**Trades Middleware Component**

**== Overview ==**

System under development is a middleware component that receives ticking trades from a stream. After every trade update the system recalculates trade statistics. For example, it recalculates the average price, largest trade by size and number of trade for a given symbol and flag.

**== Assumption ==**

It is assumed that the streaming source

- provides a subscribe method that is used to register interest in receiving trades

- provides an unsubscribe method that is used to deregister interest in trades

- calls the onMessage method to send every new trade to the subscribed classes

**== Design ==**

A pub/sub approach was used to design the program. The publishing is provided by the streaming source. This program provided the subscription part of the pub/sub.

Questions to be answered by the system are based on the symbol and flags. As such the system segregates and accumulates the statistical data per symbol. Any locks and concurrency are handled on the symbol level to ensure that data is not corrupted at that level.

Program structure consists of api, model and the statistical packages; average, count and largest. Each statistical package contains a subscriber and accumulator

- Subscriber subscribes with the streaming source in order to start receiving trades. It passes the data to the accumulator

- Accumulator segregates the trades by symbol. A single instance of the accumulator is created per symbol. For each symbol the statistics are calculated in a thread safe manner

A test driven approach was used. Unit tests were created for the statistical functions followed by implementation the logic. At each build cycle the unit tests are evaluated automatically. If any of the tests fail, the build fails.

Java 8 and maven 3 were used to develop, manage dependencies and build the program.

**== Future Development ==**

In order to extend the program, additional statistical packages will be created, subscriber and accumulator classes will be added. Subscriber classes will be registered with the streaming source.

Further extension can be achieved by separating the statistical packages into standalone projects that produce distributable jar file(s)