VARIABLES, TIPOS DE DATOS CONVERSIONES

INDICE

- Variables
- Tipos de datos
- Conversiones

VARIABLES

Una variable es una zona en la memoria del computador con un valor que puede ser almacenado para ser usado más tarde en el programa.

La variables vienen dadas por:

- **Un nombre (identificador)**, que permite al programa acceder al valor que contiene en memoria. Debe ser un identificador válido, es decir, **el primer carácter de la secuencia debe ser una letra, un símbolo de subrayado (_) o el símbolo dólar ($)**.
- **Un tipo de dato**, que especifica qué clase de información guarda la variable en esa zona de memoria.
- **Un rango de valores** que puede admitir dicha variable.

TIPOS DE VARIABLES

En un programa, podemos encontrar distintos tipos de variables. Las diferencias entre una variable y otra dependerán de varios factores, por ejemplo, el tipo de datos que representan, si su valor cambia o no a lo largo de todo el programa, o cuál es el papel que llevan a cabo en el programa.

Según el párrafo anterior, en Java, se definen los siguientes tipos de variables:

a) **Variables de tipos primitivos y variables referencia**, según el tipo de información que contengan. En función de a qué grupo pertenezca la variable, tipos primitivos o tipos referenciados, podrá tomar unos valores u otros, y se podrán definir sobre ella unas operaciones u otras.

b) **Variables y constantes**, dependiendo de si su valor cambia o no durante la ejecución del programa. La definición de cada tipo sería:

- *Variables*: Sirven para almacenar los datos durante la ejecución del programa, pueden estar formadas por cualquier tipo de dato primitivo o referencia. Su valor puede cambiar varias veces a lo largo de todo el programa.
- *Constantes o variables finales*: Son aquellas variables cuyo valor no cambia a lo largo de todo el programa.

c) **Variables miembro y variables locales**, en función del lugar donde aparezcan en el programa. La definición concreta sería:

- **Variables miembro:** Son las variables que se crean dentro de una clase, fuera de los método de esta. Pueden ser de tipos primitivos o referencias, variables o constantes.
- **Variables locales:** Son las variables que se crean y usan ***dentro de un método o, en general, dentro de cualquier bloque de código***. La variable deja de existir cuando la ejecución del bloque de código o el método finaliza. Al igual que las variables miembro, las variables locales también pueden ser de tipos primitivos o referencias.

TIPOS DE DATOS

- En los lenguajes fuertemente tipados, a todo dato (constante, variable o expresión) le corresponde un tipo que es conocido antes de que se ejecute el programa.
- El tipo de dato, limita el valor de la variable o expresión, las operaciones que se pueden hacer sobre esos valores, y el significado de esas operaciones.

Los tipos de datos en Java se dividen principalmente en dos categorías:

- **Tipos de datos sencillos o primitivos**: Representan valores simples que vienen predefinidos en el lenguaje; contienen valores únicos, como por ejemplo un carácter o un número.
- **Tipos de datos referencia**: Se definen con un nombre o referencia (puntero) que contiene la dirección en memoria de un valor o grupo de valores. Dentro de este tipo tenemos por ejemplo los vectores o arrays, que son una serie de elementos del mismo tipo, o las clases, que son los modelos o plantillas a partir de los cuales se crean los objetos.

TIPOS DE DATOS PRIMITIVOS

Los tipos primitivos son aquéllos datos *sencillos* que constituyen los tipos de información más habituales: *números, caracteres y valores lógicos o booleanos*. Al contrario que en otros lenguajes de programación orientados a objetos, en Java, **no son objetos**.

TIPOS DE DATOS PRIMITIVOS				
Tipo	**Descripción**	**Bytes**	**Rango**	**Valor por default**
byte	Entero muy corto	1	-128 a 127	0
short	Entero corto	2	-32,768 a 32,767	0
int	Entero	4	-2,147,483,648 a 2,147,483,647	0
long	Entero largo	8	-9,223,372,036,854,775,808 a 9,223,372,036,854,775,807	0L
float	Numero con punto flotante de precisión individual con hasta 7 dígitos significativos	4	+/-1.4E-45 (+/-1.4 times 10^{-45}) a +/-3.4E38 (+/-3.4 times 10^{38})	0.0f
double	Numero con punto flotante de precisión doble con hasta 16 dígitos significativos	8	+/-4.9E-324 (+/-4.9 times 10^{-324}) a +/-1.7E308 (+/-1.7 times 10^{308})	0.0d
char	Carácter Unicode 2	`\u0000` a `\uFFFF`	'\u0000'	
boolean	Valor Verdadero o Falso	1	true o false	false

Restricciones de identificadores

Identificador	Convención	Ejemplo
nombre de variable	Comienza por letra minúscula, y si tienen más de una palabra se colocan juntas y el resto comenzando por mayúsculas	numAlumnos, suma
nombre de constante	En letras mayúsculas, separando las palabras con el guión bajo, por convenio el guión bajo no se utiliza en ningún otro sitio.	TAM_MAX, PI
nombre de una clase	Comienza por letra mayúscula	String, MiTipo
nombre de función	Comienza con letra minúscula	modifica_Valor, obtiene_Valor

VARIABLES

- Un **identificador** es una secuencia ilimitada de caracteres **Unicode**.
- **Unicode** es un código de caracteres o sistema de codificación, un alfabeto que recoge los caracteres de prácticamente todos los idiomas importantes del mundo. Las líneas de código en los programas se escriben usando ese código de caracteres, esto nos permite que aplicar nuestro idioma local para facilitar el trabajo al programador. El estándar Unicode originalmente utilizaba 16 bits, pudiendo representar hasta 65.536 caracteres distintos, dos elevado a la potencia dieciséis. Actualmente Unicode puede utilizar más o menos bits, dependiendo del formato que se utilice: UTF-8, UTF-16 o UTF32.
- A cada carácter le corresponde **unívocamente** un número entero perteneciente al intervalo de 0 a 2 elevado a n, siendo n el número de bits utilizados para representar los caracteres. Por ejemplo, la letra ñ es el entero164.
- Unicode es “compatible” con el código ASCII, por lo que la conversión entre ellos es inmediata.

CONVERSIONES DE TIPO

¿Qué pasará si dividimos un número entre otro, tendrá decimales el resultado de esa división?

- Siempre que el denominador no sea divisible entre el divisor, tendremos un resultado con decimales, pero no es así.
- Si el denominador y el divisor son variables de tipo entero, el resultado será entero y no tendrá decimales. Para que el resultado tenga decimales necesitaremos hacer una **conversión de tipo**.

Las conversiones de tipo se realizan para hacer que el resultado de una expresión sea del tipo que nosotros deseamos y/o esperamos. Existen dos tipos de conversiones:

- **Conversiones automáticas**: Cuando a una variable de un tipo se le asigna un valor de otro tipo numérico *con menos bits para su representación,* se realiza una conversión automática. En ese caso, el valor se dice que es promocionado al tipo más grande, para poder hacer la asignación. También se realizan conversiones automáticas en las operaciones aritméticas, cuando estamos utilizando valores de distinto tipo, el valor más pequeño se promociona al valor más grande, ya que *el tipo mayor siempre podrá representar cualquier valor del tipo menor* (por ejemplo, de int a long o de float a double).

CONVERSIONES DE TIPO

- **Conversiones explícitas**: Cuando hacemos una conversión de un tipo con más bits a un tipo con menos bits. En estos casos debemos indicar que queremos hacer la conversión de manera expresa, ya que *se puede producir una pérdida de datos y hemos de ser conscientes de ello*. Este tipo de conversiones se realiza con el **operador cast**. El operador cast es un operador unario que se forma colocando delante del valor a convertir el tipo de dato entre paréntesis. Tiene la misma precedencia que el resto de operadores unarios y se asocia de izquierda a derecha. Debemos tener en cuenta que *un valor numérico nunca puede ser asignado a una variable de un tipo menor en rango*, si no es con una conversión explícita.

```
int a ;
byte b;
a = 12 ;
b = 12;
```

Conversiones de una cadena

Java nos facilita varios métodos para convertir una cadena (String) en alguno de los tipos de datos primitivos:

- **De String a Float**: La función parseFloat() analiza una cadena y devuelve un número de punto flotante. Ejemlo:

  Float.parseFloat(String str)

- **De String a Int**: La función parseInt() analiza una cadena y devuelve un entero. Ejemplo:

  Integer.parseInteger(String str)

  Integer.valueOf(String str);

- **De String a Double**: La función parseDouble() analiza una cadena y devuelve un número decimal doble.

  Double.parseDouble(String str)

Conversiones de una cadena

- **De String a Boolean**: La función parseBoolean() analiza una cadena y devuelve un booleano. Ejemplo:

Boolean.parseBoolean(String str)

Boolean.valueOf(String str);

- **De String a char**: La función charAt() analiza una cadena y devuelve un carácter. Ejemplo:

cadena.charAt(posicion)

Todos estos tipos de datos poseen un proceso similar de conversión a String:

String.valueOf(valorTipoDato);