**String**

# 1. Tìm hiểu về các đặc điểm và tính chất của String trong java

Khái niệm cơ bản

* Định nghĩa: String trong Java là một chuỗi các ký tự. Nó được sử dụng để biểu diễn văn bản và là một trong những kiểu dữ liệu cơ bản và quan trọng nhất trong lập trình Java.
* Đặc điểm:
  + Không thay đổi: String là một đối tượng không thể thay đổi (immutable). Khi bạn thay đổi một chuỗi, thực chất là tạo ra một chuỗi mới và gán nó cho một biến khác.
  + Là một đối tượng: Mặc dù được sử dụng như một kiểu dữ liệu cơ bản, String thực chất là một đối tượng của lớp java.lang.String. Điều này có nghĩa là nó thừa hưởng các phương thức và thuộc tính từ lớp này.
  + Được bao quanh bởi dấu nháy kép: Các chuỗi trong Java luôn được bao quanh bởi dấu nháy kép (""). Ví dụ: String str = "Hello, world!";

Các tính chất quan trọng của String:

* Các phương thức: String cung cấp rất nhiều phương thức hữu ích để thao tác với chuỗi như:
  + **length()**: Trả về độ dài của chuỗi.
  + **charAt(index)**: Trả về ký tự tại vị trí index.
  + **concat(str)**: Nối hai chuỗi lại với nhau.
  + **substring(beginIndex, endIndex)**: Trả về một chuỗi con.
  + **indexOf(str)**: Trả về vị trí đầu tiên xuất hiện của chuỗi con str.
  + **toUpperCase()**: Chuyển đổi tất cả các ký tự thành chữ in hoa.
  + **toLowerCase()**: Chuyển đổi tất cả các ký tự thành chữ thường.
  + **trim()**: Loại bỏ khoảng trắng thừa ở đầu và cuối chuỗi.
  + **equals(str)**: So sánh nội dung của hai chuỗi.
  + **compareTo(str)**: So sánh hai chuỗi theo thứ tự từ điển.
* So sánh chuỗi:
  + **==**: So sánh hai tham chiếu, tức là kiểm tra xem hai biến có cùng trỏ đến cùng một đối tượng String hay không.
  + **equals()**: So sánh nội dung của hai chuỗi.
* Chuỗi rỗng và chuỗi null:
  + **Chuỗi rỗng**: Là một chuỗi có độ dài bằng 0, được biểu diễn bằng cặp dấu nháy kép trống (" ").
  + **Chuỗi null**: Là một tham chiếu không trỏ đến bất kỳ đối tượng nào.

# 2. Có bao nhiêu cách để tạo 1 biến String

Trong Java, có nhiều cách để tạo một biến String

## 1. Sử dụng String Literal

* Đây là cách phổ biến và đơn giản nhất để tạo một chuỗi. Khi bạn sử dụng cách này, chuỗi sẽ được lưu trong String Pool.
* Nếu một chuỗi giống hệt đã tồn tại trong String Pool, Java sẽ sử dụng lại chuỗi đó thay vì tạo ra một chuỗi mới.

## 2. Sử dụng Constructor của lớp String

* Bạn cũng có thể sử dụng từ khóa new để tạo một đối tượng String. Khi sử dụng cách này, chuỗi sẽ được tạo trong **heap memory**, không phải trong String Pool.
* Mặc dù chuỗi này có cùng nội dung với chuỗi được tạo bằng literal, nhưng nó là một đối tượng khác trong bộ nhớ.

## 3. Tạo chuỗi từ StringBuilder hoặc StringBuffer

* StringBuilder và StringBuffer là các lớp có khả năng thay đổi nội dung chuỗi mà không tạo ra đối tượng mới sau mỗi thao tác. Sau khi xây dựng chuỗi,có thể chuyển nó thành đối tượng String bằng phương thức toString().

4. Sử dụng các phương thức tĩnh từ các lớp khác

* Từ số nguyên (int), số thực (double) hoặc các kiểu nguyên thủy khác
* Từ các đối tượng như Object hay Boolean

# 3. Tìm hiểu về String pool?

String pool là một vùng nhớ đặc biệt nằm trong vùng heap của JVM, được sử dụng để lưu trữ các đối tượng String. Mục đích chính của string pool là tối ưu hóa việc sử dụng bộ nhớ và tăng hiệu suất khi làm việc với các chuỗi.

**Khi khai báo một chuỗi bằng cách gán trực tiếp một chuỗi literal vào một biến:**

* JVM sẽ kiểm tra xem chuỗi đó đã tồn tại trong string pool chưa.
* Nếu chưa có, chuỗi sẽ được thêm vào string pool và biến sẽ tham chiếu đến chuỗi này.
* Nếu đã có, biến sẽ tham chiếu đến chuỗi đã tồn tại trong pool, tránh tạo ra một đối tượng String mới.

# 4. Sự khác biệt giữa String và StringBuilder/StringBuffer?

Sự khác biệt chính giữa **String**, **StringBuilder**, và **StringBuffer** trong Java nằm ở cách chúng quản lý bộ nhớ, tính bất biến (immutable) và hiệu suất khi xử lý các chuỗi.

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuộc tính** | **String** | **StringBuilder** | **StringBuffer** |
| Tính bất biến | Bất biến (Immutable) | Có thể thay đổi (Mutable) | Có thể thay đổi (Mutable) |
| Đồng bộ hóa | Không | Không | Có |
| An toàn luồng | Không | Không | Có |
| Hiệu suất | Chậm nhất do tạo đối tượng mới mỗi lần thay đổi | Nhanh nhất khi xử lý chuỗi | Chậm hơn StringBuilder vì có cơ chế đồng bộ hóa |
| Sử dụng khi nào | Khi chuỗi không thay đổi, hoặc cần tái sử dụng từ String Pool | Khi cần thay đổi chuỗi thường xuyên trong môi trường đơn luồng | Khi cần thay đổi chuỗi trong môi trường đa luồng (an toàn luồng) |
| Nơi lưu trữ chuỗi | String Pool hoặc Heap | Heap | Heap |
| Độ dài khởi tạo | Cố định (không thể thay đổi) | Có thể thay đổi | Có thể thay đổi |
| Phương thức sửa đổi | Tạo đối tượng mới khi thay đổi chuỗi | Sửa đổi trực tiếp mà không tạo đối tượng mới | Sửa đổi trực tiếp mà không tạo đối tượng mới |

# 5. Sự khác biệt giữa == và .equals() khi so sánh chuỗi?

|  |  |  |
| --- | --- | --- |
| So sánh | == | .equals() |
| Cách so sánh | So sánh tham chiếu (địa chỉ bộ nhớ) | So sánh nội dung (giá trị ký tự) |
| Thường sử dụng khi | Kiểm tra xem hai biến có cùng trỏ đến một vùng nhớ không | Kiểm tra giá trị của hai chuỗi có giống nhau không |