app.library

Journal

- granularity: int
- investedAmount: double
- totalAmount: double
- currentTime: Instant
- currentValue: double
- MIN_TRADEABLE: double
- finish: boolean
- trades: List<Trade>
- hodlTrade: Trade

... fields for Report generation ...

+ Journal(int, Instant, double)
+ setCurrentCandle(Candle): void
+ hasAmount(): boolean
+ openTrade(double): Trade
+ roundDown(double): double
+ allIn(): Trade
+ finish(): void
+ closeTrade(Trade): Trade
+ openTrades(): List<Trade>
+ closeAll(): void
+ oldestTrade(): Trade
+ getTrade(int): Trade
+ hasOpenTrades(): boolean
+ lastOpenTrade(): Trade
+ openTradesCount(): long
+ closedTradesCount(): long
+ tradesCount(): long
+ investedAmount(): double
+ availAmount(): double
+ generateReport(): Report

Trade

- entryTime: Instant
- amount: double
- entryValue: double
- exitTime: Instant
- exitValue: double
- duration: long

~ Trade(Instant, double, double)
~ close(Instant, double, int): void
+ open(): boolean
+ closed(): boolean
+ amount(): double
+ entryTime(): Instant
+ exitTime(): Instant
+ entryValue(): double
+ exitValue(): double
+ profit(): double
+ duration(): long
+ profitable(): boolean

Candle

- openTime: Instant
- open: double
- high: double
- low: double
- close: double
- volume: double
- ta: Map<String, Double>
- granularity: int

+ Candle()
+ getTa(String): double
+ getTime(): Instant
+ getCloseTime(): Instant
+ getOpen(): double
+ getHigh(): double
+ getLow(): double
+ getClose(): double
+ getVolume(): double
+ getGranularity(): int
+ getTrueRange(): double
+ getTypicalPrice(): double
+ getIncrement(): double
+ getDecrement(): double
+ getRange(): double

... setters ...

<<interface>>
ExecutableStrategy

+ name(): String
+ indicators(): List<Indicator>
+ validate(): boolean
+ init(Journal): void
+ process(Journal, Candle): void
+ finish(Journal): void

app.library.indicators

<<interface>>
FacetPipeline

+ name(): String
+ pipeline(): List<Bson>

ATR

MOM

EMA

StdDev

SMA

app.library.indicators.enums

<<enumeration>>
InputPrice

O, H, L, C, TP, TR, U, D, V

+ getShortName(): String
+ fromShortName(String): InputPrice

Indicator

+ depends(): List<Indicator>
+ compute(Candle): void
- getProjection(InputPrice): Bson
- getAccumulator(InputPrice): BsonField
- getMapDocument(InputPrice): Document
getMappingStages(InputPrice): List<Bson>

MACD

OBV

BollingerBands

RS

RSI

StochRSI

<<use>>