Strings in Java

In Java, a String is an immutable sequence of characters. It is one of the most commonly used classes in Java, defined in the java.lang package. Since strings are immutable, any modification to a string creates a new String object rather than modifying the existing one.

The immutability of strings ensures that any modification creates a new object.

Characteristics of Strings:

Immutable: Once created, the content of a String cannot be changed. For example:

String Methods

char charAt(int index)

int codePointAt(int index)

int codePointBefore(int index)

int codePointCount(int beginIndex, int endIndex)

int compareTo(String anotherString)

int compareToIgnoreCase(String str)

String concat(String str)

boolean contains(CharSequence s)

boolean contentEquals(CharSequence cs)

boolean contentEquals(StringBuffer sb)

boolean endsWith(String suffix)

boolean equals(Object anObject)

boolean equalsIgnoreCase(String anotherString)

byte[] getBytes()

byte[] getBytes(Charset charset)

byte[] getBytes(String charsetName)

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

int hashCode()

int indexOf(int ch)

int indexOf(int ch, int fromIndex)

int indexOf(String str)

int indexOf(String str, int fromIndex)

boolean isEmpty()

int lastIndexOf(int ch)

int lastIndexOf(int ch, int fromIndex)

int lastIndexOf(String str)

int lastIndexOf(String str, int fromIndex)

int length()

boolean matches(String regex)

boolean regionMatches(boolean ignoreCase, int toffset, String other, int ooffset, int len)

boolean regionMatches(int toffset, String other, int ooffset, int len)

String replace(char oldChar, char newChar)

String replace(CharSequence target, CharSequence replacement)

String replaceAll(String regex, String replacement)

String replaceFirst(String regex, String replacement)

String[] split(String regex)

String[] split(String regex, int limit)

boolean startsWith(String prefix)

boolean startsWith(String prefix, int toffset)

CharSequence subSequence(int beginIndex, int endIndex)

String substring(int beginIndex)

String substring(int beginIndex, int endIndex)

char[] toCharArray()

String toLowerCase()

String toLowerCase(Locale locale)

String toUpperCase()

String toUpperCase(Locale locale)

String trim()

static String valueOf(boolean b)

static String valueOf(char c)

static String valueOf(char[] data)

static String valueOf(char[] data, int offset, int count)

static String valueOf(double d)

static String valueOf(float f)

static String valueOf(int i)

static String valueOf(long l)

static String valueOf(Object obj)

boolean isBlank() (Java 11+)

Stream<String> lines() (Java 11+)

String strip() (Java 11+)

String stripLeading() (Java 11+)

String stripTrailing() (Java 11+)

boolean matches() (Java 11+)

String str = "Hello";

str.concat(" World"); // Creates a new string but does not modify `str`

System.out.println(str); // Output: Hello

Stored in String Pool: Java optimizes memory by storing String literals in a special memory area called the String Pool.

Implemented as a Class: String is a final class, which means it cannot be extended.

Unicode Support: Strings in Java support Unicode, allowing them to represent characters from various languages.

Creating Strings

There are two ways to create strings in Java:

Using string literals:

String str = "Hello";

This stores the string in the String Pool.

Using the new keyword:

String str = new String("Hello");

This explicitly creates a new string object in the heap.

Common String Functions in Java

Here's a list of commonly used String methods with examples:

1. Length of a String

Method: int length()

Returns the length of the string (number of characters).

String str = "Hello";

System.out.println(str.length()); // Output: 5

2. Accessing Characters

Method: char charAt(int index)

Returns the character at the specified index.

String str = "Hello";

System.out.println(str.charAt(1)); // Output: e

3. Substring

Method: String substring(int startIndex)

Returns a substring starting from the given index to the end.

String str = "Hello World";

System.out.println(str.substring(6)); // Output: World

Method: String substring(int startIndex, int endIndex)

Returns a substring starting from startIndex to endIndex - 1.

System.out.println(str.substring(0, 5)); // Output: Hello

4. String Concatenation

Method: String concat(String str)

Concatenates the specified string to the end of the current string.

String str1 = "Hello";

String str2 = "World";

System.out.println(str1.concat(" " + str2)); // Output: Hello World

Alternatively, you can use the + operator:

System.out.println(str1 + " " + str2); // Output: Hello World

5. Case Conversion

Method: String toLowerCase()

Converts all characters to lowercase.

```
String str = "Hello";
System.out.println(str.toLowerCase()); // Output: hello
```

Method: String toUpperCase()

Converts all characters to uppercase.

```
System.out.println(str.toUpperCase()); // Output: HELLO
```

6. Trim Whitespace

Method: String trim()

Removes leading and trailing spaces.

```
String str = "   Hello World   ";
System.out.println(str.trim()); // Output: Hello World
```

7. Replace Characters

Method: String replace(char oldChar, char newChar)

Replaces all occurrences of a character with another.

```
String str = "Hello";
System.out.println(str.replace('l', 'p')); // Output: Heppo
```

Method: String replace(CharSequence target, CharSequence replacement)

Replaces all occurrences of a substring with another substring.

```
String str = "Hello World";
```

System.out.println(str.replace("World", "Java")); // Output: Hello Java

8. Check for String Content

Method: boolean contains(CharSequence s)

Checks if the string contains the specified sequence of characters.

String str = "Hello World";

System.out.println(str.contains("World")); // Output: true

9. Starts or Ends With

Method: boolean startsWith(String prefix)

Checks if the string starts with the specified prefix.

String str = "Hello World";

System.out.println(str.startsWith("Hello")); // Output: true

Method: boolean endsWith(String suffix)

Checks if the string ends with the specified suffix.

System.out.println(str.endsWith("World")); // Output: true

10. Equality Check

Method: boolean equals(Object obj)

Compares two strings for exact equality.

String str1 = "Hello";

String str2 = "Hello";

System.out.println(str1.equals(str2)); // Output: true

Method: boolean equalsIgnoreCase(String anotherString)

Compares strings for equality, ignoring case.

```
String str1 = "Hello";
String str2 = "hello";
System.out.println(str1.equalsIgnoreCase(str2)); // Output: true
```

11. String Splitting

Method: String[] split(String regex)

Splits the string around matches of the given regular expression.

```
String str = "Java,Python,C++";
String[] languages = str.split(",");
for (String lang : languages) {
   System.out.println(lang);
}
// Output:
// Java
// Python
// C++
```

12. Index of Characters or Substrings

Method: int indexOf(String str)

Returns the index of the first occurrence of the specified substring.

```
String str = "Hello World";
System.out.println(str.indexOf("World")); // Output: 6
```

Method: int lastIndexOf(String str)

Returns the index of the last occurrence of the specified substring.

System.out.println(str.lastIndexOf("o")); // Output: 7

13. Check if Empty or Blank

Method: boolean isEmpty()

Checks if the string is empty (length() == 0).

String str = "";

System.out.println(str.isEmpty()); // Output: true

Method: boolean isBlank()

Checks if the string is empty or contains only whitespace.

String str = "   ";

System.out.println(str.isBlank()); // Output: true

14. Join Strings

Method: static String join(CharSequence delimiter, CharSequence... elements)

Joins multiple strings with the specified delimiter.

String result = String.join(", ", "Java", "Python", "C++");

System.out.println(result); // Output: Java, Python, C++

15. Convert to Character Array

Method: char[] toCharArray()

Converts the string into a character array.

String str = "Hello";

char[] chars = str.toCharArray();

for (char c : chars) {

```
    System.out.print(c + " ");
}
// Output: H e l l o
```

Key Notes

Strings are immutable, so most operations return a new string.

StringBuilder and StringBuffer are mutable alternatives for better performance when modifying strings frequently.

Let me know if you'd like deeper examples or comparisons with StringBuilder!

You said:

continue

ChatGPT said:

ChatGPT

Advanced String Operations

Let’s explore some additional String features and scenarios where strings are commonly used.

16. Comparing Strings

Using compareTo

The compareTo() method compares two strings lexicographically.

Returns 0 if the strings are equal.

Returns a negative value if the current string is lexicographically less than the argument.

Returns a positive value if the current string is lexicographically greater than the argument.

java

Copy

Edit

String str1 = "Apple";

String str2 = "Banana";

System.out.println(str1.compareTo(str2)); // Output: -1 (Apple < Banana)

Using compareToIgnoreCase

Compares two strings lexicographically, ignoring case differences.

java

Copy

Edit

String str1 = "Apple";

String str2 = "apple";

System.out.println(str1.compareToIgnoreCase(str2)); // Output: 0 (case ignored)

17. Interning Strings

String Intern Pool

The intern() method ensures that the string is in the String Pool.

If the string already exists in the pool, the reference is returned.

Otherwise, the string is added to the pool.

java

Copy

Edit

String str1 = new String("Hello").intern();

String str2 = "Hello";

System.out.println(str1 == str2); // Output: true (references point to the same object in the pool)

18. String Format

Using String.format

The String.format() method allows you to create formatted strings using placeholders.

java

Copy

Edit

String name = "John";

int age = 30;

String formatted = String.format("My name is %s and I am %d years old.", name, age);

System.out.println(formatted);

// Output: My name is John and I am 30 years old.

%s - String placeholder

%d - Integer placeholder

%f - Floating-point placeholder

%n - Line separator (new line)

java

Copy

Edit

double pi = 3.14159;

System.out.println(String.format("Value of pi: %.2f", pi)); // Output: Value of pi: 3.14

19. Regular Expressions with Strings

Matching Patterns

The matches() method checks if the string matches a specified regular expression.

java

Copy

Edit

String str = "hello123";

System.out.println(str.matches("[a-z]+[0-9]+")); // Output: true

Replacing with Patterns

The replaceAll() method replaces all occurrences of a pattern with a specified replacement.

java

Copy

Edit

String str = "a1b2c3";

System.out.println(str.replaceAll("[0-9]", "")); // Output: abc

Splitting with Patterns

The split() method uses a regular expression to split a string.

java

Copy

Edit

String str = "apple,orange,banana";

```
String[] fruits = str.split(",");
for (String fruit : fruits) {
   System.out.println(fruit);
}
// Output:
// apple
// orange
// banana
```

20. Convert Non-String Data Types to String

Using String.valueOf() Converts non-string data types (e.g., int, double, char) into strings.

java

Copy

Edit

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

Using Integer.toString() and Similar Methods

These methods are specialized for specific types.

java

Copy

Edit

```
int num = 123;
String str = Integer.toString(num);
System.out.println(str); // Output: 123
```

21. String Joining

Joining with Delimiters

java

Copy

Edit

String[] words = {"Java", "Python", "C++"};

String joined = String.join(", ", words);

System.out.println(joined); // Output: Java, Python, C++

22. Encoding and Decoding Strings

Converting Strings to Bytes

The getBytes() method converts a string into a byte array using the specified character encoding.

java

Copy

Edit

String str = "Hello";

byte[] bytes = str.getBytes();

for (byte b : bytes) {

  System.out.print(b + " "); // Output: 72 101 108 108 111

}

Converting Bytes to Strings

java

Copy

Edit

byte[] bytes = {72, 101, 108, 108, 111};

String str = new String(bytes);

System.out.println(str); // Output: Hello

23. Immutable Behavior of Strings

Since strings are immutable, methods like concat, replace, and others create new strings instead of modifying the original one.

java

Copy

Edit

String str = "Hello";

String modified = str.concat(" World");

System.out.println(str); // Output: Hello

System.out.println(modified); // Output: Hello World

24. StringBuffer vs StringBuilder

StringBuffer (Thread-Safe):

A mutable sequence of characters.

Thread-safe and synchronized.

java

Copy

Edit

StringBuffer sb = new StringBuffer("Hello");

sb.append(" World");

System.out.println(sb); // Output: Hello World

StringBuilder (Faster but Non-Thread-Safe):

A faster, non-synchronized alternative to StringBuffer.

java

Copy

Edit

```
StringBuilder sb = new StringBuilder("Hello");
sb.append(" World");
System.out.println(sb); // Output: Hello World
```

25. Converting a String to a Collection

Convert to List

java

Copy

Edit

```
String str = "apple,banana,orange";
List<String> list = Arrays.asList(str.split(","));
System.out.println(list); // Output: [apple, banana, orange]
```

Convert to Set

java

Copy

Edit

```
Set<String> set = new HashSet<>(Arrays.asList(str.split(",")));
System.out.println(set); // Output: [apple, orange, banana]
```

Summary

The String class in Java is incredibly versatile and offers methods for common operations like searching, modifying, formatting, and splitting text. While String is immutable, alternatives like StringBuffer and StringBuilder provide mutable options for scenarios requiring frequent modifications.

26. Splitting Strings with Limit

The split() method can take a second parameter to limit the number of substrings generated.

Example: Splitting with a Limit

java

Copy

Edit

```
String str = "apple,orange,banana,grape";

String[] parts = str.split(",", 2);

for (String part : parts) {

   System.out.println(part);

}

// Output:

// apple

// orange,banana,grape
```

In this example, the string is split into at most two parts. The second part contains the rest of the string.

27. Reversing a String

Java's String class doesn't provide a direct method for reversing strings, but it can be achieved using StringBuilder or StringBuffer.

Using StringBuilder

java

Copy

Edit

```
String str = "Hello";
String reversed = new StringBuilder(str).reverse().toString();
System.out.println(reversed); // Output: olleH
```

Using a Loop

java

Copy

Edit

```
String str = "Hello";
String reversed = "";
for (int i = str.length() - 1; i >= 0; i--) {
   reversed += str.charAt(i);
}
System.out.println(reversed);
```

Removing All Spaces

java

Copy

Edit

String str = " H e l l o ";

System.out.println(str.replaceAll("\\s", "")); // Output: Hello

31. Splitting Lines

Splitting a Multiline String

java

Copy

Edit

String str = "Line1\nLine2\nLine3";

String[] lines = str.split("\\R"); // Matches any line break

for (String line : lines) {

  System.out.println(line);

}

// Output:

// Line1

// Line2

// Line3

32. String Subsequence

Using subSequence

Extracts a subsequence of characters from the string.

java

Copy

Edit

String str = "Hello World";

CharSequence sub = str.subSequence(0, 5); // Extracts "Hello"

System.out.println(sub);

Note: subSequence and substring behave similarly, but subSequence returns a CharSequence rather than a String.

33. Creating Strings from Arrays

From a Character Array

java

Copy

Edit

char[] chars = {'H', 'e', 'l', 'l', 'o'};

String str = new String(chars);

System.out.println(str); // Output: Hello

From a Byte Array

java

Copy

Edit

byte[] bytes = {65, 66, 67};

String str = new String(bytes);

System.out.println(str); // Output: ABC

34. Strings and Streams (Java 8)

Filtering Characters

java

Copy

Edit

```
String str = "hello123";
String lettersOnly = str.chars()
    .filter(Character::isLetter)
    .collect(StringBuilder::new,
        StringBuilder::appendCodePoint,
        StringBuilder::append)
    .toString();
System.out.println(lettersOnly); // Output: hello
```

Sorting Characters

java

Copy

Edit

```
String str = "cba";
String sorted = str.chars()
    .sorted()
    .collect(StringBuilder::new,
        StringBuilder::appendCodePoint,
        StringBuilder::append)
    .toString();
System.out.println(sorted); // Output: abc
```

35. Palindrome Check

Check if a String is a Palindrome

java

Copy

Edit

```
public static boolean isPalindrome(String str) {
   int start = 0;
   int end = str.length() - 1;
   while (start < end) {
      if (str.charAt(start) != str.charAt(end)) {
         return false;
      }
      start++;
      end--;
   }
   return true;
}

public static void main(String[] args) {
   String str = "madam";
   System.out.println(isPalindrome(str)); // Output: true
}
```

36. Escaping Special Characters

Escape Characters in Strings

\n - Newline

\t - Tab

\" - Double quote

\\ - Backslash

Example:

java

Copy

Edit

```
String str = "He said, \"Java is fun!\"\nLet's code.";

System.out.println(str);

// Output:

// He said, "Java is fun!"

// Let's code.
```

37. String Immutability

The immutability of strings ensures that any modification creates a new object.

java

Copy

Edit

```
String str = "Hello";

String modified = str.replace("H", "J");

System.out.println(str);     // Output: Hello

System.out.println(modified); // Output: Jello
```

This property makes strings thread-safe and memory-efficient when used properly with String Pooling.

Here’s the complete explanation with examples for all methods of StringBuilder, as requested.

StringBuilder Methods with Examples

append Methods

StringBuilder append(Object obj)

```
StringBuilder sb = new StringBuilder("Hello");
sb.append(123);  // Appends the object's string representation
System.out.println(sb);  // Output: Hello123
```

StringBuilder append(String str)

```
StringBuilder sb = new StringBuilder("Hello");
sb.append(" World");
System.out.println(sb);  // Output: Hello World
```

StringBuilder append(char[] str)

```
char[] chars = {'J', 'a', 'v', 'a'};
StringBuilder sb = new StringBuilder("Learn ");
sb.append(chars);
System.out.println(sb);  // Output: Learn Java
```

StringBuilder append(char[] str, int offset, int len)

```
char[] chars = {'J', 'a', 'v', 'a', 'S', 't', 'r', 'i', 'n', 'g'};
StringBuilder sb = new StringBuilder("Learn ");
sb.append(chars, 0, 4);  // Appends the first 4 characters
```

```
System.out.println(sb);  // Output: Learn Java
```

StringBuilder append(CharSequence s)

```
CharSequence seq = "Programming";
StringBuilder sb = new StringBuilder("Java ");
sb.append(seq);
System.out.println(sb);  // Output: Java Programming
```

StringBuilder append(CharSequence s, int start, int end)

```
CharSequence seq = "Programming";
StringBuilder sb = new StringBuilder("Java ");
sb.append(seq, 0, 4);  // Appends "Prog" (0-4)
System.out.println(sb);  // Output: Java Prog
```

StringBuilder append(boolean b)

```
StringBuilder sb = new StringBuilder("Condition: ");
sb.append(true);
System.out.println(sb);  // Output: Condition: true
```

StringBuilder append(char c)

```
StringBuilder sb = new StringBuilder("A");
sb.append('B');
System.out.println(sb);  // Output: AB
```

StringBuilder append(int i)

```
StringBuilder sb = new StringBuilder("Number: ");
```

```
sb.append(100);
System.out.println(sb);  // Output: Number: 100
```

StringBuilder append(long lng)

```
StringBuilder sb = new StringBuilder("Big number: ");
sb.append(123456789012345L);
System.out.println(sb);  // Output: Big number: 123456789012345
```

StringBuilder append(float f)

```
StringBuilder sb = new StringBuilder("Pi: ");
sb.append(3.14f);
System.out.println(sb);  // Output: Pi: 3.14
```

StringBuilder append(double d)

```
StringBuilder sb = new StringBuilder("Value: ");
sb.append(9.81);
System.out.println(sb);  // Output: Value: 9.81
```

StringBuilder appendCodePoint(int codePoint)

```
StringBuilder sb = new StringBuilder("Emoji: ");
sb.appendCodePoint(0x1F600);  // Appends 😀
System.out.println(sb);  // Output: Emoji: 😀
```

Capacity and Length

int capacity()

```
StringBuilder sb = new StringBuilder();
```

```
System.out.println(sb.capacity());  // Default capacity: 16
sb.append("Java");
System.out.println(sb.capacity());  // Capacity remains same if string length < capacity
```

int length()

```
StringBuilder sb = new StringBuilder("Java");
System.out.println(sb.length());  // Output: 4
```

void ensureCapacity(int minimumCapacity)

```
StringBuilder sb = new StringBuilder();
sb.ensureCapacity(50);
System.out.println(sb.capacity());  // Output: 50
```

Character Access

char charAt(int index)

```
StringBuilder sb = new StringBuilder("Java");
System.out.println(sb.charAt(2));  // Output: v
```

IntStream chars()

```
StringBuilder sb = new StringBuilder("Java");
sb.chars().forEach(c -> System.out.print((char)c + " "));  // Output: J a v a
```

IntStream codePoints()

```
StringBuilder sb = new StringBuilder("Java");
sb.codePoints().forEach(c -> System.out.print(c + " "));  // Output: Unicode values
```

Get and Insert Characters

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

StringBuilder sb = new StringBuilder("HelloWorld");

char[] dst = new char[5];

sb.getChars(0, 5, dst, 0);

System.out.println(dst); // Output: Hello

Insert Methods

StringBuilder sb = new StringBuilder("Java");

sb.insert(4, "Script"); // Inserts at position 4

System.out.println(sb); // Output: JavaScript

Search

int indexOf(String str)

StringBuilder sb = new StringBuilder("JavaScript");

System.out.println(sb.indexOf("Script")); // Output: 4

int lastIndexOf(String str)

StringBuilder sb = new StringBuilder("JavaScript");

System.out.println(sb.lastIndexOf("Script")); // Output: 4

Mutations

StringBuilder delete(int start, int end)

StringBuilder sb = new StringBuilder("HelloWorld");

```
sb.delete(5, 10);
System.out.println(sb);  // Output: Hello
```

StringBuilder deleteCharAt(int index)

```
StringBuilder sb = new StringBuilder("Hello");
sb.deleteCharAt(4);
System.out.println(sb);  // Output: Hell
```

StringBuilder replace(int start, int end, String str)

```
StringBuilder sb = new StringBuilder("HelloWorld");
sb.replace(5, 10, "Java");
System.out.println(sb);  // Output: HelloJava
```

StringBuilder reverse()

```
StringBuilder sb = new StringBuilder("Hello");
sb.reverse();
System.out.println(sb);  // Output: olleH
```

void setCharAt(int index, char ch)

```
StringBuilder sb = new StringBuilder("Hello");
sb.setCharAt(0, 'J');
System.out.println(sb);  // Output: Jello
```

Substrings and Subsequence

CharSequence subSequence(int start, int end)

```
StringBuilder sb = new StringBuilder("HelloWorld");
System.out.println(sb.subSequence(0, 5));  // Output: Hello
```

String substring(int start)

```
StringBuilder sb = new StringBuilder("HelloWorld");
System.out.println(sb.substring(5));  // Output: World
```

Utility

String toString()

```
StringBuilder sb = new StringBuilder("Java");
String str = sb.toString();
System.out.println(str);  // Output: Java
```

void trimToSize()

```
StringBuilder sb = new StringBuilder(50);
sb.append("Hello");
sb.trimToSize();
System.out.println(sb.capacity());  // Output: 5
```

Would you like to dive deeper into any specific method?

Here are examples for each of the `StringBuffer` methods you mentioned:

1. `StringBuffer append(Object obj)`

Appends the string representation of the specified object to the current buffer.

```
java
Copy
StringBuffer sb = new StringBuffer("Hello ");
sb.append(123);
System.out.println(sb);  // Output: Hello 123
```

2. `StringBuffer append(String str)`

Appends the specified string to the current buffer.

```
java
```

```
Copy
StringBuffer sb = new StringBuffer("Hello ");
sb.append("World!");
System.out.println(sb);  // Output: Hello World!
```

3. StringBuffer append(char[] str)

Appends the specified character array to the current buffer.

```
java
Copy
StringBuffer sb = new StringBuffer("Hello ");
char[] arr = {'W', 'o', 'r', 'l', 'd'};
sb.append(arr);
System.out.println(sb);  // Output: Hello World
```

4. StringBuffer append(char[] str, int offset, int len)

Appends a portion of the specified character array to the current buffer.

```
java
Copy
StringBuffer sb = new StringBuffer("Hello ");
char[] arr = {'W', 'o', 'r', 'l', 'd'};
sb.append(arr, 1, 3);  // Appends "orl"
System.out.println(sb);  // Output: Hello orl
```

5. StringBuffer append(CharSequence s)

Appends the specified `CharSequence` to the current buffer.

```
java
Copy
StringBuffer sb = new StringBuffer("Hello ");
CharSequence cs = "Java!";
sb.append(cs);
```

```
System.out.println(sb);  // Output: Hello Java!
```

6. StringBuffer append(CharSequence s, int start, int end)

Appends a subsequence of the specified CharSequence to the current buffer.

```
java
Copy
StringBuffer sb = new StringBuffer("Hello ");
CharSequence cs = "Java Programming";
sb.append(cs, 0, 4);  // Appends "Java"
System.out.println(sb);  // Output: Hello Java
```

7. StringBuffer append(boolean b)

Appends the string representation of the specified boolean value to the current buffer.

```
java
Copy
StringBuffer sb = new StringBuffer("Is it true? ");
sb.append(true);
System.out.println(sb);  // Output: Is it true? true
```

8. StringBuffer append(char c)

Appends the specified character to the current buffer.

```
java
Copy
StringBuffer sb = new StringBuffer("Hello ");
sb.append('A');
System.out.println(sb);  // Output: Hello A
```

9. StringBuffer append(int i)

Appends the string representation of the specified integer to the current buffer.

```
java
Copy
StringBuffer sb = new StringBuffer("Number: ");
sb.append(456);
System.out.println(sb);  // Output: Number: 456
```

10. StringBuffer append(long lng)

Appends the string representation of the specified long value to the current buffer.

```
java
Copy
StringBuffer sb = new StringBuffer("Large number: ");
sb.append(9876543210L);
System.out.println(sb);  // Output: Large number: 9876543210
```

11. StringBuffer append(float f)

Appends the string representation of the specified float value to the current buffer.

```
java
Copy
StringBuffer sb = new StringBuffer("Float value: ");
sb.append(3.14f);
System.out.println(sb);  // Output: Float value: 3.14
```

12. StringBuffer append(double d)

Appends the string representation of the specified double value to the current buffer.

```
java
Copy
StringBuffer sb = new StringBuffer("Pi: ");
sb.append(3.14159);
System.out.println(sb);  // Output: Pi: 3.14159
```

13. StringBuffer appendCodePoint(int codePoint)

Appends the character (represented by the given code point) to the current buffer.

```
java
Copy
StringBuffer sb = new StringBuffer("Codepoint: ");
sb.appendCodePoint(9733);  // Unicode for a star character (★)
System.out.println(sb);  // Output: Codepoint: ★
```

14. int capacity()

Returns the current capacity of the `StringBuffer`.

```
java
Copy
StringBuffer sb = new StringBuffer("Hello");
System.out.println(sb.capacity());  // Output: 20 (default capacity)
```

15. char charAt(int index)

Returns the character at the specified index.

```
java
Copy
StringBuffer sb = new StringBuffer("Hello");
System.out.println(sb.charAt(1));  // Output: e
```

16. void ensureCapacity(int minimumCapacity)

Ensures that the capacity is at least the specified minimum.

```
java
Copy
StringBuffer sb = new StringBuffer();
sb.ensureCapacity(50);
```

```
System.out.println(sb.capacity());  // Output: 50
```

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

Copies characters from the specified range to a character array.

```
java
Copy
StringBuffer sb = new StringBuffer("Hello, World!");
char[] dst = new char[5];
sb.getChars(7, 12, dst, 0);
System.out.println(new String(dst));  // Output: World
```

18. int indexOf(String str)

Returns the index of the first occurrence of the specified string.

```
java
Copy
StringBuffer sb = new StringBuffer("Hello World");
System.out.println(sb.indexOf("World"));  // Output: 6
```

19. int indexOf(String str, int fromIndex)

Returns the index of the first occurrence of the specified string, starting from a specific index.

```
java
Copy
StringBuffer sb = new StringBuffer("Hello World");
System.out.println(sb.indexOf("o", 5));  // Output: 7
```

20. StringBuffer insert(int offset, char[] str, int index, int len)

Inserts a portion of the character array at the specified position.

```
java
Copy
StringBuffer sb = new StringBuffer("Hello ");
char[] arr = {'W', 'o', 'r', 'l', 'd'};
sb.insert(6, arr, 1, 3);  // Inserts "orl"
System.out.println(sb);  // Output: Hello orl
```

21. StringBuffer insert(int offset, Object obj)

Inserts the string representation of the specified object at the specified position.

```
java
Copy
StringBuffer sb = new StringBuffer("Hello ");
sb.insert(6, 123);  // Inserts the string "123"
System.out.println(sb);  // Output: Hello 123
```

22. StringBuffer insert(int offset, String str)

Inserts the specified string at the specified position.

```
java
Copy
StringBuffer sb = new StringBuffer("Hello ");
sb.insert(6, "World");
System.out.println(sb);  // Output: Hello World
```

23. StringBuffer insert(int offset, char[] str)

Inserts the specified character array at the specified position.

```
java
```

```
Copy
StringBuffer sb = new StringBuffer("Hello ");
char[] arr = {'W', 'o', 'r', 'l', 'd'};
sb.insert(6, arr);
System.out.println(sb);  // Output: Hello World
```

24. StringBuffer insert(int offset, CharSequence s)

Inserts the specified CharSequence at the specified position.

```
java
Copy
StringBuffer sb = new StringBuffer("Hello ");
CharSequence cs = "Java!";
sb.insert(6, cs);
System.out.println(sb);  // Output: Hello Java!
```

25. StringBuffer insert(int offset, CharSequence s, int start, int end)

Inserts a subsequence of the specified CharSequence at the specified position.

```
java
Copy
StringBuffer sb = new StringBuffer("Hello ");
CharSequence cs = "Java Programming";
sb.insert(6, cs, 0, 4);  // Inserts "Java"
System.out.println(sb);  // Output: Hello Java
```

26. StringBuffer insert(int offset, boolean b)

Inserts the string representation of the specified boolean at the specified position.

```
java
Copy
StringBuffer sb = new StringBuffer("Is it true? ");
sb.insert(14, false);
```

```
System.out.println(sb);  // Output: Is it true? false
```

27. StringBuffer insert(int offset, char c)

Inserts the specified character at the specified position.

```
java
Copy
StringBuffer sb = new StringBuffer("Hello ");
sb.insert(6, 'W');
System.out.println(sb);  // Output: Hello W
```

28. StringBuffer insert(int offset, int i)

Inserts the string representation of the specified integer at the specified position.

```
java
Copy
StringBuffer sb = new StringBuffer("Number: ");
sb.insert(8, 456);
System.out.println(sb);  // Output: Number: 456
```

29. StringBuffer insert(int offset, long l)

Inserts the string representation of the specified long value at the specified position.

```
java
Copy
StringBuffer sb = new StringBuffer("Large number: ");
sb.insert(15, 9876543210L);
System.out.println(sb);  // Output: Large number: 9876543210
```

30. StringBuffer insert(int offset, float f)

Inserts the string representation of the specified float value at the specified position.

```
java
Copy
StringBuffer sb = new StringBuffer("Float value: ");
sb.insert(13, 3.14f);
System.out.println(sb);  // Output: Float value: 3.14
```

31. StringBuffer insert(int offset, double d)

Inserts the string representation of the specified double value at the specified position.

```
java
Copy
StringBuffer sb = new StringBuffer("Pi: ");
sb.insert(4, 3.14159);
System.out.println(sb);  // Output: Pi: 3.14159
```

32. int lastIndexOf(String str)

Returns the index of the last occurrence of the specified string.

```
java
Copy
StringBuffer sb = new StringBuffer("Hello World, Hello!");
System.out.println(sb.lastIndexOf("Hello"));  // Output: 13
```

33. int lastIndexOf(String str, int fromIndex)

Returns the index of the last occurrence of the specified string, starting from a specific index.

```
java
Copy
StringBuffer sb = new StringBuffer("Hello World, Hello!");
System.out.println(sb.lastIndexOf("Hello", 10));  // Output: 0
```

34. int length()

Returns the length of the `StringBuffer`.

```
java
Copy
StringBuffer sb = new StringBuffer("Hello");
System.out.println(sb.length());  // Output: 5
```

35. StringBuffer delete(int start, int end)

Deletes the substring from the start index to the end index (exclusive).

```
java
Copy
StringBuffer sb = new StringBuffer("Hello World");
sb.delete(5, 11);  // Deletes " World"
System.out.println(sb);  // Output: Hello
```

36. StringBuffer deleteCharAt(int index)

Deletes the character at the specified index.

```
java
Copy
StringBuffer sb = new StringBuffer("Hello");
sb.deleteCharAt(1);  // Deletes 'e'
System.out.println(sb);  // Output: Hllo
```

37. StringBuffer replace(int start, int end, String str)

Replaces the substring from the start index to the end index with the specified string.

```
java
Copy
StringBuffer sb = new StringBuffer("Hello World");
sb.replace(6, 11, "Java");
```

```
System.out.println(sb);  // Output: Hello Java
```

38. StringBuffer reverse()

Reverses the characters in the `StringBuffer`.

```
java
Copy
StringBuffer sb = new StringBuffer("Hello");
sb.reverse();
System.out.println(sb);  // Output: olleH
```

39. void setCharAt(int index, char ch)

Sets the character at the specified index to the given character.

```
java
Copy
StringBuffer sb = new StringBuffer("Hello");
sb.setCharAt(1, 'a');  // Changes 'e' to 'a'
System.out.println(sb);  // Output: Hallo
```

40. void setLength(int newLength)

Sets the length of the `StringBuffer` to the specified value.

```
java
Copy
StringBuffer sb = new StringBuffer("Hello World");
sb.setLength(5);
System.out.println(sb);  // Output: Hello
```

41. CharSequence subSequence(int start, int end)

Returns a subsequence from the start index to the end index.

```
java
Copy
StringBuffer sb = new StringBuffer("Hello World");
System.out.println(sb.subSequence(6, 11));  // Output: World
```

42. String substring(int start)

Returns a substring starting from the specified index to the end of the string.

```
java
Copy
StringBuffer sb = new StringBuffer("Hello World");
System.out.println(sb.substring(6));  // Output: World
```

43. String substring(int start, int end)

Returns a substring from the specified start index to the specified end index.

```
java
Copy
StringBuffer sb = new StringBuffer("Hello World");
System.out.println(sb.substring(6, 11));  // Output: World
```

44. String toString()

Converts the `StringBuffer` to a `String`.

```
java
Copy
StringBuffer sb = new StringBuffer("Hello");
String str = sb.toString();
System.out.println(str);  // Output: Hello
```

45. void trimToSize()

Trims the capacity of the `StringBuffer` to the current length.

```
java
Copy
StringBuffer sb = new StringBuffer("Hello World");
sb.trimToSize();
System.out.println(sb.capacity());  // Output: 11 (equal to length)
```

Let me know if you'd like further clarification!