
&	AND evaluando ambos miembros	true sólo si ambos operandos son true
||	OR de corto circuito	true sólo si alguno de los dos operandos es true
|	OR evaluando ambos miembros	true sólo si alguno de los dos operandos es true
^	XOR (OR exclusivo))	true sólo si ambos operandos son diferentes
!	NOT (negación)	true sólo si el operando es false

```
int  m=2, n=5;
boolean res;
res =m > n && m >= n;//res=false
res =!(m < n || m != n);//res=false
```

Operadores en Java

- ***Lógicos***
  - Todos estos operadores son binarios (tiene dos operandos) excepto el de negación que es unario (tiene un sólo operando).
  - Un operador es **de corto circuito** si cuando el valor del primer operando determina el resultado, el segundo operando no se evalúa.
  - Los operadores de corto circuito hacen que la ejecución sea más rápida porque se salta pasos innecesarios

Precedencia de operadores en Java



Separadores

- Los separadores admitidos en Java son:
  - ***()*** *- paréntesis*.
    - Contiene listas de parámetros en la definición y llamada a métodos.
    - Define precedencia en expresiones
    - Contiene expresiones para control de flujo
    - Rodea las conversiones de tipo.
  - ***{}*** *- llaves*.
    - Contiene los valores de matrices inicializadas automáticamente.
    - Define un bloque de código, para clases, métodos y ámbitos locales.
  - ***[]*** *- corchetes*.
    - Declara tipos de arrays.
    - Referencia valores de arrays.
  - ***.*** *- punto*.
    - Separa nombres de paquete de subpaquetes y clases.
    - Separa una variable o método de una variable de referencia.

Separadores

- Los separadores admitidos en Java son:
  - **;** - *punto y coma*.
    - Separa sentencias.
  - **,** - *coma*.
    - Separa identificadores consecutivos en una declaración de variables.
    - Encadena sentencias dentro de una sentencia *for*.
  - ***Recuerda:***
    - Una ***expresión*** es una serie de variables/constantes/datos unidos por operadores (por ejemplo: 2*PI*radio).
    - Una ***sentencia*** es una expresión que acaba en **;** (Por ejemplo: area= 2*PI*radio;)

Comentarios

- En Java hay tres tipos de comentarios:
  - →// comentarios para una sola línea
  - →/* comentarios de una o más líneas */
  - →/** comentario de documentación, de una o más líneas */
  - Los dos primeros tipos de comentarios son los que todo programador conoce y se utilizan del mismo modo.
  - Los comentarios de documentación, colocados inmediatamente antes de una declaración (de variable o función), indican que ese comentario ha de ser colocado en la documentación que se genera automáticamente cuando se utiliza la herramienta de Java, *javadoc*.
    - Dichos comentarios sirven como descripción del elemento declarado permitiendo generar una documentación de nuestras clases escrita al mismo tiempo que se genera el código.
    - En este tipo de comentario para documentación, se permite la introducción de algunos tokens o palabras clave, que harán que la información que les sigue aparezca de forma diferente al resto en la documentación.

Conversiones de tipos (CAST)

- Conversiones implícitas
  - Se realizan de forma automática entre dos tipos de datos diferentes. Requiere que la variable destino (la colocada a la izquierda) tenga más precisión que la variable origen (situada a la derecha)

```
double dato1=23.12;
int dato2=5;
dato1=dato2; //dato1=5
dato2=dato1; //dato2=23
```

- Conversiones explicitas
  - En este caso es el programador el que fuerza la conversión mediante una operación ***cast*** con el formato.

```
int idato=5;
byte bdato;
bdato=(byte)idato;
System.out.println(bdato); //mostrará 5 por pantalla
```

- ***Recuerda:*** Como puede ser comprensible no se pueden realizar conversiones de tipos entre enteros y booleanos o reales y booleanos