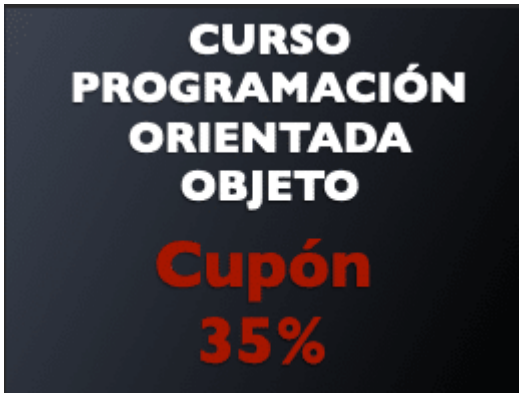
Java string literal vs string object ¿Qué diferencia existe entre las dos opciones? . Normalmente cuando trabajamos en Java solemos crear un String de forma rápida asignando una literal . Pero al tratarse de una clase tenemos la posibilidad de crearlo también como un objeto . El hecho disponer de varias posibilidades suele generar dudas sobre cual es la mejor. Vamos a entrar un poco más a detalle sobre este tema y aclarar las cosas que son un poco peculiares. Supongamos que partimos del siguiente bloque de código.




```
package com.arquitecturajava;

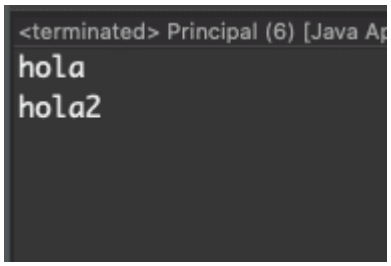
public class Principal {

	public static void main(String[] args) {
		String mensaje1="hola";
		String mensaje2="hola2";
		System.out.println(mensaje1);
		System.out.println(mensaje2);

	}

}
```

El resultado saldrá por la consola y simplemente muestra la información.



Hasta aquí no hay ningún problema se trata de dos objetos Java y cada uno emite su propia información . Podemos construir una segunda versión del programa que generará algunas dudas razonables.

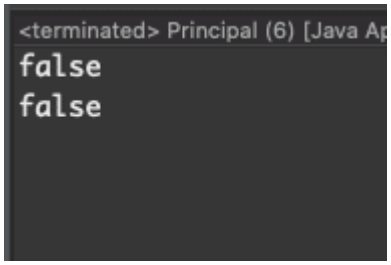
```
package com.arquitecturajava;

public class Principal {

	public static void main(String[] args) {
		String mensaje1="hola";
		String mensaje2="hola";
		System.out.println(mensaje1==mensaje2);
		System.out.println(mensaje1.equals(mensaje2));

	}

}
```



En este caso nos encontramos comprobando si los dos mensajes contienen el mismo contenido y si al ser dos objetos en memoria diferentes el comprobar la igualdad con ==

nos devuelve true o false. Todo parece bastante claro ya que son dos objetos diferentes en memoria con contenidos diferentes. Es normal que el resultado devuelva false y false.

Java string literal vs string object e igualdad

El problema real comienza cuando nosotros modificamos el código por el siguiente.

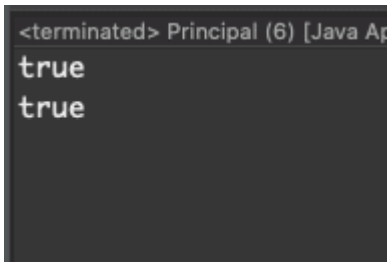
```
package com.arquitecturajava;

public class Principal {

	public static void main(String[] args) {
		String mensaje1="hola";
		String mensaje2="hola2";
		System.out.println(mensaje1==mensaje2);
		System.out.println(mensaje1.equals(mensaje2));

	}

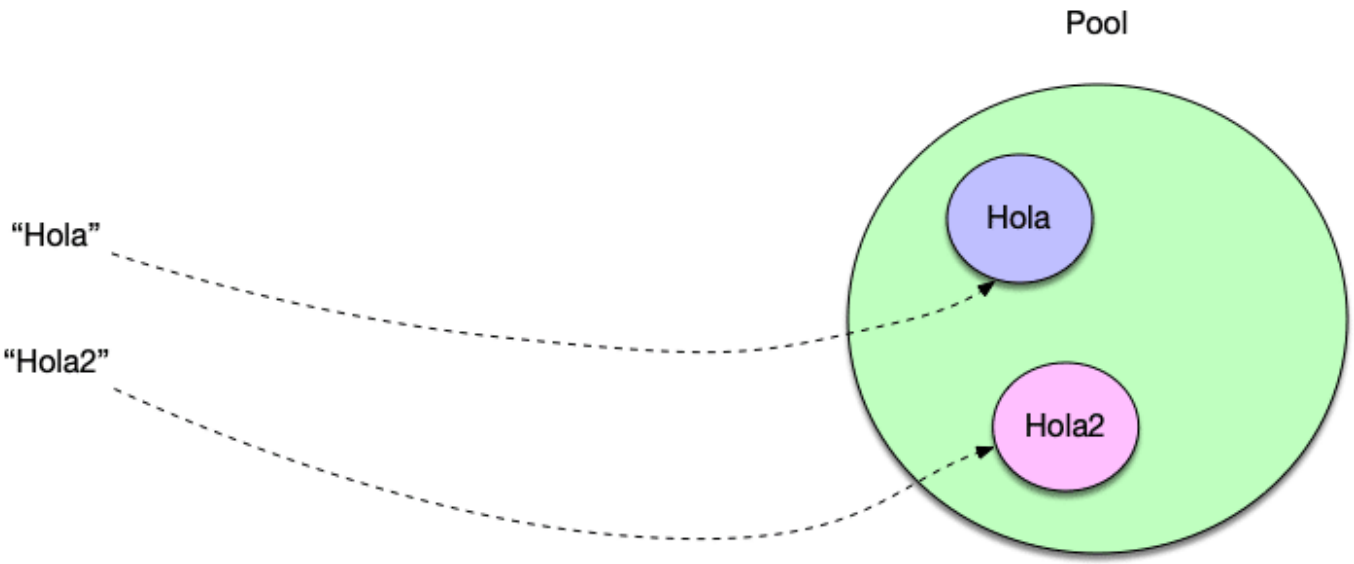
}
```



Es aquí donde las cosas nos sorprenden un poco , es fácilmente entendible que el segundo System.out devuelva true ya que los dos objetos en memoria contienen el mismo texto . Ahora bien lo que sorprende mucho es que el primer método devuelva true. ¿Cómo es esto posible? se trata de dos objetos diferentes y antes devolvía false como es que ahora simplemente cambiando el contenido de los textos el resultado es true. Para entender esto tenemos que abordar el concepto de Java String pool.

Java Pools y Strings

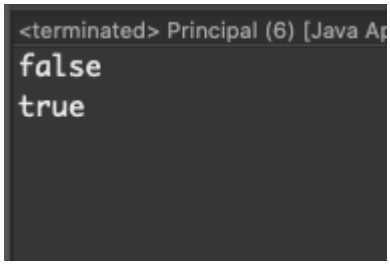
Cuando nosotros usamos literales a la hora de definir cadenas ocurre una cosa curiosa que Java interpreta que es una de las cadenas más típicas con las que vamos a trabajar y las ubica dentro de un pool especial de Strings para no tener que volver a construir ese objeto jamas .



Por ejemplo es util en concatenaciones que muchas veces usan caracteres tipo “,” esa coma irá directamente al pool de strings. Por lo tanto no se genera un objeto cada vez que definimos una cadena como literal sino que se busca primero en el pool a ver si ese objeto existe.

Java string object

No sucederá lo mismo cuando nosotros construyamos un String con el operador new . En este caso siempre se creará un objeto nuevo , por lo tanto si la nueva versión del código fuera :



Esto es mucho más común y es el comportamiento clásico cuando trabajamos con dos objetos diferentes.

Java String y curiosidades

La clase String soporta algunas curiosidades respecto a este tipo de funcionamiento ya que nos permite añadir al pool de Strings las cadenas que deseemos simplemente invocando a String.intern() .

De esta manera si un objeto se creo como literal y por temas de rendimiento lo queremos añadir al pool Java lo hará por nosotros.

```
package com.arquitecturajava;

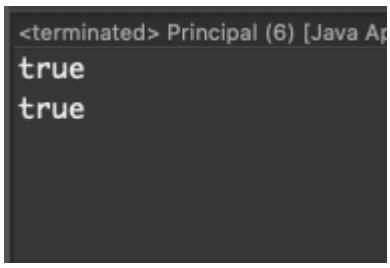
public class Principal {

	public static void main(String[] args) {
		String mensaje1=new String("hola");
		String mensaje1A=mensaje1.intern();
		String mensaje2="hola";
		System.out.println(mensaje1A==mensaje2);
		System.out.println(mensaje1.equals(mensaje2));

	}

}
```

En este código podemos observar como obligamos con el método intern a añadir una cadena al pool y la volvemos a referenciar. Acto seguido creamos un string literal con el mismo nombre que el anterior. el resultado por la consola será que ambos apuntan al mismo objeto.



Otros artículos relacionados

1. Java String pool
2. Tomcat DataSource
3. Java toString
4. Java String