# Java concurrency exercise

You have been given the task of improving an existing service. The current service implementation you need to update is provided in `com.westpac.ServiceImpl`

The service does the following:

1. Calls `lookup` on the provided `AccountLookupService` to get `accountId` by `clientId`
2. Calls `execute` on the provided `LegacyService`

Take the following into account when updating the service:

1. The service is used in a concurrent environment, where `submit()` is called from an unknown number of threads.
2. Expected request arrival rate is ~1000 requests/sec.
3. `AccountLookupService` has the following characteristics:
    1. Thread-safe
    2. Underlying thread-pool size: 30 threads
    3. Blocks if no available threads in the pool
    4. Expected avg lookup latency (excluding blocking pool wait time): 50ms
    5. You are permitted to cache results for up to 10 seconds
    6. Total client/account mappings does not exceed 100 entries
4. `LegacyService` has the following characteristics:
    1. Service will break if you make more than 25 concurrent calls. Make sure you guard against this happening.
    2. Expected latency: 20 ms
5. Only edit the `ServiceImpl` class. You cannot change any of the existing interfaces or other service implementations. If you wish, you may create new classes/interfaces and use them from your updated `ServiceImpl`.
6. Don't import any additional 3rd party libraries. Please complete your solution using just the available core Java 8 APIs.
7. Please state any reasoning or assumptions you have made in your implementation.

Your task is to improve the implementation to minimise:

1. `submit()` latency
2. end-to-end request latency

The aim is to try to achieve the best possible performance through the most efficient usage of the available resources.

The provided `IndicativeMicroBenchmark` class runs a simple benchmark to help you estimate the performance. Note it's indicative only, so won't represent actual production environment characteristics. The process will run for around a minute locally, and several minutes on Hackerrank. Please ignore the "Run Unit Tests" button in Hackerrank. We haven't included unit tests, so it won't do anything useful.

**We recommend you download the exercise and run it in your favourite IDE. This will make editing and testing much easier. Some candidates have also had issues running this on HackerRank if it works locally just submit your code**

Good luck!

```java
package com.westpac;

public interface AccountLookupService {


    interface Callback {
        /**
         * Called when lookup is complete
         *
         * @param id        request id
         * @param accountId account
         */
        void onResult(String id, String accountId);
    }

    /**
     * - Thread-safe
     * - Underlying thread-pool size: 30 threads
     * - Blocks if no available threads in the pool
     * - Expected avg lookup latency (excluding blocking pool wait time): 50ms
     * - Results can be cached for up to 10 seconds (total client/account count does not exceed 100 entries)
     *
     * @param requestId unique arbitrary request id
     * @param clientId  client id
     * @param callback  callback
     */
    void lookup(String requestId, String clientId, Callback callback);

    void shutdown();
}

```