Order Book

Objective

The objective of this individual assessment is to build an Order Book script using most of the learning outcomes of module #1

Overview - Scenario and Design

You've been asked to develop a Python script for an electronic trading platform. The script will be available to end users, stock traders/clients who can place their orders (BUY & SELL) for trading, view their order status and the trade/order history.

Class Hierarchy & Sample Design Guidelines

OrderBook:

member variable:orders

method:newOrder

method:cancelOrder

Exchange:

member variable:feeLadder

member variable:orderBooks

member variable:todaysTradeValue

SORT:

member variable:exchanges

method:executeTrade

member variable:orderBooks

SORT - trades on various exchanges based on best prices and fees. There would be at least one instance of SORT per region (eg: EMEA/APAC etc.). This class requires Exchange object to model exchange fees and how they change based on feeLadder (bulk=cheaper)

Exchange needs **orderbooks** so that we can store the exchanges orderbook and therefore decide whether an exchange has liquidity and a good price or not.

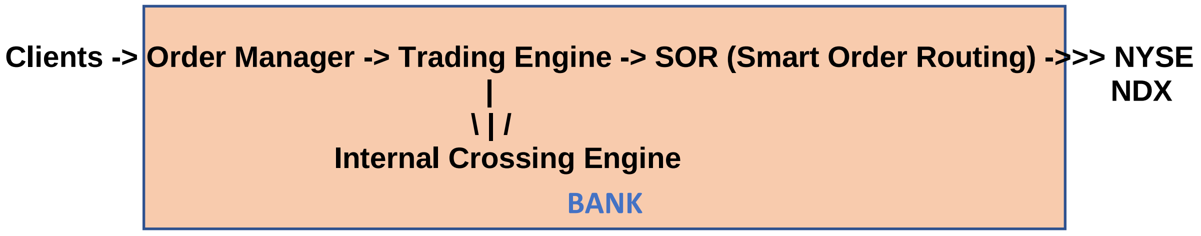
SORT requires **orderbooks** of the orders it's trading – this allows SORT to perform internal crossing. Generally, each instrument will have its own orderbook (with a collection of orders), BT has 1, VOD has 1 and so on

HINT: To start with you may consider generating sample orders 500 each (BUY and SELL) with some random prices and populate database tables

Functional Requirements

1. Ability to register and login (Use file system to store user details)
2. Ability to view BUY and SELL Orders – You may consider storing Order details in CSV or any other file formats
3. Ability to manage orders (Add new orders, cancel, or replace orders)
4. The status of all the orders (partially fill or fully filled)
5. Ability to Slice orders and send to SORT for now – avoid other ways to fill the orders other than sending them to SORT
6. Match Single orders (BUY -> SELL) to execute trades
7. Match multiple orders (BUY -> SELL) to execute trades
8. Ability to view trade history

Example - Electronic Trading System's Workflow


Clients -> Order Manager -> Trading Engine -> SOR (Smart Order Routing) ->>> NYSE
NDX
|
\ | /
Internal Crossing Engine
BANK

Marking Scheme

Order Book				
Requirements	Meets Expectations	Needs Improvement	No Submission	Marks Scored
Script demonstrated minimum suggested requirements	25	15	0	
Datatypes, operators, conditionals and loop constructs learning outcomes are demonstrated	10	5	0	
Python script uses appropriate Object-Oriented Programming principles	10	5	0	
Python collections – Python collections learning outcome demonstrated	10	5	0	
Exceptions handling learning outcome – Script demonstrates exceptions and custom exceptions	5	3	0	
File Handling learning outcome – Script uses file(s) to store data and retrive data	10	5	0	
Script uses Lambda expression for function style code	5	3	0	
Pioneer can explain Object-Oriented principles	5	3	0	
Pioneer can explain Python collections with suitable examples	10	5	0	
Coding Style – Appropriate comments, naming conventions and readable code is demonstrated	10	5	0	