System Requirements Specification Index

For

Core Java-do-while-loop

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

Table of Contents

1 Project Abstract 3

2 Assessment Tasks 3

3 Template Code Structure 6

3.1 Package: com.yaksha.assignment.DoWhileLoopAssignment 6

4 Execution Steps to Follow 6

1 Project Abstract

This project assesses knowledge of Java **do-while loops** and their practical use cases.
The tasks involve implementing loops that execute at least once, including input validation, string reversal, password checking, number guessing, and generating multiplication tables.

2 Assessment Tasks

Task 1: Repeat User Input Until Valid Response Using Do-While Loop:

- Declare an integer variable `userInput`.
- Use a **do-while loop** to keep prompting the user until a valid (positive) integer is entered:
  - ➔ If the entered number is `<= 0`, print: `"Please enter a positive integer."`.
- Repeat the loop while `userInput <= 0`.
- After the loop, print: `"Thank you! You entered: <userInput>"`.
- This ensures at least one input attempt and continues until valid input is received.

Expected Output:

```
Enter a positive integer: -1
Please enter a positive integer.
Enter a positive integer: 0
Please enter a positive integer.
Enter a positive integer: 1
Thank you! You entered: 1
```

Task 2: Reverse a Given String Using Do-While 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.
- Get the length of the string using `inputString.length()` and initialize an integer variable `i` with `length - 1`.
- Use a `do-while` loop to iterate from the end of the string to the beginning:
  - ➔ In each iteration, Use `charAt(i)` to access the character at the current index.
  - ➔ Append the character to `reversedString`.
  - ➔ Decrement `i` by 1.
- Continue the loop **while** **`i >= 0`**.

- After the loop, print: `"Reversed string: <reversedString>"`.

Expected Output:

```
Enter a string to reverse: hello
Reversed string: olleh
```

Task 3: Validate Password Using Do-While Loop:

- Declare a `String` variable `password` to store the user's input.
- Declare a `boolean` variable `validPassword` to track validation status.
- Use a `do-while` loop to repeat until a valid password is entered:
- 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 `validPassword` is `false`, print:
    - `"Password must be at least 8 characters long, contain at least one digit, and one special character."`
- Repeat the loop **while** `validPassword` is `false`.
- After 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 Do-While 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 a `do-while` loop to continue prompting until the correct guess is made:
    - ➔ 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!"`.
- Repeat the loop **while** `guess != numberToGuess`
- After 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 Using Do-While Loop:

- Prompt the user: `"Enter a number for multiplication table:"`.
- Accept the input and store it in an integer variable `number`.
- Declare an integer variable `multiplier` and initialize it to `1`.
- Use a `do-while` loop to print the table from `1` to `10`:
    - ➔ In each iteration, print:
      `<number> x <multiplier> = <number * multiplier>`
    - ➔ Increment `multiplier` by `1`.
- Repeat the loop **while** `multiplier <= 10`.

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.DoWhileLoopAssignment

Resources

Class/Interface	**Description**	**Status**
DoWhileLoopAssignment (class)	● Main class demonstrating practical use of **do-while loops** in Java. ● Includes examples of: - Validating user input. - Reversing strings. - Validating passwords. - Building number guessing games. - Generating multiplication tables.	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.DoWhileLoopAssignment"
   ```
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.**