String in Java

In Java, strings are a crucial part of programming and are used to represent sequences of characters.

Java provides a robust `String` class for manipulating strings, which is part of the `java.lang` package.

Here's a comprehensive overview of strings in Java:

1. Creating Strings

You can create strings using string literals or by using the `String` class.

String Literals:

```
String str1 = "Hello, World!";
```

Using the `String` Constructor:

```
String str2 = new String("Hello, World!");
```

2. Common String Methods

The `String` class provides a rich set of methods for string manipulation. Here are some commonly used methods:

Length of a String:

```
String str = "Hello";
int length = str.length(); // 5
```

Concatenation:

```
String str1 = "Hello";
String str2 = "World";
String result = str1 + " " + str2; // "Hello World"
```

Substring:

```
String str = "Hello, World!";
String substr = str.substring(7, 12); // "World"
```

Character at a Specific Index:

```
char ch = str.charAt(1); // 'e'
```

Index of a Substring:

```
int index = str.indexOf("World"); // 7
```

Replace Characters:

```
String str = "Hello, World!";
String replaced = str.replace("World", "Java"); // "Hello, Java!"
```

Convert to Uppercase/Lowercase:

```
String str = "Hello, World!";
String upper = str.toUpperCase(); // "HELLO, WORLD!"
String lower = str.toLowerCase(); // "hello, world!"
```

Trim Whitespace:

```
String str = "   Hello, World!   ";
String trimmed = str.trim(); // "Hello, World!"
```

Check if String Contains a Substring:

```
boolean contains = str.contains("World"); // true
```

Split String:

```
String str = "apple,banana,cherry";
String[] fruits = str.split(","); // ["apple", "banana", "cherry"]
```

3. String Immutability

Strings in Java are immutable. This means that once a `String` object is created, it cannot be changed. Any modification creates a new `String` object. For example:

```
String str1 = "Hello";
String str2 = str1.concat(", World!"); // Creates a new String "Hello, World!"
```

4. StringBuilder and StringBuffer

For mutable sequences of characters, Java provides `StringBuilder` and `StringBuffer`:

String Builder:

- **Usage:** Use `StringBuilder` when you need a mutable string and don't require synchronization (i.e., not in a multithreaded environment).
- **Example:**

```
StringBuilder sb = new StringBuilder("Hello");
sb.append(", World!");
String result = sb.toString(); // "Hello, World!"
```

String Buffer:

- **Usage:** Use `StringBuffer` when you need a mutable string and require thread safety.
- **Example:**

```
StringBuffer sb = new StringBuffer("Hello");
sb.append(", World!");
String result = sb.toString(); // "Hello, World!"
```

5. String Formatting

You can format strings using the `String.format()` method or the `printf` method in `PrintStream`.

Using `String.format()`:

```
String formatted = String.format("Name: %s, Age: %d", "Alice", 30);
        // "Name: Alice, Age: 30"
```

Using `System.out.printf()`:

```
System.out.printf("Name: %s, Age: %d%n", "Alice", 30);
```

6. String Pool

Java maintains a pool of strings to optimize memory usage. When you create a string literal, Java checks the string pool to see if an identical string already exists. If it does, it reuses the existing string object.

```
String s1 = "Java";
String s2 = "Java";

boolean isSame = (s1 == s2); // true, bcz both refer to the same object in the string pool
```

Example - Program

Here's a simple Java program that demonstrates some of the string functionalities:

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

        // Print length of string
        System.out.println("Length: " + str.length());

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

        // Replace substring
        String newStr = str.replace("World", "Java");
        System.out.println("Replaced: " + newStr);

    // Check if string contains a substring
        System.out.println("Contains 'World': " + str.contains("World"));

       // Split string
        String[] parts = str.split(", ");
        for (String part : parts) {
            System.out.println("Part: " + part);
        }
    }
}
```

This covers the basics of working with strings in Java.