HashCode

Profa. Karen Selbach Borges

Última Atualização: Jun/2021

Entenda o Hash

- Uma função hash é um procedimento bem definido ou função matemática que converte uma quantidade variável (possivelmente grande) de dados em um pequeno dado, muitas vezes um inteiro, usado como índice.

Entenda o HashValue

- Se soubermos o *hash value* de um objeto, então saberemos também sua exata posição na memória.
- Não se pode garantir que objetos, que tenham *hash values* distintos, são necessariamente **diferentes.**

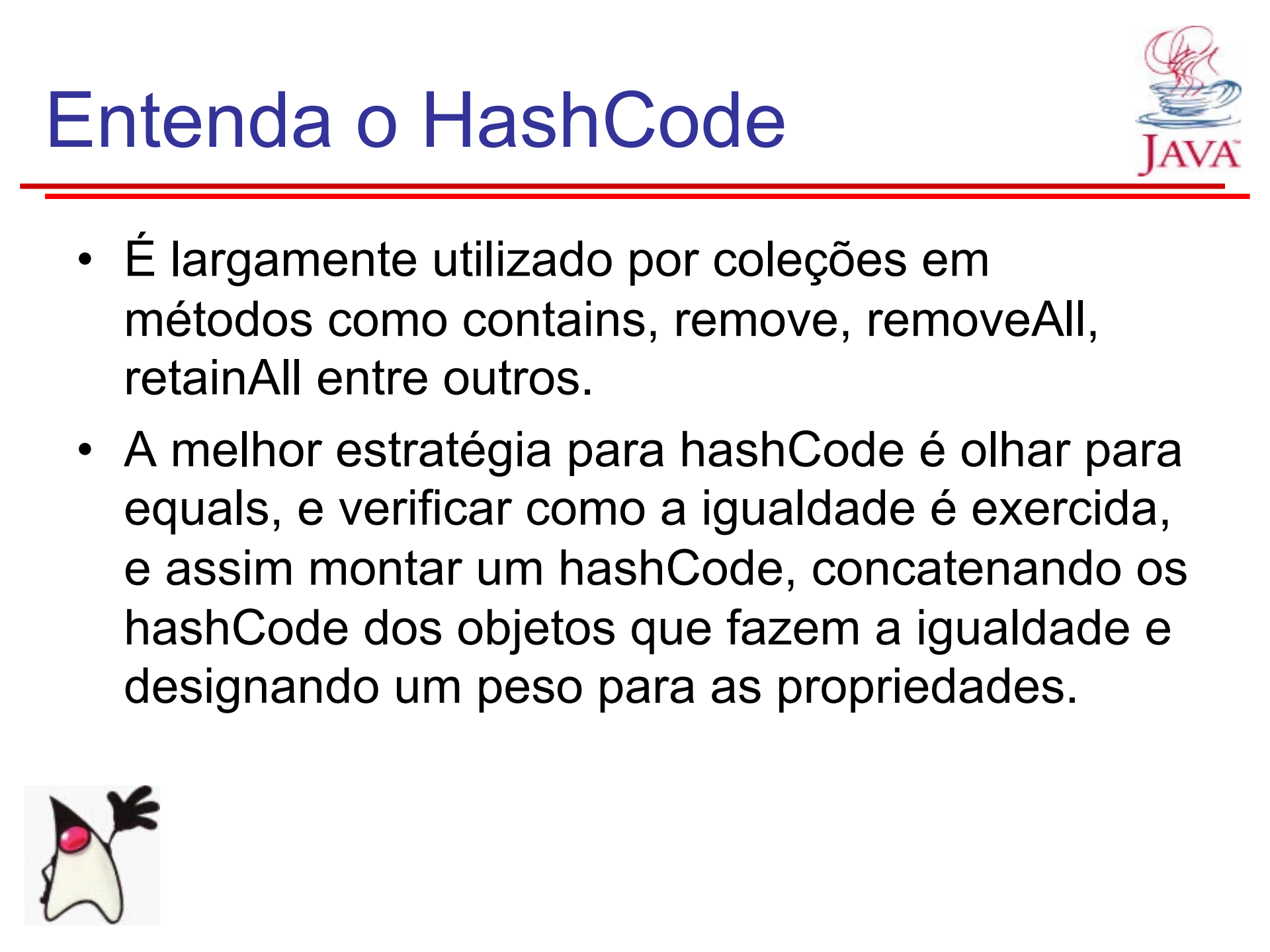
Entenda o HashValue

- O que é ser diferente ?
  - É não ser igual
  - Quem determina isso é o método equals.
  - Se a forma, como um objeto é comparado a outro, mudou, então a captura do *hash value*, desse objeto, também deve mudar.
  - Resumindo: se você sobrescreveu o método equals(), também deve fazê-lo com o hashCode() e vise-versa.

Entenda o HashCode

- O método hashCode segue o contrato descrito abaixo:
  1. **É constante**: múltiplas invocações, ao hashCode() de um determinado objeto, devem retornar sempre o mesmo valor, considerando que o objeto não foi modificado para efeito do método equals(). Por exemplo, se o método equals() considera que uma pessoa é igual a outra se seus nomes forem iguais, então uma boa implementação do hashCode() seria retornar o *hash code* do atributo em questão (nome).
  2. **Objetos iguais, *hash values* iguais**: Se dois objetos são iguais, conforme o método equals(), então esses objetos obrigatoriamente devem possuir o mesmo *hash value*.
  3. **Objetos distintos, sem regras para *hash values***: Se dois objetos são diferentes, conforme método equals(), seus *hash values* podem ser diferentes.

Entenda o HashCode



- É largamente utilizado por coleções em métodos como contains, remove, removeAll, retainAll entre outros.
- A melhor estratégia para hashCode é olhar para equals, e verificar como a igualdade é exercida, e assim montar um hashCode, concatenando os hashCode dos objetos que fazem a igualdade e designando um peso para as propriedades.

Valor de Hash

- Dada uma classe Pessoa

```
public class Pessoa {
        private String nome;
        private String sobrenome;

        public Pessoa(String nome, String sobrenome) {
                this.nome = nome;
                this.sobrenome = sobrenome;
        }

        public String toString(){
                return (nome + " " + sobrenome);
        }
}
```

Valor de Hash

```
public class TestaPessoa {

        public static void main(String[ ] args) {
                Pessoa fulano = new Pessoa("Pedro", "Rocha");
                Pessoa sicrano = new Pessoa("Pedro", "Rocha");

                System.out.println(fulano); // Pedro Rocha
                System.out.println(sicrano); // Pedro Rocha
                System.out.println(fulano.equals(sicrano)); // False ERRO !
                System.out.println(fulano.hashCode()); // 18464898
                System.out.println(sicrano.hashCode()); // 28168925
        }
}
```

Solução ...

- Sobrescrever o método equals:

```
public class Pessoa {
        ...
        public boolean equals(Object obj) {
                Pessoa outro = (Pessoa) obj;
                if (nome.equals(outro.nome)){
                        if (sobrenome.equals(outro.sobrenome)){
                                return true;
                        }
                }
                return false;
        }
}
```

Solução ...

- Fornecer uma implementação própria para o cálculo do hash code:

```
public class Pessoa {
        ...
        public int hashCode(){
                int codigo = 1;
                if (nome !=null) {
                        codigo = codigo*31+nome.hashCode();
                }
                if (sobrenome!=null) {
                        codigo = codigo*31+sobrenome.hashCode();
                }
                return codigo;
        }
}
```