IMPORTANT METHODS OF STRING CLASS

1. **length():** This method returns the length of the given String.

   Eg:

   String s1=”Mohan is here”;

   System.out.println(s1.length());

   o/p: 13

2. **intern():** Returns a canonical representation for the string object. When the intern method is invoked, if the pool already contains a string equal to this String, then the string from the pool is returned. Otherwise, this String object is added to the pool and a reference to this String object is returned.

3. **isEmpty():** This method returns true if the given String length is zero.

4. **isBlank():** This method returns true if the string is empty or contains only white space otherwise false.

5. **toLowerCase():** Converts all of the characters in this String to lower case.

   Eg:

   String s1=”Tom And Jerry”.

   System.out.println(s1.toLowerCase());

   o/p: tom and jerry

6. **toUpperCase():** Converts all of the characters in this String to upper case.
   Eg:
   String s1=”Tom And Jerry”.
   System.out.println(s1.toUpperCase());
   o/p: TOM AND JERRY

7. **trim():** Returns a string whose value is this string, with all leading and trailing space removed.

8. **concat(String str):** Concatenates the specified string to the end of this string.
   Examples:
   "care".concat("taker") returns "caretaker"
   "to".concat("get").concat("her") returns "together"

9. **charAt(int index):** This method returns the character which is available in String at the provided index.

10. **codePointAt(int index):** This method returns the ASCII value of the character which is available in String at provided index.

11. **equals(Object):** This method returns true if two strings are equal or it returns false.

12. **equalsIgnoreCase(String anotherString):** This method Compares one String to another String ignoring case considerations.

13. **compareTo(String anotherString):** Compares two strings lexicographically and returns integer value.

14. **compareToIgnoreCase(String str):** Compares two strings lexicographically ignoring case and returns integer value.

15. **startsWith(String prefix):** Tests if this string starts with the specified prefix or not.

16. **startsWith(String prefix, int toffset):** Tests if this string starts with the specified prefix from the given index or not.

17. **endsWith(String suffix):** Tests if this string ends with the specified string or not.

18. **replace(char oldChar, char newChar):** Returns a string resulting from replacing all occurrences of oldChar in this string with newChar.

    Examples:

    "mesquite in your cellar".replace('e', 'o')
    returns "mosquito in your collar"
    "the war of baronets".replace('r', 'y')
    returns "the way of bayonets"
    "sparring with a purple porpoise".replace('p', 't')
    returns "starring with a turtle tortoise"
    "JonL".replace('q', 'x') returns "JonL" (no change)

19. **indexOf(int ch):** Returns the index within this string of the first occurrence of the specified character.

20. **indexOf(int ch, int fromIndex):** Returns the index within this string of the first occurrence of the specified character, starting the search from the specified index.

21. **lastIndexOf(int ch):** Returns the index within this string of the last occurrence of the specified character.

22. **lastIndexOf(int ch, int fromIndex):** Returns the index within this string of the last occurrence of the specified character, searching backward starting from the specified index.

23. **indexOf(String str):** Returns the index within this string of the first occurrence of the specified substring.

24. **indexOf(String str, int fromIndex):** Returns the index within this string of the first occurrence of the specified substring, starting at the specified index.

25. **lastIndexOf(String str):** Returns the index within this string of the last occurrence of the specified substring.

26. **lastIndexOf(String str, int fromIndex):** Returns the index within this string of the last occurrence of the specified substring, searching backward starting from the specified index.
27. **substring(int beginIndex):** Returns a string that is a substring of this string. The substring begins with the character at the specified index and extends to the end of this string.

28. **substring(int beginIndex, int endIndex):** Returns a string that is a substring of this string. The substring begins from the specified beginIndex and extends to the character at end index-1.

29. **replace(CharSequence target, CharSequence replacement):** Replaces each substring of this string that matches the literal target sequence with the specified literal replacement sequence.

30. **replaceFirst(String regex, String replacement):** Replaces the first substring of this string that matches the given regular expression.

31. **stripLeading():** Returns a string whose value is this string, with all leading white space removed.

32. **stripTrailing():** Returns a string whose value is this string, with all trailing white space removed.

33. **getBytes():** It return byte[] for every characters of String

34. **toCharArray():** converts this String to a new character Array.

35. **valueOf(datatype x):** returns the string representation of the argument.

36. **contains(charsequence s):** Returns true if and only if this string contains the specified sequence of char values.

37. **indent(int n):** Adjusts the indentation of each line of this string based on the value of n, and normalizes line termination characters.

38. **repeat(int n):** Returns a string whose value is the concatenation of this string repeated count times.

39. **join(CharSequence delimeter, CharSequence elements):** Returns a new String composed of copies of the CharSequence elements joined together with a copy ofthe specified delimiter.
For example,

```
String message = String.join("-", "Java", "is", "cool");
// message returned is: "Java-is-cool"
```

40. **split(String regex):** Splits this string around matches of the given regular expression.

41. **matches(String regex):** Tells whether the string matches the given regular expression or not.