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

Trong Java String là một lớp đặc biệt sử dụng để biểu diễn các chuỗi ký tự.

String có những đặc điểm và tính chất sau :

* **Tính bất biến** : Một khi một đối tượng String được tạo, nội dung của nó không thể thay đổi. Nếu bạn cố gắng thay đổi chuỗi, một đối tượng mới sẽ được tạo ra thay vì thay đổi đối tượng ban đầu.
* **Thread-Safe** : Do String là bất biến, bạn có thể an tâm sử dụng nó trong nhiều luồng mà không sợ bị thay đổi giá trị .
* **Cung cấp nhiều phương thức hữu ích :** Lớp String cung cấp rất nhiều phương thức hữu ích để thao tác với chuỗi như :
  + length(): Trả về độ dài của chuỗi.
  + charAt(int index): Lấy ký tự tại vị trí chỉ định.
  + substring(int beginIndex, int endIndex): Trả về một chuỗi con.
  + indexOf(String str): Tìm vị trí của chuỗi con.
  + toLowerCase(), toUpperCase(): Chuyển đổi chuỗi thành chữ thường hoặc chữ in hoa.
  + equals(), equalsIgnoreCase(): So sánh hai chuỗi.
  + trim(): Xóa khoảng trắng ở đầu và cuối chuỗi.
* **Hỗ trợ biểu thức chính quy** : Lớp String hỗ trợ các biểu thức chính quy thông qua các phương thức như matches(), replaceAll(), split(), giúp việc xử lý chuỗi dễ dàng hơn.
* Lưu trữ tại String pool .

1. Tìm hiểu về String pool?

* Java sử dụng một khái niệm gọi là **String Pool**, một khu vực đặc biệt trong heap memory để lưu trữ các đối tượng String khi chúng được tạo ra bằng các ký tự chuỗi (string literals).
* String Pool giúp tối ưu hóa việc lưu trữ chuỗi nhằm tiết kiệm bộ nhớ .
* **Cơ chế hoạt động của String pool :**

1. Khi tạo chuỗi bằng ký tự chuỗi (String Literal):
   * Khi một chuỗi được tạo ra bằng ký hiệu " ", Java sẽ kiểm tra xem chuỗi đó có tồn tại trong String Pool chưa.
   * Nếu chuỗi đó đã tồn tại, tham chiếu của chuỗi đã có sẽ được trả về, và không tạo thêm đối tượng mới.
   * Nếu chưa tồn tại, chuỗi mới sẽ được tạo và lưu vào String Pool.
2. Khi tạo chuỗi bằng từ khóa **new**
   * Khi sử dụng từ khóa new để tạo một chuỗi, Java sẽ tạo một đối tượng mới trong **heap memory** (ngoài String Pool), ngay cả khi chuỗi đó đã tồn tại trong String Pool.
   * Điều này có nghĩa là nếu hai chuỗi có cùng nội dung nhưng một trong số đó được tạo bằng từ khóa new, chúng sẽ không tham chiếu đến cùng một đối tượng trong bộ nhớ.
   * Ta có thể dung hàm intern để đưa chuỗi đc khởi tạo bằng từ khoá new vào trong String pool.

* **Lợi ích của String Pool:**
  + **Tiết kiệm bộ nhớ:** Thay vì tạo nhiều đối tượng chuỗi giống nhau, String Pool đảm bảo rằng chỉ có một phiên bản của mỗi chuỗi được tạo ra (với điều kiện chuỗi đó được tạo bằng literal). Điều này giúp giảm bớt bộ nhớ sử dụng trong trường hợp có nhiều chuỗi giống nhau.
  + **Tăng hiệu suất:** Vì String Pool giữ lại các chuỗi đã tạo, việc truy cập và sử dụng lại các chuỗi này nhanh hơn so với việc tạo các đối tượng mới từ đầu mỗi lần

1. Có bao nhiêu cách để tạo 1 biến String ?

* Trong Java có nhiều cách để tạo 1 biến String ví dụ như sau :
  + Bằng String literal (Chuỗi chữ).
  + Bằng từ khóa new.
  + Từ mảng ký tự (char[]).
  + Thông qua phương thức valueOf().
  + Từ StringBuilder hoặc StringBuffer.
  + Sử dụng phương thức concat().

1. Làm sao để so sánh hai chuỗi trong java
2. Sử dụng toán tử ==

* Toán tử == so sánh **tham chiếu** của hai đối tượng String, tức là kiểm tra xem cả hai chuỗi có trỏ đến cùng một đối tượng trong bộ nhớ hay không.
* == chỉ trả về true nếu cả hai biến trỏ đến cùng một đối tượng trong bộ nhớ, không so sánh nội dung của chuỗi.

1. Sử dụng phương thức equals()

* Phương thức equals() so sánh nội dung của hai chuỗi. Nó trả về true nếu hai chuỗi có cùng giá trị, bất kể chúng có tham chiếu đến cùng một đối tượng trong bộ nhớ hay không.

1. Sử dụng phương thức equalsIgnoreCase()

* Phương thức equalsIgnoreCase() giống với equals(), nhưng nó không phân biệt chữ hoa và chữ thường khi so sánh nội dung.

1. Sử dụng phương thức compareTo()

* Phương thức compareTo() so sánh hai chuỗi theo thứ tự từ điển (lexicographically). Nó trả về:
  + 0 nếu hai chuỗi bằng nhau.
  + Số âm nếu chuỗi đầu tiên nhỏ hơn chuỗi thứ hai.
  + Số dương nếu chuỗi đầu tiên lớn hơn chuỗi thứ hai