Using Built-in Packages and Java API Classes

Project Abstract

The purpose of this project is to demonstrate the proper use of Built-in Packages and Java API Classes such as ArrayList, HashMap, and File. Java provides a variety of built-in packages for handling common tasks such as data structures (e.g., java.util.ArrayList and java.util.HashMap) and file operations (e.g., java.io.File). In this project, we will explore how to correctly import and utilize these built-in classes to handle data manipulation and file I/O in Java.

Tasks Overview

Task 1: Implementing ArrayList and HashMap

Objective: Demonstrate the use of ArrayList and HashMap classes from the java.util package.

- Steps:

1. Create an empty ArrayList of type String with name arrayList and call the add() method to add items to it like “Apple” and “Banana”.
2. Create a HashMap of type <String, String> with name hashMap and call the put() method to insert key-value pairs into the map like <”1”, “One”> and <”2”, “Two”>.
3. Print out the contents of both the ArrayList and HashMap as "ArrayList contents: " + arrayList and "HashMap contents: " + hashMap respectively.

Task 2: Using the File Class for File Operations

Objective: Demonstrate the use of the File class from the java.io package.

- Steps:

4. Create a File object as file.
5. Use the createNewFile() method to create a new file with name testFile.txt.
6. Handle any IOException if the file cannot be created and print an appropriate message.

Detailed Explanation of the Code Implementation

In the Main.java file, the following Java API classes are used:

ArrayList:
An ArrayList is created and populated with a few items (Apple and Banana) using the add()

method.
The contents of the ArrayList are then printed to the console.

HashMap:
A HashMap is created and populated with key-value pairs (1: "One" and 2: "Two") using the put() method.
The contents of the HashMap are then printed to the console.

File:
A File object is created for a file named testfile.txt.
The createNewFile() method is used to create the file, and the result is checked. If the file is created, a success message is printed as "File created: " + file.getName(); otherwise, a message saying "File already exists" is printed.
Any IOException that occurs during the file creation process is caught and logged as "An error occurred while creating the file.".

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. **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.**
5. **To run your project use command:**

   sudo JAVA_HOME=$JAVA_HOME /usr/share/maven/bin/mvn compile exec:java -Dexec.mainClass="com.yaksha.assignment.Main"

   ***If it asks for the password, provide password : pass@word1**

6. **To test your project test cases, use the command**

   sudo JAVA_HOME=$JAVA_HOME /usr/share/maven/bin/mvn test

   ***If it asks for the password, provide password : pass@word1**