5. Recursividad

Una **función recursiva** es aquella que se llama a sí misma. Para evitar entrar en una especie de bucle infinito que acabaría agotando la memoria del dispositivo, hay que buscar la forma de que esa llamada deje de producirse en algún momento y empiecen a devolverse los resultados a las instancias anteriores de la función.

La estructura más simple posible de una función recursiva en Java es la siguiente:

```
public static void funcion_recursiva() {
    funcion_recursiva();
}
```

Lógicamente, la función no puede quedarse de esa manera, porque se llamaría a sí misma sin hacer nada más hasta que se agotaran los recursos del dispositivo.

IMPORTANTE

Para que la recursividad funcione correctamente, debemos dejar abierta la posibilidad de, en función de una o varias condiciones, la función deje de llamarse a sí misma, resuelva un problema concreto y devuelva un resultado a la instancia anterior de la función desde la que ha sido llamada.

En realidad, la función recursiva lo que hace es descomponer un problema en problemas más sencillos hasta que llega un momento, llamado caso base, en que es capaz de resolver el problema sin descomponerlo más. Esa solución suele generar un valor que es devuelto a la función anterior, capaz ahora de resolver —gracias a dicho valor— su propia copia del problema, y así sucesivamente. De este modo, se va deshaciendo el camino realizado hasta llegar a la primera instancia de la función, que, con el dato que le ha sido devuelto, será capaz de resolver el problema inicial. Así pues, una función recursiva tendría una estructura similar a esta:

```
public static void funcion_recursiva(valorInicial) {
      if(solucionPosible) {
            return solucion;  // este sería el caso base
      }
       else {
            return funcion_recursiva(nuevoValor);
      }
}
```

El funcionamiento típico de la recursividad puede verse en el siguiente esquema:

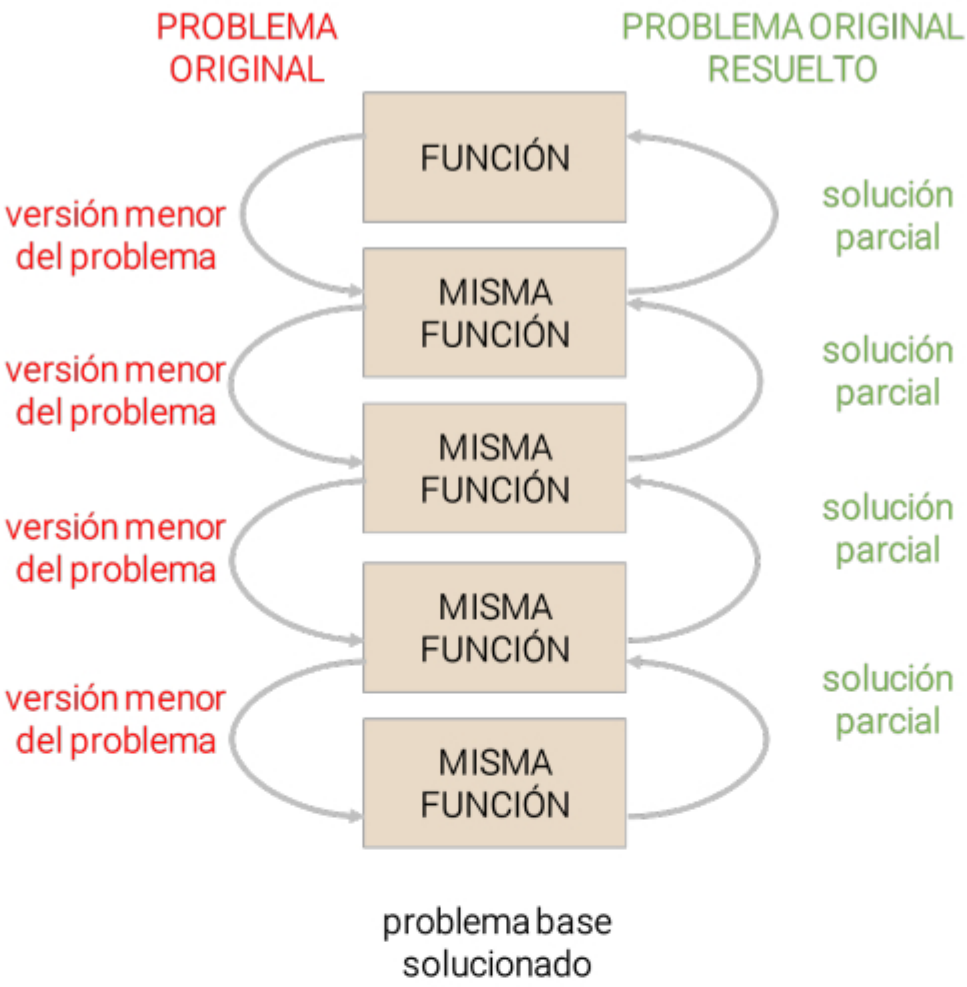

Figura 5.1. Recursividad

Como vemos, la función recursiva se llama a sí misma con una versión más reducida del problema hasta que llega un momento en que sabe cómo solucionarlo. Entonces se va devolviendo a sí misma la solución a cada versión del problema hasta que, finalmente, se consigue solucionar el problema original. Esta es la parte teórica, que puede parecer algo confusa. Vamos a aclararlo con un ejemplo práctico, que además es el ejemplo típico que suele utilizarse, porque ejemplifica muy bien los beneficios de la recursividad.

Ejemplo: cómo calcular el factorial de un número

El factorial de un número entero positivo se calcula multiplicando dicho número por todos los números menores que él hasta llegar a 1. Su fórmula teórica sería:

```
n! = n * (n-1) * (n-2) * … * 2 *1
```

Así, por ejemplo, el factorial de 5 sería 120:

```
5! = 5 * 4 * 3 * 2 * 1 = 120
```

¿Cómo nos puede ayudar en este caso la recursividad a solucionar el problema? Si nos fijamos, para hallar el factorial de 5 estamos multiplicando 5 por el factorial de 4 (4 * 3 * 2 * 1). Para hallar el factorial de 4, multiplicamos 4 por el factorial de 3 (3 * 2 * 1), y así sucesivamente. Es decir, que la fórmula puede redefinirse de la siguiente manera:

```
n! = n * (n-1)!
```

Lo cual significa que el factorial de cualquier número entero positivo puede calcularse multiplicándolo por el factorial de su número inferior. Al llegar a 0, ya no quedan números enteros positivos inferiores, con lo cual su factorial se debe poder calcular de forma directa, y así es: el factorial de 0 es 1. Ese, el factorial de 0, sería nuestro caso base que podemos resolver directamente, y cuya solución se traslada a las llamadas recursivas anteriores para que solucionen su propio caso, siguiendo así hasta llegar al caso principal.

Figura 5.2. Aplicación de la recursividad para resolver un factorial

La solución en Java se implementaría como sigue:

```
public static int factorial(int valorInicial) {
        if(valorInicial == 0) {
            return 1;
        }
        else {
            return valorInicial * factorial(valorInicial - 1);
        }
}
```

Lo podemos comprobar creando un programa en Java que la use de la siguiente manera:

```
public static void main(String[] args) {
  int numero, resultadoFactorial;
  Scanner teclado = new Scanner(System.in);
  System.out.println("Introduzca un número entero positivo: ");
  numero = teclado.nextInt();
  resultadoFactorial = factorial(numero);
  System.out.println("El factorial de " + numero + " es " +
resultadoFactorial);
}
```

En el material complementario examinaremos otros tipos de funciones recursivas y veremos más aplicaciones prácticas de la recursividad.