Java value vs reference es una de las cuestiones más habituales cuando comenzamos a programar en Java . ¿Como se pasan los valores en Java por valor o por referencia?.  Es un tema  interesante y vamos a ver un par de bloques de código que nos ayuden a clarificar.

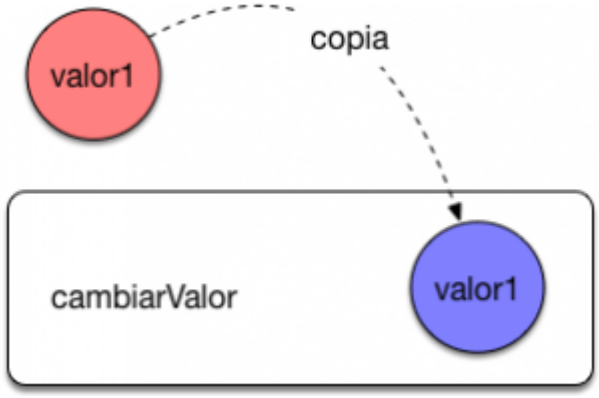


```
package com.arquitecturajava;

public class Principal {

 public static void main(String[] args) {
 // TODO Auto-generated method stub

 int valor1 = 5;

 cambiarValor(valor1);
 System.out.println(valor1);
 }

 public static void cambiarValor(int valor) {
 valor = 7;

 }
```

```
}
```

Hemos declarado una variable de un tipo básico “int” y se la hemos pasado al método cambiarValor el cual la modifica. Sin embargo cuando ejecutamos el programa el resultado por pantalla será un 5:


Esto implica que todos los tipos básicos en Java son pasados por valor y por lo tanto se realiza una copia de la variable . Si modificamos la variable estaremos modificando la copia y no el valor original.




# Java value vs reference y objetos

Vamos a pasar un objeto en vez de un tipo básico y ver que el resultado es totalmente diferente. Para ello vamos a usar la clase Persona que contiene como propiedad un nombre:

```
package com.arquitecturajava;

public class Persona {
```

```
 private String nombre;

 public String getNombre() {
 return nombre;
 }

 public void setNombre(String nombre) {
 this.nombre = nombre;
 }

 public Persona(String nombre) {
 super();
 this.nombre = nombre;
 }

}
```

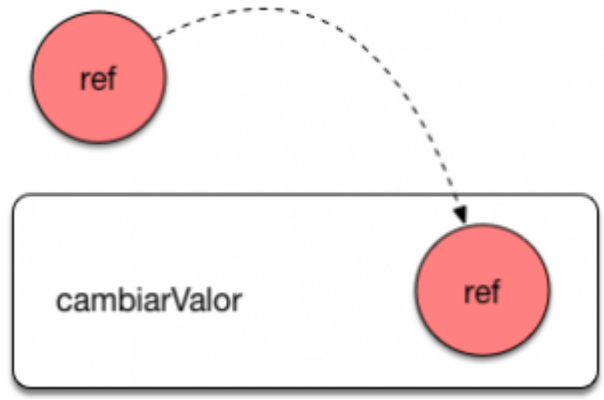
Usamos esta clase en un programa main:

```
package com.arquitecturajava;

public class Principal2 {

 public static void main(String[] args) {

 Persona persona= new Persona("pedro");
 cambiarValor(persona);
 System.out.println(persona.getNombre());
 }
```

```
 public static void cambiarValor(Persona p) {
 p.setNombre("miguel");

 }

}
```

En este caso cuando nosotros cambiamos el nombre de “pedro” a “miguel”  y luego lo imprimimos por pantalla:


El nombre se  cambiará. Por lo tanto parece que cuando nos estamos refiriendo a un objeto el paso es por referencia y no por valor .

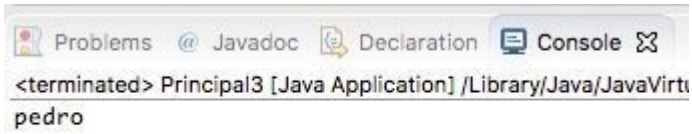



Lamentablemente las cosas no son tan sencillas como parecen . Si realizamos la siguiente modificación al código y generamos un nuevo objeto:

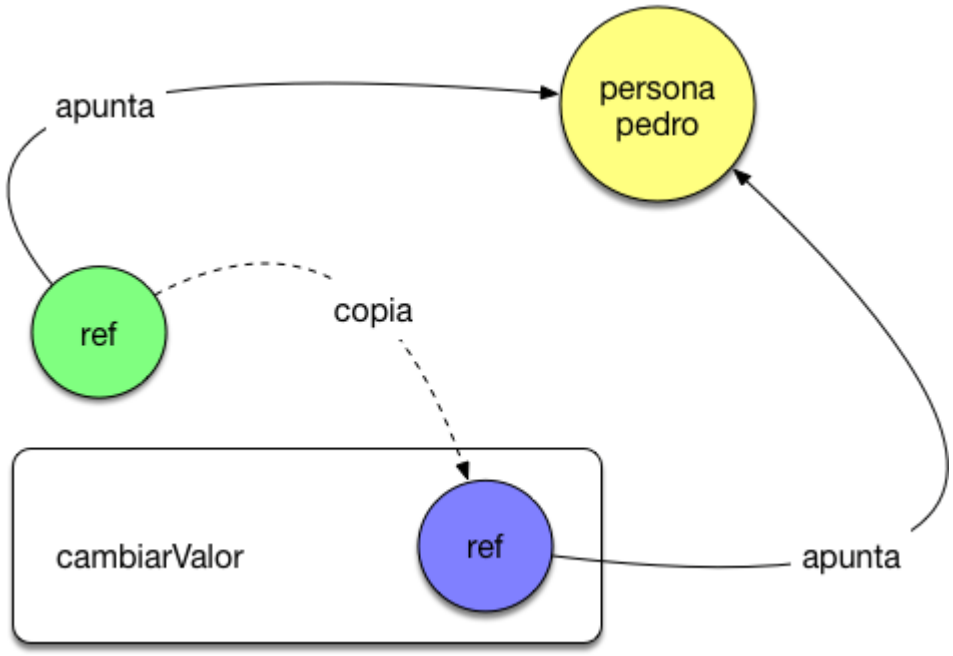
```
package com.arquitecturajava;
```

```
public class Principal3 {

public static void main(String[] args) {

Persona persona= new Persona(“pedro”);

cambiarValor(persona);
System.out.println(persona.getNombre());
}

public static void cambiarValor(Persona p) {
p= new Persona (“miguel”);

}

}
```

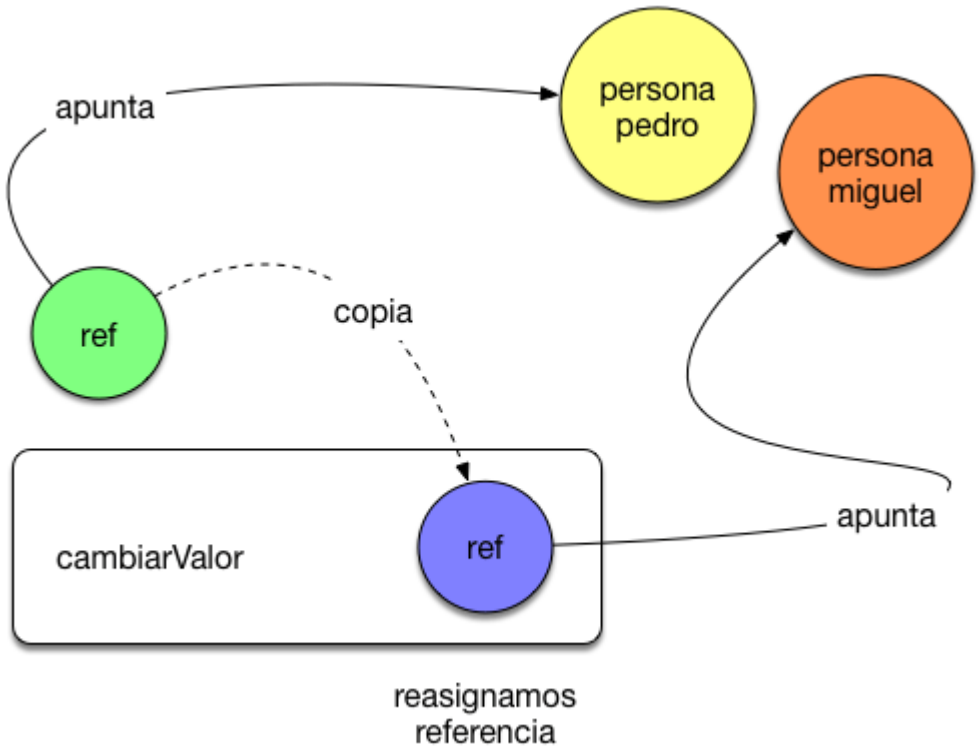
Volvemos a ejecutar el programa para nuestra sorpresa se imprimirá “pedro”.




¿Qué es lo que esta sucediendo?. Lo que esta sucediendo es que en Java todo se pasa por valor tanto los tipos básicos como las referencias a objetos. Por lo tanto cuando nosotros pasamos a una función una referencia a un objeto esta se encarga de hacer una copia de la referencia que apunta al mismo objeto.




Así pues es normal que si nosotros cambiamos el objeto al que apunta la copia de la referencia no suceda nada  nivel del programa principal  ya que la referencia original no varía.



Antes nos funcionaba correctamente porque ambas referencias apuntaban al mismo objeto y podían cambiar su nombre. El concepto de Java value vs reference genera confusiones entre los desarrolladores cuando empiezan a trabajar con el lenguaje Java . Espero que los diagramas ayuden un poco a clarificar su funcionamiento.



Otros artículos relacionados:

1. Java Boxing y sus curiosidades
2. Java this vs this()
3. Java Constructores this() y super()