Cập nhật tháng 8 năm 2024

[Bài đọc] StringBuilder và StringBuffer

1. StringBuilder và StringBuffer là gì?

- **StringBuilder** và **StringBuffer** là hai lớp trong Java được sử dụng để tạo và thao tác với các chuỗi ký tự có thể thay đổi (**mutable strings**).
- Không giống như **String** (là immutable), các thay đổi trên **StringBuilder** và **StringBuffer** không tạo ra đối tượng mới trong bộ nhớ, giúp tiết kiệm tài nguyên và tăng hiệu suất khi thao tác nhiều trên chuỗi.

2. Điểm khác biệt giữa String, StringBuilder và StringBuffer

Thuộc tính	String	StringBuilder	StringBuffer
Khả năng thay đổi	Không thể thay đổi	Có thể thay đổi	Có thể thay đổi
Thread-safe	Không thread-safe	Không thread-safe	Thread-safe
Hiệu suất	Chậm hơn nếu thay đổi chuỗi thường xuyên	Nhanh hơn (không đồng bộ)	Chậm hơn (do đồng bộ)

3. Khi nào sử dụng?

- **StringBuilder:** Sử dụng khi cần thao tác nhiều trên chuỗi và không yêu cầu thread-safe (ví dụ: xử lý chuỗi trong ứng dụng đơn luồng).
- **StringBuffer:** Sử dụng khi cần thread-safe (ví dụ: ứng dụng đa luồng hoặc yêu cầu đồng bộ hóa).

4. Các phương thức phổ biến

Phương thức	Mô tả	Ví dụ
append(String s)	Thêm chuỗi vào cuối đối tượng StringBuilder/StringBuffer.	sb.append("world"); -> Kết quả: "Hello world"
insert(int offset, s)	Chèn chuỗi tại vị trí được chỉ định.	sb.insert(5, "Java"); -> Kết quả: "HelloJava"

replace(int start, int end, String s)	Thay thế chuỗi con từ start đến end bằng chuỗi mới.	sb.replace(5, 7, "JS"); -> Kết quả: "HelloJS"
delete(int start, int end)	Xóa chuỗi con từ start đến end.	sb.delete(5, 10); -> Kết quả: "Hello"
reverse()	Đảo ngược chuỗi hiện tại.	sb.reverse(); -> Kết quả: "olleH"
capacity()	Trả về dung lượng bộ nhớ hiện tại của StringBuilder/StringBuffer.	System.out.println(sb.capacity()); -> In ra dung lượng.
ensureCapacity(int min)	Đảm bảo dung lượng tối thiểu của bộ nhớ.	sb.ensureCapacity(50);

5. **Ví dụ minh họa**

```java
public static void main(String[] args) new *
{
    // Sử dụng StringBuilder
    StringBuilder sb = new StringBuilder("Hello");

    // Thêm chuỗi
    sb.append(" World");
    System.out.println("After append: " + sb);

    // Chèn chuỗi
    sb.insert( offset: 6, str: "Java ");
    System.out.println("After insert: " + sb);

    // Thay thế chuỗi
    sb.replace( start: 6, end: 10, str: "Python");
    System.out.println("After replace: " + sb);

    // Xóa chuỗi
    sb.delete(6, 12);
    System.out.println("After delete: " + sb);

    // Đảo ngược chuỗi
    sb.reverse();
    System.out.println("After reverse: " + sb);
}
```

Kết quả chạy chương trình:

```
After append: Hello World
After insert: Hello Java World
After replace: Hello Python World
After delete: Hello  World
After reverse: dlroW  olleH
```

6. **Ưu điểm và hạn chế**

- Ưu điểm:
  - Nhanh hơn **String** khi thực hiện các thao tác thay đổi chuỗi liên tục.
  - Linh hoạt trong việc quản lý bộ nhớ, giảm số lượng đối tượng được tạo ra.
- Hạn chế:
  - **StringBuffer** có hiệu suất thấp hơn **StringBuilder** do phải đồng bộ hóa.
  - Yêu cầu quản lý bộ nhớ cẩn thận khi làm việc với các chuỗi lớn.

7. **So sánh hiệu suất**

Ví dụ so sánh hiệu suất khi thực hiện thêm chuỗi liên tục:

```
public static void main(String[] args)  new *
{
    long startTime, endTime;

    // Test String
    String str = "";
    startTime = System.currentTimeMillis();
    for (int i = 0; i < 10000; i++)
    {
        str += "a";
    }
    endTime = System.currentTimeMillis();
    System.out.println("Time taken by String: " + (endTime - startTime) + "ms");

    // Test StringBuilder
    StringBuilder sb = new StringBuilder();
    startTime = System.currentTimeMillis();
    for (int i = 0; i < 10000; i++)
    {
        sb.append("a");
    }
    endTime = System.currentTimeMillis();
    System.out.println("Time taken by StringBuilder: " + (endTime - startTime) + "ms");

    // Test StringBuffer
    StringBuffer sbf = new StringBuffer();
    startTime = System.currentTimeMillis();
    for (int i = 0; i < 10000; i++)
    {
        sbf.append("a");
    }
    endTime = System.currentTimeMillis();
    System.out.println("Time taken by StringBuffer: " + (endTime - startTime) + "ms");
}
```

Kết quả:

```
Time taken by String: 14ms
Time taken by StringBuilder: 1ms
Time taken by StringBuffer: 0ms
```

Link tài nguyên đọc thêm:

- StringBuilder: https://www.geeksforgeeks.org/stringbuilder-class-in-java-with-examples/
- StringBuffer: https://www.geeksforgeeks.org/stringbuffer-class-in-java/

Danh sách các bài học