System Requirements Specification Index

For

Ternary Operator

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

Table of Contents

USE CASE DESCRIPTION
System Requirements Specification

1 Project Abstract

This project assesses knowledge of Java conditional statements, specifically the **ternary operator**.
The tasks involve checking multiple conditions and making concise decisions based on numerical values, logical comparisons, and complex decision-making scenarios using the ternary operator.

2 Assessment Tasks

Task 1: Check if a Number is Positive, Negative, or Zero Using Ternary Operator:

- Declare an integer variable `number` with an initial value of `0`.
- Use a nested ternary operator to check the following conditions:
  - ➔ Condition 1:
    - ➢ Check if `number` is greater than `0` using the condition `(number > 0)`.
    - ➢ If true, assign the result: `"The number <number> is positive."`.
  - ➔ Condition 2:
    - ➢ Use a nested ternary operator to check if `number` is less than `0` using `(number < 0)`.
    - ➢ If true, assign the result: `"The number <number> is negative."`.
  - ➔ Else Condition:
    - ➢ Use the final `:` condition to handle the case when `number` is neither positive nor negative (i.e., it is `0`).
    - ➢ Assign the result: `"The number <number> is zero."`.
- Store the result in a String variable named `result`.
- Print the result using `System.out.println(result)`.

Task 2: Find the Smallest of Three Numbers Using Ternary Operator:

- Declare and initialize three integer variables:
  - ➔ `a` with the value `10`.
  - ➔ `b` with the value `5`.
  - ➔ `c` with the value `15`.
- Use a nested ternary operator to find the smallest number as follows:
  - ➔ Condition 1:
    - ➢ Check if `a` is less than or equal to both `b` and `c` using the condition

`(a <= b && a <= c)`.

- ➢ If true, assign the result: `<a>`.

➔ Else Condition 1:

- ➢ Use a nested ternary operator to handle the case when `a` is not the smallest:
  - Condition 2:
    - ➢ Check if `b` is less than or equal to `c` using the condition `(b <= c)`.
    - ➢ If true, assign the result: `<b>`.
  - Else Condition 2:
    - ➢ If `b` is not the smallest, assign the result: `<c>`.

- Store the result in a String variable named `smallest`.
- Print the result using `System.out.println("The smallest number is: " + smallest)`.

Task 3: Check if a Number is Divisible by 3, 5, or Both Using Ternary Operator:

- Declare an integer variable `num` with an initial value of `15`.
- Use a nested ternary operator to check the following conditions:

➔ Condition 1:

- ➢ Check if `num` is divisible by both `3` and `5` using `(num % 3 == 0 && num % 5 == 0)`.
- ➢ If true, assign the result: `"The number <num> is divisible by both 3 and 5."`.

➔ Else Condition 1:

- ➢ Use a nested ternary operator to handle the case when `num` is not divisible by both:
  - Condition 2:
    - ➢ Check if `num` is divisible by `3` only using `(num % 3 == 0)`.
    - ➢ If true, assign the result: `"The number <num> is divisible by 3."`.
  - Else Condition 2:
    - ➢ Use another nested ternary operator to handle the case when `num` is not divisible by `3`:
      - Condition 3:
        - ➢ Check if `num` is divisible by `5` only using `(num % 5 == 0)`.
        - ➢ If true, assign the result: `"The number <num> is divisible by 5."`
      - Else Condition 3:
        - ➢ If none of the conditions are met, assign the result: `"The number`

`<num> is divisible by neither 3 nor 5.`".

- Store the result in a String variable named `divisibilityResult`.
- Print the result using `System.out.println(divisibilityResult)`.

Task 4: Grade Calculation Based on Marks Using Ternary Operator:

- Declare an integer variable `marks` with an initial value of `82`.
- Use a nested ternary operator to calculate the grade as follows:
  - ➔ Condition 1:
    - ➢ Check if `marks` is greater than or equal to `90` using `(marks >= 90)`.
    - ➢ If true, assign the result: `"Grade: A"`.
  - ➔ Else Condition 1:
    - ➢ Use a nested ternary operator to handle the case when `marks` is less than `90`:
      - Condition 2:
        - ➢ Check if `marks` is greater than or equal to `75` using `(marks >= 75)`.
        - ➢ If true, assign the result: `"Grade: B"`.
      - Else Condition 2:
        - ➢ Use another nested ternary operator to handle the case when `marks` is less than `75`:
          - Condition 3:
            - ➢ Check if `marks` is greater than or equal to `50` using `(marks >= 50)`.
            - ➢ If true, assign the result: `"Grade: C"`.
          - Else Condition 3:
            - ➢ If none of the conditions are met, assign the result: `"Grade: F"`.
- Store the result in a String variable named `grade`.
- Print the result using `System.out.println(grade)`.

Task 5: Check Leap Year Using Ternary Operator:

- Declare an integer variable `year` with an initial value of `2024`.
- Use a ternary operator to check if the year is a leap year:
  - ➔ Condition:
    - ➢ Use `((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0))` to check if `year` is a leap year.
    - ➢ If true, assign the result: `"The year <year> is a leap year."`.

➔ Else Condition:

➢ If false, assign the result: `"The year <year> is not a leap year."`.

- Store the result in a String variable named `leapYear`.
- Print the result using `System.out.println(leapYear)`.

Expected Output:

The number 0 is zero.
The smallest number is: 5
The number 15 is divisible by both 3 and 5.
Grade: B
The year 2024 is a leap year.

3 Template Code Structure

3.1 Package: com.yaksha.assignment.TernaryOperatorAssignment

Resources

Class/Interface	Description	Status
TernaryOperatorAssignment (class)	• Main class demonstrating conditional checks using **ternary operators**. • Includes examples of: - Checking positive, negative, or zero using **ternary operator**. - Finding the smallest of three numbers using **nested ternary operators**. - Checking divisibility by 3, 5, or both using **ternary operator**. - Calculating grades using **nested ternary operators**. - Checking leap year using **ternary operator** logic.	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.TernaryOperatorAssignment"
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.**