Java-Loop Through A HashSet

Project Abstract

The purpose of this project is to demonstrate how to perform operations on a `HashSet` in Java. This task helps you to understand how to use the `HashSet` class to store elements and perform various operations, such as adding, removing, checking the existence of elements, and looping through the set. The project focuses on using methods such as `add()`, `contains()`, `remove()`, `size()`, and looping through the `HashSet` using an iterator.

This project focuses on:
1. Understanding how to create and use a `HashSet` in Java.
2. Adding items to a `HashSet` using the `add()` method.
3. Checking the existence of an item using the `contains()` method.
4. Removing items from a `HashSet` using `remove()`.
5. Finding the size of the `HashSet` using `size()`.
6. Iterating over a `HashSet` using an iterator.

Tasks Overview

Task 1: Add, Remove, Check Existence, and Loop Through HashSet

Objective: Perform various operations on a HashSet, including adding, removing, checking existence, and looping through elements.

Detailed Description: In this task, students will create a `HashSet` of strings and perform the following operations: adding elements to the set using `add()`, checking if an element exists using `contains()`, removing elements using `remove()`, finding the size of the set using `size()`, and looping through the `HashSet` using an iterator.

Steps:
1. **Create a HashSet:**
   - Declare a `HashSet` object named `hashSet` to store `String` values.

2. **Add Items to the HashSet:**
   - Use the `add()` method to add several items ("One", "Two", "Three") to the `hashSet` and print their values as `"HashSet after adding items: " + hashSet`.

3. **Check if an Item Exists:**
   - Use the `contains()` method to check if a specific item exists in the `hashSet` (check if "Two" exists) and print its result as `"HashSet contains 'Two': " + containsTwo`.

4. **Remove an Item from the HashSet:**
   - Use the `remove()` method to remove an item (remove the item "Three") and print the complete hashSet as `"HashSet after removing item 'Three': " + hashSet`.

5. **Find the Size of the HashSet:**
   - Use the `size()` method to get and print the size of the `hashSet` as `"Size of HashSet: " + size`.

6. **Loop Through the HashSet:**
   - Use an iterator to loop through the `hashSet` using iterator and print each element as `iterator.next()`.

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.HashSetOperations"
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.**