Strings

String Handling in Java

String handling is one of the most essential parts of Java programming because strings are used frequently in applications. Java provides a robust set of classes and methods to work with strings efficiently.

---

1. String Class

In Java, a String is an object that represents a sequence of characters. Strings in Java are immutable, meaning their values cannot be changed after they are created.

Key Features:

- Immutable (cannot be modified).
- Stored in the **String Constant Pool** for memory optimization.
- Comes with a wide range of methods for string manipulation.

Creating Strings:

String str1 = "Hello"; // String literal

String str2 = new String("World"); // Using the 'new' keyword

Common Methods in the String Class:

Method	**Description**
length()	Returns the length of the string.
charAt(index)	Returns the character at the specified index.
substring(start, end)	Extracts a substring from the string.
toUpperCase()	Converts all characters to uppercase.
toLowerCase()	Converts all characters to lowercase.
trim()	Removes leading and trailing spaces.
equals()	Compares two strings for equality (case-sensitive).
equalsIgnoreCase()	Compares two strings for equality (case-insensitive).
contains(substring)	Checks if the string contains the specified substring.
replace(oldChar, newChar)	Replaces occurrences of one character with another.

Example: String Class

```
public class StringExample {
    public static void main(String[] args) {
        String str = "  Hello, Java!  ";


        // Length of the string
        System.out.println("Length: " + str.length());
        // Trim spaces
        System.out.println("Trimmed String: '" + str.trim() + "'");

        // Convert to uppercase
        System.out.println("Uppercase: " + str.toUpperCase());

        // Substring
        System.out.println("Substring: " + str.substring(8, 12)); // Output: "Java"

        // Check equality
        System.out.println("Equals 'HELLO, JAVA!': " + str.trim().equalsIgnoreCase("HELLO, JAVA!"));
    }
}
```

Output:

```
Length: 14
Trimmed String: 'Hello, Java!'
Uppercase:   HELLO, JAVA!
Substring: Java
Equals 'HELLO, JAVA!': true
```

2. StringBuffer and StringBuilder

The StringBuffer and StringBuilder classes are used to create mutable strings, meaning their values can be modified after creation.

Key Differences Between String, StringBuffer, and StringBuilder:

Feature	**String**	**StringBuffer**	**StringBuilder**
Mutability	Immutable	Mutable	Mutable
Thread-Safe	No	Yes	No
Performance	Fast	Slower (due to thread-safety)	Fast

StringBuffer:

- Thread-safe and synchronized.
- Suitable for multithreaded applications.

Example:

```
public class StringBufferExample {
    public static void main(String[] args) {
        StringBuffer sb = new StringBuffer("Hello");

        // Append
        sb.append(" World");
        System.out.println("After Append: " + sb);

        // Insert
        sb.insert(5, ",");

        System.out.println("After Insert: " + sb);



        // Reverse

        sb.reverse();

        System.out.println("Reversed: " + sb);

    }

}
```

Output:

```
After Append: Hello World

After Insert: Hello, World

Reversed: dlroW ,olleH
```

StringBuilder:

- Not thread-safe but faster than StringBuffer.
- Suitable for single-threaded applications.

Example:

```
public class StringBuilderExample {

    public static void main(String[] args) {

        StringBuilder sb = new StringBuilder("Java");



        // Append
        sb.append(" Programming");
        System.out.println("After Append: " + sb);


        // Replace
        sb.replace(0, 4, "Python");
        System.out.println("After Replace: " + sb);


        // Delete
        sb.delete(7, 18);
        System.out.println("After Delete: " + sb);
    }
}
```

Output:

```
After Append: Java Programming
After Replace: Python Programming
After Delete: Python
```

---

3. StringTokenizer

The StringTokenizer class is used to split a string into tokens based on a delimiter. It is part of the java.util package.

Key Features:

- Older approach for splitting strings (replaced by split() method in modern Java).
- Delimiters can be specified to break the string.

Common Methods:

Method	**Description**
hasMoreTokens()	Checks if more tokens are available.
nextToken()	Returns the next token.
countTokens()	Returns the total number of tokens.

Example: StringTokenizer

```
import java.util.StringTokenizer;


public class StringTokenizerExample {
    public static void main(String[] args) {
        String data = "Java,Python,C++,JavaScript";


        // Create a tokenizer with ',' as the delimiter
        StringTokenizer st = new StringTokenizer(data, ",");


        // Print tokens
        System.out.println("Tokens:");
        while (st.hasMoreTokens()) {
            System.out.println(st.nextToken());
        }
    }
}
```

Output:

```
Tokens:
Java
Python
C++
JavaScript
```

Key Takeaways

1. **String Class** is best for immutable text processing.
2. **StringBuffer** is thread-safe for mutable strings but slower.
3. **StringBuilder** is faster and ideal for mutable strings in single-threaded applications.
4. **StringTokenizer** is useful for splitting strings based on custom delimiters but is less common in modern Java.

If you have specific questions or need more examples, feel free to ask!