Documentación JAVA

* **Día 17-05-2021**

Java es orientado a objetos, y tiene diversas clases

Todo lo que vamos programando está dentro del método main(principal)

Cuando declaramos una variable generamos un espacio de memoria

Declaramos variables de la siguiente forma:

public static void main(String[] args) {

//tipo de dato - Nombre de la variable

byte numero1;

short numero2; Tipo enteros

int numero3;

long numero4;

float numero5; Tipo decimales

double numero6;

Tenemos ciertas restricciones al declarar nuestras variables:

* No pueden empezar con un signo distinto a $ o\_
* No pueden empezar con un número

Los nombres de las variables tienen que ser significativos y que represente lo que estamos programando.

No es recomendable exigirle tanto a la variable, por ejemplo, si es de tipo “byte” no colocar 127 o 128 que es la capacidad máxima de este tipo.

Operaciones

public static void main(String[] args) {

int numero1=4, numero2=5, suma, resta;

suma=numero1+numero2;

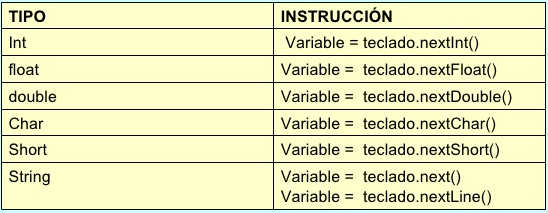
resta=numero1-numero2;

System.out.println("La suma es "+suma);

System.out.println("La resta es "+resta);

Clase Scanner

La clase de la API de Java Scanner, permite introducir a traves de teclado cualquier tipo de datos como String,int, float, etcétera. Tan solo debemos utilizar las funciones de las que dispone. A continuación podemos ver como podríamos leer números enteros desde la entrada estándar y esta clase ya se encargará de convertirlos a dicho tipo.



import java.util.Scanner

// 0 logico || && Y lógico

Java proporciona en el paquete java.util **una clase que se denomina Scanner** que nos permitirá leer datos desde ficheros de texto o incluso desde teclado. Los objetos de tipo Scanner mediante la invocación de distintos métodos **permitirán leer datos de cualquier tipo** (String, enteros, reales, etc) con diversas posibilidades de separadores. Asimismo permitirá leer un fichero de texto línea a línea, guardando cada línea en un objeto de tipo String.

Scanner se utiliza para indicar al programa que vas a ingresar un dato

<https://javadesde0.com/entrada-de-datos-con-scanner-de-java/>

**Tipos de datos entrada**

Existen diferentes tipos de datos y por ello, la clase scanner en función del tipo de dato introducido, utiliza distintos métodos para capturar un tipo de dato u otros en función del tipo de datos que se introduzca:

* **nextInt()** lee un valor int introducido por el usuario:
* **nextLong()** lee un valor long introducido por el usuario.
* **nextShort()** lee un valor short introducido por el usuario.
* **nextDouble()** lee un valor double introducido por el usuario.
* **nextByte()** lee un valor byte introducido por el usuario.
* **nextBoolean()** lee un valor boolean introducido por el usuario.
* **nextLine()**lee un valor String introducido por el usuario.

Hay varias maneras de generar números aleatorios en Java. Entre ellas, **métodos previstos por las clases Math y Random.** La terminología puede resultarnos inicialmente confusa, pues la clase Math dispone de un método denominado random(), mientras que Random es a su vez el nombre de otra clase.

Un objeto de esta clase se usa para generar una secuencia (stream) de números pseudoaleatorios a partir de un valor inicial o “semilla”. Si dos instancias de la clase Random son creadas con la misma “semilla”, y se ejecutan las mismas instrucciones para ambas instancias, se generarán y devolverán idénticas secuencias de números.

Scanner leer= new Scanner(System.in);

Random r =new Random();

int numero, ganador;

System.out.println("Ingrese su numero ");

numero =leer.nextInt();

ganador =r.nextInt(100);

if(numero==ganador){

System.out.println("Ganaste");

}

else{

System.out.println("Perdiste");

}

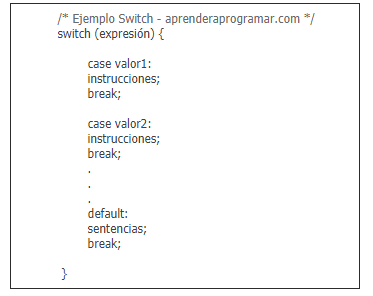
}

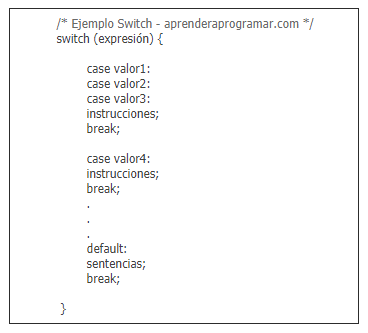
En esta clase se aprendió a utilizar las distintas clases que existen dentro de JAVA, y dentro de uno de los ejemplos que se dieron en clases fue ocupar la clase de Random para el caso de elegir un número aleatorio que es usualmente ocupado para sorteos o elegir un ganador de algún premio.

* **Día 18-05-2021**

La instrucción switch es una forma de expresión de un anidamiento múltiple de instrucciones if ... else. Su uso no puede considerarse, por tanto, estrictamente necesario, puesto que siempre podrá ser sustituida por el uso de if. No obstante, a veces nos resultará útil al introducir mayor claridad en el código.

La sintaxis será:





La cláusula default es opcional y representa las instrucciones que se ejecutarán en caso de que no se verifique ninguno de los casos evaluados. El último break dentro de un switch (en default si existe esta cláusula, o en el último caso evaluado si no existe default) también es opcional, pero lo incluiremos siempre para ser metódicos.

Busqué algunos ejercicios prácticos, los que encontré en la siguiente página

<https://javautodidacta.es/como-funciona-un-switch-en-java/>

Algunas reglas importantes para declaraciones switch:

* Los valores duplicados de los ***case*** no están permitidos.
* El valor para un ***case*** debe ser del mismo tipo de datos que la variable en el switch.
* El valor para un ***case*** debe ser una constante o un literal. Las variables no están permitidas.
* La declaración ***break*** se usa dentro del ***switch*** para finalizar una secuencia de instrucción.
* La declaración ***break*** es opcional. Si se omite, la ejecución continuará en el siguiente ***case***.
* La instrucción ***default*** es opcional, y debe aparecer al final del ***switch***.
* **Día 19-05-2021**

En esta clase vimos la función Switch en Java y revisé como ser utilizada con String.

# Sentencia switch en Java.

Como la sentencia if, esta sentencia nos permite modificar la línea de ejecucion. En vez de usar una expresión boolean, usa una expresión byte, short, char, int o un tipo enumerado. A partir de la versión de java 1.7, también se puede usar String.

## Formato de la sentencia Switch

switch (<expresión>) {

case <valor>:

<lista de sentencias separadas por punto y coma>;

case <valor>:

<lista de sentencias separadas por punto y coma>;

…

default:

<lista de sentencias separadas por punto y coma>;

}

El tipo de la <expresión> y el <valor> debe coincidir. La cláusula default es opcional. Puede haber tantas cláusulas case como queramos. El <valor> no puede ser una expresión, sólo puede ser un literal.

switch (<expresión>) {

case <valor>:

<lista de sentencias separadas por punto y coma>;

break;

case <valor>:

<lista de sentencias separadas por punto y coma>;

break;

…

default:

<lista de sentencias separadas por punto y coma>;

}

En este ejemplo, ingresamos un día de la semana en inglés y nos enumeran los días de la semana que nos quedan para terminar esa semana.

package com.edu4java.javatutorials;

import javax.swing.JOptionPane;

public class SwitchExample {

public static void main(String[] args) {

String daysLeft = "";

String weekDay = JOptionPane.showInputDialog("Ingresar el día de la semana (en ingles)");

switch (weekDay.toUpperCase()) {

case "SUNDAY":

daysLeft = daysLeft + "Sunday ";

case "MONDAY":

daysLeft = daysLeft + "Monday ";

case "TUESDAY":

daysLeft = daysLeft + "Tuesday ";

case "WEDNESDAY":

daysLeft = daysLeft + "Wednesday ";

case "THURSDAY":

daysLeft = daysLeft + "Thursday ";

case "FRIDAY":

daysLeft = daysLeft + "Friday ";

case "SATURDAY":

daysLeft = daysLeft + "Saturday ";

}

JOptionPane.showMessageDialog(null, "Quedan \"" + daysLeft + "\" para terminar la semana");

}

}

La sentencia **switch** se encarga de estructurar una selección múltiple. Al contrario del enunciado if-else que sólo podemos indicar dos alternativas, maneja un número finito de posibilidades. La estructura general del enunciado switch es la siguiente:

switch( expresión ) {

case constante1:

sentencia1;

...

break;

...

case constanteN:

sentenciaN;

...

break;

default:

sentencia;

...

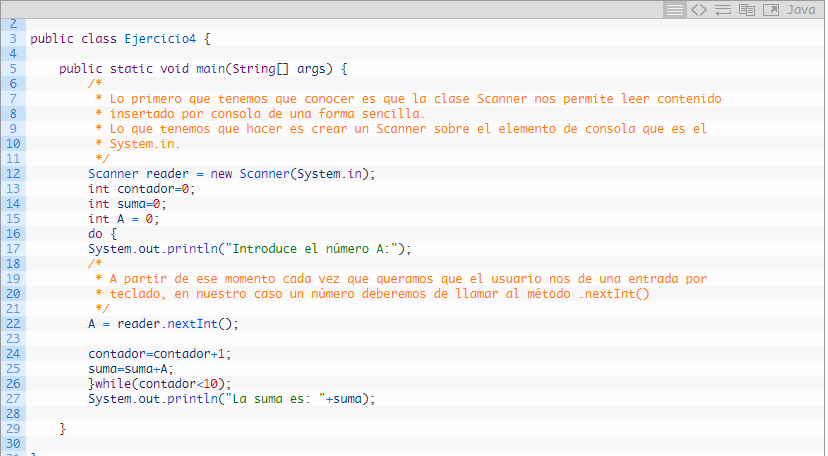
break

}

* **Día 20-05-2021**

**while**. La sentencia (hacer mientras) crea un bucle que ejecuta una sentencia especificada, hasta que la condición de comprobación **se evalúa como falsa**. La condición se evalúa después de ejecutar la sentencia, dando como resultado que la sentencia especificada se ejecute al menos una vez.

En **java** hay dos sentencias que nos permiten forzar la salida de un bucle, y estas sentencias son break (para **salir** completamente del bucle) y continue (para **salir** de la iteración actual y saltar directamente a la siguiente).





En la página <https://jesusfernandeztoledo.com/ejercicios-resueltos-iniciacion-a-java/> encontré el por qué se usan ciertas sentencias en el uso de for, while y dowhile, lo encontré muy claro y explicativo con algunos ejemplos.