Java Lambda Expression Method -

Assignment

Instructions:

You are provided with the `LambdaMethod.java` class. Your task is to implement the `main()` method using lambda expressions and method references. Below are the specific tasks you need to complete, including exact variable names and type declarations.

Task 1: Lambda Expression as a Method Argument

1. **Objective**: Use a lambda expression as an argument to a method.

2. **Details**:
   - You have to create an array list of type ArrayList<String>.
   - Your task is to create a method with name 'processList()' that accepts a List<String> and Consumer<String> and pass a lambda expression to the method `processList()` that will print each name.

3. **Steps**:
   - **Create a variable** named `names` of type `List<String>`.
   - **Add the following names** to the list: `"Alice"`, `"Bob"`, `"Charlie"`, and `"Dave"`.
   - Call the `processList()` method and pass a lambda expression to print each name.
   - The lambda expression should print the name prefixed with `"Name: "`.

Task 2: Method Reference to Simplify Lambda Expression

1. **Objective**: Use a method reference to simplify the lambda expression.

2. **Details**:
   - You need to call the same `processList()` method.
   - This time, you should use a **method reference** instead of a lambda expression.
   - The method reference should print each name from the list.

3. **Steps**:
   - **Use a method reference** to simplify the lambda expression.
   - The method reference should point to the `println` method of `System.out` to print each name.

Task 3: Lambda Expression that Returns a Result

1. **Objective**: Use a lambda expression that returns a result.

2. **Details**:

   o   You are required to implement a lambda expression that takes two integers and adds them together.

   o   The result of the addition should be returned by the lambda expression.

3. **Steps**:

   o   **Create a Calculator functional interface**. This interface should have a method add(int a, int b) that performs addition.

   o   **Declare a variable** named sum of type int. This will store the result of the addition.

   o   **Call the calculate() method**, which accepts a Calculator functional interface and two integers as parameters.

      ▪   The calculate() method should perform the addition using the lambda expression passed as the Calculator argument.

      ▪   Inside the calculate() method, you will use the lambda expression to return the sum of the two numbers.

   o   **Pass a lambda expression** to the calculate() method, where the lambda expression implements the add() method of the Calculator interface.

      ▪   The lambda should perform addition of two integers (e.g., a + b).

      ▪   Store the result in the sum variable.

   o   Finally, **print the sum** to the console as ["Sum: " + sum].

4. **Additional Information**:

   o   The Calculator interface is a functional interface, meaning it contains only one abstract method. The lambda expression will implement this method to perform the addition.

   o   The calculate() method accepts this functional interface and the two integers, then invokes the add() method of the lambda to compute the result.

Final Deliverable:

1. Implement the `main()` method in the `LambdaMethod.java` class.
2. Ensure that:
   - You **declare and initialize** the `List<String>` variable `names` with the names `"Alice"`, `"Bob"`, `"Charlie"`, and `"Dave"`.
   - You pass a **lambda expression** to the `processList()` method to print each name prefixed with `"Name: "`.
   - You use a **method reference** to simplify the lambda expression in `processList()`.
   - You declare an `int` variable `sum` and call the `calculate()` method with a lambda expression to add two numbers.
3.   Do not include any print statements outside of the lambda expressions and method references.

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.LambdaMethod"

   *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