- A **String** is an object representing a sequence of characters.
- Part of the `java.lang.String` class.
- Strings are **immutable**.

Creating Strings:

- Using String literals:

```
1 String str1 = "Hello, World!";
2
```

- Using the `new` keyword:

```
1 String str2 = new String("Hello, World!");
2
```

2. Important String Methods

- **Length**: `str.length()`
- **Concatenation**: `str1 + str2` or `str1.concat(str2)`
- **Character Access**: `str.charAt(index)`
- **Substring**: `str.substring(start, end)`
- **Case Conversion**: `str.toUpperCase()`, `str.toLowerCase()`
- **Trimming Whitespace**: `str.trim()`
- **Replacing Characters**: `str.replace(old, new)`
- **Splitting Strings**: `str.split(delimiter)`

3. Immutability of Strings

- Strings are **immutable**.
- Any modification creates a new string object.
- Why immutability?
  - Thread safety
  - Security
  - Caching

Immutability Demonstration

- Strings cannot be modified after creation.

```
String s1 = "Java";
s1.concat(" Programming"); // No change to s1
System.out.println(s1); // Output: Java

```

Example:

```
String str = "Java";
str = str + " Programming"; // Creates a new string

```

Length and Accessing Characters

- `length()` method.
- `charAt(index)` method.

```
String name = "Alice";
System.out.println(name.length()); // Output: 5
System.out.println(name.charAt(1)); // Output: l

```

Concatenation

- Using + operator or `concat()` method.

```
1 String str1 = "Good";
2 String str2 = "Morning";
3 String result = str1 + " " + str2; // Output: Good
    Morning
4
```

Substring and Searching

- `substring(startIndex, endIndex)`.
- `indexOf()` and `lastIndexOf()`.

```
1 String message = "Welcome to Java";
2 System.out.println(message.substring(0, 7)); // Output
      : Welcome
3 System.out.println(message.indexOf("a")); // Output:
      11
4
```

Case Conversion and Trimming

- `toLowerCase()` and `toUpperCase()`.
- `trim()` method.

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

String padded = "   Hello   ";
System.out.println(padded.trim()); // Output: Hello

```

Regular Expressions

- Using `matches()` and `replaceAll()` with regex.

```
String email = "test@example.com";
System.out.println(email.matches(".*@.*\\.com")); // Output: true

```

StringBuilder

- Mutable alternatives to String.

```
1 StringBuilder sb = new StringBuilder("Hello");
2 sb.append(" World");
3 System.out.println(sb.toString()); // Output: Hello
      World
4
```

- Using == instead of `.equals()`.
- Ignoring immutability (use `StringBuilder` for frequent modifications).
- Regex mistakes (escape special characters properly).

Basic Exercises

- What is the difference between == and .equals() when comparing two strings? Provide an example
- Write a program to count the number of characters in a given string, excluding spaces.
- Why does the following code not modify the original string?
  - String str = "Java";
  - str.concat(" Programming");
  - System.out.println(str); // Output: Java

Basic Exercises

- Write a program to check if a given string is a palindrome (reads the same backward as forward).
- Write a program to find the frequency of each character in a string.
- How would you find all occurrences of a substring in a given string?
- Write a program to find the length of the longest substring without repeating characters.
- Write a program to check if two strings are anagrams (contain the same characters in a different order).
- Write a program to validate an email address using regular expressions.
- Implement a Caesar cipher to encrypt and decrypt a string by shifting characters by a fixed number.

Advanced

- Write a program to validate a password based on the following criteria: At least 8 characters long. Contains at least one uppercase letter, one lowercase letter, one digit, and one special character.
- Write a program to read a text file, replace all occurrences of a specific word, and write the modified content back to the file.
- Create a custom method to reverse a string without using StringBuilder or any built-in reverse functions.

Books and Online Platforms

- Books: "Head First Java", "Java: The Complete Reference".
- Platforms: LeetCode, HackerRank, Oracle's official documentation.