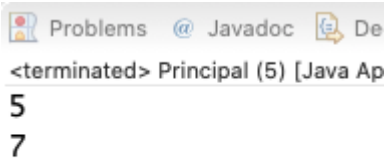
Java Object Reference es una de las preguntas más habituales cuando empezamos a trabajar en Java .¿Que se pasan las variables por valor ? ¿Se pasan por referencia? . Vamos a ver un ejemplo sencillo con unos tipos básicos.

```
package com.arquitecturajava;

public class Principal {

	public static void main(String[] args) {
		int a =5;
		int b=7;
		System.out.println(a);
		System.out.println(b);

	}

}
```

Este código nos imprime un 5 y un 7 por la consola simplemente son los datos que estamos manejando.

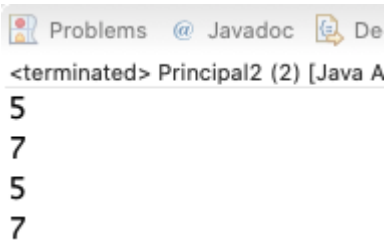


Vamos a ver ahora una pequeña modificación utilizando una función.

```
package com.arquitecturajava;

public class Principal {
```

```
        public static void main(String[] args) {
                int a =5;
                int b=7;
                System.out.println(a);
                System.out.println(b);
                cambiar(a,b);
                System.out.println(a);
                System.out.println(b);
        }

        public static void  cambiar (int a ,int b) {
                a=8;
                b=9;
        }
}
```

Este código nos cambia el contenido de las variables a y b y por lo tanto si las volvemos a imprimir nos debería cambiar el valor . Sin embargo esto no ocurre ya que los tipos básicos son pasados por valor y se realiza una copia de ellos por lo tanto los valores originales no se modifican.



Java Object Reference y funcionamiento

¿Cómo esta esto funcionando exactamente? . La realidad es que Java hace una copia de los valores que tenemos almacenados en memoria al pasárselos a la función y son estos los que se modifican.

Por lo tanto los valores originales se quedan como estaban y no se produce ningún cambio ya que hemos realizado un paso por valor.

Java Objetos y referencias

Esto funciona muy bien para los tipos básicos pero no así para los objetos.Vamos a hacer un ejemplo de la clase Persona.

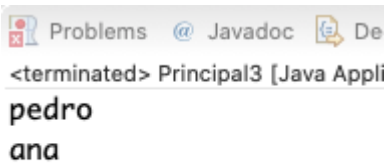
```
package com.arquitecturajava;

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;
        }
}
```

Vamos a ver ahora la clase integrada en nuestro código:

```
package com.arquitecturajava;

public class Principal3 {

        public static void main(String[] args) {
                Persona p= new Persona("pedro");
                System.out.println(p.getNombre());
                cambiar(p);
                System.out.println(p.getNombre());
        }

        public static void  cambiar (Persona p) {
                p.setNombre("ana");
        }
}
```

En este caso hemos usado objetos y todo parece coherente ya que cambiamos el valor del nombre de Pedro a Ana . Si ejecutamos el programa el comportamiento es el esperado.

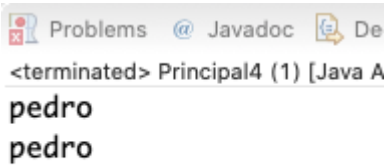


Así que parece que las variables que son objetos se pasan por referencia es decir no se copia el valor.

Lamentablemente asumir esto como válido luego nos generará fuertes quebraderos de cabeza. Vamos a ver une ejemplo que ayude a aclarar las cosas

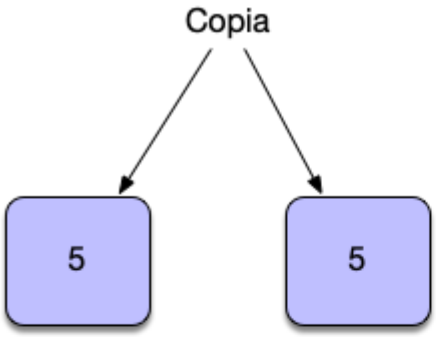
```
package com.arquitecturajava;

public class Principal4 {

	public static void main(String[] args) {
		Persona p= new Persona("pedro");
		System.out.println(p.getNombre());
		cambiar(p);
		System.out.println(p.getNombre());
	}

	public static void  cambiar (Persona p) {
		p= new Persona("ana");
	}
}
```

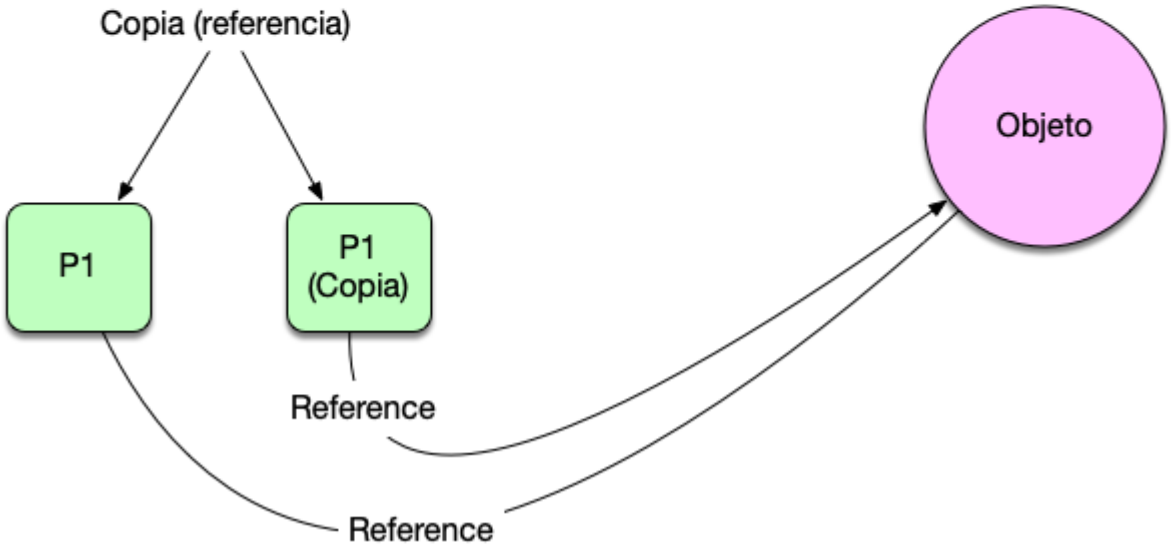
En este caso en vez de usar el método set uso el constructor y vuelvo a generar un objeto nuevo. El resultado será diferente.



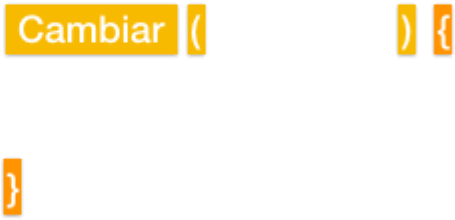
Java Object Reference????

Nos acabamos de quedar helados , ya que no es el comportamiento que estamos esperando. ¿Qué está pasando? . El código es muy similar al anterior pero la realidad es que no funciona como queremos. Mucha gente mirará al constructor a ver si existe algún problema con él . Pero no , todo es correcto. Simplemente no hemos entendido bien como Java gestiona el paso por valor y por referencia . Vamos a explicarlo un poco más a detalle. Java siempre pasa todo por valor y nunca por referencia . Los tipos básicos simplemente hace una copia de ellos y los objetos lo que hace es una copia de la referencia.





Cuando nosotros tenemos una variable como p1 , lo que tenemos es una referencia al Objeto no el objeto en sí. Cuando pasamos esta referencia a una función esta realiza una copia de la referencia y trabaja con ella dejando sin tocar la referencia inicial . Con esta copia la función es capaz de modificar el contenido del objeto pero nunca de variar la referencia original . Veámoslo más a detalle.



Por eso no nos ha funcionado el usar un constructor ya que hemos instanciado un objeto nuevo y la copia de la referencia apunta a uno nuevo.

Cursos Relacionados

- Programación Orientada a Objeto
- Java SE APIS
- Desarrollo Web en Java

Otros artículos relacionados

- Java Herencia vs interfaces
- Java Adapter Pattern
- Java Wrappers

Cecilio Álvarez Caules

Cecilio Álvarez Caules Oracle Java Certified Architech