**Session: Strings in Java**

**Introduction**

In Java, strings are one of the most commonly used data types. A string represents a sequence of characters and is immutable (cannot be changed once created). The String class in Java provides a wide variety of methods to manipulate and analyze strings.

**Session Objectives**

1. Understand how strings are created and stored in Java.
2. Learn about commonly used methods of the String class.
3. Discuss string immutability and its impact.
4. Explore examples and hands-on exercises.

**Topics Covered**

1. String Basics
2. Creating Strings
3. Common String Methods
4. String Immutability
5. Hands-on Exercises

**1. String Basics**

* A String in Java is an object that represents a sequence of characters.
* Strings are immutable, meaning once created, their values cannot be changed.
* The String class is part of java.lang package, so it doesn’t need to be imported.

**2. Creating Strings**

There are two ways to create strings:

1. **Using String Literals**:  
   Stored in the String Pool for memory efficiency.

java

String str1 = "Hello";

1. **Using the new Keyword**:  
   Creates a new string object every time in the heap memory.

java

String str2 = new String("Hello");

**Example**:

java

public class StringExample {

public static void main(String[] args) {

String s1 = "Java";

String s2 = new String("Java");

System.out.println(s1.equals(s2)); // true

}

}

**3. Common String Methods**

| **Method** | **Description** | **Example** |
| --- | --- | --- |
| length() | Returns the length of the string. | "Hello".length() → 5 |
| charAt(int index) | Returns the character at the specified index. | "Java".charAt(1) → 'a' |
| substring(int start) | Extracts a substring from the start index. | "Welcome".substring(3) → "come" |
| substring(int start, int end) | Extracts a substring from start to end (exclusive). | "Welcome".substring(0, 3) → "Wel" |
| toUpperCase() | Converts the string to uppercase. | "java".toUpperCase() → "JAVA" |
| toLowerCase() | Converts the string to lowercase. | "JAVA".toLowerCase() → "java" |
| trim() | Removes leading and trailing spaces. | " Hello ".trim() → "Hello" |
| replace(char, char) | Replaces all occurrences of one character with another. | "abc".replace('a', 'z') → "zbc" |
| contains(CharSequence) | Checks if the string contains a sequence. | "hello".contains("ell") → true |
| equals(Object) | Compares strings for equality. | "Java".equals("java") → false |
| equalsIgnoreCase(String) | Compares strings ignoring case differences. | "Java".equalsIgnoreCase("java") → true |
| startsWith(String) | Checks if the string starts with the specified prefix. | "Java".startsWith("Ja") → true |
| endsWith(String) | Checks if the string ends with the specified suffix. | "Java".endsWith("va") → true |
| indexOf(char) | Returns the first occurrence of the character. | "Hello".indexOf('l') → 2 |
| lastIndexOf(char) | Returns the last occurrence of the character. | "Hello".lastIndexOf('l') → 3 |
| split(String) | Splits the string into an array based on the regex. | "a,b,c".split(",") → ["a", "b", "c"] |
| isEmpty() | Checks if the string is empty. | "".isEmpty() → true |

**4. String Immutability**

* When you modify a string, a new string object is created, and the original remains unchanged.  
  **Example**:

java

String s = "Java";

s.concat(" Programming"); // New object created, not assigned

System.out.println(s); // Output: Java

**5. Hands-On Exercises**

1. **Task**: Find the first and last occurrence of a character in a given string.  
   **Example**: "hello" → 'l' is at index 2 and 3.

**Code**:

java

public class StringTask {

public static void main(String[] args) {

String str = "hello";

System.out.println("First 'l': " + str.indexOf('l'));

System.out.println("Last 'l': " + str.lastIndexOf('l'));

}

}

1. **Task**: Write a method to count the number of vowels in a string.  
   **Example**: "Java Programming" → 5 vowels.

**Code**:

java

public class VowelCounter {

public static void main(String[] args) {

String str = "Java Programming";

int count = 0;

for (char c : str.toLowerCase().toCharArray()) {

if ("aeiou".indexOf(c) != -1) {

count++;

}

}

System.out.println("Vowel count: " + count);

}

}

1. **Task**: Reverse a given string using StringBuilder.  
   **Code**:

java

public class StringReverse {

public static void main(String[] args) {

String str = "Java";

String reversed = new StringBuilder(str).reverse().toString();

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

}

}

**Common Interview Questions**

1. **What is the difference between == and equals() in Java?**
   * == compares memory addresses.
   * equals() compares the contents of strings.
2. **Why are strings immutable in Java?**
   * For security, caching, and thread safety.
3. **What is the difference between String, StringBuilder, and StringBuffer?**
   * String: Immutable.
   * StringBuilder: Mutable, not thread-safe.
   * StringBuffer: Mutable, thread-safe.

**Conclusion**

* Strings in Java are powerful and provide extensive functionality.
* Practice common methods to get comfortable working with strings.
* Remember the immutability aspect when designing applications.

**Any questions? Let’s discuss!**