```
package com.price.processor;

/**
 * You have to write a PriceThrottler class which will implement the
following requirements:
 * 1) Implement PriceProcessor interface
 * 2) Distribute updates to its listeners which are added through subscribe()
and
 * removed through unsubscribe()
 * 3) Some subscribers are very fast (i.e. onPrice() for them will only take
a microsecond) and some are very slow
 * (onPrice() might take 30 minutes). Imagine that some subscribers might be
showing a price on a screen and some
 * might be printing them on a paper
 * 4) Some ccyPairs change rates 100 times a second and some only once or two
times a day
 * 5) ONLY LAST PRICE for each ccyPair matters for subscribers. I.e. if a
slow subscriber is not coping
 * with updates for EURUSD - it is only important to deliver the latest rate
 * 6) It is important not to miss rarely changing prices. I.e. it is
important to deliver EURRUB if it ticks once
 * per day but you may skip some EURUSD ticking every second
 * 7) You don't know in advance which ccyPair are frequent and which are
rare. Some might become more frequent
 * at different time of a day
 * 8) You don't know in advance which of the subscribers are slow and which
are fast.
 * 9) Slow subscribers should not impact fast subscribers
 *
 * In short words the purpose of PriceThrottler is to solve for slow
consumers
 *
 */
public interface PriceProcessor {
    /**
     * Call from an upstream
     *
     * You can assume that only correct data comes in here - no need for
extra validation
     *
     * @param ccyPair - EURUSD, EURRUB, USDJPY - up to 200 different currency
pairs
     * @param rate - any double rate like 1.12, 200.23 etc
     */
    void onPrice(String ccyPair, double rate);

    /**
     * Subscribe for updates
     *
     * Called rarely during operation of PriceProcessor
     *
     * @param priceProcessor - can be up to 200 subscribers
     */
    void subscribe(PriceProcessor priceProcessor);

    /**
     * Unsubscribe from updates
     *
     * Called rarely during operation of PriceProcessor
     *
     * @param priceProcessor
     */
    void unsubscribe(PriceProcessor priceProcessor);
}
```