****

**Serviço Nacional de Aprendizagem Industrial**

**SENAI “Gaspar Ricardo Junior”**

Curso

TÉCNICO EM DESENVOLVIMENTO

DE SISTEMAS

# Métodos equals e hashCode em Java

Isabely Rodrigues Kusmitsch Marcelino

Sorocaba

Novembro– 2024



**Serviço Nacional de Aprendizagem Industrial**

**SENAI “Gaspar Ricardo Junior”**

Isabely Rodrigues Kusmitsch Marcelino

# Métodos equals e hashCode em Java

Pesquisa sobre métodos equals e hashCode e uso da biblioteca Lombok

Prof. – Emerson Magalhães

Sorocaba

Novembro - 2024

**Sumário**

[Métodos equals e hashCode em Java 1](#_Toc182205531)

[Métodos equals e hashCode em Java 2](#_Toc182205532)

[INTRODUÇÃO 4](#_Toc182205533)

[1. Fundamentos Teóricos 5](#_Toc182205534)

[2. Introdução à biblioteca Lombok 7](#_Toc182205535)

[2.1. Análise das anotações @EqualsAndHashCode e @Data 9](#_Toc182205536)

[2.1.1. @EqualsAndHashCode 9](#_Toc182205537)

[2.1.2. @Data 10](#_Toc182205538)

[2.2. Exemplo prático de implementação de uma entidade com Lombok, comparando com uma implementação manual 11](#_Toc182205539)

[2.3. Vantagens e Desvantagens de Usar Lombok para equals e hashCode 12](#_Toc182205540)

[2.3.1. Vantagens 12](#_Toc182205541)

[2.3.2. Desvantagens 13](#_Toc182205542)

[CONCLUSÃO 14](#_Toc182205543)

[BIBLIOGRAFIA 15](#_Toc182205544)

[LISTA DE FIGURAS 16](#_Toc182205545)

# 

# INTRODUÇÃO

O método hashCode é de extrema importância para a formação de tabelas de dispersão (empalhamento), cujo dados são armazenados de acordo com o cálculo de um número hash, a partir das especificações de determinada informação, ou seja, é muito simples procurar um dado especifico em meio a muitos outros dados. Assim, torna-se fácil recuperar essas informações, pois mesmo que os dados colidam, o espaço de busca é reduzido.

Quando há a colisão dos dados, ou seja, quando duas ou mais informações tem características semelhantes, é utilizado o método equals. O método equals consiste em basicamente determinar se esses objetos são iguais e se possuem o mesmo valor dentre seus atributos congêneres.

Esses métodos são de suma importância para coleções e frameworks, pois os mesmos dependem dessa comparação entre dados para que a operação a ser realizada seja feita de forma mais eficiente possível, e vale a pena mencionar que ambos serão mais funcionais ao serem utilizados juntos.

Um exemplo seria o uso desses métodos no Spring Data, cuja necessidade de realizar comparações de forma simples entre determinadas informações de diferentes objetos (entidades) é muito comum.

# Fundamentos Teóricos

Algumas das regras que governam a implementação dos métodos equal e hashCode são: quando objetos são iguais eles devem apresentar o mesmo valor em seu hashCode, o método equals deve sempre retornar “true” e em caso de comparação, se um objeto retorna “true” o outro também deve retornar esse valor. Caso o equals retorne o valor “false”, é indicado que os valores entre os objetos sejam distintos, e o valor que foi retornado pelo hashCode deve ser o mesmo enquanto o objeto existir. Caso seja modificado de modo a alterar de forma conjunta o equals e hashCode, a possibilidade de o dado em questão ficar inacessível e ser de difícil conhecimento sua localização entre as coleções e frameworks.

Ademais métodos equals e hashCode são importantes para o comportamento das coleções, pois é por meio deles que um objeto pode ser adicionado a um hashSet e caso exista algum outro dado de valor semelhante pode ser utilizado o método equals para garantir que não se trata do mesmo objeto.

Em outras palavras, caso sejam adicionados dois objetos considerados de mesmo valor pelo equals, só sera permitido a duplicação em caso de concordância entre hashCode e o equals, e não exclusivamente pelo hashSet.

Para o hashMap, o hashCode é utilizado como um localizador para identificar se o objeto retornado é de fato o objeto requisitado. Assim, se houver dois localizadores com o mesmo valor em seu hashCode, mas com diferentes características é usado o método equals para garantir que se trata de localizadores de valor igual realmente.

Assim, é de extrema importância que a implementação de equals e hashCode em entidades de aplicações Java seja feita de maneira correta, pois trata-se de um fator fundamental para a integridade das informações sejam preservadas em contexto de coleções e frameworks como o Spring Data.

Diversas coleções são integralmente dependentes de tais métodos para armazenar e realizar a comparação de determinados objetos de forma conveniente.

Um exemplo dessa má implementação seria a duplicação de objetos ou localização incorreta do mesmo, já que o hashCode é responsável por determinar o local no qual o objeto será armazenado e o equals por realizar essa comparação de mesmos valores de hashCode.

Quando esses métodos são utilizados, a comparação das informações pode acabar por tornar-se dependente de tais. Se esses não forem implementados de forma correta, ocorrerá certa dificuldade no momento de manusear tais dependências. Também há o risco de **dados acabarem se cruzando** o que torna o processo lento e menos eficiente.

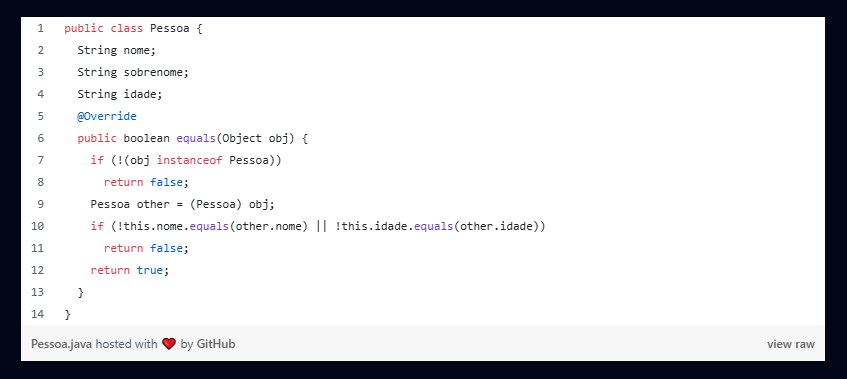


FIGURA 1- EXEMPLO EQUALS



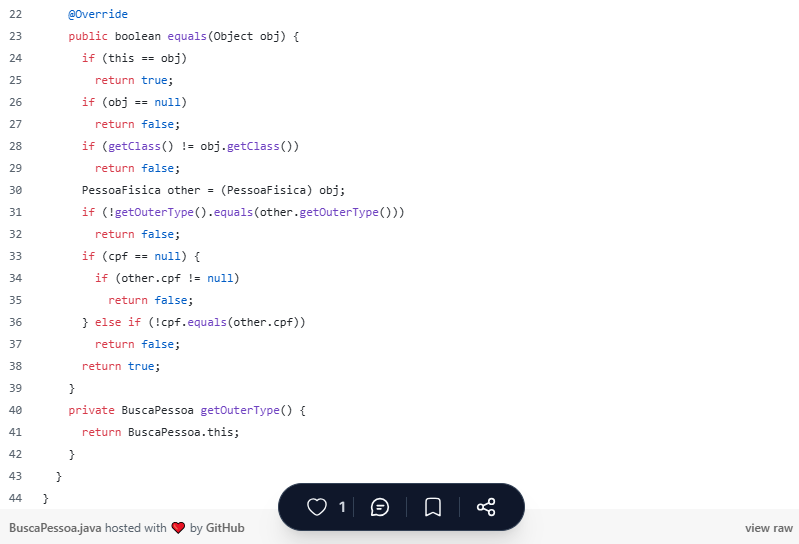


FIGURA 2- EXEMPLO HASHCODE

# Introdução à biblioteca Lombok

O Lombok é uma biblioteca Java poderosa que ajuda e permite aumentar a produtividade e reduzir o "código padrão" que possui linhas que acabam se repetindo durante o projeto. A partir dessa biblioteca, você pode adicionar anotações ao seu código e deixar que ela faça o trabalho, gerando automaticamente métodos essenciais, como *getters* , *setters* , *equals* , *hashCode* , além de métodos construtores, ajudando a poupar tempo e transformando o código em algo mais limpo e fácil de mexer.

Usar o Lombok pode auxiliar e ajudar o programador a otimizar tempo e deixar sua programação mais organizada e limpa uma vez que, você reduz a quantidade de código que precisa digitar, focando mais na lógica e funcionalidades de sua classe. Além disso, o Lombok ajuda a evitar erros simples que ocorrem ao escrever métodos como *equals* e *hashCode* manualmente que pode levar a problemas de desempenho e dificultar na procura desses erros.

Ademais, para começar a utilizar o Lombok, é preciso apenas colocá-lo como dependência em seu projeto Java, seja via *Maven* ou *Gradle*. Essa configuração inicial é simples e traz muitos benefícios ao longo do desenvolvimento, especialmente em projetos grandes, onde a manutenção de código se torna mais difícil.

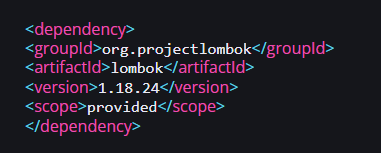


FIGURA 3- EXEMPLO MAVEN

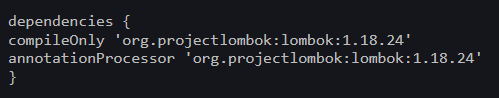


FIGURA 4- EXEMPLO GADLE

## Análise das anotações @EqualsAndHashCode e @Data

### @EqualsAndHashCode

A anotação @EqualsAndHashCode é uma das anotações do presentes na biblioteca do lombok, ela é uma das mais úteis para simplificar a implementação dos métodos equals() (compara dois objetos ) e hashCode() (retorna um valor inteiro que represente o objeto) em uma classe Java. Ao escrever esses métodos, é necessário garantir que sigam corretamente o contrato de equals e hashCode para evitar problemas em coleções que utilizam hashing, como HashMap e HashSet. Entretanto, pode-se simplesmente adicionar a anotação @EqualsAndHashCode à classe, e o Lombok se encarrega de gerar os métodos. Em resumo, a anotação @EqualsAndHashCode do Lombok é uma ferramenta útil para simplificar a implementação dos métodos equals() e hashCode() em uma classe Java. Com essa anotação, podemos deixar o Lombok se carregar de gerar os métodos para nós, tornando o desenvolvimento muito mais fácil e produtiva.

A tabela pessoa possui métodos equals() e hashCode() que são gerados automaticamente pelo Lombok, baseados nos campos id e nome.

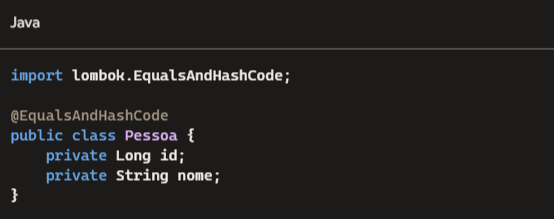


FIGURA 5- EXEMPLO @EqualsAndHashCode

### @Data

A anotação @Data é usada para gerar automaticamente os métodos getters, setters, equals(), hashCode() e toString() de uma classe Java. Isso significa que, ao invés de escrever manualmente esses métodos repetitivos, você pode simplesmente adicionar a anotação @Data à sua classe, e o Lombok se encarregará de gerar esses métodos para você. Assim, a anotação @Data substitui a necessidade de escrever manualmente cada método, reduzindo a quantidade de código escrito. Utilizar a anotação @Data facilita e agiliza o trabalho de manutenção de desenvolvimento do código uma vez que deixa o código mais fácil de entender diminuindo a possibilidade de erros que possam atrasar o processo de desenvolvimento.

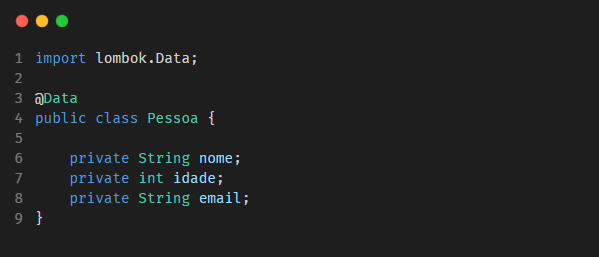
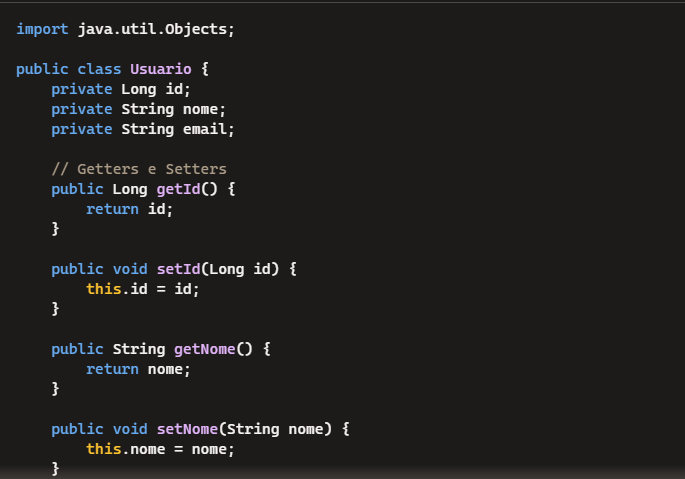
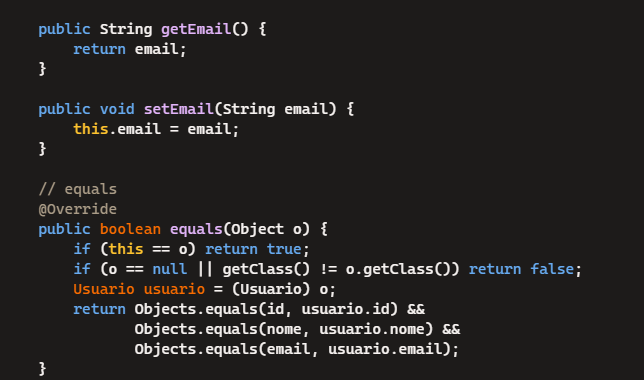


FIGURA 6- EXEMPLO @Data

A anotação @Data fará com que o Lombok gere automaticamente os métodos getters, setters, equals(), hashCode() e toString() para a classe Pessoa.

## Exemplo prático de implementação de uma entidade com Lombok, comparando com uma implementação manual





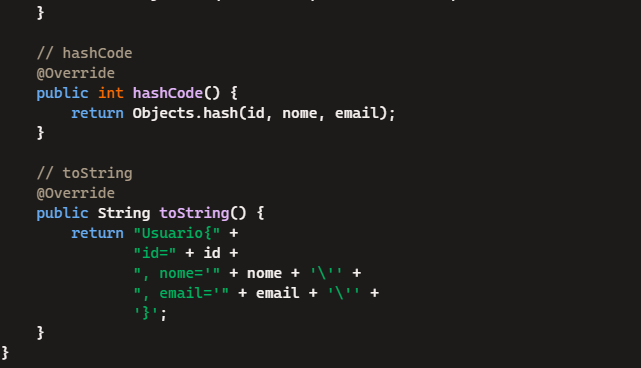


FIGURA 7- EXEMPLO DE IMPLEMENTAÇÃO MANUAL

Como já foi visto no desenvolvimento em Java, existem duas formas distintas de implementar métodos comuns: escrever esses métodos manualmente ou usar anotações do Lombok. Cada abordagem tem suas vantagens e desvantagens.

Ao optar por escrever manualmente o código, há a vantagem de ter controle total sobre a implementação dos métodos, permitindo personalizar conforme necessário. Além disso, essa abordagem não depende de bibliotecas externas, o que pode ser um ponto positivo em ambientes com restrições de dependências. Outra vantagem é a flexibilidade para adaptar os métodos às necessidades específicas do projeto. Entretanto, essa prática pode resultar em código extenso e repetitivo, o que dificulta a identificação de erros e a modificação do código, tornando o desenvolvimento mais demorado e aumentando a chance de erros.

Por outro lado, utilizar as anotações do Lombok diminui significativamente a quantidade de código que precisa ser escrito, reduzindo as repetições e tornando o código mais limpo e fácil de ler. Isso facilita a compreensão e a manutenção. Além disso, essa abordagem garante que os métodos gerados sigam um padrão, reduzindo a possibilidade de erros simples. No entanto, a utilização da biblioteca pode complicar a depuração (debugging) e a modificação do código gerado automaticamente, especialmente para desenvolvedores que não estão familiarizados com o Lombok.

## Vantagens e Desvantagens de Usar Lombok para equals e hashCode

### Vantagens

**Redução de Código Boilerplate**: Muitas vezes durante a programação se faz necessário o uso de métodos repetitivos, como getters, setters, construtores entre outros. Isto é conhecido como "código boilerplate". A biblioteca Lombok permite eliminar a necessidade de utilizar esses deixando o código mais limpo e focado no que realmente importa.

**Maior Produtividade**: A utilização dessa biblioteca possibilita perder menos enos tempo escrevendo códigos repetitivos deixando mais tempo focar na lógica do negócio, aumentando a produtividade, especialmente em projetos grandes.

**Melhor legibilidade e manutenção do código:** Com Lombok, o código fica mais curto, limpo e fácil de ler. Isso permite uma melhor visão do código pelos desenvolvedores possibilitando a ocorrência de menos erros e permitindo a manutenção do código de forma mais rápida e eficaz.

### Desvantagens

**Dependência Externa:** Adicionar o Lombok ao seu projeto significa depender de uma biblioteca externa para funcionalidades que poderiam ser implementadas diretamente em Java. Isso pode ser um problema em projetos que possuem restrições quanto ao uso de bibliotecas de terceiros.

**Dificuldade no Debugging:** Um problema comum com o Lombok é que ele gera código que não é visível diretamente no código-fonte. Isso pode dificultar a depuração, pois o código gerado só aparece após a compilação, tornando mais difícil encontrar e corrigir erros.

**Boas Práticas de Uso de Lombok em Ambientes de Produção:** Para desenvolvedores que não estão familiarizados com o Lombok, a legibilidade do código pode ser comprometida. Métodos gerados automaticamente não aparecem no código-fonte, o que pode dificultar a compreensão para novos desenvolvedores.

## CONCLUSÃO

Em conclusão, a implementação adequada dos métodos hashCode e equals é crucial para a comparação eficiente de objetos em Java. Quando esses métodos são mal implementados, podem surgir problemas como a duplicação de objetos ou a localização incorreta deles na memória, uma vez que o hashCode determina onde um objeto é armazenado e o equals compara objetos com valores de hashCode iguais. Essa dependência da comparação pode tornar o manuseio de objetos problemático, levando a dificuldades na gestão de dados e a riscos de cruzamento de informações. Como resultado, o desempenho do sistema pode ser comprometido, tornando o processo mais lento e menos eficiente.

Ademais, a utilização de bibliotecas externas como o Lombok oferece benefícios significativos, como a simplificação do código e a redução de boilerplate, permitindo que os desenvolvedores se concentrem na lógica de negócios e acelerem o desenvolvimento. No entanto, a geração automática de código pode complicar o debugging, já que o código gerado não é visível, dificultando a identificação de erros, especialmente em ambientes de produção. Além disso, a legibilidade do código pode ser afetada para desenvolvedores não familiarizados com o Lombok, criando barreiras para novos desenvolvedores. Assim mesmo existindo desvantagens a utilização da biblioteca Lombok é muito útil para o desenvolvimento de projetos tornando mais fácil e simples seua criação.

# BIBLIOGRAFIA

ANGELISKI, R. **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: 11 nov. 2024.

NORMANDES. **Entendendo o equals e o hashCode**. Disponível em: <https://blog.algaworks.com/entendendo-o-equals-e-hashcode/>. Acesso em: 11 nov. 2024.

BOMFIM, F. **Como Lombok Pode Transformar Seu Código Java**. Disponível em: <https://www.dio.me/articles/como-lombok-pode-transformar-seu-codigo-java>. Acesso em: 11 nov. 2024.

OBJECT ORIENTED DESIGN. **Quais são os benefícios e desvantagens de usar o 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: 11 nov. 2024.

LIMA, M. **Entenda as Annotations do Lombok, as 4 annotations avançadas que vão facilitar seu trabalho**. **Artefato X**, 27 mar. 2023. Disponível em: <https://artefatox.com/annotations-do-lombok/>. Acesso em: 11 nov. 2024

SOUZA, D. **Uma visão sobre o Projeto Lombok**. Disponível em: <https://www.devmedia.com.br/uma-visao-sobre-o-projeto-lombok/28321>. Acesso em: 11 nov. 2024.

**Introdução ao Lombok**. Disponível em: <https://rdrblog.com.br/java/introducao-ao-lombok/>. Acesso em: 11 nov. 2024.

# LISTA DE FIGURAS

FIGURA 1- EXEMPLO EQUALS

FIGURA 2- EXEMPLO HASHCODE

FIGURA 3- EXEMPLO MAVEN

FIGURA 4- EXEMPLO GADLE

FIGURA 5- EXEMPLO @EqualsAndHashCode

FIGURA 6- EXEMPLO @Data

FIGURA 7- EXEMPLO DE IMPLEMENTAÇÃO MANUAL