Cập nhật tháng 8 năm 2024

[Bài đọc] Giới thiệu về String

Trong Java, **String** là một kiểu dữ liệu đặc biệt được sử dụng để đại diện cho chuỗi các ký tự. String trong Java không phải là kiểu dữ liệu nguyên thủy mà là một đối tượng của lớp String trong thư viện chuẩn. Dưới đây là một số thông tin cơ bản về kiểu dữ liệu String trong Java:

1. **Khai báo và khởi tạo String:**

- **Cách khai báo String:** Để khai báo một biến kiểu String, bạn sử dụng từ khóa String theo cú pháp:

```
String str;
```

- **Cách khởi tạo String:** String có thể được khởi tạo theo 2 cách chính:
  - **Khởi tạo với giá trị chuỗi:**

```
String str = "Hello, world!";
```

Đây là cách thông dụng nhất để khởi tạo String. Khi bạn gán một chuỗi vào một biến String, Java tự động tạo ra một đối tượng String trong bộ nhớ.

- **Khởi tạo với từ khóa new:**

```
String str = new String( original: "Hello, world!");
```

Phương pháp này sẽ tạo một đối tượng String mới trong bộ nhớ, dù chuỗi đã tồn tại trong pool của String.

2. **Tính năng của String:**

- **String là bất biến (Immutable):** Điều này có nghĩa là khi một chuỗi String đã được khởi tạo, giá trị của nó không thể thay đổi. Nếu bạn thay đổi giá trị của một String, một đối tượng String mới sẽ được tạo ra.
- **String pool:** Java sử dụng một khu vực nhớ đặc biệt gọi là "String pool" để lưu trữ các chuỗi String. Khi bạn khai báo một chuỗi String với cú pháp "text", Java sẽ kiểm tra trong String pool xem chuỗi đó đã tồn tại chưa. Nếu tồn tại, Java sẽ sử dụng chuỗi đó, tránh tạo ra bản sao mới.

3. **Một số phương thức phổ biến của String:**

- **length()**: Trả về độ dài của chuỗi.

```
String str = "Hello";
int length = str.length();  // Kết quả: 5
```

- **charAt(int index)**: Trả về ký tự tại vị trí chỉ định.

```
String str = "Hello";
char ch = str.charAt(1);  // Kết quả: 'e'
```

- **substring(int start, int end)**: Trả về một phần chuỗi từ vị trí start đến end.

```
String str = "Hello";
String subStr = str.substring(0, 3);  // Kết quả: "Hel"
```

- **equals(String other)**: So sánh 2 chuỗi xem chúng có giống nhau không.

```
String str = "Hello";
boolean isEqual = str.equals("Hello");  // Kết quả: true
```

4. **Ví dụ về khai báo và khởi tạo String trong Java:**

```
public static void main(String[] args)  new *
{
    // Khai báo và khởi tạo String
    String greeting = "Hello, Java!";  // Khởi tạo bằng literal
    String name = new String( original: "John");  // Khởi tạo bằng từ khóa new

    // Sử dụng các phương thức String
    System.out.println("Greeting: " + greeting);
    System.out.println("Length of greeting: " + greeting.length());
    System.out.println("First character of name: " + name.charAt(0));
}
```

5. **Lưu ý khi sử dụng String:**

- **Chú ý về hiệu suất:** Vì String là đối tượng bất biến, việc thay đổi nhiều lần giá trị của String có thể gây tốn kém tài nguyên, đặc biệt trong các vòng lặp lớn. Trong trường hợp này, bạn có thể sử dụng lớp StringBuilder hoặc StringBuffer để tăng hiệu suất khi thay đổi chuỗi.

Link tài nguyên đọc thêm: https://www.geeksforgeeks.org/strings-in-java/

Danh sách các bài học