Diferencias entre parámetros por valor y por referencia.

1) **Por valor:** significa que la función (o subrutina) recibe sólo una copia del valor que tiene la variable, o sea que no la puede modificar.
2) **Por referencia:** significa que se pasa la posición de memoria donde esta guardada la variable, por lo que la función puede saber cuánto vale, pero además puede modificarla de cualquier manera.

En la mayoría de lenguajes existen estas dos formas de pasar variables a una función.

Sin embargo toda esta diferencia queda escondida casi totalmente por el compilador, por lo que uno puede pasarse media vida programando sin darse cuenta de esto.

Parámetros por referencia en Java

En los lenguajes de programación suelen existir dos formas de pasar los parámetros a los métodos. Parámetros por valor (dónde se realiza una copia de las variables) o parámetros por referencia (dónde se pasa una referencia a la variable original).

En Java solo ha paso de parámetros por copia. Pero entonces, por qué la gente habla del paso de parámetros por referencia en Java. Veamos en detalle por qué parece que realizamos un paso de parámetros por referencia en Java.

Paso de parámetros por valor

Lo primero que tenemos que ver es que para los datos primitivos en Java se realiza claramente una copia.

```
1. public void metodo(int p) {
2.   p=3;
3. }
4.
5. int p1=2;
6. metodo(p1);
```

```
7.
8.  System.out.println(p1); //p1 = 2
```

Paso de parámetros "por referencia": referencia de objetos

Pero ahora pasemos a manejar un objeto como parámetro. Lo que sucede al manejar los objetos en Java es que las variables mantienen una referencia al objeto, por lo tanto cuando pasamos un objeto como parámetro se está realizando una copia de la referencia. Así tenemos dos variables diferentes apuntando al mismo objeto.

Creamos una clase básica llamada MiClase:

```
1.  public class MiClase {
2.   public int valor;
3.  }
```

Y ahora un método que modifica ese valor:

```
1.  public static void metodo_referencia(MiClase m) {
2.    m.valor =3;
3.  }
```

Veamos cómo se pasa por valor, aunque parece que hay una referencia:

```
1.  MiClase m1 = new MiClase();
2.  m1.valor = 2;
3.  System.out.println(m1.valor); // Devuelve 2
4.  metodo_referencia(m1);
5.  System.out.println(m1.valor); // Devuelve 3
```

Hemos instanciado con un valor de 2 el atributo de la clase y el método lo cambia a 3. Como la variable copia m en el método sigue manteniendo la referencia al objeto original, se produce un cambio en dicho objeto. Por lo tanto es cuando tenemos la sensación del paso por referencia de los objetos.

Es decir, que el paso por parámetros es por valor, aunque lo que se copia es una referencia. Es por ello que tenemos la "falsa sensación" de que estamos pasando los parámetros por referencia, aunque realmente es por valor.

Desmitificando el paso de parámetros "por referencia"

Pero para verlo mejor veamos otro caso. Ahora lo que vamos a hacer es crear un nuevo objeto y cambiar la referencia de la variable pasada por copia.

```
1. public static void metodo_referencia2(MiClase m1) {
2.   MiClase m2 = new MiClase();
3.   m1 = m2;
4.   m1.valor = 3;
5. }
```

Lo que sucede es que ahora m1, que mantenía una referencia al objeto inicial, pasa a tener una referencia a un nuevo objeto. Por lo tanto los cambios que hagamos en m1 no afectan ya al objeto inicial.

Si volvemos a realizar la misma secuencia de salida vemos que no hay cambios en el objeto inicial.

```
1. MiClase m2 = pr.new MiClase();
2. m2.valor = 2;
3. System.out.println(m2.valor); // Devuelve 2
4. metodo_referencia2(m2);
5. System.out.println(m2.valor); // Devuelve 2
```

Por lo tanto siempre debemos de tener en cuenta que los parámetros en Java se pasan por valor. Pero que existen las referencias a los objetos y por lo tanto podemos tener la falsa sensación de que hay paso de parámetros por referencia en Java.