DIFERENCIA DE PARÁMETROS POR REFERENCIA Y POR VALOR

Java almacena las variables en memoria. Básicamente de dos tipos de variables: primitivas y objetos.

TIPOS PRIMITIVOS	REFERENCIA DE OBJETOS
int	Integer
double	Double
float	Float
char	String
Boolean	Matrices
	Cualquier instancia de una clase propia o de la biblioteca de Java.

Las variables primitivas siempre son almacenadas en la memoria stack, sin embargo, en el caso de los objetos estos son almacenados en dos etapas, los objetos actuales son almacenados en la memoria heap y la referencia a objetos se encuentra almacenada en la memoria stack que apunta a los objetos.

Variables por valor:

Significa que cuando un argumento se pasa a una función, la función recibe una copia del valor original. Por lo tanto, si la función modifica el parámetro, sólo la copia cambia y el valor original permanece intacto.

Entendemos como asignación de variables por valor cuando se hace **una copia del valor** de una variable a otra. Esto lo vemos algo así:

```
public static void main(String[] args) {
        int a = 10; //es una variable
        int b = 20; //es otra variable
        int c = a;  //la variable "c" tiene la copia del valor
de "a"
        a = 30; //"a" tiene otro valor
        System.out.format("a=%d, b=%d, c=%d", a, b, c);
//imprime a=30, b=20, c=10
    }
```

Variables por referencia:

Significa que cuando un argumento se pasa a una función, la función recibe la dirección de memoria del valor original, no la copia del valor. Por lo tanto, si la función modifica el parámetro, el valor original en el código que llamó a la función cambia.

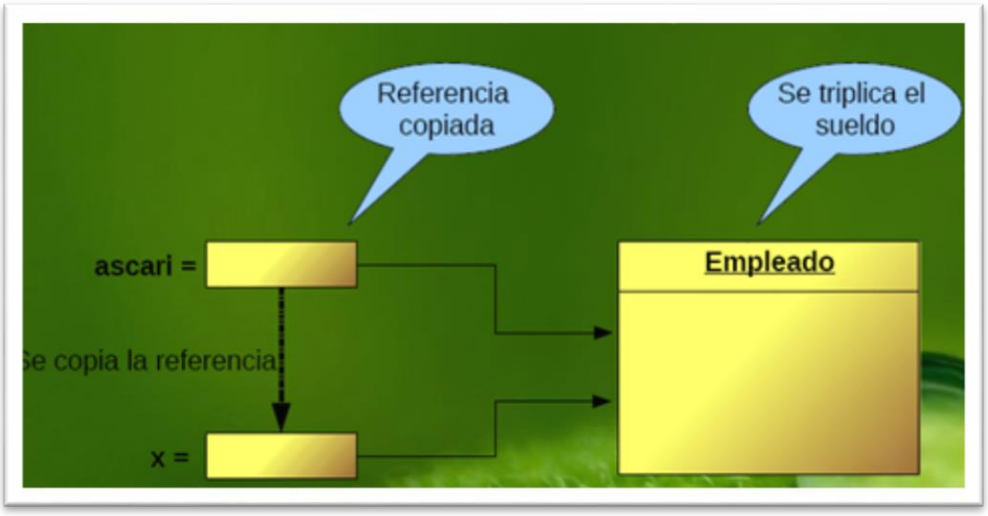
¿Como son pasados los argumentos en Java?

En Java, los argumentos son siempre pasados por valor independientemente del tipo de variable. Cada que el método es invocado, pasa lo siguiente:

- Una copia del argumento es pasado en la memoria stack y la copia es pasada en el método
  - Si la variable original es primitiva, es simple, una copia de la variable es creada en el stack y esta se pasa al método
  - Si la variable original no es primitiva, entonces una nueva referencia o apuntador es creado dentro de la memoria stack que apunta al objeto actual y la nueva referencia es pasada al método, (en esta etapa, 2 referencias apuntan al mismo objeto).

Parámetros por referencia en Java:

Los objetos son pasados por valor, le guste a quien le guste así es y será por los siglos de los siglos. Lo que realmente sucede es que se pasa una **copia de la referencia.**



Como se ve en la imagen, cuando creamos al objeto “ascari”, este es solo una referencia a un objeto Empleado. Cuando invocamos al método ***triplicarSueldo*** y le pasamos como parámetro al objeto ascari, no le estamos pasando la dirección de memoria de “ascari”. Lo que sucede es que se hace una copia de la referencia “ascari” y esta copia se pone en “x”.

Lo anterior es la piedra de tropiezo para muchos, y es lo que les causa confusión. El hecho de que Java maneje el concepto de “referencia de objeto” es lo que hace pensar que Java hace un paso de parámetros por referencia, es decir, que se le pasa la dirección de memoria. Sin embargo, una referencia de objeto es un apuntador a un Objeto ubicado en algún otro lugar de memoria.

cuando “ascari” es pasado al método triplicarSueldo, no quiere decir que a “x” le sea asignada la dirección de memoria de “ascari”. ¡Como lo ilustra la figura de arriba, se hace una copia de la referencia! Por lo tanto, existen dos referencias que apuntan a un objeto Empleado.

Técnicamente está mal dicho si decimos “modificar x” o “modificar ascari” ya que en ningún momento estamos modificando a “x” o “ascari” ya que estas son solo referencias a un objeto Empleado, lo que realmente se modifica es el objeto Empleado.