CENG 785 - Algorithmic Trading and Quantitative Strategies

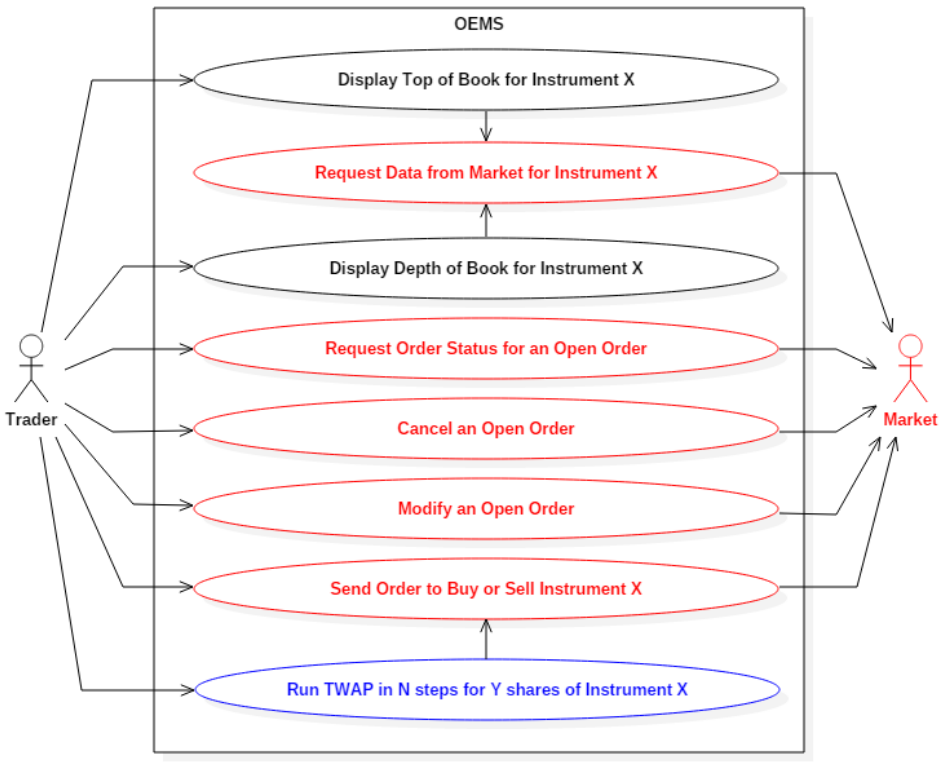
Fall 2016 - Homework 1

Order / Execution Management System

Selim Temizer

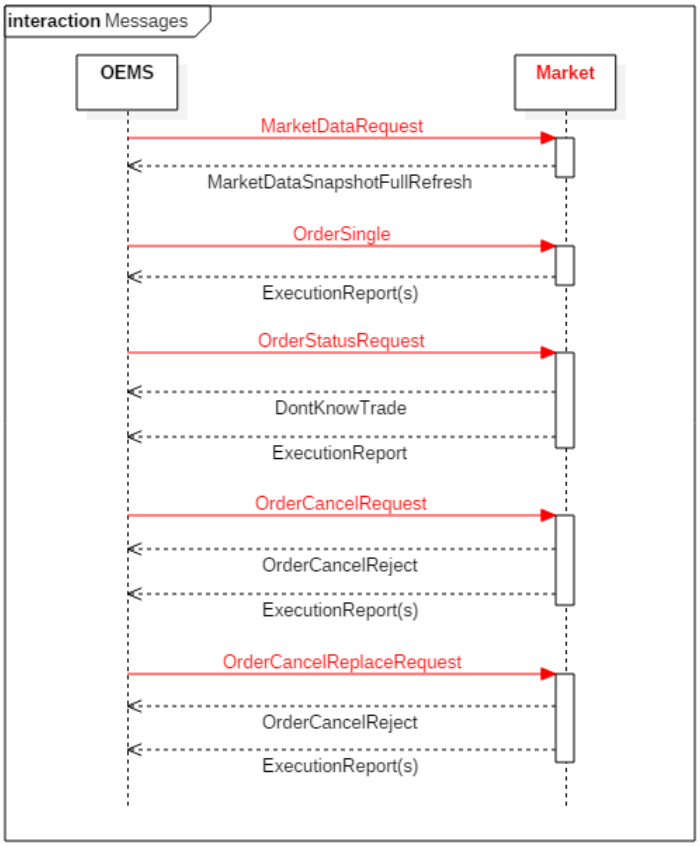
Feedback : Between November 14th and November 18th, 2016
Due date : November 20th, 2016 (Submission through COW by 23:55)

In this homework, we will build a software application that will basically enable us to send orders to a hypothetical market electronically using FIX protocol (version 4.2). The application should also allow the user to fetch live market data and to cancel or modify submitted open orders. In this aspect, our application will demonstrate the features of a small scale Execution Management System (EMS). The market may respond with cancels, partial fills or complete fills for the submitted orders. Our application should also inform us about the status of our open orders and display the unfilled amounts, thereby demonstrating the features of a small scale Order Management System (OMS). In addition, we want the application to help the trader with executing TWAP orders as well. A use case diagram that demonstrates the 7 functionalities (with an extra internal functionality) that should be provided to the trader is shown below:

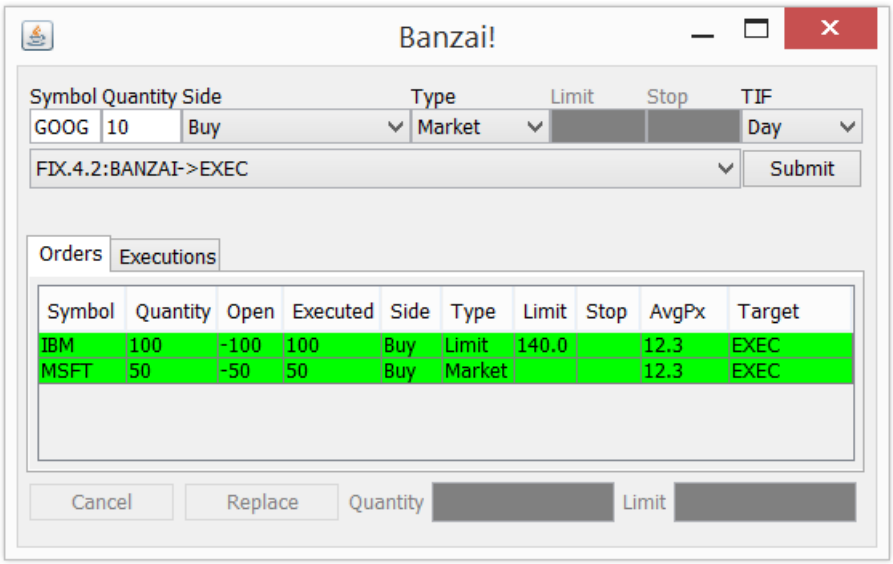


For the FIX engine part, we will use QuickFIX engine (your choice of the C++, Java or .NET versions) which is an open source and high quality implementation of the FIX protocol, supporting all versions (including version 4.2). The sequence chart below depicts the

interactions between our OEMS application and the market software in terms of the FIX 4.2 messages exchanged:



You may want to check out the ***Banzai*** application that comes as an open source example with the QuickFIX distribution. A screenshot of Banzai can be seen below. Our OEMS software will mostly be similar in functionality to Banzai. However, you are not required to build a GUI for your application (a nice and practical GUI would provide bonus points, though), your whole application might be text based and running from the command line. Please note that the set of functionalities provided in Banzai is a subset of the functionalities that we need to build for our application (such as order cancelling and market data display).



The following order parameters will be supported by a sample market simulator implementation that I plan to build (therefore, your application does not need to support any other parameters) :

Side	Buy Sell
TimeInForce	Day Good_Till_Date Good_Till_Cancel Fill_Or_Kill Immediate_Or_Cancel
OrdType	Market Limit
HandlInst	Automated_Execution_Order_Public

Simple OEMS and Market software implementations in Java programming language are also provided (along with QuickFIX configuration files and also output transcripts from a sample run) as an example for you. Both the OEMS and the Market software print out all messages that they exchange (you may copy and paste the message texts in some online FIX message parsers, and get more detailed information during implementation and debugging). In addition, the OEMS application shows how to build and send a message to the market, and the Market application shows how to crack a received message.

As I have not finished building the market simulator and a sample solution to this homework assignment yet, there might be certain details that we need to work out and clarify together during lectures and/or through the course newsgroup. Feel free to post any questions, ideas, suggestions, etc.

There are a number of design decisions (for example: provide upmost flexibility in terms of the possible parameters of the required OEMS services) and opportunities for visual enhancements (for example: build a GUI similar to or better than Banzai) and creative extensions (for example: additional trading algorithms) that are deliberately left open-ended in this homework specification. We have enough time until the deadline to discuss your suggestions and make further clarifications as necessary. There will be bonuses awarded for all types of extra effort. Late submissions will NOT be accepted, therefore, try to have at least a working baseline system by the deadline. Good luck.

What to submit? (Use ***only ASCII characters*** when naming all of your files and folders)

1. A short (1 or 2 pages) description of how to compile, start, and use your application, your design decisions that you made during implementation, and any other information that is necessary to run your code (as a ***doc*** or ***pdf*** file) in a directory named "***Docs***". You may add any other documentation, UML diagrams, etc. related to your solution as you see fit.
2. Your OEMS application source code (all in a single directory named "***Source***"). If you are using an object oriented language, then do not use name spaces and/or packages that may cause difficulty in compiling. Do not submit IDE created project files or other binary files. Just submit your source code files.
3. Your compiled and ready-to-run application (as a jar, exe or elf file) with all required run-time libraries (in a single directory named "***OEMS***").
4. Note that you do not need to create and submit any Market software implementation. However, if you build simple Market implementations to test your OEMS code, you may put your source code and runnable files for your Market implementation in a directory named "***Market***".

Zip the 3 directories (or 4 if you want to submit Market) above (tar also works, but I prefer Windows zip format if possible), name the compressed file as <ID>_<FullNameSurname> (with the correct extension of .zip or .tar) and submit it through COW. For example:

e1234567_SelimTemizer.zip