CURSO SPRING BOOT
GRATIS
APUNTATE!!

El concepto de Java override hashcode es una de las preguntas clásicas de los exámenes de certificación de Java Programmer. ¿Cómo funcionan los hashcodes y para que sirven?. Los Java HashCode se encargan de generar un hash para cada uno de nuestros objetos. Vamos a ver un ejemplo sencillo:

```
package com.arquitecturajava.ejemplo1;

public class Libro {

	private String titulo;

	public String getTitulo() {
		return titulo;
	}
	public void setTitulo(String titulo) {
		this.titulo = titulo;
	}

	public Libro(String titulo) {
		super();
		this.titulo = titulo;
	}
```

```
	@Override
	public boolean equals(Object obj) {
		// TODO Auto-generated method stub
		Libro l= (Libro)obj;
		return titulo.equals(l.getTitulo());
	}

	@Override
	public int hashCode() {
		// TODO Auto-generated method stub
		return titulo.hashCode();
	}

}
```

En este caso hemos generado los hash para la clase Libro apoyandonos en los títulos.La primera pregunta que nos podemos hacer es ¿Qué devuelve el método hashcode? .La respuesta es un numero entero.

```
package com.arquitecturajava.ejemplo1;

public class Principal2 {

	public static void main(String[] args) {

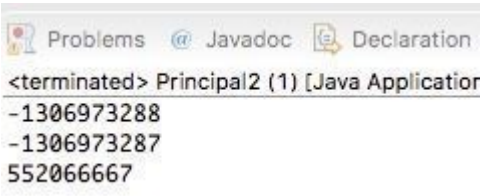
		Libro libroA= new Libro("titulo1");
		System.out.println(libroA.hashCode());
		Libro libroB= new Libro("titulo2");
```

```
                System.out.println(libroB.hashCode());
                Libro libroC= new Libro("2titulo");
                System.out.println(libroC.hashCode());

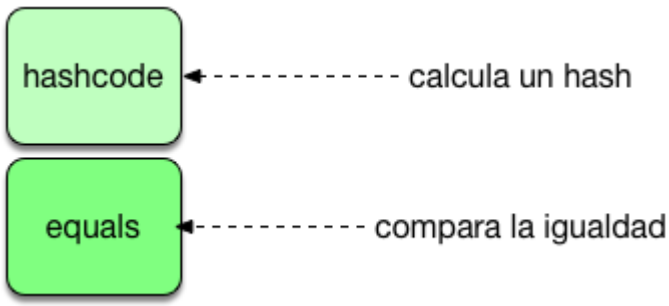
        }

}
```

El resultado se imprime por la consola:



Cada objeto genera su propio hashcode . Si dos objetos generan hashcodes diferentes , los objetos son diferentes . Si dos objetos tienen el mismo hashcode pueden todavía ser diferentes y hay que comprobar con el método equals para confirmar su igualdad.



Java Override HashCode y Colecciones

¿Cómo se usan los hashcodes a nivel del framework de colecciones? . Los hashcodes se utilizan cuando trabajamos con HashSet ,HashMap o colecciones similares. Estas colecciones guardan internamente una estructura de Tabla Hash . ¿Qué es una Tabla Hash? . Es una estructura que permite organizar la información en una forma matricial en este

caso apoyándose en los hashcodes.

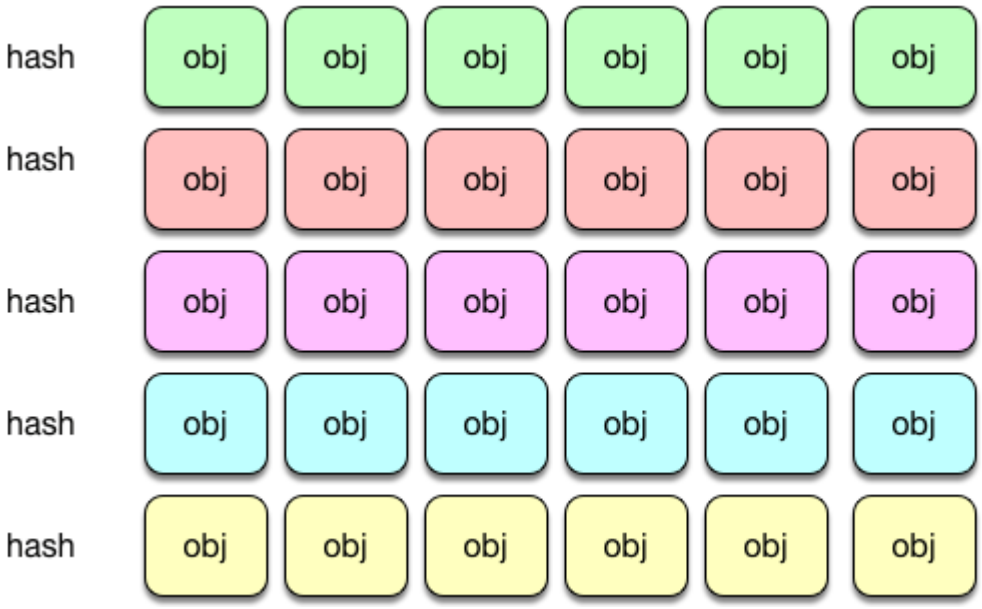
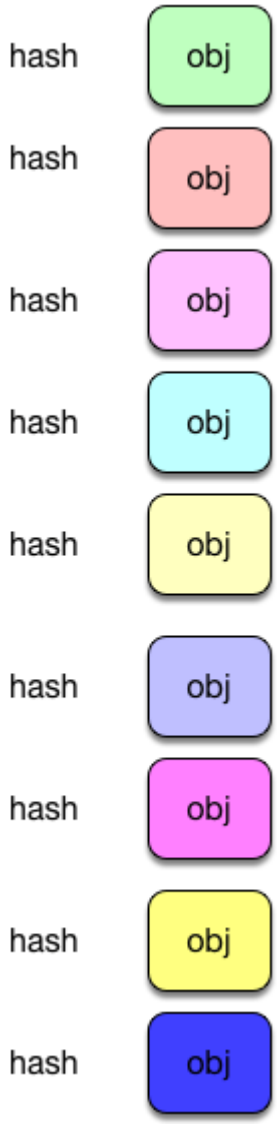
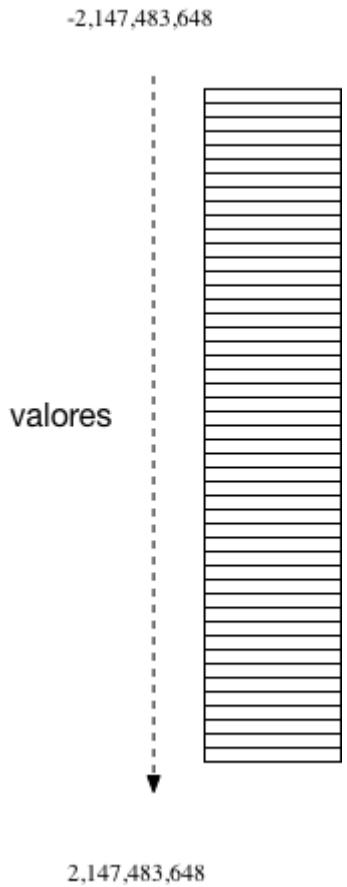


Tabla Hash

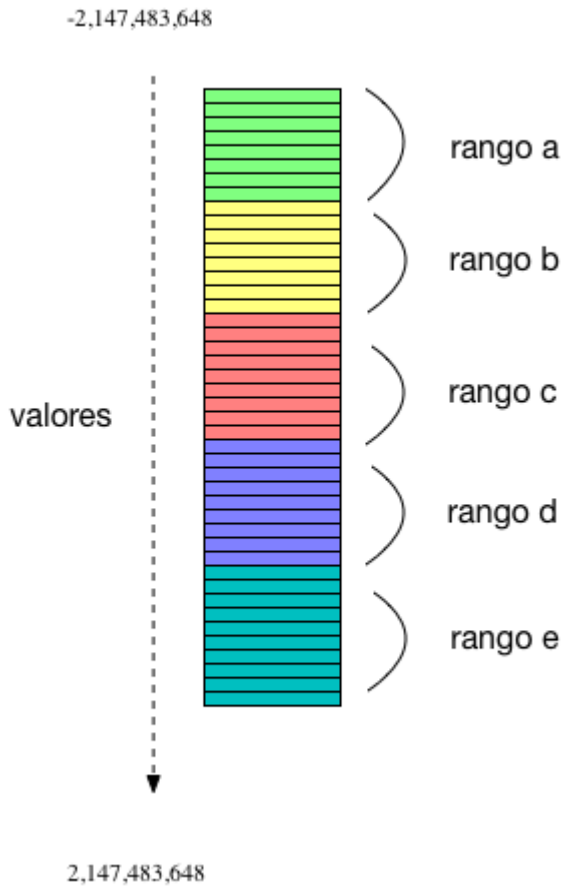
Cada objeto que insertemos se ubica en la tabla en relación a su hashcode. De esta forma en vez de recorrernos todos los elementos por completo basta con generar su hash y recorrer una sublista. El problema muchas veces es entender la implementación a detalle. Si cada vez que generamos un hash diferente se generara una nueva una sublista. la estructura no será como la que aparece en el diagrama anterior . Sino algo más similar a lo siguiente:



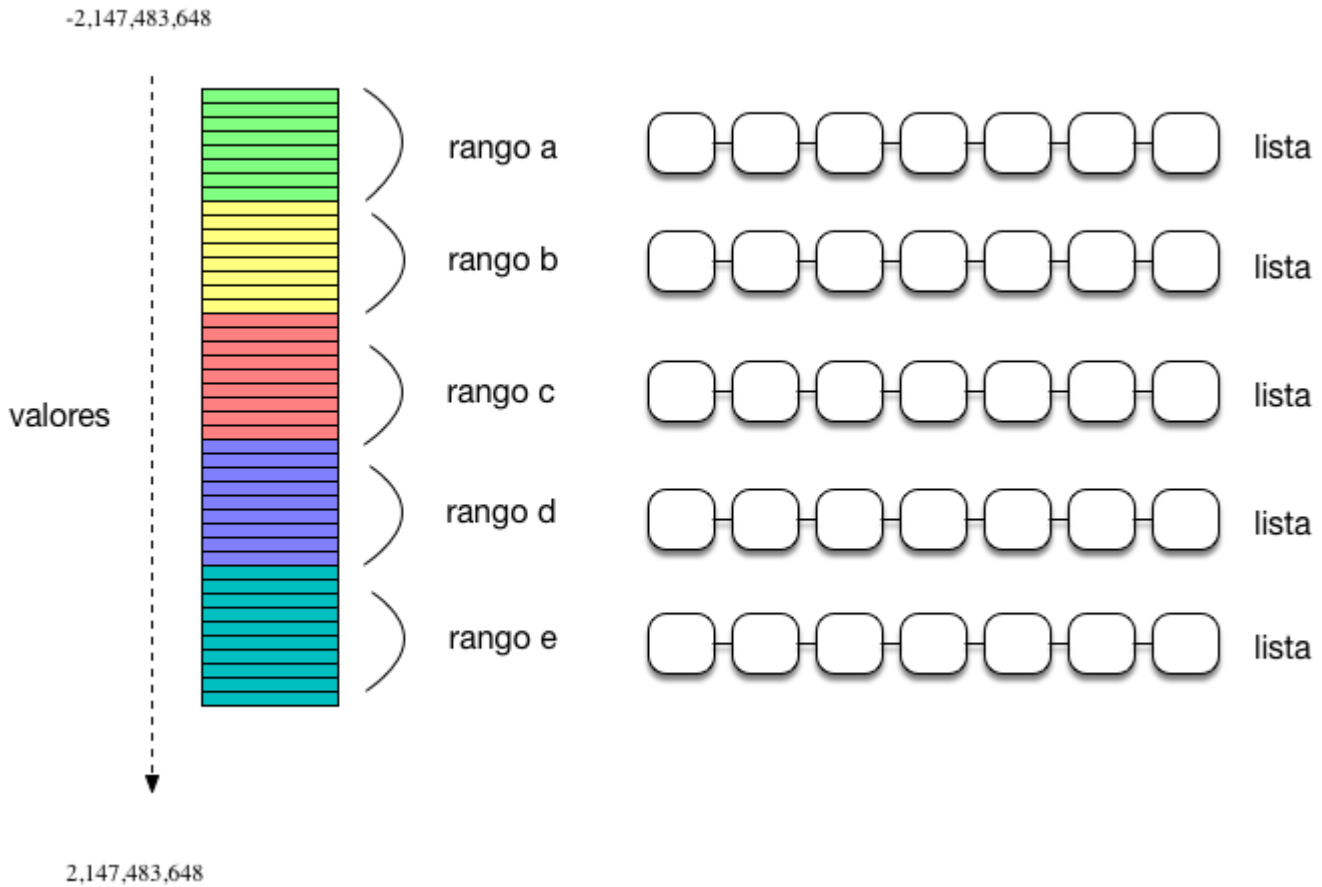
No parece una solución muy sólida,¿Cada elemento genera una una sublista? . Realmente parece poco acertado. ¿Entonces que es lo que no estamos entendiendo? . ¿Cada hash diferente crea una nueva sublista?¿ Un grupo amplio de objetos genera el mismo hash y comparten lista?. Ninguna de las dos opciones es cierta. Vamos a explicarlo un poco más a detalle. Lo primero que tenemos que entender es que un hashcode es un número entero y soporta un rango de valores.



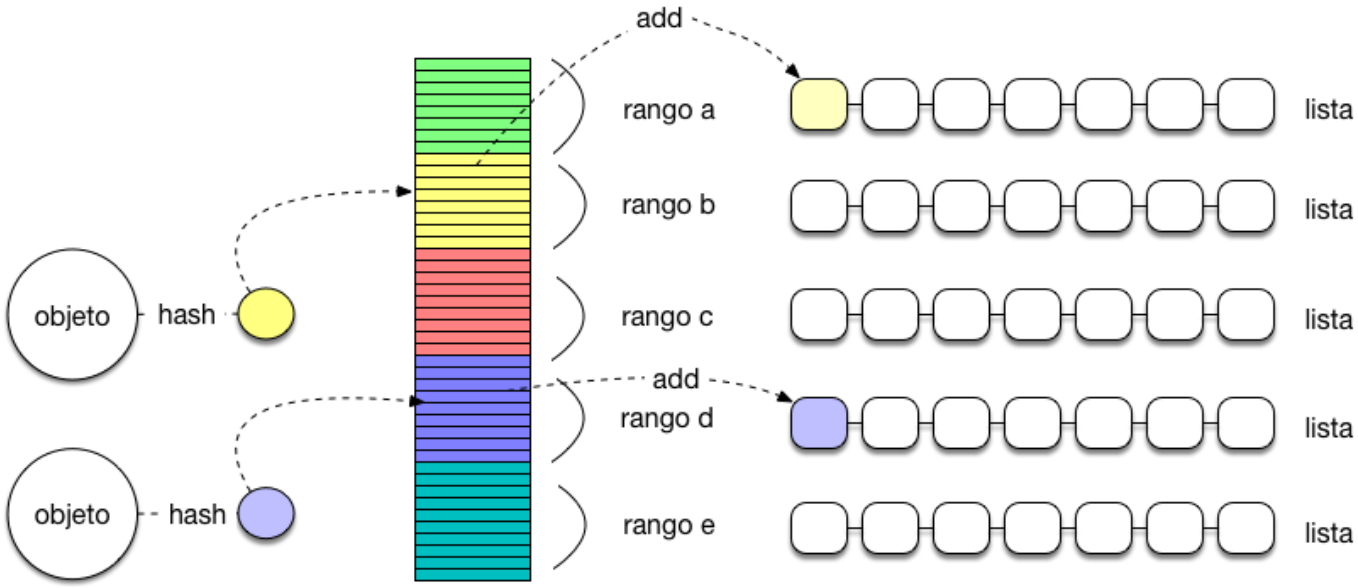
Este rango de valores se puede dividir en zonas (buckets). De tal forma que cada zona agrupe un subrango de valores:



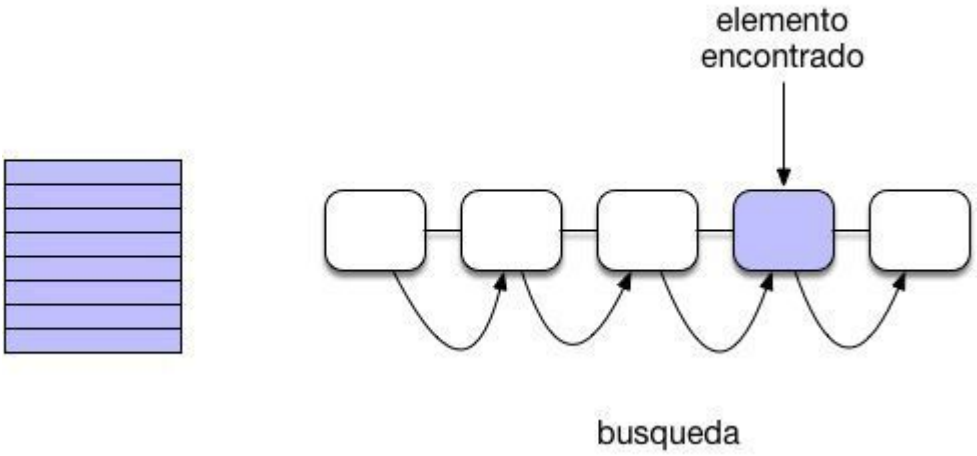
Cada subrango de valores tiene asociada una lista en la cual nosotros insertaremos cada objeto. Por lo tanto muchos objetos con diferentes hash irán en la misma sublista.



Cada vez que queremos añadir un elemento al HashSet tendremos que generar su hashcode y una vez generado el hash añadirle en la sublista que corresponda.



Esta es la forma en la que funciona por ejemplo un HashSet o un HashMap con Java override HashCode. En estas estructuras de información es más lento añadir elementos ya que se necesita generar el hash . Pero luego las búsquedas se acelerarán ya que buscaremos en una única sublista.



De ahí que sea fundamental en Java sobreescribir de forma simultanea hashcode y equals. Sino nuestras clases no se comportarán de la forma adecuada cuando las usemos en algunas clases del framework de colecciones. Los hashcodes no son usados por Java cuando recorremos un ArrayList, en este caso simplemente se compara el método equals .

Otros artículos relacionados:

CURSO SPRING BOOT

GRATIS

APUNTATE!!

1. Comparando java == vs equals
2. Java Override y encapsulación
3. Java Constructores this() y super()
4. Java HashCode