Java-Sorting An Arraylist

Project Abstract

The purpose of this project is to demonstrate how to perform sorting operations on an ArrayList in Java. This task helps you to understand how to use the `ArrayList` class to store elements and how to sort them in both ascending and descending order using the `Collections.sort()` method. The project focuses on adding items to an ArrayList, sorting it, and demonstrating different sorting orders.

This project focuses on:
1. Understanding how to create and use an ArrayList in Java.
2. Learning how to add items to an ArrayList.
3. Sorting an ArrayList in ascending and descending order using `Collections.sort()`.
4. Demonstrating the impact of sorting operations on an ArrayList.

Tasks Overview

Task 1: Add Items and Sort ArrayList

Objective: Add elements to an ArrayList and sort it in both ascending and descending order.

Detailed Description: In this task, you will create an `ArrayList` of integers, add some items to it, and then perform sorting operations. You will first sort the ArrayList in ascending order and then in descending order, using the `Collections.sort()` method. The program will display the ArrayList after each operation to show the results of sorting.

Steps:
1. **Create an ArrayList:**
   - Declare an `ArrayList` object named `arrayList` to store `Integer` values.

2. **Add Items to the ArrayList:**
   - Use the `add()` method of the `ArrayList` class to add several integers (3, 1, 4, 2) to the `arrayList`.

3. **Display the ArrayList:**
   - Print the `arrayList` to show the items before sorting as `"ArrayList after adding items: " + arrayList`.

4. **Sort the ArrayList in Ascending Order:**
   - Use the `Collections.sort()` method to sort the `arrayList` in ascending order.
   - Print the `arrayList` again to show the result after sorting in ascending order as `"ArrayList after sorting in ascending order: " + arrayList`.

5. **Sort the ArrayList in Descending Order:**
   - Use the `Collections.sort()` method with `Collections.reverseOrder()` to sort the `arrayList` in descending order.
   - Print the `arrayList` again to show the result after sorting in descending order as `"ArrayList after sorting in descending order: " + arrayList`.

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.ArrayListSortingOperations"
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.**