System Requirements Specification Index

For

String Comparison and Conversion

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

Table of Contents

USE CASE DESCRIPTION

System Requirements Specification

1 Project Abstract

This project will assess knowledge of string comparison and conversion methods in Java. You need to implement string comparison and conversion operations using built-in Java methods.

2 Assessment Tasks

Task 1:

1. **Declare 2 variables:**
   - A variable named `str1` of `String` datatype, initialized with the value `"hello"`.
   - A variable named `str2` of `String` datatype, initialized with the value `"Hello"`.

2. **Perform String Comparison:**
   Use the string variables `str1` and `str2` to perform the following comparisons:

   - **Equals (`equals()`)**:
     1) Compare `str1` and `str2` using `equals()`, which checks for case-sensitive equality.
     2) Store the result in a variable named `isEqual` of `boolean` datatype.
   - **Equals Ignore Case (`equalsIgnoreCase()`)**:
     1) Compare `str1` and `str2` using `equalsIgnoreCase()`, which ignores case while comparing.
     2) Store the result in a variable named `isEqualIgnoreCase` of `boolean` datatype.
   - **Lexicographical Comparison (`compareTo()`)**:
     1) Compare `str1` and `str2` using `compareTo()`, which returns:
        - `0` if both strings are equal.
        - A negative value if `str1` is lexicographically smaller than `str2`.
        - A positive value if `str1` is lexicographically greater than `str2`.
     2) Store the result in a variable named `comparison` of `integer` datatype.

   Print the Results:
   - Print the results of each string comparisons i.e, `isEqual`, `isEqualIgnoreCase`, and `comparison` with appropriate labels in separate lines as shown in the expected output.

Task 2:

3. **Declare a new string variable:**
    - A variable named `str` of `String` datatype, initialized with the value: `"  Trim me  "`.

4. **Perform String Conversion Operations:**
   Use the string variable `str` to perform the following conversions:

    - **Trim (`trim()`)**:
        1) Remove leading and trailing spaces from `str` using `trim()`.
        2) Store the result in a variable named `trimmed` of `String` datatype.
    - **Split (`split(" ")`)**:
        1) Split `str` into an array of words using `split(" ")`, which separates the string based on spaces.
        2) Store the result in an array named `split` of `String` datatype.
    - **Convert to Character Array (`toCharArray()`)**:
        1) Convert `str` into a character array using `toCharArray()`.
        2) Store the result in an array named `charArray` of `char` datatype.
    - **Convert to String Using `valueOf()`**:
        1) Convert an integer `123` into a string using `String.valueOf(123)`.
        2) Store the result in a variable named `strValue` of `String` datatype.

   Print the Results:
    - Print the results of each string conversion operation i.e, `trimmed`, `split`, `charArray`, and `strValue` with appropriate labels in separate lines as shown in the expected output.

Expected Output:

```
Equals: false
Equals Ignore Case: true
CompareTo: 32
Trimmed: 'Trim me'
Split: , , Trim, me
CharArray: [C@6c8338d0
String Value of 123: 123
```

3 Template Code Structure

3.1 Package: com.yaksha.assignment.StringComparisonConversionAssignment Resources

Class/Interface	Description	Status
StringComparisonConversionAssignment (class)	• Main class demonstrating string comparison operations such as: `equals`, `equalsIgnoreCase`, `compareTo`. • And string conversion operations like: `valueOf`, `trim`, `split`, and `toCharArray`.	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.StringComparisonConversionAssignment"

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.**

9. **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.**