CHRIST (Deemed to be University)
Department of Computer Science
Master of Artificial Intelligence and Machine Learning

Course: MAI271 – JAVA Programming
Exercise No: LAB Exercise – 6
Date: 30 – 11 – 2024
Duration: 2 Hrs

Question 1 (4 Marks)

You are provided with an integer array `coins[]` of size N, representing different denominations of currency, and an integer `sum`. The task is to find the number of ways you can make the given sum by using different combinations from the coins[] array. Assume that you have an infinite supply of each type of coin, and you can use any coin as many times as needed.Incorporate multithreading concepts to optimize the computation of different combinations. Design a concurrent solution that utilizes multiple threads to explore and calculate the various combinations efficiently.

Example 1:
Input:
N = 3, sum = 4
coins = {1, 2, 3}
Output: 4
Explanation: Four possible ways are: {1, 1, 1, 1}, {1, 1, 2}, {2, 2}, {1, 3}.

Example 2:
Input:
N = 4, Sum = 10
coins = {2, 5, 3, 6}
Output: 5
Explanation: Five possible ways are: {2, 2, 2, 2, 2}, {2, 2, 3, 3}, {2, 2, 6}, {2, 3, 5}, and {5, 5}.

Question 2 (6 Marks)

You are building a coffee shop simulation where baristas (producers) prepare coffee orders, and customers (consumers) pick them up. The counter can hold a maximum of 3 coffee cups at any time. Additionally, there is a **coffee reviewer** (an observer task) who randomly samples a coffee from the counter to rate its quality.

- Use multithreading to manage the interaction between baristas, customers, and the reviewer using wait() and notify() for synchronization.
- Baristas should stop making coffee if the counter is full, and customers should wait if the counter is empty.

- The reviewer should only attempt to sample when there is at least one coffee available. If the counter is empty, the reviewer should also wait.
- Define a custom exception CounterEmptyException that is raised if a customer or the reviewer tries to take a coffee when none are available.

Write a program to simulate this enhanced producer-consumer scenario, ensuring all interactions are properly synchronized.

Sample Test Inputs and Expected Output

Inputs:

1. **Baristas' tasks (producers):**
    - Barista 1: Prepares 2 coffees.
    - Barista 2: Prepares 3 coffees.
2. **Customers' tasks (consumers):**
    - Customer 1: Picks up 1 coffee.
    - Customer 2: Picks up 2 coffees.
    - Customer 3: Picks up 1 coffee.
3. **Coffee Reviewer task (observer):**
    - Samples 1 coffee for review.

Expected Sequence of Events:

1. **Barista 1** prepares the first coffee. (Counter: 1 coffee)
2. **Barista 1** prepares the second coffee. (Counter: 2 coffees)
3. **Barista 2** prepares the third coffee. (Counter: 3 coffees)
    - **Barista 2 waits** because the counter is now full.
4. **Customer 1** picks up 1 coffee. (Counter: 2 coffees)
    - **Barista 2 is notified** and resumes preparing coffee.
5. **Barista 2** prepares another coffee. (Counter: 3 coffees)
    - **Barista 2 waits** again as the counter is full.
6. **Customer 2** picks up 2 coffees. (Counter: 1 coffee)
    - **Barista 2 is notified** and resumes preparing the last coffee. (Counter: 2 coffees)
7. **Customer 3** picks up 1 coffee. (Counter: 1 coffee)
8. **Coffee Reviewer** samples 1 coffee for review. (Counter: 0 coffees)
    - **Barista 1 and Barista 2 are notified** as the counter is now empty.

Expected Output:

Barista 1 prepared coffee. Counter: 1
Barista 1 prepared coffee. Counter: 2

Barista 2 prepared coffee. Counter: 3
Barista 2 is waiting. Counter is full.
Customer 1 picked up coffee. Counter: 2
Barista 2 prepared coffee. Counter: 3
Barista 2 is waiting. Counter is full.
Customer 2 picked up coffee. Counter: 2
Customer 2 picked up coffee. Counter: 1
Barista 2 prepared coffee. Counter: 2
Customer 3 picked up coffee. Counter: 1
Coffee Reviewer sampled coffee. Counter: 0
Barista 1 is notified. Counter is empty.
Barista 2 is notified. Counter is empty.

General Instruction:

1. Ensure that your code includes relevant comments to enhance readability and understanding. Subsequently, upload your code to GitHub for version control and collaborative access.
2. Include descriptive comments within the code, explaining its functionality and logic.
3. In the Google Classroom submission, include the GitHub URL where your code is hosted.
4. Attach a PDF document named "your_register_number_exercise_No.pdf" to the submission.The PDF document should include screenshots of the code and the output screen.
5. Upload the answer document&GitHub URL in Google Classroom on or before the deadline mentioned.Evaluation will not be considered for late submission