A Importância dos Métodos `equals` e `hashCode` em Java (Com Ajuda do Lombok)

Para quem programa em Java, entender e usar os métodos `equals` e `hashCode` é essencial, especialmente quando trabalhamos com coleções como `HashMap` e `HashSet`. Eles ajudam o Java a comparar e armazenar objetos de forma eficiente. Esse resumo explica por que esses métodos são importantes, como eles funcionam juntos e como o Lombok pode facilitar a vida dos desenvolvedores.

1. O |equals| e |hashCode|

O que é

O contrato entre `equals` e `hashCode` é um conjunto de regras que garantem que objetos iguais (segundo o método `equals`) também tenham o mesmo valor de `hashCode`.

Por que é importante

Sem uma implementação correta, um objeto que foi adicionado a uma coleção pode acabar "invisível" ou duplicado porque o Java depende desses métodos para localizar e organizar objetos. Em coleções como `HashSet` e `HashMap`, o `hashCode` decide onde o objeto vai ser armazenado, e o `equals` confirma se dois objetos são realmente iguais.

Impacto nos Frameworks

Em frameworks como o Spring, esses métodos são essenciais para operações de caching e persistência. Sem `equals` e `hashCode` corretos, o Spring pode ter problemas para reconhecer se uma entidade já foi armazenada ou se é um novo objeto.

2. Exemplos de Uso com Coleções e Spring

Coleções Java (`HashSet` e `HashMap`)

No `HashSet`, o `hashCode` ajuda a encontrar o local exato do objeto, e `equals` garante que não haverá duplicatas. Já no `HashMap`, o `hashCode` é usado para encontrar o "bucket" correto, e o `equals` para confirmar a chave exata. Isso torna o processo rápido e eficiente.

Exemplo com Spring

No Spring, essas regras ajudam no gerenciamento de entidades, especialmente em operações de caching e persistência de dados. Implementar `equals` e `hashCode` corretamente garante que o Spring não duplique entidades acidentalmente ou perca o vínculo entre objetos.

3. Lombok: Simplificando `equals` e `hashCode`

O que é Lombok

Lombok é uma biblioteca Java que gera automaticamente métodos como `equals`, `hashCode`, `toString`, e outros métodos repetitivos. Isso ajuda a reduzir o código repetitivo (boilerplate), deixando o código mais direto e fácil de entender.

Anotações Importantes

- `@EqualsAndHashCode`: Gera automaticamente `equals` e `hashCode` com base nos campos da classe.
- `@Data`: Essa anotação combina `@EqualsAndHashCode`, `@Getter`, `@Setter`, e outros, automatizando a criação de métodos básicos.

Comparação com Implementação Manual

Em uma classe que representa uma entidade, o Lombok com `@Data` ou `@EqualsAndHashCode` reduz muito o trabalho manual. Sem o Lombok, o desenvolvedor teria que escrever esses métodos do zero, o que é trabalhoso e sujeito a erros.

4. Vantagens e Desvantagens do Lombok

- **Vantagens:**
  - Redução de Código Repetitivo: O Lombok ajuda a manter o código mais limpo e direto, sem métodos desnecessários.
  - Melhor Legibilidade e Manutenção: Código mais curto e claro é mais fácil de ler e de dar manutenção.

- Desvantagens:

- Dependência Externa: Lombok é uma biblioteca extra que adiciona dependências ao projeto.
- Debugging Mais Difícil: Como o Lombok gera código automaticamente, isso pode dificultar o processo de depuração, especialmente em ambientes de produção.

5. Conclusão

O uso correto de `equals` e `hashCode` é essencial para coleções e frameworks em Java, garantindo que o gerenciamento de objetos seja eficiente. Lombok facilita muito esse processo, ajudando a reduzir o código repetitivo e melhorando a legibilidade. No entanto, é importante equilibrar suas vantagens e desvantagens, especialmente em ambientes de produção.

6. EX(img):

Simplificando `equals`, `hashCode`, `HashSet` e `HashMap`;

Vamos criar a classe abaixo:

```
ublic class Produto {

 private String sku;
 private String nome;

 public Produto(String sku, String nome) {
   this.sku = sku;
   this.nome = nome;
 }

 // getters e setters

 @Override
 public String toString() {
   return "Produto [sku=" + sku + ", nome=" + nome + "]";
 }
```

1;

```
public class CadastradorProdutos {

  public static void main(String[] args) {
    Collection produtos = new ArrayList<>();

    System.out.println("##### Cadastro de produtos #####\n

    try (Scanner entrada = new Scanner(System.in)) {
      String continuar = "s";
      while ("s".equalsIgnoreCase(continuar)) {
        System.out.print("SKU: ");
        String sku = entrada.nextLine();

        System.out.print("Nome: ");
        String nome = entrada.nextLine();

        Produto produto = new Produto(sku, nome);
        if (produtos.contains(produto)) {
          System.err.println("Esse produto já foi adiciona
        } else {
          produtos.add(produto);
          System.out.println("Produto adicionado.");
        }

        System.out.print("Deseja adicionar mais algum prod

        continuar = entrada.nextLine();
      }
```

2;

Para ensinar a classe `Produto` a comparar dois objetos com o *sku*, vamos sobrescrever o método `equals()` com o código abaixo:

```
1  public boolean equals(Object obj) {
2    Produto outro = (Produto) obj;
3    return this.sku.equals(outro.getSku());
4  }
```

3;

Vamos fazer uma alteração na nossa classe de teste e usar um `HashSet` no lugar do `ArrayList`.

```
Collection produtos = new HashSet<>();
```

4;

Vamos criar um código baseado na primeira letra do sku. Veja o método `hashCode` abaixo:

```
public int hashCode() {
  return this.sku.charAt(0);
}
```

5;

HashMap;

```
// Criando um HashMap de nomes e idades
Map<String, Integer> mapaDeIdades = new HashMap<>();

// Adicionando pares chave-valor ao HashMap11
mapaDeIdades.put("Alice", 25);
mapaDeIdades.put("Bob", 30);
mapaDeIdades.put("Charlie", 28);

// Acessando valores usando chave
int idadeDaAlice = mapaDeIdades.get("Alice");
System.out.println("Idade da Alice: " + idadeDaAlice);

boolean temChaveBob = mapaDeIdades.containsKey("Bob");
System.out.println("Tem a chave Bob? " + temChaveBob);

mapaDeIdades.remove("Charlie");
System.out.println("Tamanho do HashMap: " + mapaDeIdades.size());
```

Fontes:

- IA.
- https://rdrblog.com.br/java/introducao-ao-lombok/#:~:text=O%20Lombok%20%C3%A9%20uma%20ferramenta,depend%C3%AAncia%20de%20uma%20biblioteca%20externa.
- https://jdevtreinamento.com.br/metodos-equals-e-hashcode/#:~:text=Tal%20compara%C3%A7%C3%A3o%20tem%20a%20finalidade,tabela%20hash%20de%20modo%20correto.
- https://blog.geekhunter.com.br/spring-framework/
- https://blog.algaworks.com/entendendo-o-equals-e-hashcode/