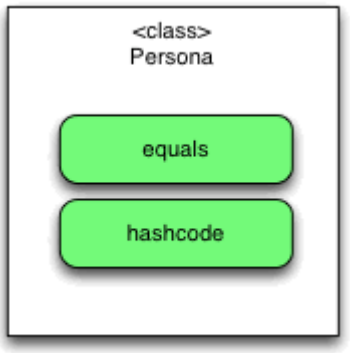
Todos los días cuando programamos usamos objetos y en muchas ocasiones necesitamos comparar unos con otros. Para ello en muchas ocasiones usamos los métodos de de Java equals y hashcode. Estos métodos generan muchas dudas entre los desarrolladores a la hora de usarlos.



Vamos a crear una implementación por defecto para la clase Persona que realice un override sobre ambos métodos para entenderlos mejor.

```
package com.arquitecturajava;

public class Persona {

private String nombre;
 private int edad;
```

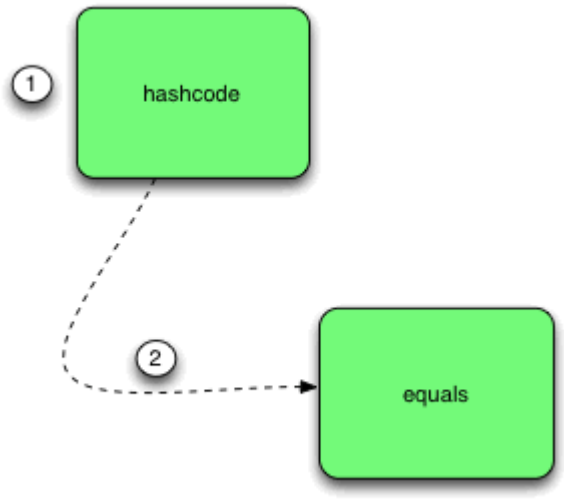
```
public String getNombre() {
return nombre;
}
public void setNombre(String nombre) {
this.nombre = nombre;
}
public int getEdad() {
return edad;
}
public void setEdad(int edad) {
this.edad = edad;
}
@Override
public int hashCode() {
final int prime = 31;
int result = 1;
result = prime * result + ((nombre == null) ? 0 : nombre.hashCode());
return result;
}
@Override
public boolean equals(Object obj) {
if (this == obj)
return true;
if (obj == null)
return false;
if (getClass() != obj.getClass())
return false;
Persona other = (Persona) obj;
if (nombre == null) {
if (other.nombre != null)
return false;
```

```
 } else if (!nombre.equals(other.nombre))
 return false;
 return true;
 }

}
```

El método equals es el más sencillo de entender ya que comprueba si los dos objetos son del mismo tipo y si su nombre coincide. En tal caso el resultado será true y en todos los demás false. Para ello el método realiza una serie de comparaciones.

El método hashcode

Este método viene a complementar al método equals y sirve para comparar objetos de una forma más rápida en estructuras Hash ya que únicamente nos devuelve un número entero. Cuando Java compara dos objetos en estructuras de tipo hash (HashMap, HashSet etc) primero invoca al método hashcode y luego el equals. Si los métodos hashcode de cada objeto devuelven diferente hash no seguirá comparando y considerará a los objetos distintos. En el caso en el que ambos objetos compartan el mismo hashcode Java invocará al método equals() y revisará a detalle si se cumple la igualdad. De esta forma las búsquedas quedan simplificadas en estructuras hash.

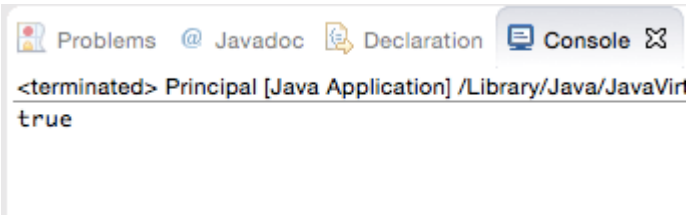


Manejando colecciones

Muchas veces se nos olvida que la invocación a los métodos equals y hashcode forma parte intrínseca del framework de colecciones. Por ejemplo si construimos dos objetos de tipo Persona y los añadimos a un HashSet, podemos comprobar que la Persona existe utilizando el método contains dentro del conjunto.

```
package com.arquitecturajava;

import java.util.HashSet;

public class Principal {

public static void main(String[] args) {
Persona p1= new Persona();
p1.setNombre("david");

Persona p2= new Persona();
p2.setNombre("miguel");
```

```
HashSet&amp;amp;amp;lt;Persona&amp;amp;amp;gt; conjunto= new
HashSet&amp;amp;amp;lt;Persona&amp;amp;amp;gt;();
conjunto.add(p1);
conjunto.add(p2);
System.out.println(conjunto.contains(p1));

}

}
```



Esto nos devolverá true ya que el HashSet contiene este elemento. Ahora bien si nosotros sobreescribimos de forma incorrecta el hashcode con el siguiente código:

```
@Override
 public int hashCode() {

 return (int)(Math.random()*1000);

 }
```

Estaremos calculando al azar el hashcode y dos objetos iguales devolverán hashcodes diferentes .El HashSet nos devolverá false cuando invoquemos el método contains aunque sabemos que el elemento existe en el conjunto.

```
<terminated> Principal [Java Application] /Library/Java/JavaVirtu
false
```

Esto se debe a que Java comparará dos elementos primero por su hashcode y como no coinciden directamente devolverá falso . Por lo tanto si queremos sobreescribir correctamente los métodos equals y hashcode debemos asegurarnos de que cuando dos objetos sean iguales devuelvan el mismo hashcode. Eso sí aunque nos parezca curioso dos objetos diferentes pueden tener mismo hashcode. Esto no está en constradicción con lo anteriormente expuesto.