por Nacho Iborra

Entornos de Desarrollo
Bloque 2
Tema 2: Estructuras de control. Condiciones y bucles

En esta sección vamos a ver las diferenctes estructuras de selección e iteración que se pueden usar en Java. En todas ellas se deben usar las llaves `{...}` para agrupar aquello que estará contenido en la instrucción correspondiente, de la misma forma que lo hacemos por ejemplo en C#.

2.2.1. Estructuras selectivas

En Java se pueden usar estructuras `if`, `if..else` o `switch` para seleccionar el camino de ejecución dependiendo de una condición.

2.2.1.1. if .. else if .. else

Se puede utilizar la estructura `if` básica, y `if.. else if ..else if ..else` como en otros muchos lenguajes. La primera nos permite ejecutar un trozo de código solo si se cumple una determinada condición.

```
if (age >= 18)
{
        System.out.println("You are old enough");
}
```

La segunda estructrua nos permite elegir entre diferentes caminos dependiendo de la condición de cada una de ellas. Sólo un camino sería posible.

```
if (number > 0)
{
        System.out.println("It is positive");
}
else if (number < -10)
{
        System.out.println("It is under -10");
}
else
{
        System.out.println("It is between -10 and 0");
}
```

2.2.1.2. switch

Además, existe la sentencia `switch` con sus correspodientes `case` y `default`. Los datos que puede manejar esta instrucción deben tipos primitivos, las cadenas (tipo String) no estarían permitidos en versiones antiguas de Java (Java 6 y anteriores). La clausula `break` al final de cada caso no es obligatoria, si no se pone se pasa al siguiente caso.

```
switch(number)
{
        case 0: System.out.println("It is 0"); break;
        case 1: System.out.println("It is 1");
        case 2: System.out.println("It is 2"); break;
        default: System.out.println("Unknown number");
}
```

En el ejemplo anterior, si *number* es 1, aparecerán los mensajes "It is 1" y "It is 2", ya que no hay claúsula `break` en el *case 1*.

Ejercicios propuestos:

2.2.1.1. Crea un programa llamado *MarkCheck* que pida al usuario que introduzca 3 notas. El programa mostrará por pantalla uno de estos mensajes, dependiendo de las notas:

- *Todas las notas son mayores o iguales a 4*
- *Algunas notas no son mayores o iguales a 4*
- *Ninguna nota es mayor o igual a 4*

2.2.1.2. Crea un programa llamado *GramOunceConverter* que mejorará el ejercicio 2.1.6.2 del tema anterior. En este caso, el usuario introducirá un peso (float), y una unidad de medida (`g` para gramos, `o` para libras *ounces*). Entonces, dependiendo de la unidad elegida, el programa convertirá el peso en la otra unidad de medida. Por ejemplo, si el usuario introduce un peso de 33 y

> elige `o`, entonces el programa covertirá 33 libras a gramos. Se debe resolver usando la estructura `switch`. Si la unidad introducida es diferente a `g` o `o`, entonces el programa deberá mostrar un mensaje de error: "Unidad no reconocida" y ningún resultado.

2.2.2. Bucles

En Java existen la mayoría de las estructuras iterativas que tienen casi todos los lenguajes de programación. `while`, `do..while` y `for`. Tienen una estructura similar a la de C o C#.

2.2.2.1. while

Usaremos este tipo de bucle cuando no sabemos cuantas iteraciones se van a realizar y ni siquiera si se realizara alguna. Este ejemplo cuenta de 1 a 10:

```
int n = 1;
while (n <= 10)
{
        System.out.println(n);
        n++;
}
```

2.2.2.2. do..while

Usaremos este bucle cuando no sabemos cuantas iteraciones se van a realizar pero sabemos que al menos se realizará una vez. Es muy usual para pedir al usuario algún dato y comprobarlo para volver a pedírselo en caso de no ser correcto. El ejemplo anterior usando `do..while` sería:

```
int n = 1;
do
{
        System.out.println(n);
        n++;
} while (n <= 10);
```

2.2.2.3. for

Usaremos este bucle cuando sabemos cuantas iteraciones se van a realizar. El ejemplo de contar de 1 a 10 es preferible hacerlo con este tipo de estructura, y sería así:

```
for (int n = 1; n <= 10; n++)
{
        System.out.println(n);
}
```

Nótese que se pueden declarar variables en un `for` (y en medio de otro código, como en otros lenguajes como C#).

2.2.2.4 Otro "for"

Hay otra forma de usar una instrucción `for`, aplicada a una colección o arrays. Consiste en usar una variable con el mismo tipo de dato, de esta forma:

```
for (int number: numbers)
        System.out.println("" + number);
```

donde `numbers` debe ser una colección o array de enteros. Esta estructura es equivalente a la estructura `foreach` de otros lenguajes como C#, y se usa solo para consultar valores y no modificarlos.

> **Ejercicios propuestos**:
>
> **2.2.2.1.** Crea un programa llamado *GroupPeople* que pida al usuario que introduzca cuanta gente va a acudir a una conferencia. El programa debe crear grupos de preferiblemente 50 personas. Cuando esto no sea posible, se probará a crear grupos de 10 y finalmente grupos de 1 persona. El programa deberá mostrar cuantos grupos de cada tipo serían necesarios. Por ejemplo, si van a ir 78 personas a la conferencia, entonces necesitaremos 1 grupo de 50 personas, 2 grupos de 10 personas y 8 grupos de 1 persona.
>
> **2.2.2.2.** Crea un progrma llamado *SumDigits* que pida al usuario que introduzca números enteros hasta que se introduzca un 0. El programa deberá sumar todos los números introducidos y mostrar el resultado final, así como indicar el número de dígitos que tiene. Por ejemplo, si el usuario introduce 12,20,60,33,99 y 0, entonces el programa deberá mostrar: "El resultado es 224 y tiene 3 dígitos".

2.2.3. Ejercicios adicionales: Retos de Java

2.2.3.1. Introducción a los retos de Java

A medida que vamos aprendiendo nuevos conceptos de Java, podemos aplicarlos para resolver algunos retos interensantes de Acepta el Reto, para ello debes registrarte en este sitio web. Aquí puedes encontrar muchos retos de programación, agrupados por categoría. En este tema y en adelante se os anima a resolver algunos.

2.2.3.2. Reto de ejemplo: Hello world

Veamos por ejemplo este reto de *Acepta el reto*. En él se pide leer un número N y sacar por pantalla la cadena "Hola mundo" N veces. Para resolver este reto, se podría implementar algo parecido a esto en Java:

```
import java.util.Scanner;

public class Challenge116_Hola_mundo
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        int times = sc.nextInt();

        for (int i = 0; i < times; i++)
            System.out.println("Hola mundo.");
    }
}
```

Prueba a subir este código a *Acepta el reto* y observa como es aceptado.

> **Ejercicios propuestos**:
>
> **2.2.3.1.** Intentad resolver estos retos de *Acepta el reto*:
>
> - Aburrimiento en las sobremesas
> - Último dígito del factorial