System Requirements Specification Index

For

For Loop

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

Table of Contents

1 Project Abstract

This project assesses knowledge of Java looping constructs, specifically the **for loop**.
The tasks involve using for loops to perform iterative operations such as validating inputs, reversing strings, password validation, implementing number guessing game logic, and printing multiplication tables.

2 Assessment Tasks

Task 1: Repeat User Input Until Valid Response Using For Loop:

- Use a `for` loop with no initialization, condition, or increment (`for (;;)`) to create an infinite loop.
- Inside the loop:
  - ➔ Prompt the user: `"Enter a positive integer:"`.
  - ➔ Accept the input and store it in an integer variable `userInput`.
  - ➔ Check if `userInput` is less than or equal to `0`:
    - If true, print: `"Please enter a positive integer."` and repeat the loop.
- Else, break the loop when a positive integer is entered.
- After breaking the loop, print: `"Thank you! You entered: <userInput>"`.

Expected Output:

```
Enter a positive integer: -3
Please enter a positive integer.
Enter a positive integer: 5
Thank you! You entered: 5
```

Task 2: Reverse a Given String Using For Loop:

- Prompt the user: `"Enter a string to reverse:"`.
- Accept the input and store it in a `String` variable `inputString`.
- Declare an empty `String` variable `reversedString` to store the reversed result.
- Use a `for` loop starting from `inputString.length() - 1` to `0` (decrementing):
  - ➔ In each iteration, append the character at the current index to `reversedString`.
- After the loop, print: `"Reversed string: <reversedString>"`.

Expected Output:

Enter a string to reverse: hello
Reversed string: olleh

Task 3: Validate Password Using For Loop:

- Declare a variable `password` of `String` type and a `boolean` type variable `validPassword`.
- Use an infinite `for` loop (`for (;;)`) to repeatedly prompt the user.
- Inside the loop:
  - ➔ Prompt the user: `"Enter a password:"`.
  - ➔ Accept the input and store it in a `String` variable `password`.
  - ➔ Create and call the helper method `isValidPassword(password)` to check if the password is valid.
  - ➔ The password is valid if:
    - It is at least 8 characters long.
    - Contains at least one digit.
    - Contains at least one special character (`!@#$%^&*()`).
  - ➔ If invalid, print:
    - `"Password must be at least 8 characters long, contain at least one digit, and one special character."`
    - and repeat the loop.
    - Else, break the loop when the password is valid.
- After breaking the loop, print: `"Password accepted."`.

Helper Method: Password Validation

- A method `isValidPassword` is defined to check if the password is valid:
  - ➔ Checks if the length of the password is `>= 8`.
  - ➔ Checks if it contains at least one digit using regex `.*\\d.*`.
  - ➔ Checks if it contains at least one special character using regex `.*[!@#$%^&*()].*`.
- Returns `true` if all conditions are met, else returns `false`.

Expected Output:

```
Enter a password: abc
Password must be at least 8 characters long, contain at least one digit, and one
special character.
Enter a password: abcd123!
Password accepted.
```

Task 4: Number Guessing Game Using For Loop:

- Declare an integer variable `guess` to store the user's guesses.
- Generate a random number between `1` and `100` and store it in `numberToGuess` of integer datatype.

- Use an infinite `for` loop (`for (;;)`) to allow repeated guessing:
  - ➔ Prompt the user: `"Guess the number (1-100):"`.
  - ➔ Accept the input and store it in the `guess` variable.
  - ➔ Check the following conditions:
    - If `guess` is greater than `numberToGuess`, print: `"Too high, try again!"`.
    - If `guess` is less than `numberToGuess`, print: `"Too low, try again!"`.
    - If `guess` equals `numberToGuess`, break the loop.
- After breaking the loop, print: `"Correct! You guessed the number."`.

Expected Output:

```
Guess the number (1-100): 50
Too low, try again!
Guess the number (1-100): 80
Too high, try again!
Guess the number (1-100): 73
Correct! You guessed the number.
```

Note: The actual output values will vary since they are generated randomly

Task 5: Print Multiplication Table of a Number Using For Loop:

- Prompt the user: `"Enter a number for multiplication table:"`.
- Accept the input and store it in an integer variable `number`.
- Use a `for` loop with a `multiplier` variable of integer type starting from `1` to `10`:
  - ➔ In each iteration, calculate and print:
    `"<number> x <multiplier> = <number * multiplier>"`.

Expected Output:

```
Enter a number for multiplication table: 4
4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36
4 x 10 = 40
```

Note: The actual output values will vary depending on the user input.

3 Template Code Structure

3.1 Package: com.yaksha.assignment.ForLoopAssignment

Resources

Class/Interface	Description	Status
ForLoopAssignment (class)	● Main class demonstrating iterative operations using **for loops**. ● Includes examples of: - Repeating user input until valid using infinite **for loop**. - Reversing a string using **for loop**. - Validating password using **for loop**. - Implementing number guessing game logic. - Printing multiplication table.	Need to be implemented.

4 Execution Steps to Follow

1. **All actions like build, compile, running application, running test cases will be through Command Terminal.**
2. **To open the command terminal the test takers, need to go to Application menu (Three horizontal lines at left top) Terminal New Terminal.**
3. **This editor Auto Saves the code.**
4. **If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to use CTRL+Shift+B-command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.**

5. **These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.**
6. **To run your project use command:**
   ```
   mvn compile exec:java -Dexec.mainClass="com.yaksha.assignment.ForLoopAssignment"
   ```
7. **To test your project test cases, use the command**
   ```
   mvn test
   ```
8. **You need to use CTRL+Shift+B - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.**