CASE 2: PAIRS TRADING

1. Introduction

This case examines how an understanding of the relationship between securities can be used to attain profits when prices deviate from the equilibrium.

Each team will be allowed to trade a portfolio of stocks in each round of the case. Your algorithm should be able to identify the profitable trading pairs within the portfolio of securities, discover the underlying relationship and profit from it, at the same time managing your exposure to secure the profit before the deviation reverts.

The case will consist of three rounds with 1,000 ticks each. (1) The first round will consist of only two securities with a pre-established relationship. Your job is to study it and profit from any deviations from its equilibrium. (2) In the second round, there will be three tradable securities so your job extends to being able to identify the *one* trading pair (and the third security, which has no meaningful relationship with the other two) and profit from it. (3) The third round will have five tradable securities available with *two* distinct trading pairs for your algorithm to tackle.

Securities	S1	S2	S3	S4	S5
Starting Price for Round 1	X	Y	-	-	-
Starting Price for Round 2	X	Y	Z	-	-
Starting Price for Round 3	X	Y	Z	W	R

2. Addtional Details

To discourage running algorithm that only trades one particular security instead of attempting to discover a profitable pair, data for each individual security will be generated from a process where the expected returns are 0.

In addition, you must maintain a net zero position at all times. Hence, when you buy a security, you have to simultaneously sell another security (and vice versa).

An absolute position limit will also be enforced so there will be a limit to how large a directional position may be. This constraint means your algorithm should allocate your exposure to where it thinks the most profitable trades will lie.

There will be a fixed bid-ask spread around the price of each security.

Participants will be given sample data generated from our algorithm to test run their program, but we may change the parameters used on the competition day. You will be provided with adequate capital at the beginning of each round to establish positions within the limits set forth above. Positions and PnL are not carried over between rounds. Any open positions will be liquidated at the closing price of each security at the end of each round. Teams will have time between rounds to make changes to their parameters used in their algorithm if they choose.

3. Scoring

Participants' scores in each round of the case will be determined based on PnL, where PnL is defined to include profits and losses from closed positions, as well as profits and losses from any open positions at the end of the round marked to the closing price of each security of that round.

The weighting of the scores from each round to calculate the final score of the case will be announced prior to the start of the competition.

4. Case Objects & Interface

The following Java class objects are implemented in the util package and should be used in your program based on the interface defined:

```
public enum Ticker {
    S1, S2, S3, S4, S5
}
```

Only the securities relevant will be available during each round.

```
public static class Quote {
    public Quote(Ticker ticker, double bid, double offer)
}

public static class Order {
    public Order (Ticker ticker, int quantity)
}
```

Your program only needs to expose the following functions:

```
public Order[] getNewQuotes(Quote[] quotes);
```

This method is called when new quotes for the stocks are distributed. New price information will be passed on via the quotes array argument. You implementation should return an array of orders indicating your actions for each of the stocks. 1 for buy, -1 for sell and 0 for inaction. Please make sure the order of the securities in your returned array be the same as in the Quotes array you receive, which will be consistent.

```
public void orderFill (Order[] orders);
```

This function is invoked to confirm that the orders you submitted were filled. The reason that this second function is needed is due to the case that the order of securities in the returned array is different from the `Quote` the system sends out.