String

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

- String là một class trong Java chứ không phải kiểu nguyên thủy, khi khai báo sẽ làm một đối tượng đại diện cho một chuỗi các ký tự

- String là bất biến (**immutable**) tức là không thể thay đổi giá trị khi đã tạo. Khi nào thay đổi giá trị của bất kỳ chuỗi nào thì một instance mới được tạo ra. Đối với chuỗi có thể thay đổi, có thể sử dụng các lớp StringBuffer và StringBuilder.

- **String Constant Pool** trong java được dùng để tối ưu bộ nhớ. Ví dụ:

String a = “abc”;

String b = “abc”;

=> a và b sẽ cùng trỏ tới một đối tượng trong pool, không tạo mới. Nhưng nếu dùng **new String(“abc”)** thì sẽ luôn tạo object mới trong heap.

- So sánh String:

- Dùng == để so sánh tham chiếu
- Dùng .equals() để so sánh giá trị

- Các phương thức phổ biến:

- **length():** Trả về độ dài của chuỗi.
- **charAt(int index):** Lấy ký tự tại một vị trí cụ thể.
- **substring(int beginIndex, int endIndex):** Cắt một chuỗi con.
- **equals(Object anotherObject):** So sánh nội dung của hai chuỗi (phân biệt chữ hoa/thường).
- **equalsIgnoreCase(String anotherString):** So sánh nội dung không phân biệt chữ hoa/thường.
- **concat(String str):** Nối một chuỗi vào cuối chuỗi hiện tại.
- **replace(char oldChar, char newChar):** Thay thế ký tự.
- **toLowerCase() / toUpperCase():** Chuyển đổi chuỗi thành chữ thường hoặc chữ hoa.
- **trim():** Loại bỏ khoảng trắng ở đầu và cuối chuỗi.

- **split(String regex):** Tách chuỗi thành một mảng các chuỗi con dựa trên một biểu thức chính quy.

2. Có bao nhiêu cách để tạo 1 biến String

- **String Literal**: Để làm cho Java sử dụng bộ nhớ hiệu quả hơn (vì không có đối tượng mới nào được tạo nếu nó đã tồn tại trong nhóm hằng chuỗi). Ví dụ:

String str1 = “Java”;

Nếu đã có “Java” trong pool, biến sẽ tham chiếu tới object có sẵn. Nếu chưa có, JVM sẽ tạo mới trong pool.

- Sử dụng toán tử **new**: Luôn tạo ra một object mới trong heap, ngay cả khi giá trị đó đã có trong pool

String str2 = new String(“Java”);

System.out.println(str1 == str2); // false;

- Tạo String từ mảng ký tự (**char[]**):

```
char[] chars = {'J', 'a', 'v', 'a'};
String str3 = new String(chars);
```

- Tạo String từ mảng byte (**byte[]):**

```
byte[] bytes = {65, 66, 67}; // ASCII của A, B, C
String str4 = new String(bytes);
```

- Sử dụng các phương thức tĩnh khác như: String.valueOf(), String.format(), String.join(), …

3. Tìm hiểu về String pool?

- String pool là một vùng nhớ đặc biệt nằm trong vùng nhớ Heap (Heap memory), dùng để lưu trữ các biến được khai báo theo kiểu String.

- String pool giúp tối ưu hoá việc lưu trữ và sử dụng vùng nhớ khi khai báo biến String, giúp hạn chế tình trạng tràn bộ nhớ Java Heap Space.
- Cách thức làm việc của String pool:

Cách 1:

String str1 = “abcd”;

String str2 = “abcd”;

Java sẽ truy cập vào String Pool, rồi tìm ở trong Pool ô nhớ nào có cùng giá trị với nó, nếu tìm thấy thì sẽ tham chiếu đến địa chỉ của ô nhớ đó, còn không thì nó sẽ tạo ô nhớ mới ở trong Pool rồi sẽ thực hiện việc tham chiếu.

- Khai báo biến String str1 = “abcd”, nó sẽ tạo mới trong String Pool một ô nhớ có giá trị là “abcd” vì lúc này ở trong Pool chưa có ô nhớ nào giá trị đó.
- Lúc khai báo String str2 = “abcd”; nó tìm thấy trong Pool giá trị một ô nhớ có giá trị là “abcd”, str2 chỉ việc tham chiếu đến địa chỉ của ô nhớ này. Như vậy str2 đã có giá trị là “abcd” nhờ vào việc tham chiếu mà không cần phải tạo 1 ô nhớ khác.

=> System.out.println(str1 == str2); // true

Cách 2:

String str3 = new String(“abcd”);

- Java sẽ không tạo ô nhớ mới ở bộ nhớ String Pool mà sẽ tạo ở Java Heap Space. Và khi đó nó sẽ luôn luôn tạo ô nhớ mới cho dù đã có sẵn những ô nhớ khác có cùng giá trị.
- String str3 = new String(“acbd”), dòng này sẽ tạo 1 ô nhớ ở Java Heap Space có giá trị là “abcd”, biến s3 này sẽ tham chiếu địa chỉ của ô nhớ đó.
- Nếu có: String str4 = new String(“abcd”), tương tự dòng này cũng sẽ tạo 1 ô nhớ khác ở Java Heap Space cũng có giá trị là “abcd” (không giống ở trên là sẽ tìm ô nhớ nào có giá trị như vậy rồi tham chiếu đến nhé), biến str4 này sẽ tham chiếu đến địa chỉ của ô nhớ mới.
  => System.out.println(str3 == str4); // false

4. Làm sao để so sánh hai chuỗi trong java

- Toán tử == so sánh sự tham chiếu của đối tượng, sự giống nhau về vùng nhớ. Vì thế, nếu 2 đối tượng string a và b cùng tham chiếu đến một literal trong string pool, hoặc cùng tham chiếu đến một object trong vùng nhớ heap thì a == b sẽ trả về true. Ngược lại, sẽ trả về false

- Phương thức equals() được override trong lớp String. Nó kiểm tra giá trị của chuỗi kí tự lưu trữ trong string object. Vì thế, nếu a và b cùng chứa chuỗi kí tự như nhau thì a.equals(b) luôn trả về true, bất kể chúng có tham chiếu tới đâu đi nữa.

- Phương thức equalsIgnoreCase():Tương tự equals() nhưng **không phân biệt** chữ hoa và chữ thường. Trả về true nếu nội dung giống nhau (bỏ qua hoa/thường), false nếu khác.

- So sánh theo thứ tự từ điển: compareTo()**:** So sánh hai chuỗi theo thứ tự từ điển. Trả về một số nguyên (int):

- **0**: nếu hai chuỗi bằng nhau.
- **Số âm**: nếu chuỗi gọi phương thức đứng *trước* chuỗi trong tham số.
- **Số dương**: nếu chuỗi gọi phương thức đứng *sau* chuỗi trong tham số.