Java Integer es la clase del lenguaje Java que nos permite convertir un tipo básico int en un objeto. Esta clase contiene varios métodos estáticos que permiten realizar conversiones comunes de una forma rápida entre int e Integer o entre Integer y String. Veamos unos ejemplos sencillos.

```
package com.arquitecturajava;

public class Principal {

	public static void main(String[] args) {
		int numero1=7;
		Integer numero2= new Integer(numero1);
		System.out.println(numero2);
		int numero3=Integer.parseInt("8");
		System.out.println(numero3);
		Integer numero4=Integer.valueOf(9);
		System.out.println(numero4);
		String numero5=Integer.toString(10);
		System.out.println(numero5);

	}

}
```

Java Integer Wrapper

Acabamos de utilizar los métodos más comunes

Integer() : El constructor que permite recibir un tipo entero básico y convertirlo en un objeto Integer

parseInt(): El método que nos permite convertir un String a tipo básico int.

valueOf(): Convierte un tipo básico o un String a un objeto Integer.

Hasta aquí no tenemos ningún problema ya que se trata de algo muy elemental con lo que nos manejamos. Sin embargo el otro día me llego una pregunta curiosa sobre el manejo de int e Integers. ¿Qué sucede si ejecutamos el siguiente código?

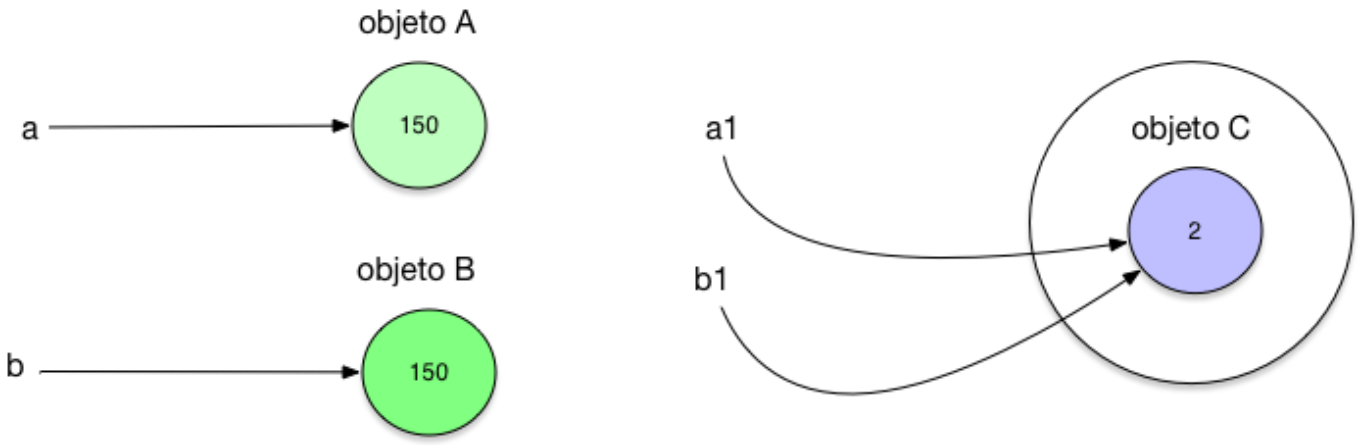
```
package com.arquitecturajava.composite1;

public class JavaInteger {

	public static void main(String[] args) {

		Integer a = 150;
		Integer b = 150;
		System.out.println(a == b); //false

		Integer a1 = 2;
		Integer b1 = 2;
		System.out.println(a1 == b1); //true
	}

}
```

El resultado por la consola es cuando menos inquietante:

¿Qué es lo que esta pasando? . Deberían devolver false ambas casuisticas ya que el operador == compara referencias en memoria. Por lo tanto tenemos 4 objetos diferentes que todos tienen referencias diferentes . El resultado esperado tendría que ser false, false.

JVM y curiosidades

En este caso estamos ante una de las optimizaciones de la máquina virtual . Acabamos de construir 4 objetos ,sin embargo existe una diferencia entre ellos . Aunque les consideramos Integers a los 4 resulta que dos de ellos tienen el tamaño de un tipo “byte”. Ante esta situación la maquina virtual realiza una optimización y decide que para los números muy pequeños del tamaño de un byte generará un pool de objetos. Recordemos que los objetos Integer son inmutables, la optimización parece razonable. Así pues aunque nosotros en nuestro código pensamos que hemos construido 4 objetos. Realmente solo se han construido 3.



Por eso cuando nosotros imprimimos las igualdades por la consola el segundo resultado es true . Ambas referencias apuntan al mismo objeto. En cuanto cambiemos los números e incrementemos su valor a 2oo por ejemplo este efecto desaparecerá.

Otros artículos relacionados

1. Java BigInteger
2. Java String Pool , un concepto importante
3. Java Diamond Operator y Genéricos

Externos

1. Java Tipos básicos