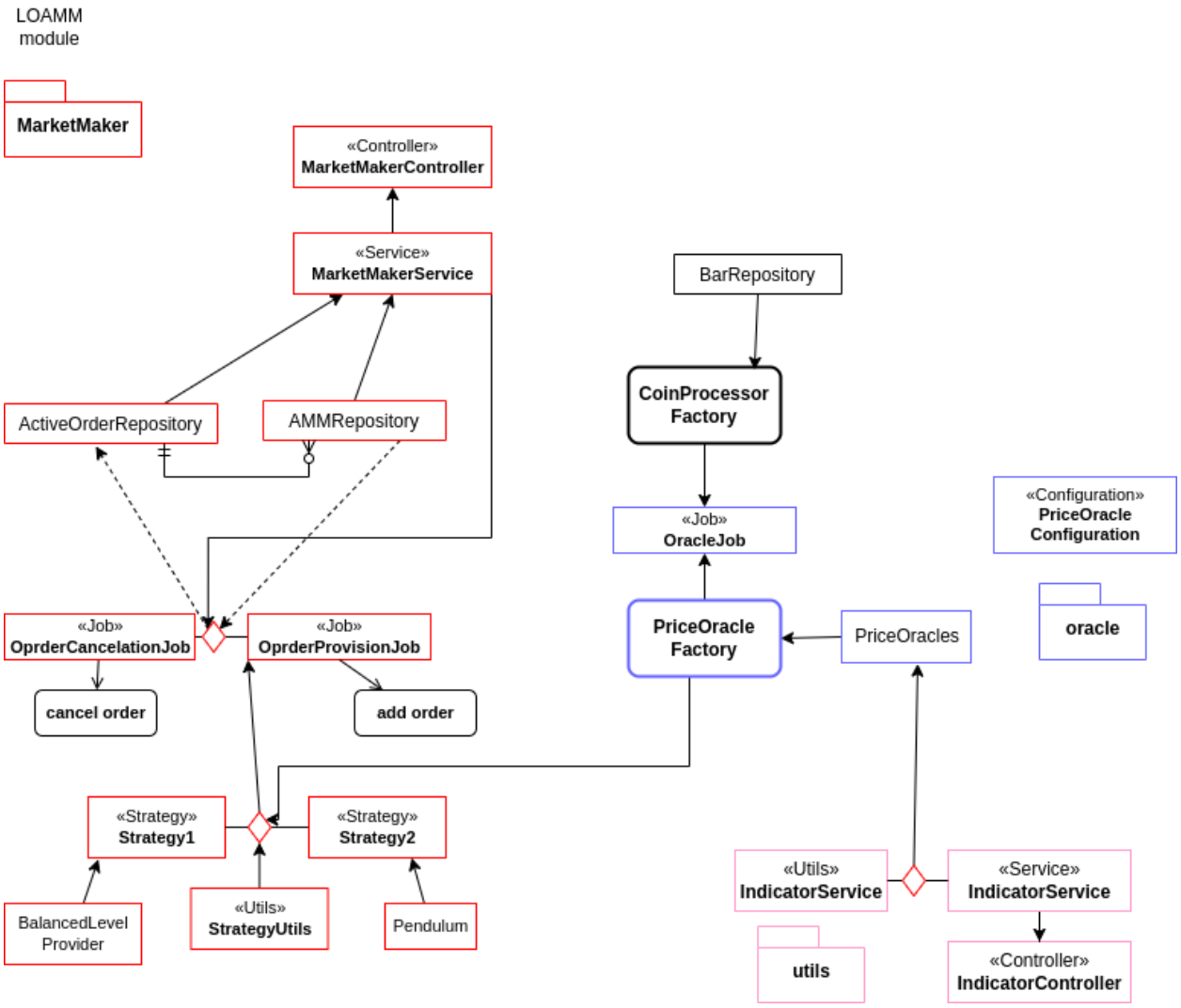
Rule-based **Automated Market Maker** and **Price oracle**

Market-makers use different strategies to provide orders to handle the market's liquidity and trading rate. Currently, two strategies use **price oracle** to predict the market behavior and then create some orders.

- Strategies are implemented by Strategy.java, and new strategies can be added.
- PriceOracles are implemented by PriceOracle.java, and new oracles can be added.



---

1. Price Oracle:

Oracles use indicators to predict the market's behavior. Each of the price oracles extends PriceOracleBase.java. Indicators are available at utils.service.IndicatorService. At last, the **oracle** method creates a Map<Duration, Num> which contains oracles of different time scales.

OracleJob updates the oracles according to their time scale.

1.1. PriceOraclAuto:

It uses MACD and RSI indicators. The simplest prediction by the RSI is a Buy/Sell signal by crossing the boundaries. Another one is Swing rejection, and bypassing 4 different steps, it can report a Buy/Sell signal. Also, the Rapid movement signal is generated by both RSI and MACD. For MACD, the oracle considers the Crossing points, Confirmations, and Divergences. Each of these features has a different effect on the final report. Also, the time distance can reduce the oracle strength.

$$-1 < O_{MACD} + O_{RSI} < 1$$

$$O_{totla} = O_{MACD} e^{-k\Delta t_m} + O_{RSI} e^{-k\Delta t_r}$$

---

2. AutomatedMarketMaker:

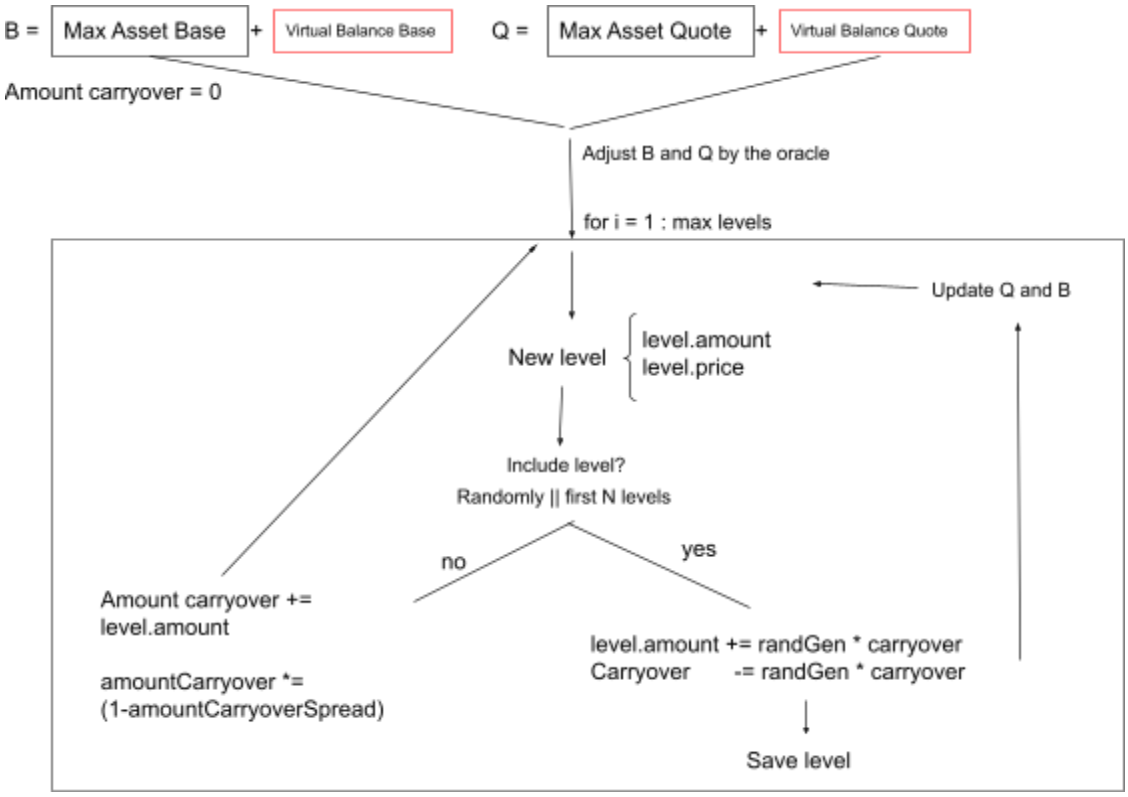
- String **id**
- String **Symbol**
- String **baseCoin**
- String **quoteCoin**
- boolean **enable:** market maker works will it is enabled.
- StrategyType **type:** identifies the strategy that the AMM uses.
- TradeSpeed **speed:** AMM submits its orders by specific (rate/rates).
- long **memberId**
- String **features:** strategy features are stored in String (JSON).
- int **group:** to decrease traffic, each AMM is randomly assigned to one of the five groups.

AMMs are implemented to be semi-automated. Operators can manually change strategy features and add/cancel orders.

3. Strategy1:

BalanceLevelProvider (from Kelp):

- First, get the oracle form **PriceOracleAuto**, which contains 4 different scores ([bearish] -1<score<1 [bullish]).
- Adapts the investment risk by the oracle.
- Uses the below mechanism to generate levels of order.
- Sends the orders to addOrder method (?).



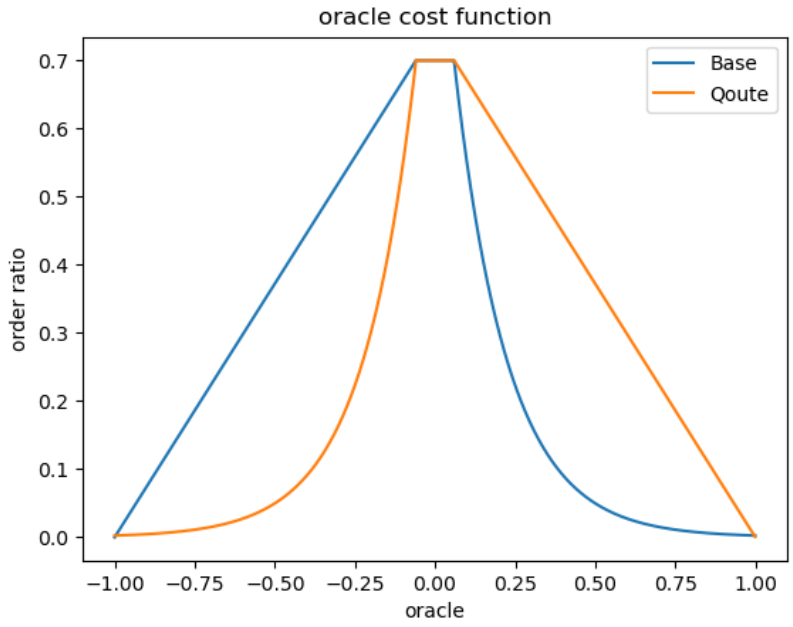
$$level.amount = maxAsset \times \frac{2s}{4+s}$$

$$level.price = \frac{maxAssetBase + virtualBase}{maxAssetQuote + virtualQuote} \rightarrow higher\ virtuals \Rightarrow more\ stable\ price$$

- According to the oracle-risk ratio (plot 1), only a part of the amm's balance is included.

$$Ratio_{Base} = \begin{cases} 0.7 & -0.06 < x < 0.06 \\ 0.7 * (1 - (oracle - 0.06)/.94) & x <- 0.06 \\ e^{-6*oracle} & 0.06 < x \end{cases}$$

$$Ratio_{Qoute} = \begin{cases} 0.7 & -0.06 < x < 0.06 \\ 0.7 * (1 + (oracle + 0.06)/.94) & 0.06 < x \\ e^{6*oracle} & x <- 0.06 \end{cases}$$



---

4. Strategy2:

Pendulum:

To make the balance, pendulum strategy enters the market when the price is 10% above the Support line (both S1 and S2) and quits partially when the price is 35%, 50%, 65%, and 90% above the Support line.


Euro / U.S. Dollar · 1h · OANDA
O1.12267 H1.12302 L1.12166 C1.12208 −0.00060 (−0.05%)
USD
Resistance
T2
T1
Resistance turned Support
SL
1.11890
34:28
1.11880
1.11832
1.11712
1.11539
1.11512
12:00
27
12:00
28
12:00
29
12:00
Jun
12:00
2