String(Module-23)
Assignment

Q.1 import java.util.Scanner;

```
public class StringInput {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in); // Create a Scanner object for input

        System.out.print("Enter your name: ");
        String name = scanner.nextLine(); // Read a line of text (including spaces)

        System.out.print("Enter your age: ");
        int age = scanner.nextInt(); // Read an integer

        scanner.nextLine();//Consume newline left-over

        System.out.print("Enter your favorite quote: ");
        String quote = scanner.nextLine(); // Read another line of text

        System.out.println("\nYour Information:");
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
        System.out.println("Favorite Quote: " + quote);

        scanner.close(); // Close the scanner to release resources (important!)
    }
}
```

Q.2 There are several ways to concatenate (join together) two or more strings in Java:

1. Using the + Operator (Concatenation Operator)

- This is the most common and easiest way to concatenate strings.
- The + operator, when used with strings, performs string concatenation rather than addition.

  Ex:

  ```
  String str1 = "Hello";

  String str2 = "World";

  String result = str1 + " " + str2; // Concatenates str1, a space, and str2

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

```
String message = "The number is: " + 10; // Concatenating a String and an int

System.out.println(message); // Output: The number is: 10


String multiConcat = "One" + "Two" + "Three";

System.out.println(multiConcat); // Output: OneTwoThree
```

2. Using the `concat()` Method

- The `String` class provides the `concat()` method, which takes another string as an argument and returns a new string that is the concatenation of the two strings

```
String str1 = "Hello";

String str2 = "World";

String result = str1.concat(" ").concat(str2); // Concatenates str1, a space, and str2

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

3. Using `StringBuilder` (For Efficient Concatenation in Loops)

- If you are concatenating strings in a loop or performing many concatenation operations, using `StringBuilder` is much more efficient than using the `+` operator or `concat()`.
- `StringBuilder` is a mutable class, meaning it modifies the string in place rather than creating new String objects each time

```
StringBuilder sb = new StringBuilder(); // Creates an empty StringBuilder

sb.append("Hello"); // Appends "Hello"

sb.append(" ");     // Appends a space

sb.append("World"); // Appends "World"

String result = sb.toString(); // Converts the StringBuilder to a String

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


//Example in a loop:
```

```
StringBuilder messageBuilder = new StringBuilder("Numbers: ");

for (int i = 1; i <= 5; i++) {

    messageBuilder.append(i).append(" ");

}

String finalMessage = messageBuilder.toString();

System.out.println(finalMessage); // Output: Numbers: 1 2 3 4 5
```

Q.3 In Java, you can find the length of a string using the `length()` method.

Explanation:

- The `length()` method is a built-in method of the `String` class.
- It returns an integer value representing the number of characters in the string.
- Empty strings have a length of 0.

Example:

```
public class StringLengthExample {

    public static void main(String[] args) {

        String str = "Hello, World!";

        int length = str.length();



        System.out.println("The length of the string is: " + length);

        // Output: The length of the string is: 13

    }

}
```

In this example:

1. `str` is a String variable that holds the value "Hello, World!".
2. `str.length()` calls the `length()` method on the `str` object, which returns the number of characters in the string (13).
3. The output will be: "The length of the string is: 13"

Q.4 You can compare two strings in Java primarily using these methods:

1. `equals()` Method (For Content Equality)

- This is the most common and recommended way to compare strings in Java.
- The `equals()` method compares the *content* of the strings, character by character.
- It returns `true` if the strings have the same sequence of characters and `false` otherwise.
- It's case-sensitive.

```
String str1 = "Hello";

String str2 = "Hello";

String str3 = "hello";



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

System.out.println(str1.equals(str3)); // Output: false (different case)
```

2. `equalsIgnoreCase()` Method (For Case-Insensitive Comparison)

- This method is similar to `equals()`, but it ignores the case of the characters.
- It returns `true` if the strings have the same sequence of characters, regardless of case, and `false` otherwise.

```
String str1 = "Hello";

String str3 = "hello";



System.out.println(str1.equalsIgnoreCase(str3)); // Output: true (same content, ignoring case)
```

3. `==` Operator (For Reference Equality)

- The `==` operator compares the *references* (memory addresses) of the String objects, not their content.
- It returns `true` if the two variables refer to the *same* String object in memory, and `false` otherwise.
- Using `==` for String comparison is generally *not recommended* unless you specifically need to check if two variables point to the exact same object.

- String str1 = "Hello";   // String literal (goes to String Pool)
- String str2 = "Hello";   // String literal (points to the same object in the String Pool)
- String str3 = new String("Hello"); // Creates a new String object in the heap
-
- System.out.println(str1 == str2); // Output: true (same object in String Pool)
- System.out.println(str1 == str3); // Output: false (different objects in memory)

Q.5 program:

```
public class StringLength {

    public static void main(String[] args) {

        String str = "refrigerator";

        int length = str.length();


        System.out.println("The length of the string \"" + str + "\" is: " + length);

        // Output: The length of the string "refrigerator" is: 12

    }

}
```

Q.6 Program:

```
  public class CheckLetter {

    public static void main(String[] args) {

        String word = "Umbrella"; // The given word

        char letterToCheck = 'e'; // The letter to check


        // Check if the letter 'e' is present in the word

        if (word.contains(String.valueOf(letterToCheck))) {

            System.out.println("The letter '" + letterToCheck + "' is present in the word '" + word +
"'.");
```

```
        } else {
            System.out.println("The letter '" + letterToCheck + "' is not present in the word '" + word +
"'.");
        }
    }
}
```

Q.7 Program:

```
public class RemoveConsonants {
    public static void main(String[] args) {
        String input = "Hello, have a good day"; // The input string
        StringBuilder result = new StringBuilder(); // To store the result

        // Iterate through each character in the string
        for (int i = 0; i < input.length(); i++) {
            char c = input.charAt(i);

            // Check if the character is a vowel (a, e, i, o, u) or not an alphabet
            if ("aeiouAEIOU".indexOf(c) != -1 || !Character.isLetter(c)) {
                result.append(c); // Keep vowels and non-alphabet characters
            }
        }

        // Print the result
        System.out.println("String after removing consonants: " + result.toString());
    }
}
```