Strings Introduction

=> String is the sequence of characters OR
   String is an array of characters.
=> For example :-
    char[] ch = {'d', 'e', 'e', 'p', 'a', 'k'};
=> To handle Strings, Java has provided one interface i.e. CharSequence

=> To create Strings, java has provided some classes i.e.
    1. String (java.lang package)
    2. StringBuffer (java.lang package)
    3. StringBuilder (java.lang package)
    4. StringTokenizer (java.util package)

String Class

=> String is a pre-defined class in Java present in java.lang package
=> Syntax :-
```
    public final class String
                 extends Object
                 implements CharSequence, Serializable, Comparable
    {
        // constructors
        // methods
    }
```

=> String is an index based DS

=> Different ways to create String object
    1. By using new keyword
        >> String str = new String("deepak");
    2. By String literal
        >> String str = "deepak";

=> Points to note :-
    1. Whenever we create String objects, they occupy memory in heap area and
       String literal objects will occupy memory in String Literal Pool or String Constant Pool
    2. Garbage Collection process is not applicable in String Constant Pool (SCP)
    3. String objects are immutable

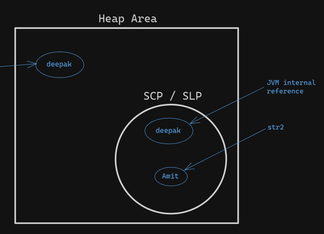
=> Difference between creating String object using new keyword and using String literal :-
    -> By using new keyword, it will create an object in Heap Area as well as in SCP
    -> By using String literal, it will create an object in SCP, not in Heap Area

    -> By using new keyword, it will create 2 objects
    -> By using String literal, it will create only 1 object

    -> Diagram

```
String str1 = new String("deepak");  // 2 objects

String str2 = "Amit";  // 1 object
```



=> Concept related to String Constant Pool or String Literal Pool

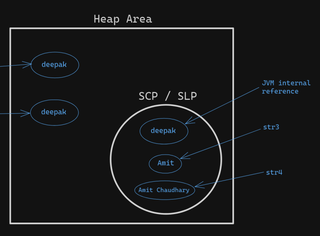
```
String str1 = new String("deepak");  // 2 objects

String str2 = new String("deepak");  // 1 object

String str3 = "Amit";  // 1 object

String str4 = "Amit";  // 0 object

str4 = "Amit Chaudhary";
```



=> How memory management occurs in SCP ?

=> String class Constructors & Methods :-

-> String()
-> String(String original)
-> String(char[] value)
-> String(char[] value, int offset, int count)
   etc

-> Methods :-
    1. Validating User Input
        >> length()
        >> isEmpty()
        >> trim()
    2. Comparing two Strings:
        >> equals()
        >> equalsIgnoreCase()
        >> compareTo()
        >> compareToIgnoreCase()
    3. Concatinating two String:
        >> concat()
        >> + operator
    4. Get sub-strings:
        >> subString()
        >> subSequence()
    5. Replacing or Removing characters:
        >> replace()
        >> replaceFirst()
        >> replaceAll()
    6. Searching characters:
        >> indexOf()
        >> lastIndexOf()
        >> contains()
        >> charAt()
        >> endsWith()
        >> startsWith()
    7. Case conversion:
        >> toLowerCase()
        >> toUpperCase()
    8. Type Conversion:
        >> valueOf()
        >> toCharArray()
    9. Other methods :-
        >> split()

Interview Questions

=> Java Interview Questions – String Class Basics
1. What is the String class in Java?
2. Why is String considered a special class in Java?
3. Is String a primitive type in Java? Why or why not?
4. Why is the String class declared as final in Java?
5. What interfaces does String implement?
6. What are some commonly used constructors of the String class?
7. How does Java handle memory allocation for String objects?

=> String Immutability
8. What is meant by immutability of String in Java?
9. Why are strings immutable in Java?
10. What are the advantages of making String immutable?
11. Does immutability of String improve security? How?
12. Can we make our own immutable class in Java similar to String?
13. What are the disadvantages of immutable strings?

=> String Constant Pool (SCP)
14. What is the String Constant Pool (SCP) in Java?
15. How are String literals stored in the SCP?
16. What happens when we create a String using a literal versus using new?
17. Can two different String literals point to the same object? Why?
18. What is the role of the intern() method in String?
19. What happens when we call intern() on a String created with new?
20. Where is the String Constant Pool located in JVM memory?

=> String Literal vs new Keyword
21. What is the difference between creating a String using literals and using the new keyword?
22. How many objects are created in memory with String s = new String("Java");?
23. Does new String("Java") use the String Constant Pool? Explain.
24. What is the best practice: using String literals or new String()? Why?

=> String Methods
25. What are some commonly used methods in the String class?
26. What is the difference between == and equals() when comparing Strings?
27. What is the difference between equals() and equalsIgnoreCase()?
28. What is the difference between compareTo() and compareToIgnoreCase()?
29. What is the difference between substring() and subSequence()?
30. What is the difference between split() and tokenize() methods?
31. How does the replace() method work in Strings?
32. What is the difference between replace() and replaceAll()?
33. What is the difference between trim(), strip(), and stripLeading()/stripTrailing()?
34. How does the matches() method work with regex in Strings?
35. What is the difference between contains() and indexOf() methods?
36. What is the difference between isEmpty() and isBlank()?
37. How does join() method work in Java Strings?
38. What is the difference between String.valueOf() and toString()?

=> Other String-related Questions
39. Why is String hashCode cached once it is calculated?
40. How does String concatenation work internally in Java?
41. What is the difference between compile-time and runtime String concatenation?
42. How does + operator work on Strings in Java?
43. Why is String often used as a key in HashMap?
44. Can a String be null in Java? What happens if we call a method on a null String?
45. What is the difference between a null String, empty String "", and a blank String " "?
46. How does the JVM optimize String literals at runtime?
47. Can we override methods of the String class? Why or why not?