NETWORK MANAGEMENT SYSTEM

ASSIGNMENT-4: **INPUT HANDLING AND BASIC FUNCTIONALITY**

NAME: PAVITHRAA E S

CLASS AND SECTION: CSE-B

REG. NO: 211721104099

ROLL NO: 202101100

COLLEGE NAME: RAJALAKSHMI INSTITUTE OF TECHNOLOGY

TRAINER: Mr. Kripal Tripati

INPUT HANDLING AND BASIC FUNCTIONALITY

OBJECTIVE:

The objective of this assignment is to implement a program that can handle user inputs by taking in a hostname and an IP address, replacing placeholders in a template with these inputs, and returning a formatted output. Additionally, the assignment requires the development of test cases to validate the program's behaviour across various scenarios, ensuring correct functionality even with missing or invalid inputs.

INTRODUCTION:

In many applications, handling user inputs is crucial to providing a smooth and reliable user experience. This assignment focuses on creating a simple input-handling function that processes a hostname and IP address by integrating them into a predefined template. By doing so, the program should demonstrate input validation, error handling, and correct placeholder replacement. To confirm the robustness of this functionality, we will also develop test cases for several scenarios, including both valid and invalid inputs. This assignment emphasizes foundational skills in input handling, validation, and testing—key components in building resilient and user-friendly software solutions.

DESIGNING THE SOLUTION:

In this solution, we'll use a method in Java to handle user inputs for `hostname` and `IP address`, validate the inputs, and replace placeholders in a template string. Below are the main components:

1. Template Creation

We'll use a string template with placeholders for `hostname` and `IP address`. In Java, placeholders can be managed by using `String.format()` for easier substitution of values. Here's an example template:

```
java

"Hostname: %s, IP Address: %s"
```

The `%s` placeholders represent the `hostname` and `IP address` respectively. They will be replaced with actual values based on user input.

2. Code Explanation and Design

i. **Class and Method Overview:**

This code defines a single class, PlaceholderReplacer, with a main method that serves as the entry point. The main method uses a Scanner object to capture user input for three items:

a) A customizable template string with placeholders.
b) A hostname.
c) An IP address.

ii. **Process Breakdown:**

a) User Input: The program prompts the user to enter the template, hostname, and IP address. It uses `Scanner.nextLine()` to read each input.

b) Input Validation: The program checks if `hostname` or `ip_address` is empty. If either is empty, it prints an error message and exits.

c) **Placeholder Replacement:** If both inputs are valid, the program uses `String.replace()` to substitute `{hostname}` and `{ip_address}` placeholders in the template with the actual user-provided values.

d) **Output:** The formatted string is displayed as the final output.

iii.

```
java

public class PlaceholderReplacer {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);


// Prompt user for template, hostname, and IP address

System.out.print("Enter the template string: ");

String template = scanner.nextLine();


System.out.print("Enter the hostname: ");

String hostname = scanner.nextLine();


System.out.print("Enter the IP address: ");

String ipAddress = scanner.nextLine();


// Validate inputs for hostname and IP address

if (hostname.isEmpty()) {

    System.out.println("Hostname cannot be empty.");

    return;

}
```

EXPLANATION OF KEY COMPONENTS

i. User Input Collection: -

The program uses `Scanner` to take input from the user for `template`, `hostname`, and `IP address`.

`System.out.print()` is used to prompt the user for each input, while `scanner.nextLine()` reads the full line entered by the user.

ii. Input Validation: -

To ensure all necessary information is provided, the program checks if `hostname` or `ipAddress` is empty using `.isEmpty()`.

If `hostname` is empty, it outputs `"Hostname cannot be empty."` and terminates the program.

If `ipAddress` is empty, it outputs `"IP address cannot be empty."` and terminates the program.

iii. Placeholder Replacement: -

The program uses `String.replace()` twice to substitute `{hostname}` and `{ip_address}` with user-provided values.

First, `template.replace("{hostname}", hostname)` replaces `{hostname}` with the actual `hostname`.

Next, `.replace("{ip_address}", ipAddress)` replaces `{ip_address}` with the actual `ipAddress` in the same modified string.

iv. Output: -

The formatted result is printed with `System.out.println("Result: " + result);`, showing the template with placeholders replaced by the user's input values.

OUTPUTS

Main.java

```java
import java.util.Scanner;
public class PlaceholderReplacer {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the template string: ");
        String template = scanner.nextLine();
        System.out.print("Enter the hostname: ");
        String hostname = scanner.nextLine();
        System.out.print("Enter the IP address: ");
        String ipAddress = scanner.nextLine();
        // Validate input
        if (hostname.isEmpty()) {
            System.out.println("Hostname cannot be empty.");
            return;
        }
        if (ipAddress.isEmpty()) {
            System.out.println("IP address cannot be empty.");
            return;
        }
        // Replace placeholders
        String result = template.replace("{hostname}", hostname).replace
            ("{ip_address}", ipAddress);

        System.out.println("Result: " + result);
    }
}
```

Output

```
java -cp /tmp/pVs2ZC5I0n/PlaceholderReplacer
Enter the template string: Server example.com has IP address 192.168.1.1.
Enter the hostname: example.com
Enter the IP address: 192.168.1.1
Result: Server example.com has IP address 192.168.1.1.

=== Code Execution Successful ===
```

Main.java

```java
import java.util.Scanner;
public class PlaceholderReplacer {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the template string: ");
        String template = scanner.nextLine();
        System.out.print("Enter the hostname: ");
        String hostname = scanner.nextLine();
        System.out.print("Enter the IP address: ");
        String ipAddress = scanner.nextLine();
        // Validate input
        if (hostname.isEmpty()) {
            System.out.println("Hostname cannot be empty.");
            return;
        }
        if (ipAddress.isEmpty()) {
            System.out.println("IP address cannot be empty.");
            return;
        }
        // Replace placeholders
        String result = template.replace("{hostname}", hostname).replace
            ("{ip_address}", ipAddress);

        System.out.println("Result: " + result);
    }
}
```

Output

```
java -cp /tmp/6dlKXkllf6/PlaceholderReplacer
Enter the template string: Server {hostname} has IP address 192.168.1.1.
Enter the hostname:
Enter the IP address: 192.168.1.1
Hostname cannot be empty.

=== Code Execution Successful ===
```

Main.java

```java
import java.util.Scanner;
public class PlaceholderReplacer {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the template string: ");
        String template = scanner.nextLine();
        System.out.print("Enter the hostname: ");
        String hostname = scanner.nextLine();
        System.out.print("Enter the IP address: ");
        String ipAddress = scanner.nextLine();
        // Validate input
        if (hostname.isEmpty()) {
            System.out.println("Hostname cannot be empty.");
            return;
        }
        if (ipAddress.isEmpty()) {
            System.out.println("IP address cannot be empty.");
            return;
        }
        // Replace placeholders
        String result = template.replace("{hostname}", hostname).replace
            ("{ip_address}", ipAddress);

        System.out.println("Result: " + result);
    }
}
```

Output

```
java -cp /tmp/xStadh40Yh/PlaceholderReplacer
Enter the template string: Server example.com has IP address {ip_address}.
Enter the hostname: example.com
Enter the IP address:
IP address cannot be empty.

=== Code Execution Successful ===
```

EXAMPLE EXECUTION AND TEST SCENARIOS

i. Example Execution:

Given the following inputs: -

- Template: `"Hostname: {hostname}, IP Address: {ip_address}"`

- Hostname: `"example.com"`

- IP Address: `"192.168.1.1"`

The program output would be: -

Result: Hostname: example.com, IP Address: 192.168.1.1

ii. Test Scenarios:

1. Test Case 1: Valid Hostname and IP Address

- Inputs: Template = `"Hostname: {hostname}, IP Address: {ip_address}"`, Hostname = `"example.com"`, IP Address = `"192.168.1.1"`

- Expected Output: `Result: Hostname: example.com, IP Address: 192.168.1.1`

2. Test Case 2: Empty Hostname

- Inputs: Template = `"Hostname: {hostname}, IP Address: {ip_address}"`, Hostname = `""`, IP Address = `"192.168.1.1"`

- Expected Output: `Hostname cannot be empty.`

3. Test Case 3: Empty IP Address

- Inputs: Template = `"Hostname: {hostname}, IP Address: {ip_address}"`, Hostname = `"example.com"`, IP Address = `""`

- Expected Output: `IP address cannot be empty.

CONCLUSION

Summary of Accomplishments

1. **Input Handling**:

   Successfully captured and validated user input for a hostname and IP address.

2. **Placeholder Replacement:**

   Implemented dynamic replacement of placeholders in a template string.

3. **Error Handling:**

   Checked for missing inputs and provided clear error messages.

4. **Testing:**

   Demonstrated functionality with examples and test cases to ensure the program behaves as expected under various conditions.