Controladores de flujo y clases de Java

Emmanuel Cruz Hernández
emmanuel_cruzh@ciencias.unam.mx

22 de octubre de 2020

Contenido

Introducción

¿Por qué necesitamos controladores de flujo?

Dentro de un método debemos llevar a cabo un algoritmo que nos lleve de un estado inicial a otro final, donde existe una serie de tareas a ejecutar en cierto orden. [1]

Existen cuatro maneras de organizar a los enunciados o líneas de un algoritmo: secuencial, iteración, ejecución condicional y recursividad.

Secuencial

La ejecución prosigue, en orden, con cada línea, una después de la otra y siguiendo la organización física.

Las tareas se ejecutarán exactamente en el orden que están listadas.

Iteración

Marca a un cierto conjunto de enunciados secuenciales e indica la manera en que se van a repetir.

En *Java* hay tres formas de generar iteración. Usando las siguientes estructuras de control: **while**, **do... while** y **for**.

while

El ciclo **while** se utiliza para ejecutar repetidamente una secuencia de líneas de código, mientras la evaluación de una condición booleana se mantenga como verdadera (true).

La forma de hacer un **while** es la siguiente:

while (<Condición>){

    ...

}

do... while

El ciclo **do** permite que las líneas de código que encierra se ejecuten al menos una vez, antes de realizar la evaluación de la expresión booleana.
Si tal evaluación resulta con un valor verdadero (true), entonces el ciclo se vuelve a repetir.
En caso contrario, la ejecución continúa con la siguiente instrucción.

La forma de declarar una expresión **do** es la siguiente:

do {
    ...
} (<Condición>);

for I

El ciclo for permite la repetición controlada de una secuencia de líneas de código. La repetición se controla mediante una serie de expresiones:

- La expresión inicial debe dar como resultado un valor con el cual el ciclo for comienza su iteración.

for II

- La expresión booleana determina la condición de terminación del ciclo, es decir, la ejecución continúa en el ciclo si la evaluación de la expresión booleana es verdadera (true) y termina si es falsa (false).
- La expresión de actualización permite cambiar el valor dado por la expresión inicial, idealmente hasta que la expresión booleana se evalúa como falsa.

for III

La forma de declarar una expresión **for** es la siguiente:

for (<Inicio> ; <Condición> ; <Actualización>) {
    ...
}

Ejecución condicional

Se elige una de dos o más opciones de grupos de enunciados.

Hay dos formas de declarar estas estructuras: **if**, **switch**.

if

El enunciado **if** permite una selección en tiempo de ejecución del programa acerca de qué líneas de código se ejecutan o se omiten de la ejecución. En el caso de que la evaluación de la expresión booleana sea verdadera (true), el código que sigue se ejecuta, y en caso contrario (que se evalúe como false), se omite.

La forma de declarar una expresión **if** es la siguiente:

if (<Condición>) {
    ...
}

if... else...

Por otro lado, el enunciado **if** puede extenderse mediante añadirle una parte llamada **else**, que corresponde a una secuencia de código que se ejecuta en caso de que la evaluación de la expresión booleana sea false.

La forma de declarar una expresión **if** es la siguiente:

if (<Condición>) {
    ...
} **else** {
    ...
}

switch I

El enunciado **switch** permite escoger una entre varias opciones de secuencias de código a ser ejecutado.
Su funcionamiento se basa en considerar una expresión que se va secuencialmente comparando con cada valor constante propuesto (o **case**) como resultado de su evaluación.

switch II

Si el resultado es verdadero (true) se ejecuta el código correspondiente a esa opción. La evaluación se lleva a cabo en el orden en que los valores propuestos se escriben.
Si se desea que el código correspondiente a un sólo un valor propuesto sea ejecutado, es necesario añadir un enunciado de control de flujo **break** para que la estructura **switch** no evalúe los demás valores propuestos.

switch III

Al final, se añade una opción que recoge cualquier otra posibilidad de resultado de la evaluación de la expresión, bajo la etiqueta **default**.

switch IIII

La forma de declarar una expresión **switch** es la siguiente:

switch (<Expresión>) {
    case <Constante> *char/byte/short/int/String* **:**
        ...
    break;
    ...
    default:
        ...
}

Operador Condicional I

El operador condicional **no es una estructura de control**, como su nombre lo dice, **es un operador**. Sin embargo, tiene un funcionamiento condicional y es importante mencionarlo.

El operador condicional es un operador ternario (toma tres argumentos). Esencialmente, se comporta como una versión comprimida del enunciado **if**, y permite la elección en la evaluación de dos expresiones.

La estructura es como sigue:

<Condición> **?** <casoTrue> **:** <casoFalse> **;**

Operador Condicional II

Existen tres operandos: el primero, delante del caracter **?**, es una expresión booleana.
El segundo y tercero siguen después, y se separan entre sı por **:**.
Ambas representan expresiones del mismo tipo, o al menos de tipos compatibles.

Operador Condicional III

Primero, la expresión booleana es evaluada. Si tiene el valor *true*, la expresión inmediatamente después de **?** se evalúa y su resultado se retorna como el resultado del operador.
Si por el contrario, tiene el valor *false*, la expresión después de **:** se evalúa y retorna su resultado como el resultado del operador condicional.[2]

Operador Condicional IIII

Puedes consultar más en el siguiente enlace:

`https://codenowprogramming.000webhostapp.com/entradas/Java/9-operadorCondicional.php`

Recursividad

Es cuando un enunciado está escrito en términos de sí mismo.

Pero eso es algo de lo que hablaremos más adelante...

Clases proporcionadas por Java

Java ofrece clases ya definidas y programadas. Estas clases cuentan con atributos y métodos (si son necesarios), como una clase creada por nosotros.

Para poder usar una de las clases de *Java* se utiliza la palabra reservada **import** de la siguiente manera:

import <*Nombre de la clase de Java*> ;

NOTA: todas las clases importadas se deben especificar al comienzo del programa. Es la primera línea de código.

String

La clase **String** representa cadenas de caracteres. Todos los literales de cadena en los programas *Java*, como *abc*, se implementan como instancias de esta clase.[3]

Para usar esta clase **no es necesario importarla**.

Métodos de String

Algunos métodos interesantes de la clase **String** son *charAt(int)*, *contains(CharSequence)*, *equals(Object)*, *indexOf(int)*, *length()*, *replace(char,char)*, *substring(int)*, *substring(int,int)*, *toLowerCase()*, *toUpperCase()*, *trim()*.

Puedes consultar más métodos: `https://docs.oracle.com/javase/8/docs/api/java/lang/String.html`.

Scanner

Para poder usar esta clase en necesario agregar la siguiente línea:

import *java.util.Scanner*;

Usando un Scanner

Un escáner divide su entrada en tokens utilizando un patrón delimitador, que por defecto coincide con los espacios en blanco. Los tokens resultantes se pueden convertir en valores de diferentes tipos utilizando los siguientes métodos.[4]

Para definirlo se usa la siguiente línea:

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

Métodos del Scanner

Los métodos más importantes, por ahora, son los que nos permiten leer un texto como entrada desde una terminal.

Algunos de estos métodos son *nextInt()*, *nextDouble*, *nextLine()* y *close()*.

Puedes consultar más métodos: `https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html`.

Bibliografía


Elisa Viso G. Canek Peláez V.
Introducción a las Ciencias de la Computación con Java.
Segunda edition, 2012.


Jorge L. Ortega Arjona.
Notas de Introducción al Lenguaje de Programación Java.
2004.


Oracle.
Class string.
urlhttps://docs.oracle.com/javase/8/docs/api/java/lang/String.html,
2020.


Oracle.
Class scanner.
urlhttps://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html,
2020.