Java: Hàm xử lý chuỗi (String)

Lớp String cung cấp khá nhiều hàm (hay phương thức) dùng cho việc thao tác chuỗi (hay xâu ký tự). Ta cần lưu ý rằng một biến chuỗi hay một hằng chuỗi thì đều được coi là một đối tượng, và vì vậy nó có thể có quyền sử dụng tất cả các hàm được xây dựng sẵn trong lớp String. Do đó, nếu một hàm nào đó trả về một chuỗi thì ta có quyền sử dụng trực tiếp một hàm nữa ngay sau hàm đó vì chuỗi trả về là một đối tượng mới.

Giả sử ta có một chuỗi `str` được khai báo và gán như sau: `String str="ABCDEF"`, thì dưới đây sẽ trình bày các hàm chủ yếu mà ta có thể áp dụng trên chuỗi này.

Danh sách các hàm xử lý chuỗi

1. charAt(int index):

Hàm này trả về một ký tự ứng với chỉ số **index** trong chuỗi hiện thời. Ví dụ, `str.charAt(3)` sẽ trả về ký tự 'D'. Lưu ý rằng ký tự đầu tiên của chuỗi có chỉ số là 0.

2. compareTo(String st):

Hàm này dùng để so sánh hai đối tượng kiểu String là chuỗi hiện thời và chuỗi **st**. Phép so sánh sẽ trả về một số nguyên là hiệu của cặp ký tự cuối cùng đem so sánh giữa ký tự của chuỗi hiện thời với chuỗi **st**. Thao tác so sánh dựa trên giá trị Unicode của từng ký tự trong hai chuỗi bắt đầu từ bên trái hai chuỗi. Cụ thể, kết quả trả về là một số âm khi chuỗi **st** có thứ tự trong bảng chữ cái lớn hơn chuỗi hiện thời. Nếu kết quả trả về là số 0 thì **st** giống hệt chuỗi hiện thời. Còn nếu kết quả trả về một số dương thì chuỗi hiện thời lớn hơn chuỗi **st**.

```
String st = "ABcd";

str.compareTo(st); //Trả về số -32 vì ký tự 'C' của str có mã là 67, còn ký tự 'c' trong chuỗi đối số có mã là 99.
```

Lưu ý: Unicode là một chuẩn trong đó nó cung cấp một số duy nhất cho mỗi ký tự mà không quan tâm đến nền tảng, chương trình, hay ngôn ngữ. Chuẩn Unicode được áp dụng trong lập trình bởi các hãng công nghệ lớn như Google, IBM, Apple, Microsoft, và những hãng khác.

3. compareToIgnoreCase(String st):

Phương thức này tương tự phương thức compareTo(), chỉ khác ở một điểm là nó không phân biệt chữ hoa hay thường giữa hai chuỗi đem so sánh.

```
"IJK".compareToIgnoreCase("ijk"); //Trả về 0.
```

4. concat(String st):

Hàm này có nhiệm vụ nối chuỗi **st** vào sau chuỗi hiện thời và trả về chính chuỗi hiện thời sau khi nối.

```
str.concat("123"); //Kết quả là chuỗi str chứa "ABCDEF123".
```

5. copyValueOf(char[] data):

Hàm sẽ sao chép dữ liệu từ mảng data sang chuỗi hiện thời.

```
char[] data = {'a', 'b', 'c'};

str = String.copyValueOf(data); /* Sau câu lệnh này thì biến str sẽ chứa chuỗi "abc" */
```

6. copyValueOf(char[] data, int offset, int count):

Hàm này có nhiệm vụ sao chép các ký tự của mảng **data** bắt đầu từ vị trí **offset** với số lượng **count** ký tự.

```
str = String.copyValueOf(data, 1, 2); //Sao chép vào biến chuỗi str hai ký tự 'b' và 'c' của mảng data
```

7. endsWith(String suffix):

Kiểm tra xem chuỗi có kết thúc bằng xâu **suffix** hay không, nếu đúng thì trả về true, ngược lại thì trả về false.

```
str.endWith("HI"); //Trả về true vì chuỗi str kết thúc là "HI"
```

8. indexOf(int ch):

Trả về một số nguyên là vị trí (theo chỉ số) xuất hiện đầu tiên của ký tự có giá trị mã Unicode là **ch** trong chuỗi hiện thời. Miền giá trị của **ch** là từ 0 đến 0xFFFF (0 đến 65536). Ta cũng có thể thay **ch** bằng một ký tự cụ thể. Nếu không tìm thấy thì hàm trả về -1. Ví dụ, nếu chuỗi s có nội dung là "XYZYX" thì ta có như sau:

9. indexOf(int ch, int fromIndex):

Hàm này tương tự như hàm trên, chỉ khác ở một điểm là việc tìm kiếm và lấy chỉ số bắt đầu từ vị trí có chỉ số **fromIndex**.

```
s.indexOf('Y', 2); //Sẽ trả về 3 vì vị trí bắt đầu tìm kiếm là từ chỉ số 2, tức là từ ký tự thứ 2
```

10. indexOf(String st):

Hàm này sẽ tìm chuỗi **st** trong chuỗi hiện thời, trả lại vị trí xuất hiện đầu tiên của chuỗi **st** trong chuỗi khi tìm thấy, trả về -1 nếu không tìm thấy.

```
str.indexOf("CD"); //Trả về 2 vì chuỗi "CD" có trong chuỗi "ABCDEFGHI" ở vị trí chỉ số 2.
```

11. indexOf(String st, int fromIndex):

Hàm này tương tự hàm trên, có điểm khác là việc tìm kiếm chuỗi **st** bắt đầu từ vị trí chỉ số **fromIndex**.

```
str.indexOf("CD", 3); //Trả về -1 vì từ vị trí chỉ số 3 không tìm thấy chuỗi "CD"
```

12. lastIndexOf(int ch):

Hàm này trả lại vị trí chỉ số của ký **ch** cuối cùng trong chuỗi.

```
System.out.print("Đào tạo Lập trình viên".lastIndexOf('t')); /* In ra giá trị là 12 vì ký tự 't' cuối cùng của chuỗi có chỉ số 12 */
```

13. lastIndexOf(int ch, int lastIndex):

Trả lại vị trí xuất hiện cuối cùng của ký tự có mã **ch** (từ 0 đến 65536) trong xâu nhưng bắt đầu từ vị trí 0 đến vị trí **lastIndex**. Bạn cũng có thể sử dụng **ch** là một ký tự cụ thể.

```
String s3 = "ab123ab321";

s3.lastIndexOf(97, s3.length()); //Trả về 5, 97 là mã của ký tự 'a'

s3.lastIndexOf('b', s3.length()); //Trả về 6
```

14. lastIndexOf(String st):

Hàm này ngược với hàm thứ 10 ở trên, có nghĩa là hàm trả về vị trí xuất hiện cuối cùng của chuỗi **st** trong chuỗi hiện thời.

```
String s1="abc123bca";

s1.lastIndexOf("bc"); //Trả về số 6
```

15. lastIndexOf(String st, int lastIndex):

Hàm trả lại vị trí xuất hiện cuối cùng của chuỗi **st** trong chuỗi hiện thời bắt đầu từ vị trí 0 đến **lastIndex**.

```
String s2="abacad";

s2.lastIndexOf("a", 3); //Trả về 2
```

16. length():

Hàm này dùng để lấy chiều dài của chuỗi hiện thời.

```
String s4 = "1a2b3c";

s4.length(); //Trả về 6
```

17. replace(char oldChar, char newChar):

Hàm có nhiệm vụ thay thế tất cả các ký tự **oldChar** của chuỗi hiện thời bằng các ký tự **newChar** và trả về chuỗi mới tương ứng.

```
String s5 = "a1a2a3";

s5.replace('a', 'A'); //Sau câu lệnh này, s5 sẽ chứa chuỗi "A1A2A3"
```

18. startsWith(String prefix):

Hàm này sẽ kiểm tra xem xâu có bắt đầu bằng xâu **prefix** không, trả lại true nếu đúng, ngược lại thì trả lại false.

```
String s6 = "V1Study";
```

Hàm có nhiệm vụ trả về một chuỗi con trong chuỗi hiện thời bắt đầu từ vị trí chỉ số **beginIndex**.

```
String s7 = "America";

s7.substring(3); //Trả về chuỗi "rica"
```

20. substring(int beginIndex, int endIndex):

Hàm trả lại một xâu con trong xâu hiện tại bắt đầu từ vị trí **beginIndex** đến vị trí **endIndex-1**.

s7.substring(1, 5); //Trả về "meri"

21. toCharArray():

Hàm có tác dụng chuyển chuỗi hiện thời sang một mảng ký tự nếu bạn muốn thao tác chuỗi hiện thời theo dạng mảng ký tự.

22. toLowerCase():

Chuyển tất cả các ký tự trong chuỗi hiện thời sang chữ thường.

```
String s8 = "VIETNAM";

s8.toLowerCase(); //s8 sẽ chứa chuỗi "vietnam"
```

23. toUpperCase():

Hàm chuyển tất cả các ký tự của chuỗi sang chữ hoa.

```
String s9 = "programming";

s9.toUpperCase(); //s9 sẽ chứa chuỗi "PROGRAMMING"
```

24. trim():

Tác dụng của hàm này là xoá tất cả các ký tự trắng (Space) ở đầu và cuối chuỗi hiện thời.

```
String s10 = "   Thuy   ";

s10.trim(); //Sau câu lệnh này thì s10 sẽ chứa "Thuy"
```

25. valueOf(giá_trị):

Nếu bạn muốn chuyển một **giá_trị** bất kỳ nào đó sang dạng chuỗi thì bạn sử dụng đến hàm này. Ví dụ dưới đây sẽ chuyển giá trị *true* (có kiểu *boolean*) thành chuỗi.

```
String s11 = String.valueOf(true); //Sau câu lệnh này thì s11 sẽ chứa chuỗi "true"
```

26. codePointAt(int index):

Hàm này trả về một số chính là thứ tự ký tự trong bảng mã ASCII của ký tự tương ứng với chỉ số index.

```
String s = "abcdef";
System.out.println(s.codePointAt(2)); //In ra: 99
```

27. codePointBefore(int index):

Hàm này tương tự như hàm codePointAt(), nhưng trả về thứ tự ký tự trong bảng mã ASCII của ký tự trước ký tự có chỉ số index.

```
String s = "abcdef";
System.out.println(s.codePointBefore(2)); //In ra: 98
```

28. contains(CharSequence s):

Hàm này dùng để tìm kiếm chuỗi con s trong chuỗi hiện thời. Nếu tìm thấy trả về true, không tìm thấy trả về false.

```
String str = "V1Study.com";
System.out.println(str.contains("V1")); //In ra: true
System.out.println(str.contains("VS")); //In ra: false
```

29. contentEquals(CharSequence s):

Hàm có nhiệm vụ so sánh chuỗi s với chuỗi hiện thời, nếu giống nhau thì hàm trả về true, ngược lại sẽ trả về false.

30. equals(Object obj):

Nhiệm vụ của hàm này là để so sánh đối tượng obj với chuỗi hiện thời, nếu giống nhau thì trả về true, ngược lại sẽ trả về false. Lưu ý là việc so sánh là có tính đến ký tự hoa thường.

```
String s = "Study";
int n = 10;
System.out.println(s.equals("V1Study")); //In ra: false
System.out.println(s.equals(n)); //In ra: false
System.out.println(s.equals("Study")); //In ra: true
```

31. equalsIgnoreCase(String str):

Hàm này dùng để so sánh chuỗi str với chuỗi hiện thời mà không quan tâm đến chữ hoa hay thường.

```
String s = "abc";
System.out.println(s.equals("ABC")); //In ra: true
```

32. getBytes():

Hàm này dùng để mã hóa chuỗi hiện thời sử dụng bảng mã mặc định của hệ thống và trả về một mảng kiểu byte.

```
System.out.println("V1Study".getBytes()); //In ra: [B@1db9742
```

33. void getChars(int srcBegin, int srcEnd, char[] dst, int dstBegin):

Hàm này dùng để copy nội dung của chuỗi hiện thời vào mảng ký tự dst. Nội dung copy bắt đầu từ vị trí có chỉ số srcBegin đến vị trí có chỉ số srcEnd-1, và nội dung copy được sẽ được đặt tại vị trí bắt đầu từ chỉ số dstBegin của mảng dst.

```
String str = "V1Study";
char[] ch = new char[30];
str.getChars(0, 3, ch, 0);
System.out.println(ch); //Sẽ in ra: V1S
str.getChars(3, str.length(), ch, 3);
System.out.println(ch); //Sẽ in ra: V1Study
```

34. hasCode():

Hàm này trả về mã băm cho chuỗi hiện thời.

```
String str = "V1Study";
System.out.println(str.hashCode()); //In ra: 498957294
```

35. isEmpty():

Hàm này sẽ trả về true nếu chuỗi hiện thời có kích thước bằng 0, ngược lại thì hàm trả về false.

```
String str = "V1Study";
System.out.println(str.isEmpty()); //In ra: false
str = "";
System.out.println(str.isEmpty()); //In ra: true
```

Dưới đây chúng ta sẽ tìm hiểu một số ví dụ áp dụng các hàm xử lý chuỗi trong Java.

Ví dụ về hàm xử lý chuỗi

Ví dụ 1:

Viết chương trình nhập vào một chuỗi str và đếm xem trong chuỗi đó có bao nhiêu ký tự 'a'. Chương trình được viết như sau:

```
import java.util.Scanner;
public class DemKyTu {
 public static void main(String[] args) {
  Scanner boNhap = new Scanner(System.in); //tạo một bộ nhập
  int count = 0; //khai báo và khởi tạo biến đếm ký tự
  String str = new String(); //tạo một chuỗi
  System.out.print("Mời bạn nhập 1 chuỗi: ");
  str = boNhap.nextLine(); //tiến hành nhập một chuỗi
```

```
      count++; //thì tăng biến đếm lên 1
     }
    }
    System.out.printf("Chuỗi \"%s\" có %d ký tự 'a'.%n", str, count); //in ra kết quả
  }
}
```

Ví dụ 2:

Yêu cầu người dùng nhập vào một chuỗi rồi đếm xem chuỗi đó có bao nhiêu từ (mỗi từ cách nhau bằng một hoặc nhiều dấu cách). Chương trình được viết như sau:

```
import java.util.Scanner;
public class DemTu {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    int count = 0;
    System.out.print("Nhập một chuỗi bất kỳ: ");
    String str = input.nextLine();
    for (int i = 0; i < str.length() - 1; i++) { /* lấy từng ký tự của chuỗi từ ký tự đầu tiên đến ký tự gần cuối */
      if (str.charAt(i) == ' ' && str.charAt(i + 1) != ' ') { /* nếu ký tự đem lấy là dấu cách và đồng thời ký tự sau đó không phải dấu cách */
        count++; /* thì tăng biến đếm lên 1 (đếm được 1 từ) */
      }
    }
    if(str.charAt(0) != ' ') { /* nếu ký tự đầu tiên không phải dấu cách (không thỏa mãn thuật toán tại vòng lặp for) */
      count++; //thì đếm nốt từ đầu tiên của chuỗi
    }
    System.out.printf("Chuỗi \"%s\" có %d từ.%n", str, count);
  }
}
```

Ví dụ 3:

Nhập vào họ và tên của một người. Kiểm tra xem người đó có phải họ "Dang" không? Sau đây là đoạn mã thực hiện yêu cầu:

```
import java.util.Scanner;
public class CheckTu {
  public static void main(String[] args) {
    Scanner nhapLieu = new Scanner(System.in);
    System.out.print("Mời bạn nhập họ và tên: ");
    String hovaten = nhapLieu.nextLine();
    hovaten = hovaten.trim(); //Cắt tất cả các dấu cách ở đầu và cuối chuỗi
    // Kiểm tra xem từ ở đầu có phải là "Dang" không?
    if (hovaten.startsWith("Dang")) {
      System.out.println("Họ của bạn là \"Dang\"");
    }
    else {
      System.out.println("Bạn không mang họ \"Dang\"");
    }
  }
}
```

« Prev Thích 4 Chia sẻ Next »

Đặng Trần Long

Một bạn có hỏi: MSMH: gồm 8 ký tự, theo định dạng sau:
[A-Z][A-Z][A-Z][0-9][0-9][0-9][0-9][0-9]
Ví dụ: NCH01234, DLA11223, TRA11112
Với kiểu này bạn có thể cho mình 1 ví dụ cụ thể không

Thích · Phản hồi · 24 tuần