### Bài toán 1: Bank Demo với ExcutorFramework

**Mô tả: Giả sử bạn đang xây dựng một hệ thống ngân hàng với nhiều tài khoản. Mỗi tài khoản có số dư ban đầu. Nhiều thread sẽ thực hiện các giao dịch chuyển tiền giữa các tài khoản. Bạn cần đảm bảo rằng các giao dịch được thực hiện một cách an toàn mà không có lỗi do truy cập đồng thời vào tài khoản.**

**Các yêu cầu:**

1. Mỗi thread sẽ thực hiện một giao dịch chuyển tiền từ một tài khoản này sang tài khoản khác.
2. Giao dịch cần được đồng bộ hóa để tránh tình trạng hai thread cùng lúc truy cập và thay đổi số dư của một tài khoản.
3. Sử dụng wait() và notify() để tránh việc rút tiền từ tài khoản nếu số dư không đủ, và chờ đợi cho đến khi tài khoản đó có đủ tiền.
4. Tạo và quản lý các thread bằng cách sử dụng ExecutorService với một ThreadPool.

Mô tả các lớp:

Tài khoản (Account): Mỗi tài khoản có số dư (balance) và phương thức để rút (withDraw) và nạp tiền (deposit). Đảm bảo đồng bộ hóa khi thực hiện giao dịch trên tài khoản.

Ngân hàng (Bank): Quản lý nhiều tài khoản và xử lý các giao dịch chuyển tiền (transfer) giữa các tài khoản.

Giao dịch (Transaction): Mỗi Transaction là một giao dịch chuyển tiền được thực hiện nhiệm vụ chuyển tiền giữa hai tài khoản, bao gồm cả kiểm tra số dư.

ExecutorService: Sử dụng để tạo nhiều thread và quản lý việc thực hiện các giao dịch.

**Gợi ý:**

**Chi tiết bài tập:**

1. **Account Class:**
   * **Mỗi tài khoản có một số dư và hai phương thức: deposit() để nạp tiền và withdraw() để rút tiền.**
   * **Sử dụng từ khóa synchronized để đồng bộ các phương thức này, đảm bảo rằng không có nhiều thread cùng truy cập và thay đổi số dư cùng lúc.**
   * **Sử dụng wait() để thread chờ nếu không đủ số dư để rút và notifyAll() để thông báo khi có đủ tiền.**
2. **Bank Class:**
   * **Quản lý một mảng các tài khoản.**
   * **Phương thức transfer() để thực hiện giao dịch giữa hai tài khoản. Phương thức này đảm bảo rằng các giao dịch được thực hiện một cách an toàn và đồng bộ.**
3. **Transaction Class:**
   * **Mỗi transaction là một thread chạy trong nền để thực hiện chuyển tiền giữa các tài khoản. Nó sử dụng Bank.transfer() để thực hiện giao dịch.**
4. **Main Class:**
   * **Tạo một ExecutorService với một ThreadPool để quản lý và thực hiện nhiều giao dịch đồng thời.**
   * **Tạo 10 giao dịch ngẫu nhiên và chờ tất cả giao dịch hoàn thành trước khi in ra số dư cuối cùng của tất cả các tài khoản.**

### Bài toán 2: Hệ thống sản xuất và vận chuyển hàng hóa

**Mô tả:** Xây dựng một hệ thống sản xuất và vận chuyển hàng hóa với các yêu cầu sau:

1. Một nhà máy có nhiều dây chuyền sản xuất (Producer) tạo ra hàng hóa.
2. Hàng hóa được lưu trữ trong một kho lưu trữ (Warehouse) có kích thước giới hạn. Nếu kho đầy, các dây chuyền sản xuất phải tạm dừng cho đến khi có không gian trống.
3. Nhiều xe vận chuyển (Consumer) lấy hàng hóa từ kho và vận chuyển ra ngoài. Nếu kho trống, các xe vận chuyển phải chờ cho đến khi có hàng hóa mới.
4. Hệ thống phải quản lý số lượng hàng hóa trong kho, đảm bảo các xe vận chuyển không lấy nhiều hơn số hàng hiện có.
5. Sử dụng wait() và notifyAll() để đảm bảo sự đồng bộ giữa việc sản xuất và vận chuyển hàng hóa.
6. Sử dụng ExecutorService để quản lý các thread của dây chuyền sản xuất và xe vận chuyển.

**Yêu cầu:**

* Tạo một **Warehouse** để lưu trữ hàng hóa với số lượng giới hạn.
* Các **Producer** tạo ra hàng hóa và lưu trữ vào Warehouse nếu còn chỗ trống.
* Các **Consumer** lấy hàng từ Warehouse và vận chuyển ra ngoài.
* Hệ thống phải an toàn cho nhiều thread cùng truy cập.

Phần này không có gợi ý để mọi người tự tư duy. Tương tự bài số 1 thôi. Nếu mà tư duy thấy khoai quá thì ping riêng anh. Anh sẽ gửi gợi ý nhé