



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**

Helena Padilha Bueno

Sorocaba
Novembro – 2024



SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL

SENAI “GASPAR RICARDO JUNIOR”

Helena Padilha Bueno


Métodos equals e hashCode em Java e o uso de Lombok

Pesquisa da matéria de backend, sobre
equals hashCode e Lombok
Prof. – Emerson Magalhães

Sorocaba
Novembro – 2024

SUMÁRIO

INTRODUÇÃO.....	2
1. Contextualização dos Métodos equals e hashCode	3
1.1. Importância para Coleções e Frameworks como Spring	3
2. Introdução ao Lombok.....	4
2.1. Fundamentos Teóricos.....	4
2.1.1. Contrato entre equals e hashCode	4
2.1.2. Como o Contrato afeta o Comportamento das Coleções.....	4
2.1.3. Importância da Implementação Correta em Entidades.....	5
3. Utilização Prática em Coleções Java e no Spring.....	5
4. Exemplo com o Spring	6
5. Lombok: Simplificação do Código	6
6. Vantagens e Desvantagens de Usar Lombok para equals e hashCode.....	8
CONCLUSÃO.....	9
BIBLIOGRAFIA	10



Métodos equals e hashCode em Java e o uso de Lombok

INTRODUÇÃO

No desenvolvimento em Java, os métodos equals e hashCode são essenciais para a comparação e organização de objetos, especialmente em coleções como HashSet e HashMap, além de serem importantes em frameworks como o Spring para operações de caching e persistência. Uma implementação incorreta desses métodos pode gerar duplicações e falhas na recuperação de dados. A biblioteca Lombok oferece uma forma prática de gerar automaticamente equals e hashCode, reduzindo o código repetitivo e facilitando a manutenção, mas com algumas limitações, como dependência externa e desafios na depuração. Esta pesquisa explora como equals, hashCode e Lombok impactam o desenvolvimento em Java.

1. Contextualização dos Métodos equals e hashCode

Em Java, todo objeto herda métodos fundamentais da classe Object, entre eles equals e hashCode. O método equals serve para verificar a igualdade entre dois objetos, enquanto hashCode retorna um valor inteiro que representa o “código hash” do objeto. Esses métodos são especialmente importantes em coleções baseadas em hashing, como HashMap e HashSet, pois influenciam diretamente na forma como os objetos são armazenados e recuperados nessas estruturas.

1.1. Importância para Coleções e Frameworks como Spring

Para coleções que utilizam hashing, como HashSet e HashMap, equals e hashCode determinam se um objeto já existe na coleção e onde ele será armazenado. Em frameworks como o Spring, esses métodos são essenciais para garantir que as entidades Java sejam corretamente manipuladas, comparadas e gerenciadas em operações de cache e persistência.

2. Introdução ao Lombok

Lombok é uma biblioteca que ajuda a reduzir o código repetitivo (boilerplate) em Java, fornecendo anotações para gerar automaticamente métodos como equals, hashCode, toString, entre outros. Ao facilitar a criação desses métodos, Lombok otimiza o processo de desenvolvimento, mantendo o foco na lógica de negócios.

2.1. Fundamentos Teóricos

2.1.1. Contrato entre equals e hashCode

O contrato entre equals e hashCode define que:

- Regra 1: Se dois objetos são iguais (usando equals), eles devem ter o mesmo valor de hashCode.
- Regra 2: Se dois objetos têm valores de hashCode diferentes, equals não deve considerá-los iguais.
- Regra 3: Se dois objetos têm o mesmo hashCode, eles podem ou não ser iguais; no entanto, eles devem estar armazenados no mesmo bucket em coleções baseadas em hashing.

Essas regras são cruciais para o funcionamento correto das coleções, pois o Java utiliza o hashCode para determinar o bucket de armazenamento de um objeto. Uma implementação inadequada desses métodos pode resultar em comportamentos inesperados, como objetos duplicados em coleções ou falhas na recuperação.

2.1.2. Como o Contrato afeta o Comportamento das Coleções

Em coleções como HashMap e HashSet, o valor do hashCode de um objeto determina em qual bucket ele será armazenado. Em um HashSet, por exemplo, dois objetos com o mesmo hashCode serão armazenados no mesmo bucket, e equals será utilizado para determinar se eles são considerados iguais. Caso o contrato entre equals e hashCode não seja seguido, a coleção pode armazenar objetos duplicados ou falhar ao localizar objetos existentes.

2.1.3. Importância da Implementação Correta em Entidades

Em aplicações que utilizam frameworks como o Spring, as entidades Java muitas vezes precisam implementar `equals` e `hashCode` para garantir uma gestão eficiente de objetos em operações de persistência e caching. Um erro nesses métodos pode afetar a integridade dos dados e a eficiência da aplicação

3. Utilização Prática em Coleções Java e no Spring

Exemplo Prático com Coleções Java (HashSet e HashMap)

1. HashSet:

```
class Produto {
    private int id;
    private String nome;

    // equals e hashCode implementados
    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null || getClass() != obj.getClass()) return false;
        Produto produto = (Produto) obj;
        return id == produto.id;
    }

    @Override
    public int hashCode() {
        return Objects.hash(id);
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        HashSet<Produto> produtos = new HashSet<>();
        produtos.add(new Produto(1, "Produto A"));
        produtos.add(new Produto(1, "Produto A"));

        System.out.println(produtos.size()); // Saída: 1
    }
}
```

Nesse exemplo, mesmo que adicionemos dois objetos Produto com os mesmos dados, o HashSet mantém apenas um, pois equals e hashCode foram implementados corretamente.

2. HashMap:

Em um HashMap, o hashCode ajuda a localizar rapidamente a chave no mapa, enquanto equals verifica se a chave é realmente igual:

```
HashMap<Produto, Integer> estoque = new HashMap<>();
Produto produto1 = new Produto(1, "Produto A");
Produto produto2 = new Produto(1, "Produto A");

estoque.put(produto1, 100);
System.out.println(estoque.get(produto2)); // Saída: 100
```

4. Exemplo com o Spring

No Spring, uma entidade Produto em um sistema de persistência que utilize Hibernate, por exemplo, precisa de uma implementação consistente de equals e hashCode para garantir que duas instâncias do mesmo produto sejam tratadas como iguais. Isso é crucial para a eficiência de operações de caching e consultas.

5. Lombok: Simplificação do Código

Lombok reduz o código boilerplate ao gerar métodos como equals, hashCode, toString, getters, e setters. Em vez de implementar esses métodos manualmente, uma anotação Lombok pode gerar o código automaticamente, melhorando a legibilidade e a manutenção.

Anotações @EqualsAndHashCode e @Data

- @EqualsAndHashCode: Gera os métodos equals e hashCode automaticamente com base nos campos da classe. Pode ser configurada para incluir ou excluir campos específicos.
- @Data: Gera equals, hashCode, toString, getters e setters.

Exemplo Prático com Lombok

1. Código sem Lombok:

```
class Produto {  
    private int id;  
    private String nome;  
  
    @Override  
    public boolean equals(Object obj) { /* implementação */ }  
    @Override  
    public int hashCode() { /* implementação */ }  
  
    // getters e setters  
}
```

2. Código com Lombok:

```
@Data  
class Produto {  
    private int id;  
    private String nome;  
}
```

Usando `@Data`, Lombok gera todos os métodos necessários, incluindo `equals` e `hashCode`, simplificando o desenvolvimento

6. Vantagens e Desvantagens de Usar Lombok para equals e hashCode

- Vantagens:
 - Reduz código boilerplate e melhora a legibilidade.
 - Facilita a manutenção e promove a produtividade.
- Desvantagens:
 - Depende de uma biblioteca externa.
 - Pode dificultar a depuração, pois os métodos gerados automaticamente nem sempre são exibidos diretamente no código.
- Em alguns casos, a geração automática pode não atender a necessidades específicas da aplicação.

CONCLUSÃO

A correta implementação de `equals` e `hashCode` é crucial para o funcionamento eficiente de coleções e frameworks em Java. Lombok simplifica essa implementação, ajudando a manter o código enxuto e mais fácil de gerenciar. No entanto, o uso de Lombok exige cautela em projetos maiores devido à dependência de uma biblioteca externa e possíveis dificuldades na depuração. No final, combinar uma boa implementação de `equals` e `hashCode` com Lombok permite desenvolver aplicações Java mais robustas e escaláveis.

BIBLIOGRAFIA

Project Lombok: <https://projectlombok.org/>

Introduction to Project Lombok: <https://www.baeldung.com/intro-to-project-lombok>

Dev media: <https://www.devmedia.com.br/sobrescrevendo-o-metodo-equals-em-java/26484>

Baeldung: [https://www.baeldung.com/javahashcode#:~:text=Simply%20put%2C%20hashCode\(\)%20returns,to%20return%20different%20hash%20codes.](https://www.baeldung.com/javahashcode#:~:text=Simply%20put%2C%20hashCode()%20returns,to%20return%20different%20hash%20codes.)