JAVA STRING METHODS

1. charAt()

- **Description**: Returns the character at the specified index (0-based).
- **Return Type**: `char`
- **Example**:

```
String str = "Hello";
char ch = str.charAt(1); // Returns 'e'
System.out.println(ch);
```

2. codePointAt()

- **Description**: Returns the Unicode code point of the character at the specified index.
- **Return Type**: `int`
- **Example**:

```
String str = "Hello";
int codePoint = str.codePointAt(1); // Returns 101 (Unicode for 'e')
System.out.println(codePoint);
```

3. codePointBefore()

- **Description**: Returns the Unicode code point of the character before the specified index.
- **Return Type**: `int`
- **Example**:

```
String str = "Hello";
int codePoint = str.codePointBefore(2); // Returns 108 (Unicode for 'l')
System.out.println(codePoint);
```

4. codePointCount()

- **Description**: Returns the number of Unicode code points in the specified substring.
- **Return Type**: `int`
- **Example**:

```
String str = "Hello😊";
int count = str.codePointCount(0, str.length()); // Returns 6 (5 letters + 1 emoji)
System.out.println(count);
```

5. compareTo()

- **Description**: Compares two strings lexicographically (based on Unicode values).
- **Return Type**: `int` (negative if less, 0 if equal, positive if greater)
- **Example**:

```
String str1 = "apple";
String str2 = "banana";
int result = str1.compareTo(str2); // Returns negative value (apple < banana)
System.out.println(result);
```

6. compareToIgnoreCase()

- **Description**: Compares two strings lexicographically, ignoring case.
- **Return Type**: `int`
- **Example**:

```
String str1 = "APPLE";
String str2 = "apple";
int result = str1.compareToIgnoreCase(str2); // Returns 0 (equal ignoring ca
se)
System.out.println(result);
```

---

7. concat()

- **Description**: Appends a string to the end of another string.
- **Return Type**: `String`
- **Example**:

```
String str1 = "Hello";
String str2 = str1.concat(" World"); // Returns "Hello World"
System.out.println(str2);
```

---

8. contains()

- **Description**: Checks if a string contains a specified sequence of characters.
- **Return Type**: `boolean`
- **Example**:

```
String str = "Hello World";
boolean hasWorld = str.contains("World"); // Returns true
System.out.println(hasWorld);
```

---

9. contentEquals()

- **Description**: Checks if a string has the same sequence of characters as a CharSequence or StringBuffer.
- **Return Type**: `boolean`
- **Example**:

```
String str = "Hello";
StringBuffer sb = new StringBuffer("Hello");
```

```
boolean result = str.contentEquals(sb); // Returns true
System.out.println(result);
```

10. copyValueOf()

- **Description**: Creates a string from a character array.
- **Return Type**: `String`
- **Example**:

```
char[] chars = {'H', 'e', 'l', 'l', 'o'};
String str = String.copyValueOf(chars); // Returns "Hello"
System.out.println(str);
```

11. endsWith()

- **Description**: Checks if a string ends with the specified suffix.
- **Return Type**: `boolean`
- **Example**:

```
String str = "Hello.txt";
boolean ends = str.endsWith(".txt"); // Returns true
System.out.println(ends);
```

12. equals()

- **Description**: Checks if two strings are equal (case-sensitive).
- **Return Type**: `boolean`
- **Example**:

```
String str1 = "Hello";
String str2 = "Hello";
boolean equal = str1.equals(str2); // Returns true
System.out.println(equal);
```

13. equalsIgnoreCase()

- **Description**: Checks if two strings are equal, ignoring case.
- **Return Type**: `boolean`
- **Example**:

```
String str1 = "HELLO";
String str2 = "hello";
boolean equal = str1.equalsIgnoreCase(str2); // Returns true
System.out.println(equal);
```

---

14. format()

- **Description**: Creates a formatted string using a format string and arguments.
- **Return Type**: `String`
- **Example**:

```
String str = String.format("Hello, %s!", "World"); // Returns "Hello, World!"
System.out.println(str);
```

---

15. getBytes()

- **Description**: Converts a string into an array of bytes (using platform's default charset or specified charset).
- **Return Type**: `byte[]`
- **Example**:

```
String str = "Hello";
byte[] bytes = str.getBytes(); // Converts to byte array
System.out.println(Arrays.toString(bytes)); // Prints byte values
```

---

16. getChars()

- **Description**: Copies characters from a string to a character array.

- **Return Type**: `void`
- **Example**:

```
String str = "Hello";
char[] dest = new char[5];
str.getChars(0, 5, dest, 0); // Copies "Hello" to dest array
System.out.println(Arrays.toString(dest)); // Prints [H, e, l, l, o]
```

---

17. hashCode()

- **Description**: Returns the hash code of a string.
- **Return Type**: `int`
- **Example**:

```
String str = "Hello";
int hash = str.hashCode(); // Returns hash code
System.out.println(hash);
```

---

18. indexOf()

- **Description**: Returns the index of the first occurrence of a specified substring or character.
- **Return Type**: `int`
- **Example**:

```
String str = "Hello World";
int index = str.indexOf("World"); // Returns 6
System.out.println(index);
```

---

19. intern()

- **Description**: Returns a canonical representation of the string from the string pool.
- **Return Type**: `String`

- **Example**:

```
String str1 = new String("Hello").intern();
String str2 = "Hello";
boolean same = str1 == str2; // Returns true (same object in pool)
System.out.println(same);
```

---

20. isEmpty()

- **Description**: Checks if a string is empty (length == 0).
- **Return Type**: `boolean`
- **Example**:

```
String str = "";
boolean empty = str.isEmpty(); // Returns true
System.out.println(empty);
```

---

21. join()

- **Description**: Joins multiple strings with a specified delimiter.
- **Return Type**: `String`
- **Example**:

```
String result = String.join("-", "Hello", "World"); // Returns "Hello-World"
System.out.println(result);
```

---

22. lastIndexOf()

- **Description**: Returns the index of the last occurrence of a specified substring or character.
- **Return Type**: `int`
- **Example**:

```
String str = "Hello World Hello";
int index = str.lastIndexOf("Hello"); // Returns 12
```

```
System.out.println(index);
```

23. length()

- **Description**: Returns the length of a string.
- **Return Type**: `int`
- **Example**:

```
String str = "Hello";
int len = str.length(); // Returns 5
System.out.println(len);
```

24. matches()

- **Description**: Checks if a string matches a regular expression.
- **Return Type**: `boolean`
- **Example**:

```
String str = "abc123";
boolean matches = str.matches("[a-z]+[0-9]+"); // Returns true
System.out.println(matches);
```

25. offsetByCodePoints()

- **Description**: Returns the index offset by a specified number of Unicode code points.
- **Return Type**: `int`
- **Example**:

```
String str = "Hello😊";
int index = str.offsetByCodePoints(0, 2); // Returns 2 (after "He")
System.out.println(index);
```

26. regionMatches()

- **Description**: Tests if two string regions are equal, optionally ignoring case.
- **Return Type**: `boolean`
- **Example**:

```
String str1 = "Hello World";
String str2 = "world";
boolean matches = str1.regionMatches(true, 6, str2, 0, 5); // Returns true
System.out.println(matches);
```

---

27. replace()

- **Description**: Replaces all occurrences of a specified character or sequence with another.
- **Return Type**: `String`
- **Example**:

```
String str = "Hello World";
String result = str.replace("World", "Java"); // Returns "Hello Java"
System.out.println(result);
```

---

28. replaceAll()

- **Description**: Replaces all substrings matching a regular expression with a replacement string.
- **Return Type**: `String`
- **Example**:

```
String str = "Hello123 World456";
String result = str.replaceAll("[0-9]+", "#"); // Returns "Hello# World#"
System.out.println(result);
```

---

29. replaceFirst()

- **Description**: Replaces the first substring matching a regular expression with a replacement string.

- **Return Type**: `String`
- **Example**:

```
String str = "Hello123 World456";
String result = str.replaceFirst("[0-9]+", "#"); // Returns "Hello# World456"
System.out.println(result);
```

---

30. split()

- **Description**: Splits a string into an array of substrings based on a delimiter.
- **Return Type**: `String[]`
- **Example**:

```
String str = "apple,banana,orange";
String[] fruits = str.split(","); // Returns ["apple", "banana", "orange"]
System.out.println(Arrays.toString(fruits));
```

---

31. startsWith()

- **Description**: Checks if a string starts with a specified prefix.
- **Return Type**: `boolean`
- **Example**:

```
String str = "Hello World";
boolean starts = str.startsWith("Hello"); // Returns true
System.out.println(starts);
```

---

32. subSequence()

- **Description**: Returns a subsequence of characters from the string.
- **Return Type**: `CharSequence`
- **Example**:

```
String str = "Hello World";
CharSequence sub = str.subSequence(0, 5); // Returns "Hello"
System.out.println(sub);
```

33. substring()

- **Description**: Returns a substring from the specified index (or between two indices).
- **Return Type**: `String`
- **Example**:

```
String str = "Hello World";
String sub = str.substring(6); // Returns "World"
System.out.println(sub);
```

34. toCharArray()

- **Description**: Converts a string to a character array.
- **Return Type**: `char[]`
- **Example**:

```
String str = "Hello";
char[] chars = str.toCharArray(); // Returns ['H', 'e', 'l', 'l', 'o']
System.out.println(Arrays.toString(chars));
```

35. toLowerCase()

- **Description**: Converts all characters in a string to lowercase.
- **Return Type**: `String`
- **Example**:

```
String str = "HELLO";
String lower = str.toLowerCase(); // Returns "hello"
System.out.println(lower);
```

36. toString()

- **Description**: Returns the string itself (useful in subclasses or for consistency).
- **Return Type**: `String`
- **Example**:

```
String str = "Hello";
String result = str.toString(); // Returns "Hello"
System.out.println(result);
```

37. toUpperCase()

- **Description**: Converts all characters in a string to uppercase.
- **Return Type**: `String`
- **Example**:

```
String str = "hello";
String upper = str.toUpperCase(); // Returns "HELLO"
System.out.println(upper);
```

38. trim()

- **Description**: Removes leading and trailing whitespace from a string.
- **Return Type**: `String`
- **Example**:

```
String str = "  Hello  ";
String trimmed = str.trim(); // Returns "Hello"
System.out.println(trimmed);
```

39. valueOf()

- **Description**: Converts a value (e.g., int, double, object) to its string representation.

- **Return Type**: `String`
- **Example**:

```
int num = 123;
String str = String.valueOf(num); // Returns "123"
System.out.println(str);
```
