SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL

SENAI “GASPAR RICARDO JUNIOR”

Curso

TÉCNICO EM DESENVOLVIMENTO

DE SISTEMAS

Métodos equals e hashCode em Java e o uso de Lombok para otimizar código em ambientes de desenvolvimento

Lauren Miguel Norberto

Sorocaba
Novembro 2024

SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL

SENAI “GASPAR RICARDO JUNIOR”

Lauren Miguel Norberto

Métodos equals e hashCode em Java e o uso de Lombok para otimizar código em ambientes de desenvolvimento

Elaboração de pesquisa acerca
do funcionamento dos métodos
equals e hashCode na linguagem
de programação Java
Prof. Emerson Magalhães

Sorocaba
Novembro 2024

Sumário

INTRODUÇÃO

O método equals é necessário na programação Java para comparar valores de diferentes variáveis no lugar de usar um condicional, seguido de um operador. Isto porque, ao usar um condicional e um operador o programa compara os endereços das variáveis na memória e não propriamente o conteúdo das variáveis. Já que este não é o comportamento que buscamos, é necessário utilizar o método booleano equals, que recebe uma String como parâmetro. Ele é responsável por comparar o valor do conteúdo do objeto no qual está sendo chamado com o valor que está recebendo em seu parâmetro.

Enquanto isso, o método hashCode é uma ferramenta da Java Vitual Machine que retorna um valor hash, ou seja, um valor inteiro que representa um objeto. Isso se explica por que esses objetos foram distribuídos uniformemente em uma tabela hash, onde os elementos são referenciados por uma chave chamada de “número hash”. Consequentemente, as estruturas de dados baseadas em hash permitem um acesso rápido a esses elementos.

A combinação destes elementos em Spring é importante para a consistência do código, prevenindo comportamento indesejado ou incorreto, ou ainda objetos duplicados, além disso, são frequentemente usados para a identificação correta de beans.

Por outro lado, o Lombok também é um importante framework para programação em Java que simplifica a escrita de código, pois remove a verbosidade. Com ele, é possível gerar durante a compilação, métodos getters e setters, construtores, padrões builder, entre outras funcionalidades.

1. FUNDAMENTOS TEÓRICOS

Em uma programação onde é necessário cadastrar vários objetos diferentes, o método equals é eficiente para evitar repetições. Entretanto, é preciso também sobrescrever o método hashCode, pois sua funcionalidade é retornar um valor inteiro que representará o agrupamento de objetos que serão comparados pelo método equals. Ou seja, códigos hash agrupam objetos semelhantes e atribuem um código a cada uma dessas coleções. Depois disso, métodos equals serão capazes de comparar objetos dessas coleções.

Sendo assim, não é possível implementar o método equals sem o método hashCode, ou o contrário.

1.1. Contrato de equals

Segundo a Documentação do Java, o contrato de equals segue as seguintes diretrizes:

1. **Reflexividade:** Para qualquer objeto não-nulo de valor referência x, x.equals() deve retornar true.
2. **Simetria:** Para qualquer objeto não-nulo de valor de referência x e y, x.equals(y) deve retornar true, se y.equals(x), retornar true também.
3. **Transitividade:** Para mais quaisquer objetos não-nulos de valor de referência x,y e z, x.equals(y) deve retornar true, se y.equals(z) retornar true e z.equals(x) também retornar true.
4. **Consistência:** Os resultados de equals() mudarão somente quando os campos nele incluídos, mudarem. Do contrário, os resultados deverão ser sempre os mesmos.
5. **Desigualdade com null:** Para qualquer objeto, x.equal(null) deve retornar falso.

1.2. Contrato de hashCode

Quanto ao método hashCode, o contrato possui os seguintes requisitos:

1. Se dois objetos forem iguais, ou seja, o método equals() retornar true, eles deverão ter o mesmo código hash.
2. Todas as vezes que um método hashCode for chamado em um mesmo objeto, ele deverá retornar o mesmo número.
3. Dois objetos podem ter o mesmo código hash.

1.3. Como o contrato entre equals e hashCode afeta o comportamento das coleções

Observando que uma coleção HashSet não permite itens duplicados e que o HashMap armazena itens com uma chave e um valor (relação chave/ valor), percebe-se que não há como as funcionalidades serem executadas plenamente sem a operação conjunta dos métodos hashCode e equals. Isso porque o hashCode é responsável por gerar valores inteiros que serão as chaves dos objetos e o equals, por meio da comparação, determinará se o objeto chamado será o objeto correspondente. Ou seja, na busca de objetos específicos dentro das coleções, os métodos são indispensáveis.

1.4. Utilização Prática em Coleções Java e no Spring

```java
public int hashCode()
{
    return getNome().length() * 8;
}
```

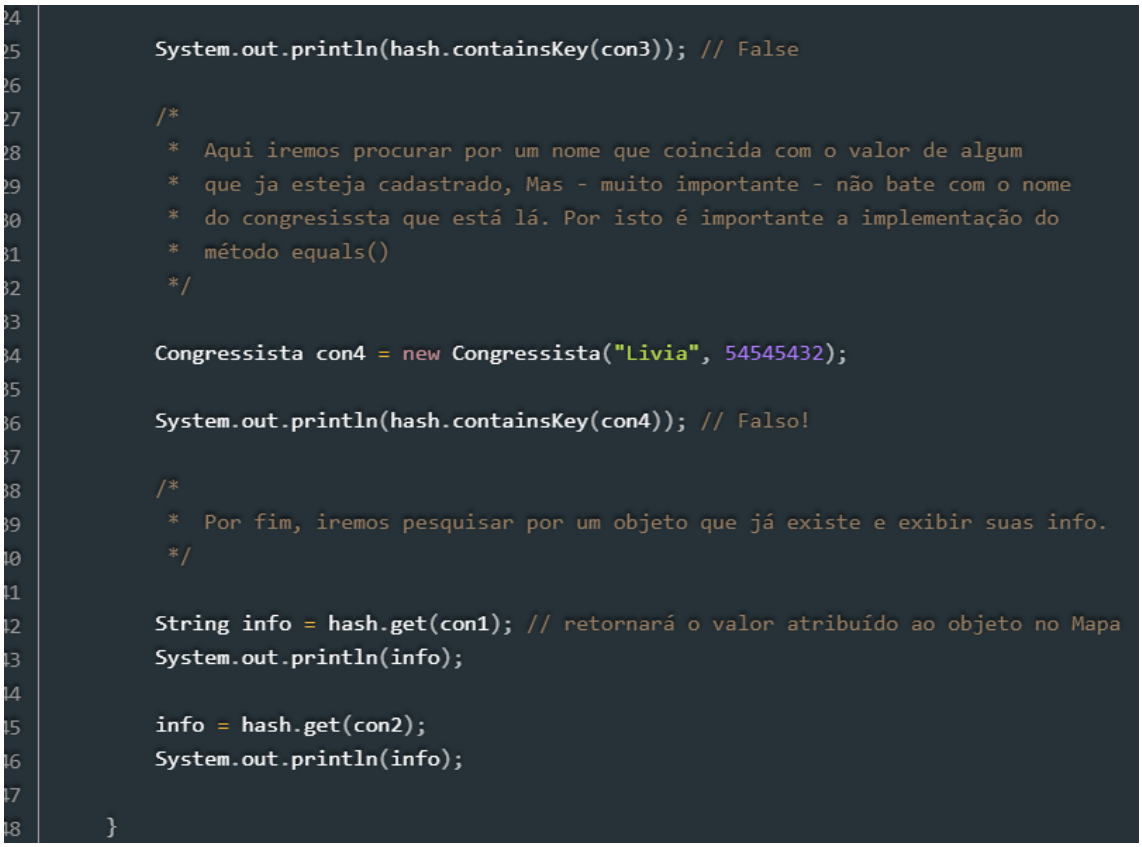
FIGURA 1 – IMPLEMENTAÇÃO DO MÉTODO HASHCODE

```java
public boolean equals(Object o)
{
    if ((o instanceof Congressista) &&
            ((Congressista) o).getNome().equals(this.getNome()))
    {
        return true;
    }else
        return false;
}
```

FIGURA 2 – IMPLEMENTAÇÃO DO MÉTODO EQUALS

```java
import java.util.HashMap;

public abstract class Teste{

    public static void main(String[] args) {

        Congressista con1 = new Congressista("Tiago", 1234567);
        Congressista con2 = new Congressista("Caio", 76543221);

        HashMap<Congressista,String> hash = new HashMap<Congressista,String>();

        // Inserção dos congressistas no HashMap
        hash.put(con1, "Info importante sobre Tiago");
        hash.put(con2, "Info importante sobr Caio");

        /*
         * Criaremos um terceiro congressista e iremos efetuar uma busca
         * para ver se o algoritimo está, de fato, funcionando.
         * Não será encontrado o objeto porque Primeiro: não o inserimos no HashMap
         * Segundo: o tamanho do nome não bate com nenhum tamanho de nome inserido.
         */

        Congressista con3 = new Congressista("Felipe", 123123123);
```

FIGURA 3– EXEMPLO PRÁTICO EM HASHMAP (PARTE 1)



```
24
25          System.out.println(hash.containsKey(con3)); // False
26
27          /*
28           *  Aqui iremos procurar por um nome que coincida com o valor de algum
29           *  que ja esteja cadastrado, Mas - muito importante - não bate com o nome
30           *  do congresissta que está lá. Por isto é importante a implementação do
31           *  método equals()
32           */
33
34          Congressista con4 = new Congressista("Livia", 54545432);
35
36          System.out.println(hash.containsKey(con4)); // Falso!
37
38          /*
39           *  Por fim, iremos pesquisar por um objeto que já existe e exibir suas info.
40           */
41
42          String info = hash.get(con1); // retornará o valor atribuído ao objeto no Mapa
43          System.out.println(info);
44
45          info = hash.get(con2);
46          System.out.println(info);
47
48      }
```

FIGURA 4 – EXEMPLO PRÁTICO EM HASHMAP (PARTE 2)

2. LOMBOK: SIMPLIFICAÇÃO DO CÓDIGO

2.1. Vantagens de usar Lombok em projetos Java

O Lombok é um framework para Java que reduz significamente a verbosidade do código e tornando-o menos repetitivo. Para isso, ele gera anotações em tempo de compilação que geram métodos construtores, getters e setters, métodos equals e hashcode, entre outros. Além disso, o Lombok é bem suportado por várias IDEs populares (como IntelliJ IDEA, Eclipse e VSCode) através de plugins, tornando sua utilização ainda mais conveniente.

2.2. Análise das anotações @EqualsAndHahsCode e @Data

Em um código Java, usando o framework Lombok, qualquer classe pode ser anotada com @EqualsAndHashCode e vai gerar os métodos equals() e hashCode() durante a compilação, como segue o código abaixo:

```
1  @Entity
2  @NoArgsConstructor @AllArgsConstructor
3  @EqualsAndHashCode
4  public class User {
5
6      @Id @GeneratedValue
7      @Getter private Long id;
8      @Getter @Setter private String firstName;
9      @Getter @Setter private String lastName;
10     @Getter @Setter private String email;
11     @Getter @Setter private Date age;
12     @Getter @Setter private String gender;
13 }
```

FIGURA 5 – EXEMPLO DE ANOTAÇÃO @EQUALSANDHASHCODE

Nesse modelo de código, também visualizamos as anotações para geração métodos getters e setters do mesmo framework.

Já a anotação @Data reúne os métodos getters, setters, equals, hashCodes e toStrings em uma única anotação. Segue o exemplo:

```java
@Data
public class User {

    @Id @GeneratedValue
    private Long id;
    private String firstName;
    private String lastName;
    private String email;
    private Date age;
    private String gender;
}
```

FIGURA 6 – EXEMPLO DE ANOTAÇÃO @DATA

2.3. Vantagens e Desvantagens

O Lombok reduz o esforço e otimiza o tempo do desenvolvedor. Ele diminui o código, mantendo as funcionalidades dos métodos nas classes e garantindo consistência e boas práticas dentro da programação. Entretanto, a dependência em uma biblioteca externa pode ser um problema no que se refere às alterações ou incompatibilidades do Lombok com o código. Também é mais complexo e obscuro depurar o código, porque é preciso consultar o código-fonte para entender seu código diretamente no IDE. Para utilização do Lombok, é necessário confiar e sua documentação.

CONCLUSÃO

Sendo assim, métodos hashCode e equals devem ser implementados em conjunto, principalmente nas coleções HashMap e HashSet. Suas funcionalidades principais são armazenar objetos evitando que sejam duplicados. Contanto, o framework Lombok pode auxiliar na implementação destes métodos através das anotações @EqualsAndHashCode ou @Data, de forma que a primeira implementa apenas estes métodos e a segunda, implementa getters, setters, construtores e ainda, equals e hashCode. O uso do Lombok reduz o código e auxilia na consistência, entretanto também pode resultar em problemas no que se refere a dependência a uma biblioteca externa.

BIBLIOGRAFIA

Portal DIO. O Método Equals em Java: Comparando Conteúdo, Não Referências. Disponível em: <https://www.dio.me/articles/o-metodo-equals-em-java-comparando-conteudo-nao-referencias>. Acesso em: 08 nov. 2024.

Portal JDEV Treinamento. MÉTODOS EQUALS E HASHCODE. Disponível em: <https://www.jdevtreinamento.com.br/metodos-equals-e-hashcode/#:~:text=J%C3%A1%20o%20hashCode()%20%C3%A9,tabela%20hash%20de%20modo%20correto >. Acesso em: 08 nov. 2024.

Portal DIO. Como usar o Lombok em projetos Java. Disponível em: <https://www.dio.me/articles/como-usar-o-lombok-em-projetos-java> Acesso em: 08 nov. 2024.

Portal AlgaWorks. Entendendo o equals e hashCode. Disponível em: <https://blog.algaworks.com/entendendo-o-equals-e-hashcode/> Acesso em: 09 nov. 2024.

Portal JavaRush. Método equals e hashCode: prática de uso. Disponível em: <https://javarush.com/pt/groups/posts/pt.2179.mtodos-equals-e-hashcode-prtica-de-uso>. Acesso em: 09 nov. 2024.

Portal Angeliski. Equals e HashCode: O que é, porque usar e como implementar. Disponível em: <https://angeliski.com.br/equals-e-hashcode?x-host=angeliski.com.br>. Acesso em: 09 nov. 2024.

Portal DEVMEDIA. Sobrescrevendo o método hashCode() em Java. Disponível em: <https://www.devmedia.com.br/sobrescrevendo-o-metodo-hashcode-em-java/26488>. Acesso em: 09 nov. 2024.

Portal W3Schools. Java HashSet. Disponível em: <https://www.w3schools.com/java/java_hashset.asp>. Acesso em: 09 nov. 2024.

Portal W3Schools. Java HashMap. Disponível em: <https://www.w3schools.com/java/exercise.asp?x=xrcise_hashmap1>. Acesso em: 09 nov. 2024.

Portal iMasters. Projeto Lombok: Escrevendo menos código em Java. Disponível em: <https://imasters.com.br/back-end/projeto-lombok-escrevendo-menos-codigo-em-java>. Acesso em: 10 nov. 2024.

Portal Linkedin. Quais são os benefícios e desvantagens de usar Lombok para gerar métodos equals e hashCode? Disponível em: <

https://pt.linkedin.com/advice/1/what-benefits-drawbacks-using-lombok-generate?lang=pt>. Acesso em: 10 nov. 2024.

LISTA DE FIGURAS