El paso por valor de los argumentos en Java

En Java todos los argumentos se pasan por valor. El paso por valor de los argumentos de un método en Java tiene varias consecuencias. El paso por valor significa que al método en la variable del argumento le llega una copia del valor en el caso de un tipo primitivo de datos o una copia del puntero a la dirección de memoria del objeto. En el paso por referencia el argumento contiene un puntero con la dirección de memoria de la variable. En el paso por valor al asignar un valor a la variable del argumento no modifica el valor de la variable usada para invocar al método, esto ocurre tanto para argumentos de tipo primitivo y para objetos.

La palabra reservada *final* en los argumentos sirve para impedir asignar un nuevo valor a una variable, una variable *final* es una variable cuyo valor es una constante ya que en caso de intentar asignar a la variable un nuevo valor el compilador produce un error de compilación. La variable no pude cambiar de valor, sin embargo, si la variable es una referencia a un objeto el objeto si puede cambiar de estado, para que un objeto no pueda cambiar ha de ser inmutable.

Algunas clases como la clase String en Java son inmutables, esto significa que al manipular el objeto se devuelve una nueva instancia de la clase en vez de modificar la original. En Java para manipular un *String* y obtener la misma referencia en vez de una nueva instancia hay que utilizar la clase StringBufffer o StringBuilder. Algunas instancias de listas obtenidas con la API de colecciones son inmutables como List.of, y sus métodos add y remove lanzan la excepción en caso de ser invocados.