In-Class Exercises: Multi-threading

Exercise 1: The "Greeting" Threads (Runnable Interface)

Goal: Practice creating and naming threads using the `Runnable` interface.

Task:

1. Create a class `GreeterTask` that implements `Runnable`.
2. In the `run()` method, print "Hello from [Thread Name]" 5 times, with a 500ms sleep between each print.
3. In your `main` method, create two threads: "Lehman-Thread-1" and "Lehman-Thread-2".
4. Start both and observe the interleaved output.

Exercise 2: Visualizing Thread States

Goal: Catch a thread in different states using `getState()`.

Task:

1. Create a thread that sleeps for 2 seconds.
2. In the main thread, print the state of the thread:
    - Immediately after creation.
    - Immediately after calling `start()`.
    - While it is sleeping (use `Thread.sleep(500)` in main to ensure the child is asleep).
    - After it has finished.

Exercise 3: The "Bank Account" Race Condition

Goal: Witness data corruption and fix it with `synchronized`.

Task:

1. Create a class `BankAccount` with a `balance` variable (start at $1000).
2. Create a method `withdraw(int amount)` that checks if the balance is sufficient, then subtracts the amount.
3. Create two threads (Husband and Wife) that both try to withdraw $700 at the exact same time.
4. **Observation:** Notice how the balance can go negative if they both pass the "check" before either "subtracts."

5. **Fix:** Use the `synchronized` keyword to ensure only one person can access the ATM at a time.

Exercise 4: Coordination with join()

Goal: Ensure a main thread waits for children to finish.

Task:

1. Create a thread that performs a "Heavy Calculation" (looping to 1 billion).
2. The main thread should print "Calculation Finished: " + result.
3. Use `thread.join()` to ensure the main thread doesn't print the result until the calculation is actually done.