O que é um Objeto Imutável?

Um **objeto imutável** é um objeto cujo estado não pode ser alterado após a sua criação. Isso significa que todos os seus campos ou atributos são definidos apenas uma vez e não podem ser modificados posteriormente. A imutabilidade é uma propriedade importante que ajuda a evitar erros de programação e a tornar o código mais fácil de entender e manter.

Características de Objetos Imutáveis:

- **Imutabilidade de Estado**: Uma vez que o objeto é criado, o seu estado não pode ser alterado. Qualquer tentativa de modificar o objeto resulta na criação de um novo objeto.
- **Segurança em Threads**: Objetos imutáveis são inerentemente seguros para uso em múltiplas threads, pois não há risco de que uma thread modifique o objeto enquanto outra está lendo ou escrevendo.
- **Facilidade de Raciocínio**: Como o estado do objeto não muda, o comportamento do código que o utiliza é mais previsível e fácil de entender.

Exemplos de Objetos Imutáveis em Java

Um dos exemplos mais comuns de um objeto imutável em Java é a classe `String`.

```
public class ObjetosImutaveis {
public static void main(String[] args) {
   String texto = "Olá";
   // Impressão do hash code de identidade do objeto String original
   System.out.println("Hash code original: " + System.identityHashCode(texto));
   // Concatenação que cria um novo objeto String
   texto = texto.concat(", mundo!");
    // Impressão do hash code de identidade do novo objeto String
    System.out.println("Hash code após concatenação: " + System.identityHashCode(texto));
   // Saída do novo valor da String
   System.out.println(texto); // Saída: "Olá, mundo!"
   String result = "";
   System.out.println("Hash code variável result antes for...next : "
   + System.identityHashCode(result));
   for (int i = 0; i < 5; i++) {
      result += i; // Cada operação de `+` cria um novo objeto `String`
      System.out.println("Hash code variável result : " + System.identityHashCode(result));
   }
   System.out.println(result); // Output: 01234
   }
}
```

No exemplo de código acima, quando fazemos `texto.concat(", mundo!")`, um novo objeto `String` é criado com o valor "Olá, mundo!". Na sequência realiza-se a concatenação de números inteiros em uma variável `String` em um laço `for...next`.

O método `System.identityHashCode` em Java retorna o hash code de um objeto. O hash code é um número inteiro que identifica unicamente um objeto na memória. Ele é usado para verificar se duas referências se referem ao mesmo objeto.

O resultado do console para esta execução de código será:

```
Hash code original: 1365202186
Hash code após concatenação: 212628335
Olá, mundo!
Hash code variável result antes for...next : 1579572132
Hash code variável result : 359023572
Hash code variável result : 305808283
Hash code variável result : 2111991224
Hash code variável result : 292938459
Hash code variável result : 917142466
01234
```

A diferença nos códigos hash confirma que a concatenação não modificou o objeto original texto, mas criou um novo objeto String com o novo valor.

Isso demonstra a imutabilidade das Strings em Java e como a linguagem lida com operações de manipulação de texto. No laço `for...next` também cinco objetos String temporários são criados, resultando em ineficiência em termos de memória e desempenho.

Uso do `StringBuilder`

`StringBuilder` é uma classe mutável para construção de strings. Ele é projetado especificamente para operações que envolvem muita manipulação de strings, como concatenação em loops.

Características do `StringBuilder`:

- **Mutável**: Diferente de `String`, ele pode ser modificado sem criar novos objetos.
- **Eficiente em Memória**: Todas as operações de manipulação ocorrem no mesmo objeto, evitando a criação de várias instâncias temporárias.

- **Desempenho**: `StringBuilder` é mais eficiente para operações repetitivas de concatenação.

Veja o código a seguir:

```
public class UsandoStringBuilder {
public static void main(String[] args) {
   StringBuilder sb = new StringBuilder();
   System.out.println("Hash code variável sb antes laço for...next : "
     + System.identityHashCode(sb));
   for (int i = 0; i < 5; i++) {
      sb.append(i); // Operação de `append` adiciona ao mesmo objeto
      System.out.println("Hash code variável sb : " + System.identityHashCode(sb));
   }
   System.out.println(sb.toString()); // Output: 01234
   }
}
```

Nesse exemplo, o `StringBuilder` modifica a mesma instância em cada iteração, resultando em um único objeto final com a string concatenada. O resultado em console:

```
Hash code variável sb antes laço for...next : 1365202186
Hash code variável sb : 1365202186
Hash code variável sb : 1365202186
Hash code variável sb : 1365202186
Hash code variável sb : 1365202186
Hash code variável sb : 1365202186
01234
```

Perceba que o hascode do objeto não se alterou, indicando que estamos lidando com o mesmo objeto durante o laço `for...next`.

O uso de `StringBuilder` em vez de concatenações repetidas com o operador `+` é uma prática recomendada em Java para garantir eficiência em termos de memória e desempenho. Essa abordagem é especialmente benéfica em contextos onde a manipulação de strings é frequente e extensa.