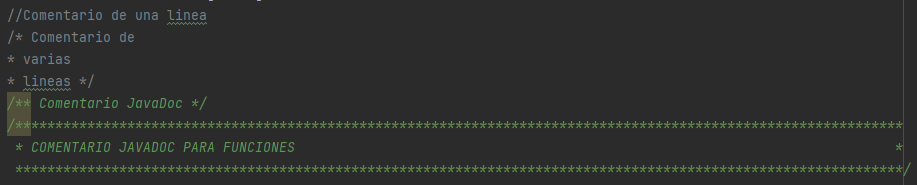
**SINTAXIS BÁSICA**

**COMENTARIOS**

En Java tenemos 3 tipos de comentarios: comentarios de una línea, comentarios multilínea y comentarios de JavaDoc o de documentación.

* Comentarios de una línea. Los comentarios comienzan con una doble barra inclinada //. Comentarios para realizar un comentario sobre una variable (por ejemplo, // Variable para guardar el número del usuario) o sobre las secciones de un método (// DECLARACIÓN DE VARIABLES).
* Comentarios multilínea. Los comentarios comienzan con una barra inclinada y un asterisco (/\*) y terminan con un asterisco y una barra inclinada (\*/). Comentarios para explicar de forma más extendida lo que realiza cierta parte de un método o algún elemento como un bucle o un condicional.
* Comentarios JavaDoc. Los comentarios comienzan con una barra inclinada y dos asteriscos (/\*\*) y terminan con un asterisco y una barra inclinada (\*/). Generalmente se utilizan en forma de caja. Comentarios dirigidos a explicar funciones, se encuadran al inicio de la función y dentro de ellos se explica lo que va a realizar la función, los parámetros que se le pasan, el autor de la función, la fecha de última actualización, etc.

**OPERADORES**

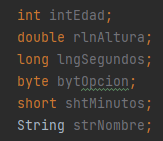
En Java utilizaremos los siguientes operadores:

|  |  |
| --- | --- |
| OPERADOR | SIGNIFICADO |
| = | Operador de asignación.  (int intNum = 1) |
| > / >= | Mayor que / Mayor o igual que.  (intNum > 18) |
| < / <= | Menor que / Menor o igual que.  (intNum < 18) |
| || | O lógico  (intNum || intNum2) |
| && | Y lógico  (intNum && intNum2) |
| != | Distinto de  (intNum != 18) |
| == | Igual que  (intNum == 18) |
| ! | La operación es false  (!blnPar) |
| + | Operador suma  (intNum1 + intNum2) |
| - | Operador resta  (intNum1 - intNum2) |
| \* | Operador multiplicación  (intNum1 \* intNum2) |
| / | Operador división  (intNum1 / intNum2) |
| % | Operador módulo (resto de la división entera)  (intNum1 % intNum2) |
| ++ | Suma 1 al operando  ++intNum1 (antes del uso de intNum1)  intNum1++ (después del uso de intNum1) |
| -- | Resta 1 al operando  --intNum1 (antes del uso de intNum1)  intNum1-- (después del uso de intNum1) |
| ?: | Operador ternario |

**DECLARACIÓN DE VARIABLES**

Podemos usar el nombre que queramos para declarar una variable, lo mejor es seguir el siguiente patrón:

tipoVariable abreviaturaVariableNombreVariable

Por ejemplo:

Principalmente usaremos tenemos los siguientes tipos:

* int 🡪 Números enteros. Abreviatura: int.
* double 🡪 Números reales. Abreviatura: rln.
* boolean 🡪 Variables booleanas (true/false). Abreviatura: bln.
* String 🡪 Cadenas de caracteres. Abreviatura: str.
* Char 🡪 Caracteres únicos. Abreviatura: chr.
* tipoVariable[] 🡪 Arrays. Abreviatura: tipoVariableArr (por ejemplo, intArr).

Para iniciar variables utilizamos la sintaxis:

tipo variable nombreVariable = valor;

En las variables String la inicialización es entre comillas dobles y en las variables char la inicialización es entre comillas simples.

**CREACIÓN Y UTILIZACIÓN DE UN OBJETO DE CLASE**

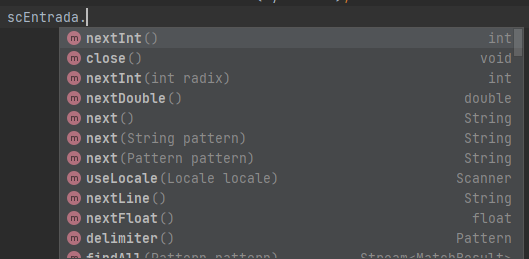
Para crear un objeto de clase se seguirá la siguiente sintaxis:

NombreClase nombreObjeto = new NombreClase (parametros);

Esta es la sintaxis que utilizaremos por ejemplo al crear un Scanner para la entrada de datos:

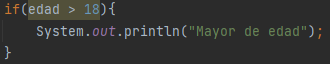
Para utilizar los métodos de un objeto de clase usaremos la siguiente sintaxis:

nombreObjeto.funcionDeClase(parametros);

Por ejemplo, en el caso del Scanner estos son algunos de los métodos de la clase:

**CONDICIONALES**

* Condicional simple: La sintaxis es la siguiente:

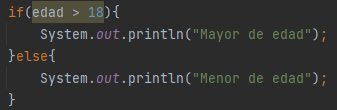


if(condicion){

instrucciones;

}

* Condicional doble: La sintaxis es la siguiente:



if(condicion){

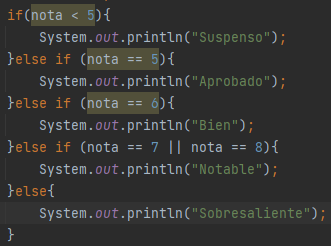
instrucciones;

}else {

Instrucciones;

}

* Condicional múltiple (else if): La sintaxis es la siguiente:



if(condicion){

instrucciones;

}else if(condicion){

instrucciones;

}else if(condicion){

Instrucciones;

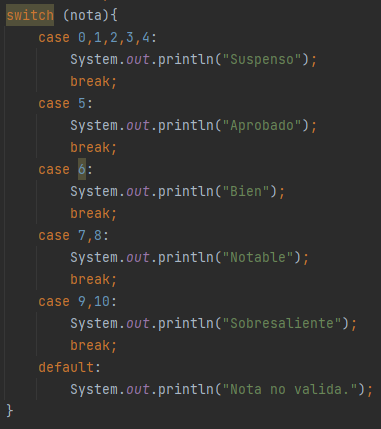
}else{

Instrucciones;

}

* Switch: Tenemos varias opciones de sintaxis:

Opción 1:

switch(valor){

case valor1:

instrucciones;

break;

case valor2:

instrucciones;

break;

case valor3:

instrucciones;

break;

...

case valorN:

instrucciones;

break;

default:

instrucciones;

}

Opción 2: SÓLO CUANDO HAYA UNA ÚNICA INSTRUCCIÓN POR CASE

switch(valor){

case valor1 -> instruccion;

case valor2 -> instruccion;

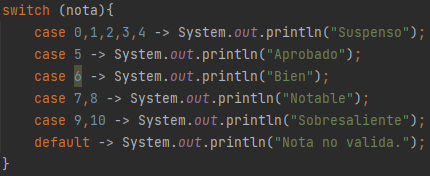
case valor3 -> instruccion;

...

case valorN -> instruccion;

default -> instruccion;

}



\*En ambas opciones el default es opcional. Además, si 2 o más valores comparten instrucciones pueden agruparse en un único case (ver foto).

* Operador ternario: Tiene la siguiente sintaxis:

condicion ? instruccionTrue : instruccionFalse;

