Java String Methods

give me all frequently used methods for String for problem solving

Java String – Frequently Used Methods for Problem Solving

Below are **most used String methods** in coding questions, interviews, and competitive programming.

---

1. Length and Character Access

- **`length()`**
  → String ki length return karta hai.
- **`charAt(int index)`**
  → Specific index ka character deta hai.

Use: character traversal, palindrome, frequency count.

```
String s = "hello";
int len = s.length();      // 5
char ch = s.charAt(1);     // 'e'
```

---

2. Comparison Methods

- **`equals(String s)`**
  → content compare karta hai (case-sensitive).
- **`equalsIgnoreCase(String s)`**
  → case ignore karke compare.
- **`compareTo(String s)`**
  → lexicographical comparison.

Use: string matching, sorting.

```
"abc".equals("abc");            // true
"abc".equalsIgnoreCase("ABC");  // true
```

---

3. Searching Methods

- **`contains(CharSequence s)`**
  → substring present hai ya nahi.
- **`indexOf(char or String)`**
  → first occurrence index.

- `lastIndexOf(char or String)`
  → last occurrence index.
- `startsWith(String prefix)`
- `endsWith(String suffix)`

Use: substring search, pattern detection.

```
String s = "banana";
s.indexOf('a');       // 1
s.lastIndexOf('a');   // 5
s.contains("nan");    // true
```

---

4. Substring and Slicing

- `substring(int start)`
- `substring(int start, int end)`

Use: part extraction, sliding window, parsing.

```
String s = "program";
s.substring(1,4);   // "rog"
```

---

5. Case Conversion

- `toLowerCase()`
- `toUpperCase()`

Use: case-insensitive comparison.

```
"Hello".toLowerCase();   // "hello"
```

---

6. Trimming and Cleaning

- `trim()`
  → starting aur ending spaces remove karta hai.

Use: input cleaning.

```
"  hi  ".trim();   // "hi"
```

---

7. Replace Methods

- `replace(char old, char new)`
- `replace(String old, String new)`
- `replaceAll(regex, replacement)`

Use: character modification, pattern replace.

```
"apple".replace('p','x');      // "axxle"
"1a2b3".replaceAll("\\d",""); // "ab"
```

---

8. Splitting Strings

- **`split(String regex)`**

Use: words count, tokenization.

```
String s = "a,b,c";
String[] arr = s.split(",");
```

---

9. Conversion Methods

- **`toCharArray()`**
  → string → char array.
- **`valueOf()`**
  → primitive → string.

Use: manual character operations.

```
char[] arr = "hello".toCharArray();
```

---

10. Checking Empty or Blank

- **`isEmpty()`**
  → length == 0
- **`isBlank()`** (Java 11+)
  → spaces only string bhi blank.

```
"".isEmpty();        // true
"   ".isBlank();     // true
```

---

11. Concatenation

- **`concat(String s)`**
- `+` operator

```
"hi".concat(" there");   // "hi there"
```

Most Important for Problem Solving (Top 10)

Yeh **90% string questions** me use hote:

1. **`length()`**
2. **`charAt()`**
3. **`equals()`**

4. `indexOf()`
5. `substring()`
6. `toCharArray()`
7. `contains()`
8. `split()`
9. `replace()`
10. `trim()`

Typical Problem Types Where Used

- Palindrome check → `charAt()`, `length()`
- Anagram → `toCharArray()`, `sort`
- Word count → `split()`
- Substring search → `indexOf()`
- Frequency count → `charAt()`

If you want, I can give:

- **Top 25 string interview problems**
- **Pattern-wise string problem list**
- **Cheat sheet PDF style summary**

eh **90% string questions** me use hote: 1. `length()` 2. `charAt()` 3. `equals()` 4. `indexOf()` 5. `substring()` 6. `toCharArray()` 7. `contains()` 8. `split()` 9. `replace()` 10. `trim()`
table bana ke do and use cases and input and output bhi

Java String – Top 10 Methods (Problem Solving Table)

#	Method	Kya karta hai (How it works)	Use Case (Kaha use hota hai)	Input Example	Output
1	`length()`	String ki total length return karta hai	Palindrome, loops, size check	`"hello".length()`	`5`
2	`charAt(i)`	Given index ka character	Character traversal, frequency count	`"hello".charAt(1)`	`'e'`

#	Method	Kya karta hai (How it works)	Use Case (Kaha use hota hai)	Input Example	Output
		return karta hai			
3	`equals(s)`	Do strings ka content compare karta hai	Password check, string match	`"abc".equals("abc")`	`true`
4	`indexOf(x)`	Character ya substring ka first index deta hai	Substring search, position find	`"banana".indexOf('a')`	`1`
5	`substring(s,e)`	String ka part extract karta hai	Sliding window, parsing	`"program".substring(1,4)`	`"rog"`
6	`toCharArray()`	String ko char array me convert karta hai	Sorting, manual operations	`"cat".toCharArray()`	`['c','a','t']`
7	`contains(s)`	Check karta hai substring present hai ya nahi	Pattern detection, validation	`"hello".contains("ell")`	`true`
8	`split(regex)`	String ko parts me todta hai	Word count, tokenization	`"a,b,c".split(",")`	`["a","b","c"]`
9	`replace(a,b)`	Character ya substring replace karta hai	Cleaning, transformation	`"apple".replace('p','x')`	`"axxle"`
10	`trim()`	Starting aur ending spaces remove karta hai	Input cleaning, form data	`"  hi  ".trim()`	`"hi"`

Quick Mapping (Method → Typical Problem)

- `length()`, `charAt()` → palindrome, reverse
- `equals()` → login, string compare
- `indexOf()` → substring search
- `substring()` → window problems

- `toCharArray()` → anagram, sorting
- `contains()` → pattern check
- `split()` → word count
- `replace()` → remove characters
- `trim()` → input cleaning

Agar chaho to main **string problems ka pattern-wise table** bhi bana sakta hoon.