System Architecture Report

Common Wealth Hedge Fund

FE545 Final Project Report

Common Wealth Hedge Fund
Portfolio Management

Long/Short Strategy

Instructor: Xugong Li

Group Member:
Zi Lang Wong
Yuxuan Xia
Xiao'ao Song
Arya Nasir Tafreshi

Submission: May 7th. 2017

Table of contents

Long/Short Strategy

1. Project Objective

The Project objective is to learn design patterns in designing the system, as well as understand how to run an asset management firm.

2. Project members

a. Arya Nasir Tafreshi
b. Zi Lang Wong
c. Xiao'Ao Song
d. Yuxuan Xia

3. Trading Strategies

The long short strategy focused on mimicking the investment process that has been used by hedge funds. The investment strategy involves universe ranking, long top ranked securities, and short bottom ranked securities. The main objective of the strategy is to focus on quality of security's fundamental value, and eliminate market risk by achieving 0 beta in comparison to our benchmark – Nasdaq100.

4. Long Short Strategy Algorithm

a. The strategy has the Nasdaq100 as the benchmark and investment universe, which mean the strategy will only invest among the stocks that fall in Nasdaq100. The portfolio management strategy follows the algorithm below:
b. Import Nasdaq100 constituents
c. Collect fundamental value from Nasdaq (Momentum, Valuation metrics)
d. Calculate an average score based on the fundamental value for each stock, and rank the stocks accordingly in the universe
e. Sort the universe according to the stock's ranking respectively
f. Select the top 20 ranked stocks and place them into the Long Basket, which

means the basket to buy

g. Select the bottom 20 ranked stocks and place them into the Short Basket, which means the basket to sell
h. Calculate the current portfolio value by importing the latest pricing in our portfolio (The portfolio starts with an initial investment of $1,000,000)
i. Invest 2.5% of the current portfolio value to each individual stock in our basket (Ex. The current portfolio value is $1,025,000, and the value to invest in each stock is $1,025,000 * 0.025 = $25,625)
j. Calculate the units to buy and sell, since the Long Basket and Short Basket are generated, the units to buy/sell can be calculated by the value to invest in each stock divided by the current market price (Ex. $25,625 / $142.06 = 180 Units to buy/sell)
k. Calculate the actual units to be executed in the public market, since the portfolio might already held a fraction of the units, we need to calculate the remaining units to be executed in the market. (Ex. 168 Units of AAPL stocks were already held previously, the remaining units to be bought in the market is calculated by 180 – 168 = 12 Units)
l. Execute trades and rebalance portfolio

5. Financial Instruments that the Project will work on

a. Equity Market
b. Constituents in NASDAQ100

6. Task Assignment of Members

a. Project Selection Report - Zi Lang Wong
b. Trading Strategy Design - Zi Lang Wong
c. Nasdaq Data Crawling, - Yuxuan Xia
d. Design Patterns Programming - Xiao'Ao, Arya
e. System Architect - Xiao'Ao
f. Presentation - Xiao'Ao, Zi Lang, Yuxuan, Arya
g. Final Report - Zi Lang, Xiao'Ao, Arya, Yuxuan

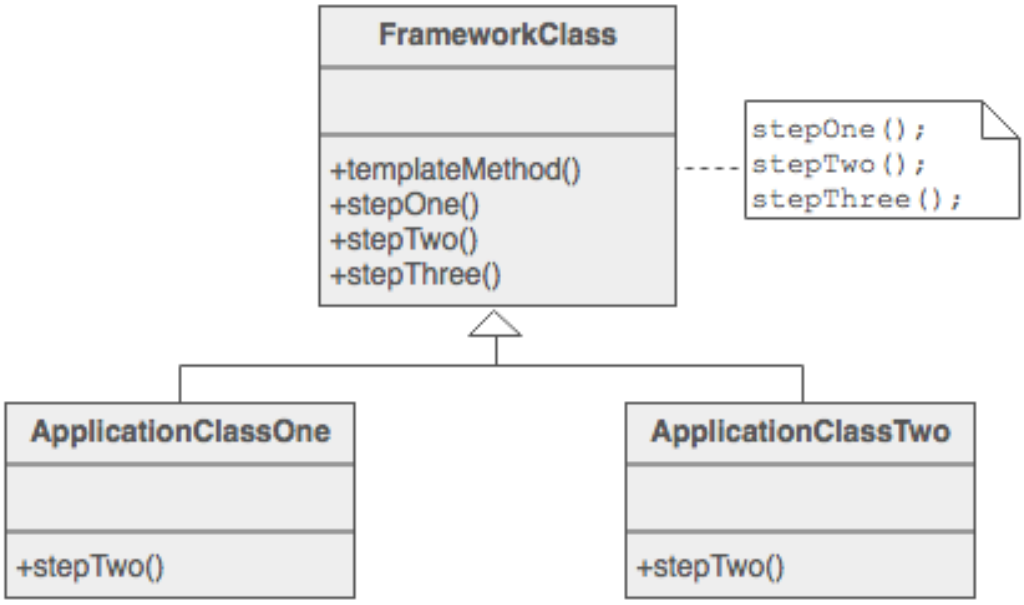
7. Five Design Patterns that include structure, and definition

a) Template Method

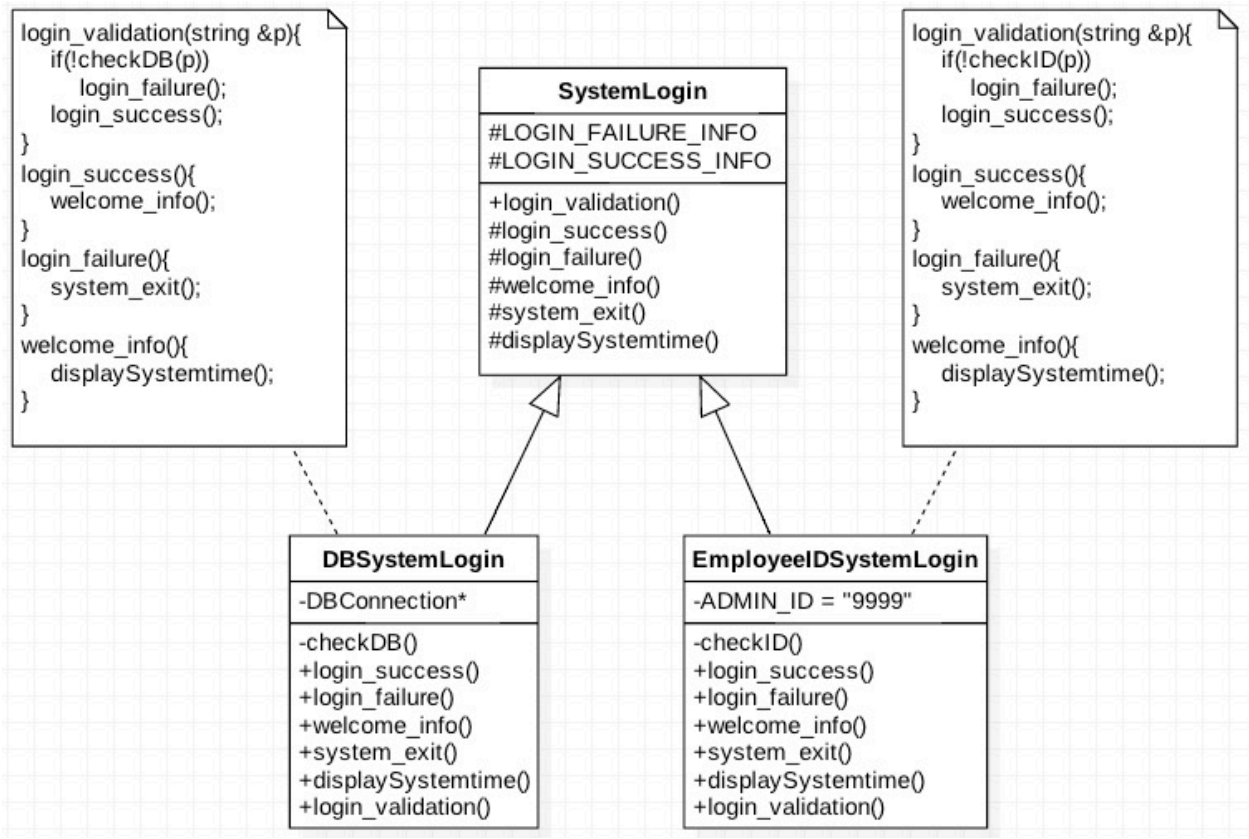
Intent:

- Define the skeleton of an algorithm in an operation, deferring some steps to client subclasses. Template Method lets subclasses redefine certain steps of an algorithm without changing the algorithm's structure.
- Base class declares algorithm 'placeholders', and derived classes implement the placeholders.
  Two different components have significant similarities, but demonstrate no reuse of common interface or implementation. If a change common to both components becomes necessary, duplicate effort must be expended.

Conceptual Graph：



Common Wealth HF System Code Sample:

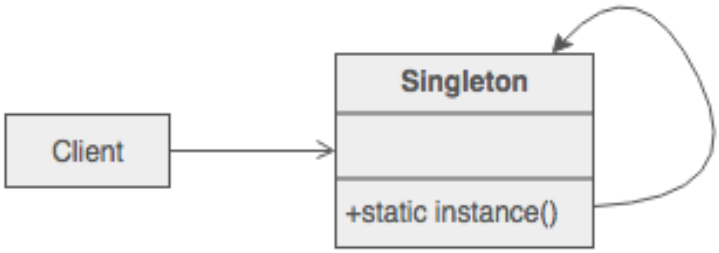


b) Singleton Method

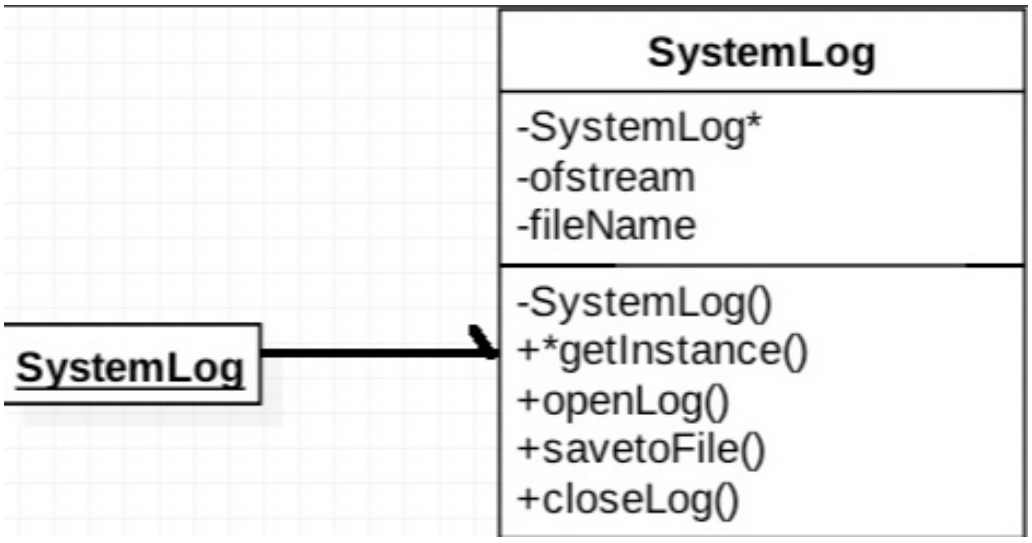
Intent:

- Ensure a class has only one instance, and provide a global point of access to it.
- Encapsulated "just-in-time initialization" or "initialization on first use". Application needs one, and only one, instance of an object. Additionally, lazy initialization and global access are necessary.

Conceptual Graph：



Common Wealth HF System Code Sample:



c) Strategy Method

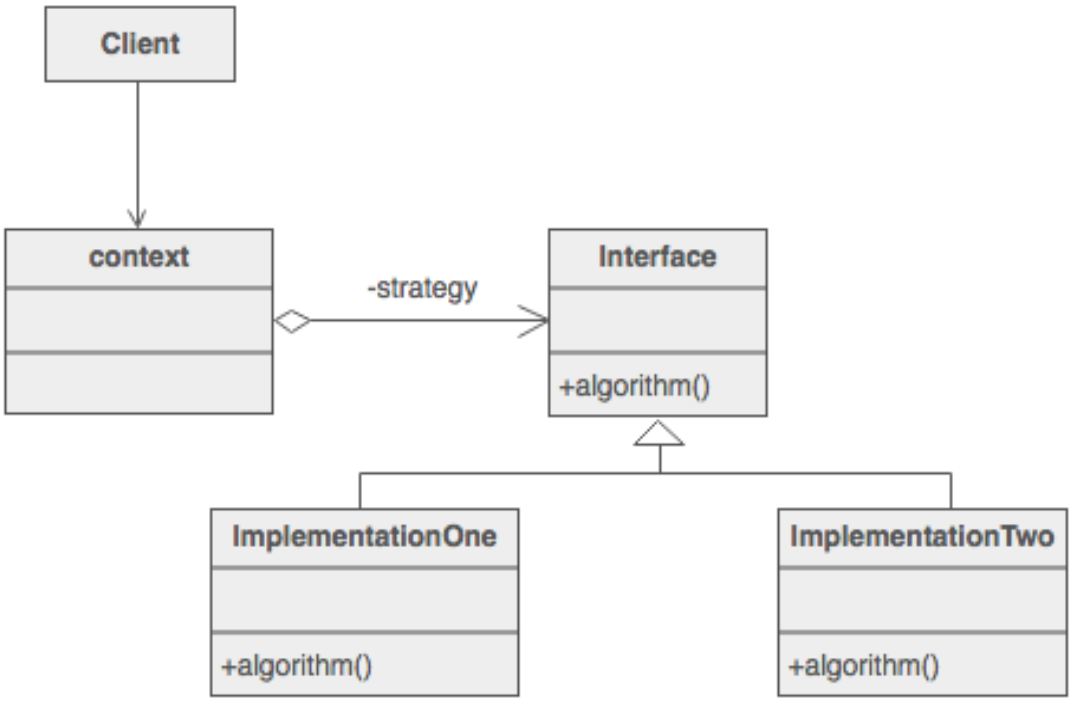
Intent:

- Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from the clients that use it.
- Capture the abstraction in an interface; bury implementation details in derived classes. A generic value of the software community for years has been, "maximize cohesion and minimize coupling". The object-oriented design approach shown in figure is all about minimizing coupling. Since the client is coupled only to an abstraction (i.e. a useful fiction), and not a particular realization of that abstraction, the client could be said to be practicing "abstract coupling" . an object-oriented variant of the more generic exhortation "minimize coupling".

  A more popular characterization of this "abstract coupling" principle is "Program to an interface, not an implementation".

  Clients should prefer the "additional level of indirection" that an interface (or an abstract base class) affords. The interface captures the abstraction (i.e. the "useful fiction") the client wants to exercise, and the implementations of that interface are effectively hidden.

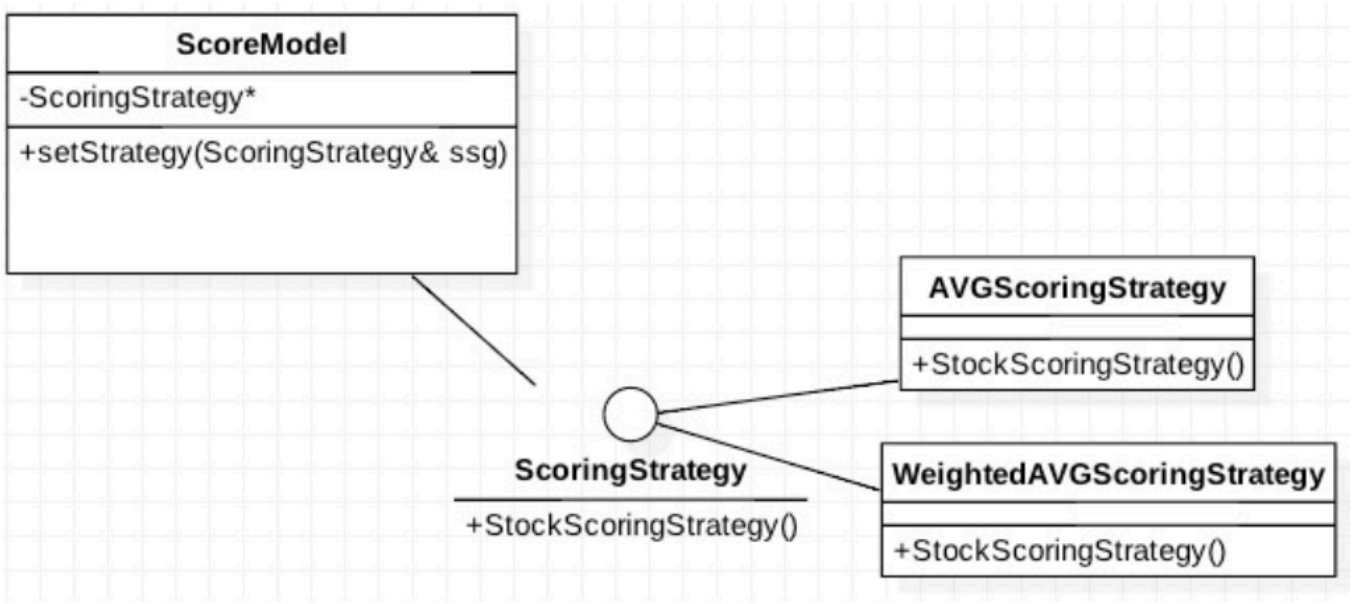
Conceptual Graph：



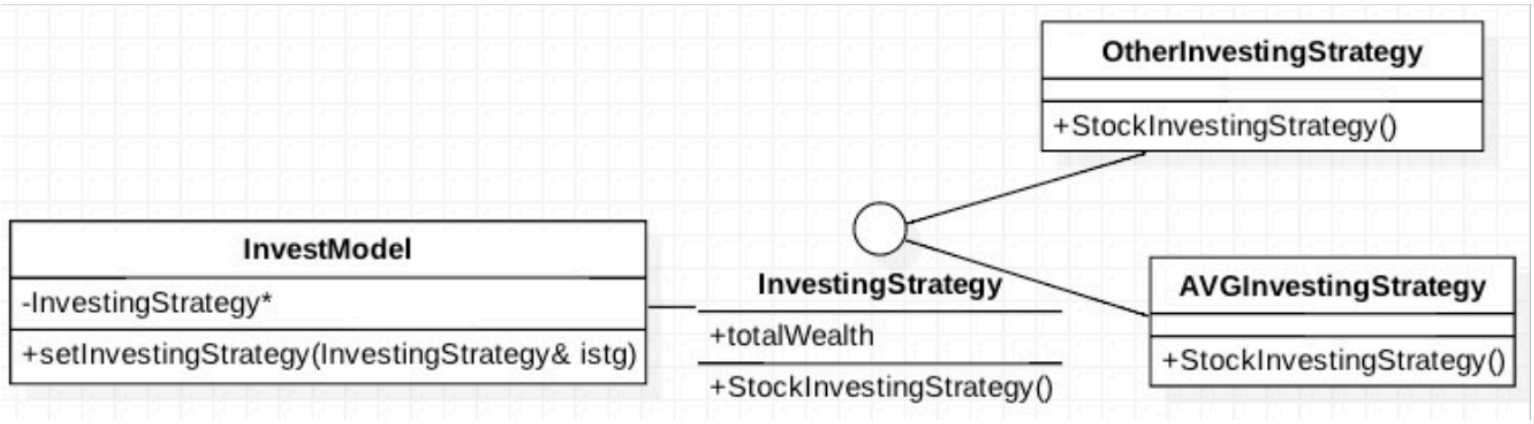
Common Wealth HF System Code Sample:

The core part of our HF System incorporates two models: stock scoring model and stock investing model. Therefore, we use strategy design pattern for both models.

ScoreModel and Scoring Strtegy



InvestModel and Investing Strtegy



d) Object Pool Method

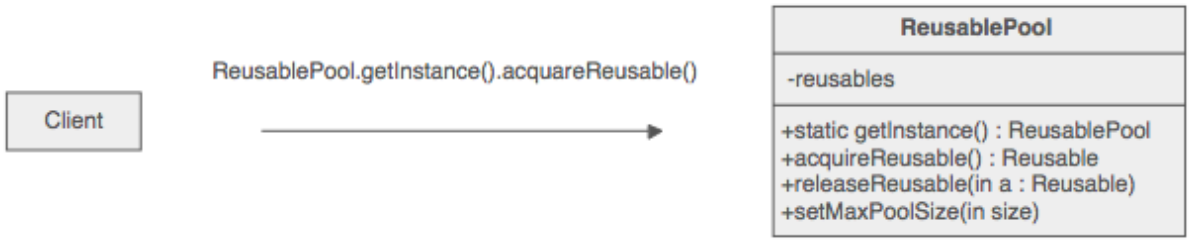
Intent:

Object pooling can offer a significant performance boost; it is most effective in situations where the cost of initializing a class instance is high, the rate of instantiation of a class is high, and the number of instantiations in use at any one time is low.

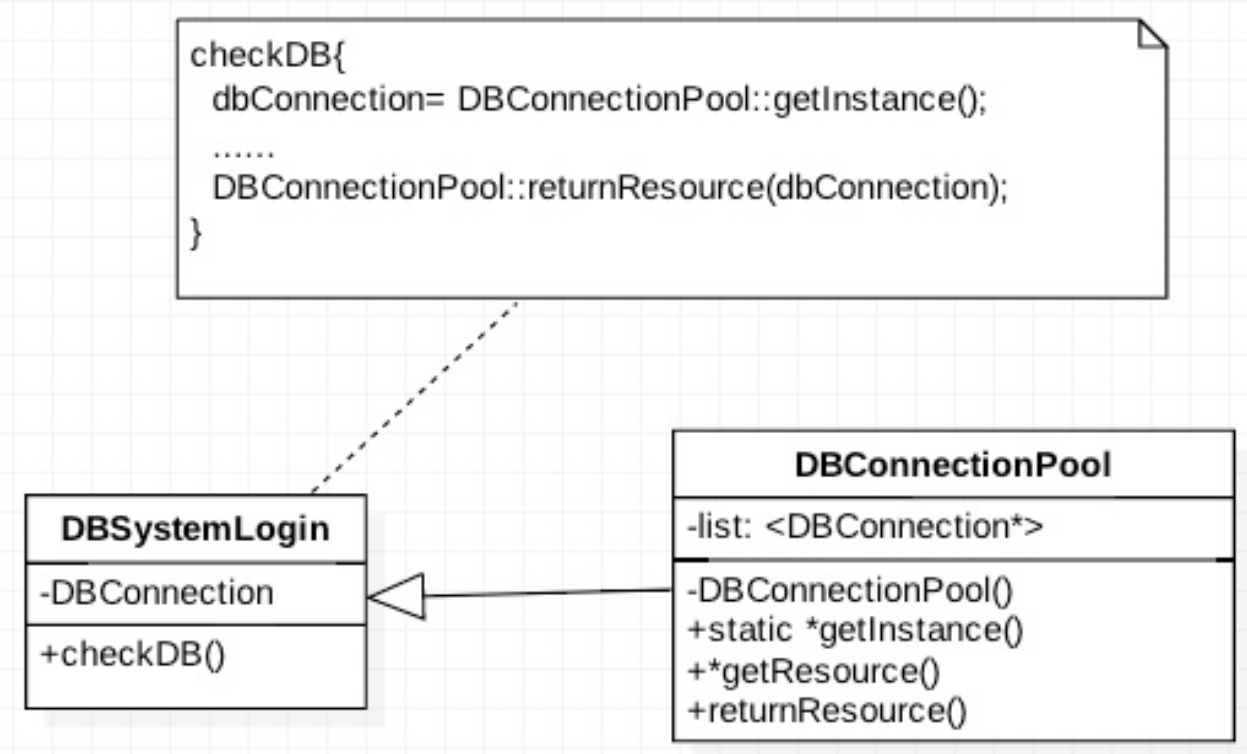
Object pools (otherwise known as resource pools) are used to manage the object caching. A client with access to a Object pool can avoid creating a new Objects by simply asking the pool for one that has already been instantiated instead. Generally the pool will be a growing pool, i.e. the pool itself will create new objects if the pool is empty, or we can have a pool, which restricts the number of objects created.

It is desirable to keep all Reusable objects that are not currently in use in the same object pool so that they can be managed by one coherent policy. To achieve this, the Reusable Pool class is designed to be a singleton class.

Conceptual Graph：



Common Wealth HF System Code Sample:

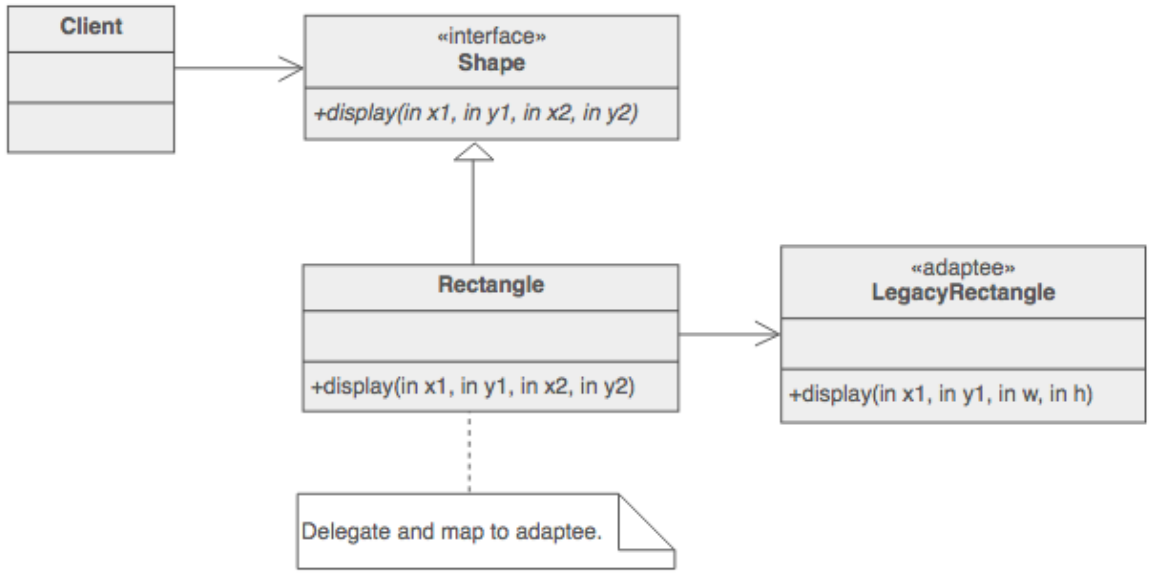


e) Adapter Method

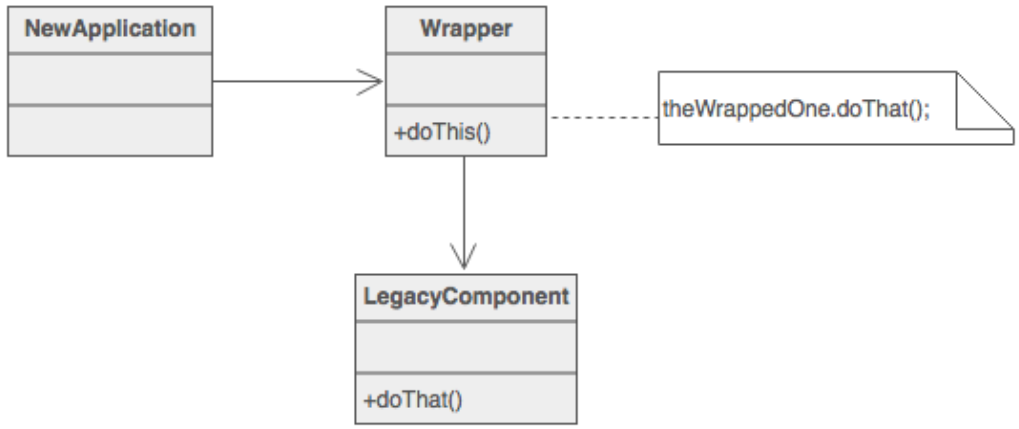
Intent:

- Convert the interface of a class into another interface clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interfaces.
- Wrap an existing class with a new interface.
- Impedance match an old component to a new system

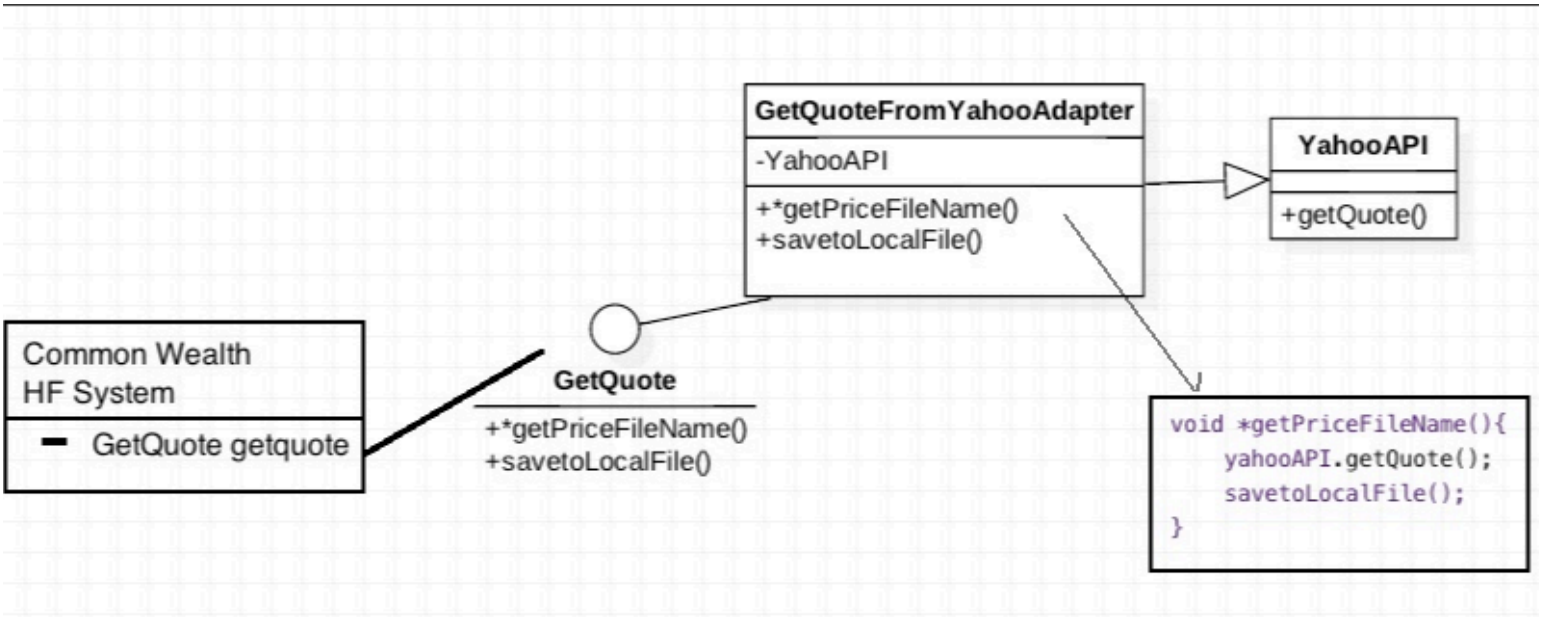
Conceptual Graph：



The Adapter could also be thought of as a "wrapper".



Common Wealth HF System Code Sample:

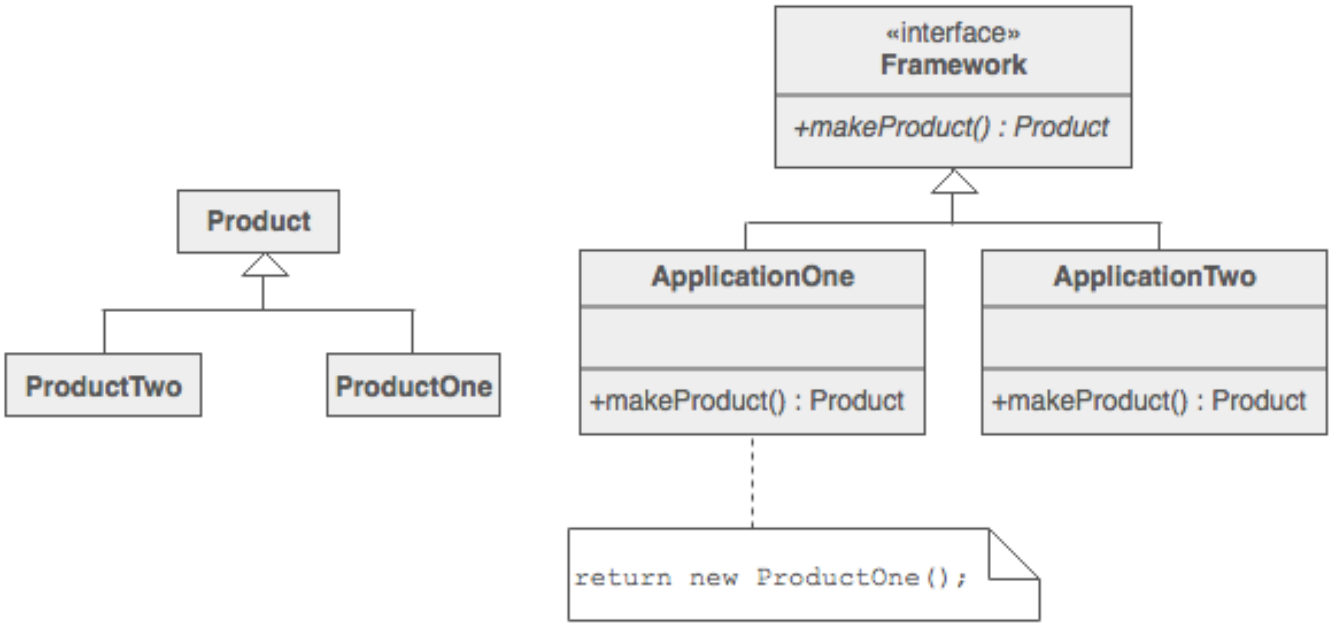


f) Factory Method

Intent:

- Define an interface for creating an object, but let subclasses decide which class to instantiate. Factory Method lets a class defer instantiation to subclasses.
- Defining a "virtual" constructor.
- The new operator considered harmful.
  A framework needs to standardize the architectural model for a range of applications, but allow for individual applications to define their own domain objects and provide for their instantiation.

Conceptual Graph：



Common Wealth HF System Code Sample:

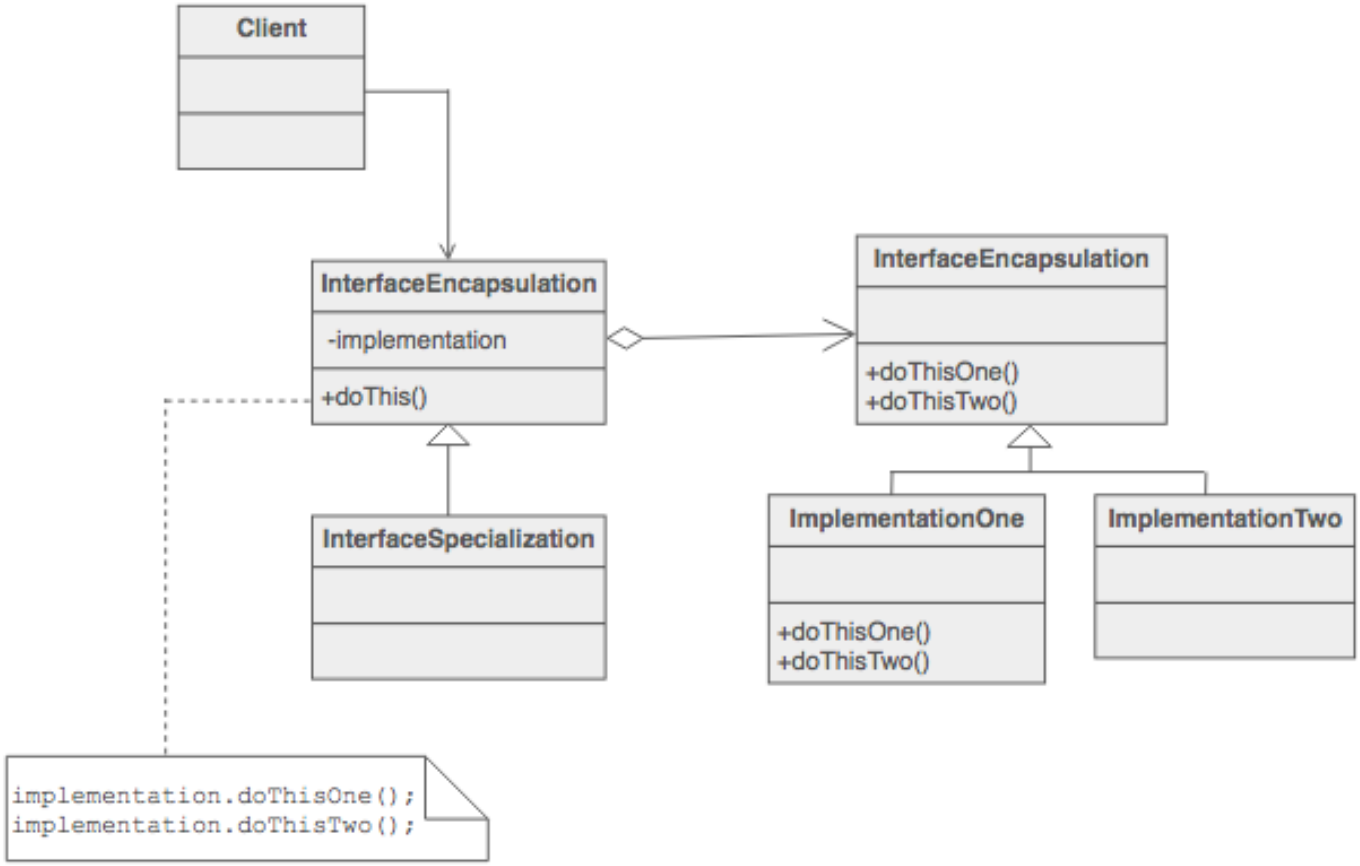
This design pattern will be used in second phrase of our system construction.

g) Bridge Method

Intent:

- Decouple an abstraction from its implementation so that the two can vary independently.
- Publish interface in an inheritance hierarchy, and bury implementation in its own inheritance hierarchy.
- Beyond encapsulation, to insulation
  "Hardening of the software arteries" has occurred by using subclassing of an abstract base class to provide alternative implementations. This locks in compile-time binding between interface and implementation. The abstraction and implementation cannot be independently extended or composed.

Conceptual Graph：



Common Wealth HF System Code Sample:

This design pattern will be used in second phrase of our system construction.

8. The input and results of the project

a. Input:

i. Nasdaq Factor Data
ii. YahooAPI Stock Quotes
iii. Client Investment Fund

b. Results:

i. Invested Portfolio
ii. Profit and Loss on a daily basis
iii. Limit Order Book

Example: These are the orders generated from the system to rebalance our portfolio, and ready to be executed in the market.

aaon	| -99
abmd	| -137
avhi	| 23
flws	| 16
jobs	| 37
srce	| -205

Here is the screenshot by our system:

```
finishing investing stocks.... and going to submit the order book now
the limit order book is:
aaon|-99
abmd|-137
avhi|23
flws|16
jobs|37
srce|-205
log is closing

Process finished with exit code 0
```

9. Connections

a. YahooAPI

In this project we use Yahoo API to get real data. Yahoo Finance provides a great and simple way to download free stock quotes.

This service returns stock data in .CSV (comma delimited format, you can just open it in Excel if you like) and we can use this link in any other programing language.

The base URL you're going to call is *finance.yahoo.com/d/quotes.csv*

Then you add a ?s= and the stock symbols you're interested in such as APPL, GOOG and MSFT like so *finance.yahoo.com/d/quotes.csv?s=AAPL+GOOG+MSFT*

Then you specify the info you want. There is a large list of stuff you can specify, just look at the list below for more info.

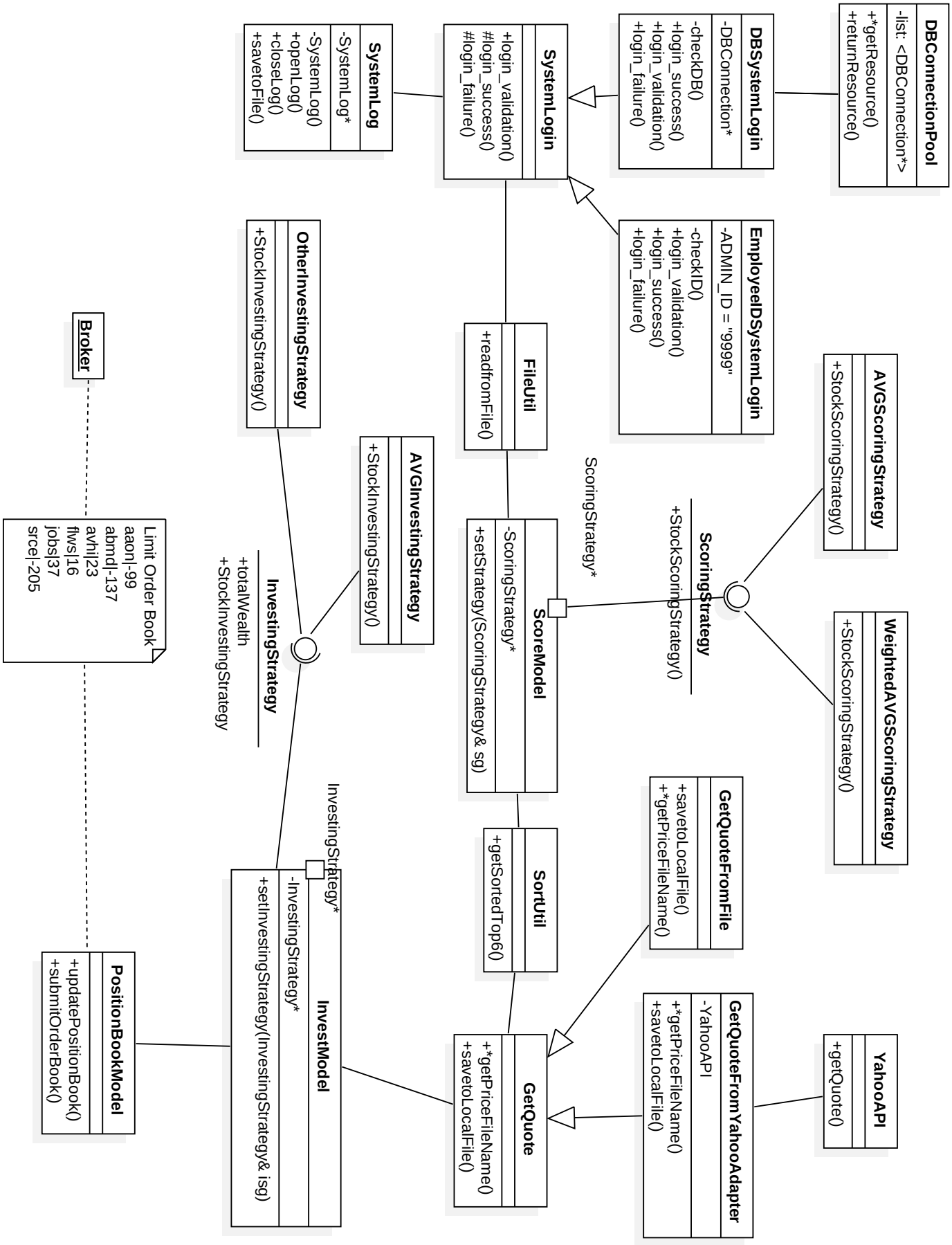
This is the sample list of Yahoo API:

a	Ask	a2	Average Daily Volume	a5	Ask Size
b	Bid	b2	Ask (Real-time)	b3	Bid (Real-time)
b4	Book Value	b6	Bid Size	c	Change & Percent Change
c1	Change	c3	Commission	c6	Change (Real-time)
c8	After Hours Change (Real-time)	d	Dividend/Share	d1	Last Trade Date
d2	Trade Date	e	Earnings/Share	e1	Error Indication (returned for symbol changed / invalid)
e7	EPS Estimate Current Year	e8	EPS Estimate Next Year	e9	EPS Estimate Next Quarter
f6	Float Shares	g	Day's Low	h	Day's High
j	52-week Low	k	52-week High	g1	Holdings Gain Percent
g3	Annualized Gain	g4	Holdings Gain	g5	Holdings Gain Percent (Real-time)
g6	Holdings Gain (Real-time)	i	More Info	i5	Order Book (Real-time)
j1	Market Capitalization	j3	Market Cap (Real-time)	j4	EBITDA
j5	Change From 52-week Low	j6	Percent Change From 52-week Low	k1	Last Trade (Real-time) With Time

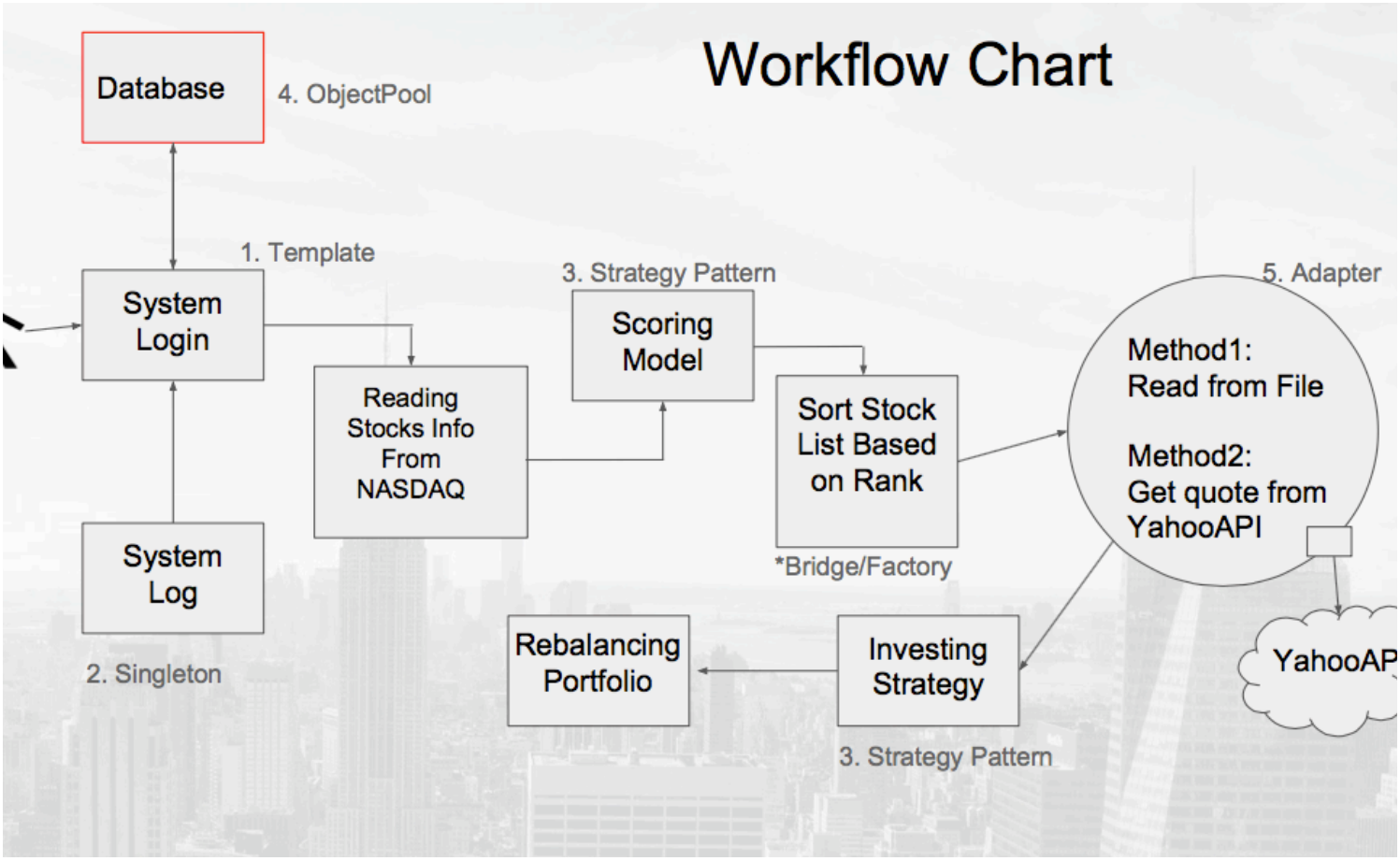
b. others

Since the **Adapter design pattern** is applied in our HF system, the system will also support other quote resources in subsequent versions such as Thomson Reuters and etc.

10. UMLDiagram



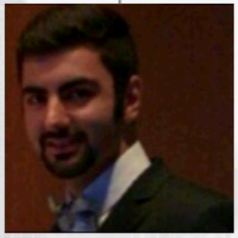
Workflow Chart



Company Structure

CEO (Xugong Li)

Data Specialist
YuXuan Xia

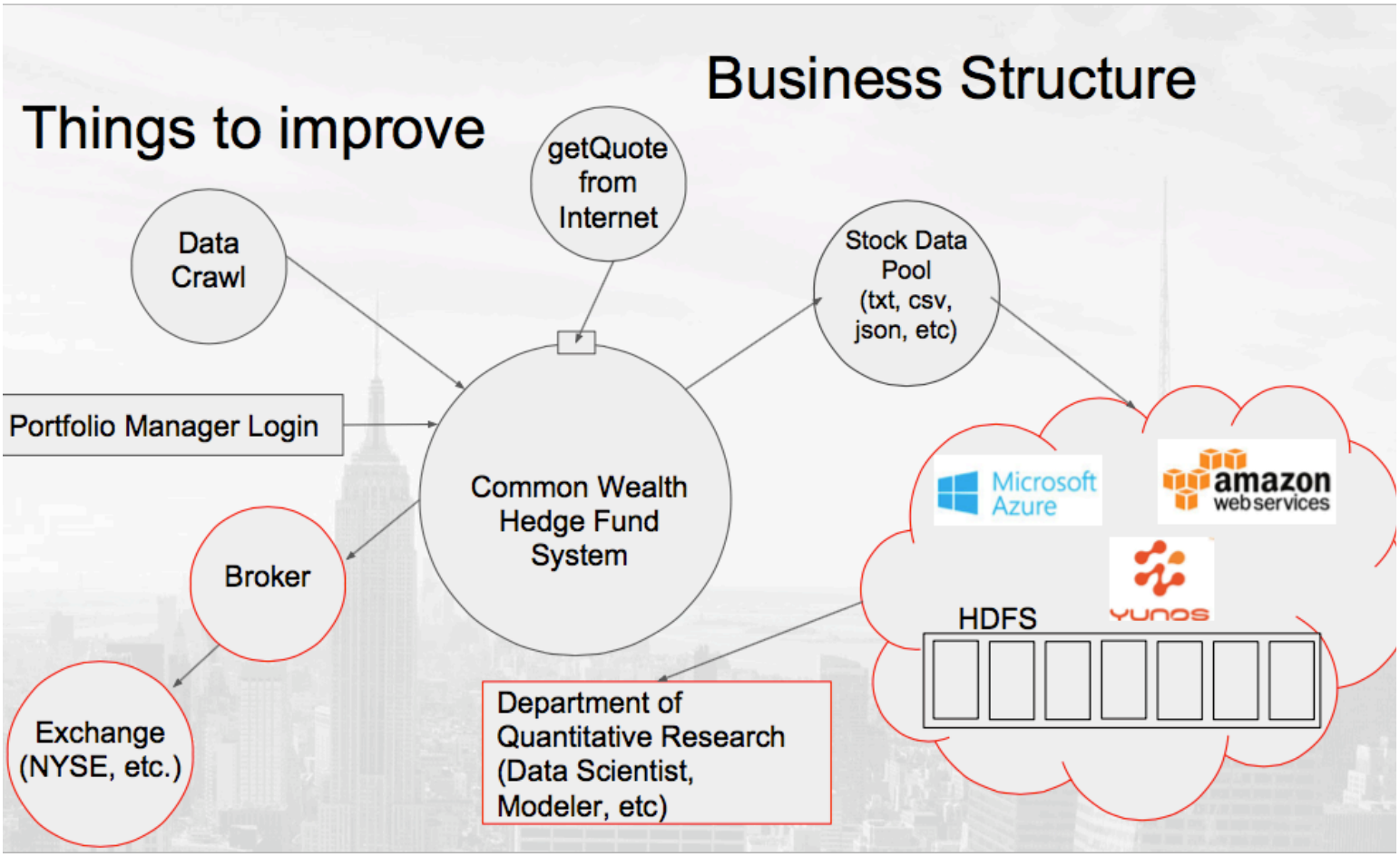


COO
Arya Nasir Tafreshi

System Architect
Xiao'Ao Song

Portfolio Manage
Zi Lang Wong

11. How can we improve



As we illustrated in the graph above, the things we can improve in the future include synchronizing stock data to the cloud and create our own data cloud storage, and therefore use by the department of Quantitative Research in the future to process and extract useful information.

Appendix

Construction of Our Dataset

Intent

Our system is aimed to provide valuable data that our client can use these data make their own portfolio or do some research on it. Since we are a tiny company who can not afford expensive cost from financial data provider such as bloomberg which cost $20,000 annually.

In order to obtain reliable real-time financial information which is critical to added-value financial services, the web is our first order data source.

We all know the web contains valuable information, but the untapped potential of information that is out there, but not structured or used correctly, is enormous.

Any doubts about that statement were put to rest recently when a web-data firm, Selerity, uncovered Twitter's Q1 earnings results about an hour before the scheduled release. Twitter's stock promptly fell 5 percent, losing roughly $1.6 billion of market capitalization. The stock continued to tumble throughout the day, as the earning results were lower than expected, and investors who knew and acted on the information first profited enormously.

A day later, a similar story unfolded with Salesforce.com. When news reports surfaced that Salesforce.com had hired bankers to help review bids from potential acquirers, Salesforce shares jumped 11 percent, adding more than $5.3 billion to Salesforce's total market value.

The stocks in these stories moved in different directions, but the takeaway is the same – early insights into company health and financial data can be enormously valuable.

Why Crawler?

Firstly, most of U.S financial instruments do not have an alternative such as API.

Secondly, not all person or company can afford the expense of market-move data. Finance professionals can judge such expenses because the sums they trade are so large. But individual investors and developers working on a custom trading algorithm typically lack such resources.

In the Twitter example, Twitter's investor relations website posted the company's earnings announcements to a URL that followed the same pattern as all other announcements. The firm was counting on the fact that nobody was looking at that URL, but with the ability to easily monitor URLs of interest, a web crawler like kimono can easily detect that change in base data.

While laws and regulations are in place to prevent insider knowledge from

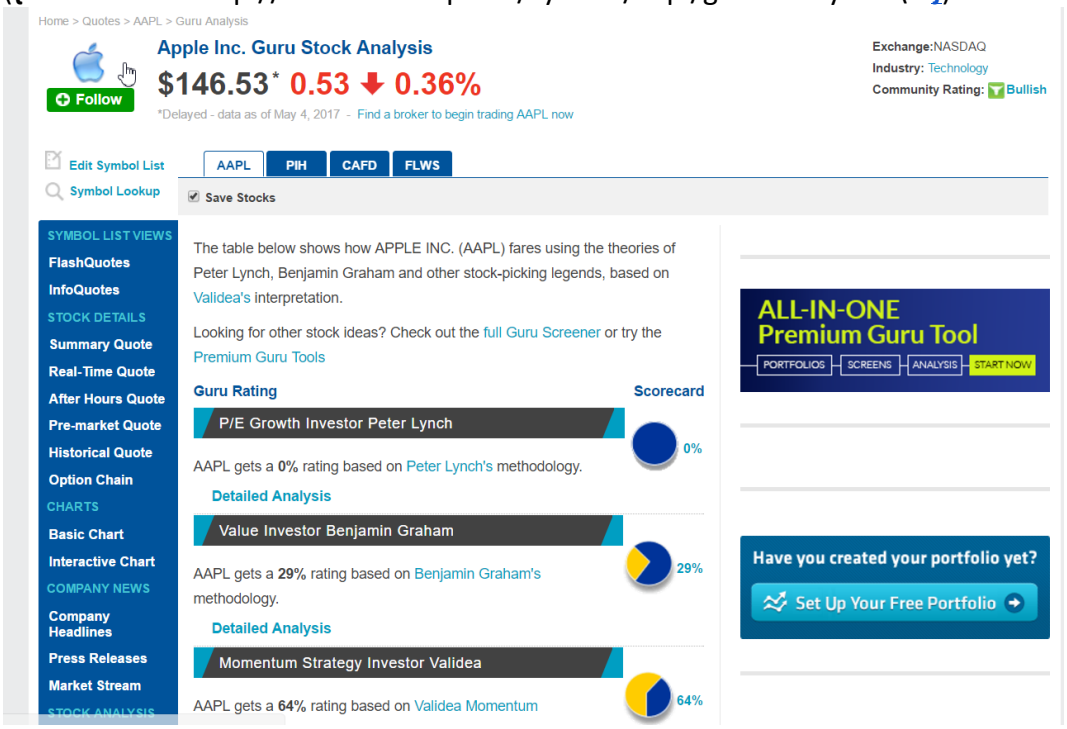
swaying the markets, there is nothing that prevents using public information smarter and faster than your competitors.

Guru Analysis

Nasdaq's Guru Analysis offers an analysis of any stock using their interpretation of the strategies of Wall Street Legends like Warren Buffett, Peter Lynch and Benjamin Graham.

The screenshot of the website is displayed below.

({ HYPERLINK "http://www.nasdaq.com/symbol/aapl/guru-analysis" \h })



As there are 8 guru rating scores in each stock's page, we can utilize these scores as we believe there is a relationship between guru scores and the future performance. One benefit of data mining is that we don't need to know the exact theories behind these scores. Because result-oriented test (such as learning and allocating) can optimize our portfolio since the booming of data science with the machine learning, artificial intelligence and hardware technique innovation such as GPU computing technique.

What's more, it's never a wrong strategy by standing on the shoulder of giants. And that is the theorem basis about why we choose guru analysis as our first market-move data.

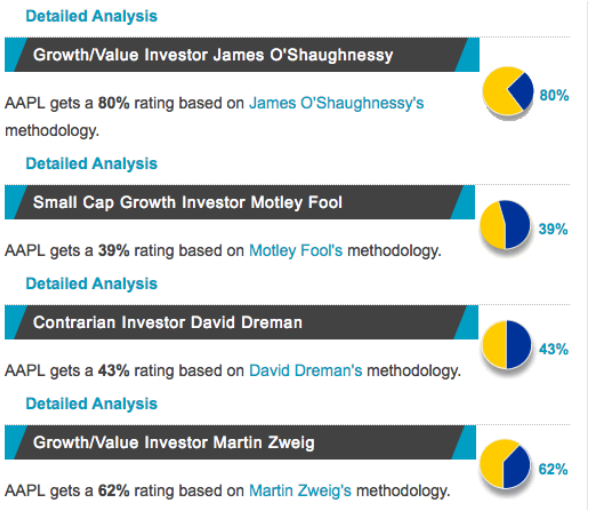
A Glimpse about Scrapy

Scrapy is an application framework for crawling web sites and extracting structured data which can be used for a wide range of useful applications, like data mining, information processing or historical archival. In this report, we choose these framework to construct our own web crawlers.

Crawling algorithm

1. Pick website: I'm going to scrape
   http://www.nasdaq.com/symbol/<ticker>/guru-analysis

   in which <ticker> can be replaced as stock ticker in Nasdaq 100 list such as AAPL.



2. We need to inspect the structure of these pages. In this case, we use chrome browser and open our inspection toolkit by press ctrl+shift+I. We then find that scores is wrappered in html tags be of the form: <a>scores we need<a>.

Because the data in this page is well structured, we can retrieve our scores easily, and here is the result using scrapy shell.

```
In [14]: response.xpath("//a[@href]/b/text()")
Out[14]:
[<Selector xpath='//a[@href]/b/text()' data='0%'>,
 <Selector xpath='//a[@href]/b/text()' data='29%'>,
 <Selector xpath='//a[@href]/b/text()' data='64%'>,
 <Selector xpath='//a[@href]/b/text()' data='80%'>,
 <Selector xpath='//a[@href]/b/text()' data='39%'>,
 <Selector xpath='//a[@href]/b/text()' data='43%'>,
 <Selector xpath='//a[@href]/b/text()' data='62%'>,
 <Selector xpath='//a[@href]/b/text()' data='10%'>]
```

With all those preparation, we could start our crawling.

+Problems and How to prevent getting blacklisted while scrapping.

Since web crawlers, scrapers don't really bring in human website traffic and seemingly affect the performance of the site, some site administrators do not like spiders and try to block their access.

At first, without any optimization,I just used my own IP address without any user agent(browser information), the server detected the scraper behavior and blocked my IP very quickly so that we scrapped successfully no more than 20 pages.

In order to prevent website from detecting our crawler, we generate the following strategies:

1. Make the crawling slower, do not slam the server, treat them nicely
2. User-agent spoofing
3. Disguise our requests by random IPs and Proxy servers

Solutions and Practices

In order to slower our crawler, the best way is using auto throttling mechanisms which will automatically throttle the crawling speed based on the load on both the spider and the website that you are crawling. And adjust the spider to an optimum crawling speed after a few trial runs and do this periodically because the environment does change over time.

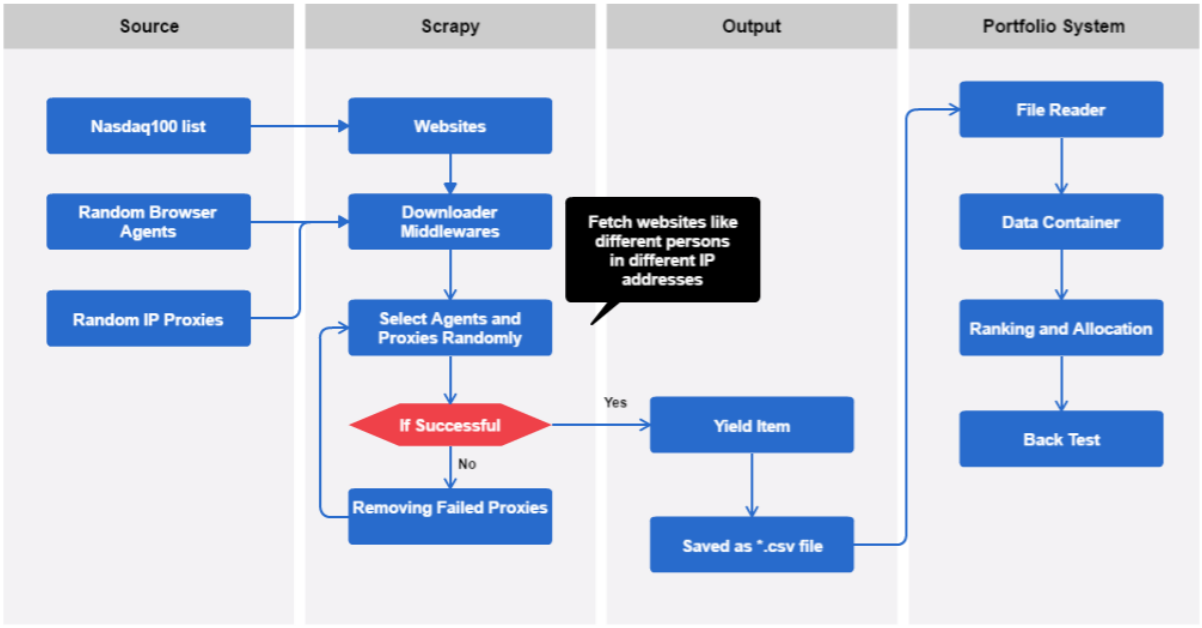
However, this method is inefficient since we are going to scrape enormous data, time delay is in a lower priority.

So, we focused on user-agent spoofing and IP disguising.

User-agent is easy to implemented if we have a list of user-agents such as: *"Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; AcooBrowser; .NET CLR 1.1.4322; .NET CLR 2.0.50727)".*

However, random IP pools is not that simple and explicit to obtain as the most proxies downloaded from the Web are usually out-of-date. As a result, we write another crawler to retrieve free proxies agents. After we have both user-agent pools and IP pools, the remaining thing is just construct our middlewares which handle the request and response using these technique.

Here is the flow chart about our Nasdaq crawler.



In this chart, we can see, we firstly select a user-agent and a IP randomly into our request. If we scraped successfully, we put the data into an item and yield it, if not, we delete our failed IP address from IP pool and try again. From this protocol, our crawler disguised itself each time like different clients in different addresses.

Crawler Logger

A singleton logger is used to keep watch on the program running and print important information while crawling, we can see, by using random IPs we failed the most time caused by either timeout or access problem(400 301 …). That's why we need a pool of IP to overcome it.

```
2017-05-07 11:55:47 [scrapy.downloadermiddlewares.retry] DEBUG: Retrying <GET http://www.nasdaq.com/symbol/VRSK/guru-analysis> (
failed 1 times): User timeout caused connection failure.
2017-05-07 11:55:47 [scrapy.proxies] INFO: **********************using agency: Mozilla/5.0 (Windows; U; MSIE 9.0; Windows NT 9.
0; en-US)

2017-05-07 11:55:47 [scrapy.proxies] INFO: proxy:http://45.55.128.212:8799
2017-05-07 11:55:47 [scrapy.proxies] INFO: url:http://www.nasdaq.com/symbol/VRSK/guru-analysis
2017-05-07 11:55:47 [scrapy.proxies] INFO: Removing failed proxy <http://45.55.128.212:8799>, 23 proxies left
2017-05-07 11:55:47 [scrapy.proxies] DEBUG: Proxy user pass not found
2017-05-07 11:55:48 [scrapy.downloadermiddlewares.retry] DEBUG: Retrying <GET http://www.nasdaq.com/symbol/STX/guru-analysis> (f
ailed 8 times): User timeout caused connection failure: Getting http://www.nasdaq.com/symbol/STX/guru-analysis took longer than
3.0 seconds..
2017-05-07 11:55:48 [scrapy.proxies] INFO: **********************using agency: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_3) A
ppleWebKit/535.20 (KHTML, like Gecko) Chrome/19.0.1036.7 Safari/535.20

2017-05-07 11:55:48 [scrapy.proxies] INFO: proxy:http://51.255.15.148:24631
2017-05-07 11:55:48 [scrapy.proxies] INFO: url:http://www.nasdaq.com/symbol/STX/guru-analysis
2017-05-07 11:55:48 [scrapy.proxies] INFO: Removing failed proxy <http://51.255.15.148:24631>, 22 proxies left
2017-05-07 11:55:48 [scrapy.proxies] DEBUG: Proxy user pass not found
2017-05-07 11:55:48 [scrapy.downloadermiddlewares.redirect] DEBUG: Redirecting (301) to <GET http://www.nasdaq.com/symbol/rost/g
uru-analysis> from <GET http://www.nasdaq.com/symbol/ROST/guru-analysis>
2017-05-07 11:55:48 [scrapy.proxies] INFO: **********************using agency: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.
0; Acoo Browser; SLCC1; .NET CLR 2.0.50727; Media Center PC 5.0; .NET CLR 3.0.04506)

2017-05-07 11:55:48 [scrapy.proxies] INFO: proxy:http://144.217.51.133:80
2017-05-07 11:55:48 [scrapy.proxies] INFO: url:http://www.nasdaq.com/symbol/rost/guru-analysis
2017-05-07 11:55:48 [scrapy.proxies] INFO: Removing failed proxy <http://144.217.51.133:80>, 21 proxies left
2017-05-07 11:55:48 [scrapy.proxies] DEBUG: Proxy user pass not found
2017-05-07 11:55:48 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://www.nasdaq.com/symbol/msft/guru-analysis> (referer: No
ne)
2017-05-07 11:55:48 [scrapy.core.engine] DEBUG: Crawled (502) <GET http://www.nasdaq.com/symbol/sbac/guru-analysis> (referer: No
ne) ['partial']
2017-05-07 11:55:48 [scrapy.core.scraper] DEBUG: Scraped from <200 http://www.nasdaq.com/symbol/msft/guru-analysis>
{'PE_growth_PL': 0,
 'contrarian_DD': 43,
 'growth_value_JO': 80,
 'growth_value_MZ': 62,
 'momentum_strategy_V': 71,
 'price_sale_KF': 10,
 'small_cap_growth_MF': 48,
 'symbol': 'msft',
 'value_BG': 57}
2017-05-07 11:55:48 [scrapy.spidermiddlewares.httperror] INFO: Ignoring response <502 http://www.nasdaq.com/symbol/sbac/guru-ana
2017-05-07 11:56:00 [scrapy.core.engine] INFO: Closing spider (finished)
2017-05-07 11:56:00 [scrapy.extensions.feedexport] INFO: Stored csv feed (60 items) in: guru4.csv
2017-05-07 11:56:00 [scrapy.statscollectors] INFO: Dumping Scrapy stats:
{'downloader/exception_count': 764,
 'downloader/exception_type_count/twisted.internet.error.ConnectionRefusedError': 4,
 'downloader/exception_type_count/twisted.internet.error.TimeoutError': 674,
 'downloader/exception_type_count/twisted.web._newclient.ResponseFailed': 2,
 'downloader/exception_type_count/twisted.web._newclient.ResponseNeverReceived': 84,
 'downloader/request_bytes': 380590,
 'downloader/request_count': 1210,
 'downloader/request_method_count/GET': 1210,
 'downloader/response_bytes': 7053117,
 'downloader/response_count': 446,
 'downloader/response_status_count/200': 61,
 'downloader/response_status_count/301': 81,
 'downloader/response_status_count/307': 3,
 'downloader/response_status_count/400': 120,
 'downloader/response_status_count/401': 20,
 'downloader/response_status_count/403': 46,
 'downloader/response_status_count/404': 56,
 'downloader/response_status_count/407': 7,
 'downloader/response_status_count/429': 6,
 'downloader/response_status_count/500': 1,
 'downloader/response_status_count/501': 6,
 'downloader/response_status_count/502': 29,
 'downloader/response_status_count/503': 8,
 'downloader/response_status_count/504': 2,
 'finish_reason': 'finished',
 'finish_time': datetime.datetime(2017, 5, 7, 15, 56, 0, 388271),
 'item_scraped_count': 60,
 'log_count/DEBUG': 2482,
 'log_count/INFO': 4584,
 'response_received_count': 103,
 'scheduler/dequeued': 1156,
 'scheduler/dequeued/memory': 1156,
 'scheduler/enqueued': 1156,
 'scheduler/enqueued/memory': 1156,
 'start_time': datetime.datetime(2017, 5, 7, 15, 51, 7, 483676)}
2017-05-07 11:56:00 [scrapy.core.engine] INFO: Spider closed (finished)
(py36)
```

Saved as Dataset

After scrapping, we saved our items in a *.csv file, here is a partial of stocks using the theories of Peter Lynch, Benjamin Graham and other stock-picking legends, based on Validea's interpretation.

	A	B	C	D	E	F	G	H	I
1	PE_growth_PL	contrarian_DL	growth_value_J	growth_value_M	momentum_strategy_V	price_sale_KF	small_cap_growth_MF	symbol	value_BG
2	0	21	25	15	57	0	21	adsk	14
3	0	50	40	62	71	40	48	adbe	29
4	0	50	40	69	71	10	61	adi	57
5	0	36	60	54	89	40	45	amzn	29
6	0	57	80	69	50	30	21	googl	57
7	0	29	25	8	50	20	1	bmrn	57
8	0	43	40	54	7	30	19	alxn	43
9	0	29	40	38	36	30	19	ca	29
10	74	64	60	77	89	20	68	amat	43
11	0	57	40	62	71	30	25	adp	57
12	72	43	60	69	71	20	21	amgn	71
13	0	50	80	62	36	10	52	aapl	29
14	0	43	100	31	29	10	28	csco	57
15	0	43	40	38	57	30	48	chtr	29
16	0	36	40	46	36	20	41	avgo	29
17	0	50	40	46	43	38	19	ctsh	57
18	0	43	60	62	71	10	61	atvi	29
19	0	76	50	54	0	50	12	bbby	86
20	0	43	40	69	57	30	19	cern	43
21	0	69	60	38	36	50	15	aal	43
22	0	50	80	54	71	38	25	cmcsa	43
23	0	57	40	69	71	20	55	ctxs	29
24	0	57	50	77	50	50	19	dltr	43

Conclusion for Data Crawling

Although the first glimpse of this algorithm is quite simple, but the implementation is not that easy. A good crawler engineer should always prepared himself/herself to encounter different anti-crawler strategies. Here are some points I have learned in this project.

First of all, make sure to follow the Terms of Use and policies related to web-scraping. Most importantly, do not bring in human website traffic and seemingly affect the performance of the sites.

Secondly, be aware of that many websites have different mechanics to detect a scraper from a normal user. Do not use your own IP address to take risk, because your IP may be blocked. The binning can be temporary or permanent. Temporary blocks can last minutes or hours. Permanent bans go against the open nature or the Internet but some Sites resort to this “scorch the internet” measure. So, change the proxy in request params and retry. If it doesn’t work, you have switch to a different IP.

Thirdly, do not follow the same crawling pattern. Only robots follows the same crawling pattern because unless specified, otherwise, program robot follow the logic which is usually very specific.

Reference

Design Patterns (n.d.). In *Source Making*. Retrieved April, 2017, from { HYPERLINK "https://sourcemaking.com/design_patterns" \h **}**

Joshi, M. S. (2004). *C++ Design Patterns and Derivatives Pricing (Mathematics, Finance and Risk)* (2nd ed.). N.p.: Cambridge University Press.

Massaron, L., & Boschetti, A. (2016). *Regression Analysis with Python*. N.p.: Packt Publishing - ebooks Account (February 29, 2016)

Python 3.6.1 documentation (2016). In *Python 3.6.1 documentation*. Retrieved March, 2017, from { HYPERLINK "https://docs.python.org/3/" \h **}**

ScrapeHero. (2014, July 31). In *How to prevent getting blacklisted while scraping*. Retrieved from { HYPERLINK "https://www.scrapehero.com/how-to-prevent-getting-blacklisted-while-scraping/" \h **}**

Scrapy 1.3 documentation (2017, March 10). In *Scrapy* . Retrieved from { HYPERLINK "https://doc.scrapy.org/en/latest/intro/tutorial.html" \h **}**

Shalloway, A., & Trott, J. (2001). *Design Patterns Explained: A New Perspective on Object-Oriented Design*. N.p.: Addison-Wesley Professional.

Yahoo Api cheat codes : https://greenido.wordpress.com/2009/12/22/work-like-a-pro-with-yahoo-finance-hidden-api/