Fundamental Analysis of Securities Trading (II) Data

Kuan-Lun WANG¹

National Taiwan University

Preliminary Draft: March 19, 2019

¹E-mail: polyphonicared@gmail.com

Kuan-Lun Wang is a doctoral student majoring in generalized pairs trading. The main goal of his research is to develop an algorithmic trading mechanism based on statistical arbitrage. His areas of expertise include automatic search procedures for model selection, multivariate co-integration approach, and structural change test.
Research Interests

Kuan-Lun Wang’s research interests comprise time series models, simulation modeling, and portfolio choice. The central themes of his application are the study of multivariate pairs trading in real time, search for assets with a long-run equilibrium, and building of riskless portfolios. Much of his current work involves conducting structural change analysis and co-integration test of the finite order vector autoregressive process and estimating the probability of mean reversion. Such methods are important in a variety of applications, including economic indicators and hedging. One such application is index funds being tied to indexes with very low costs and risks.

Source: TWSE (Data E-Shop)

The Data E-Shop of the TWSE provides a way for you to obtain the trading data more easily.


Warning: TWSE (Data E-Shop)

Some data may be only provided in the Traditional Chinese (中文) language page.

https://eshop.twse.com.tw/zh/category/all
MAGIC! Why they are so different?
Anyway, we must search for the needed data.
Daily Quotes (每日收盤行情)

An inexplicable word. We cannot search the explanation for this word by Google. Does anyone know it?

Daily Quotes: https://eshop.twse.com.tw/en/product/detail/ef7b7785e2cb4793baca3644c8a74d4e

Use of Information and Price (用途及價格)

- Internal Use (內部使用): NTD 1,000/Per month
- External Use (外部使用): NTD 1,500/Per month

Cost

If we want to backtest a strategy in $2 + 1 = 3$ years, then the cost is NTD $3 \times 12 \times 1,000 = 36,000$. That is inexpensive!?

So, buy data if we have a not bad strategy.

But, in the class, we had no money.

Are trading data public information? YES.
Public Information: Daily Quotes

In TWSE web, we can find three subpages provide free data in csv-format.

English:

中文:

日本語:

使用條款

六、智慧財產權的保護
本網站所使用之軟體或程式、網站上所有內容，包括但不限於著作、圖片、檔案、資訊、資料、網站架構、網站畫面的安排、網頁設計，均由臺灣證券交易所或其他權利人依法擁有其智慧財產權。任何人不得逕自使用、修改、重製、公開播送、改作、散布、販售、發行、公開發表、進行還原工程、解編或反向組譯。若您欲引用或轉載前述軟體、程式或網站內容，必須依法先取得臺灣證券交易所或其他權利人之事前書面同意，並請註明資料來源同時確保資料之完整性，不得任意增刪。尊重智慧財產權是您應盡的義務。

臺證寶字第1070016756號
WEB, HTML, and Web Browser

WEB, HTML, and Web Browser

- WEB is a computer programming system created by Donald E. Knuth.
- Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications.
- Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages.

https://en.wikipedia.org/wiki/WEB
https://en.wikipedia.org/wiki/HTML

We use web browser to display WEB page in HTML.

URL

A Uniform Resource Locator (URL), colloquially termed a web address, is a reference to a web resource that specifies its location on a computer network and a mechanism for retrieving it.

https://en.wikipedia.org/wiki/URL

Example: TWSE

We can use Chrome browser to display a TWSE home page, is identified by a URL http://www.twse.com.tw/en/.
Example: Download a Daily Quotes File (1/2)

TWSE provides Daily Quotes data in a web page. The page can update CSV-files download link when we send a query. If we click this link in Chrome browser, then browser saves CSV-file.


Example: Download a Daily Quotes File (2/2)

If we open the chrome download page (CTRL+J), then the page show the browser get a CSV-file from a URL http://www.twse.com.tw/en/exchangeReport/MI_INDEX?response=csv&date=20190226&type=ALL. Moreover, we can change the URL to get other CSV-file. Such as, we change 20190226 to 20190225. Okay, we only care about this URL.
Example: Download a Security Code List File

TWSE provides security code list in a web page. The page does not provide CSV-file. We need to convert the HTML table to a form file, e.g. CSV-file.

So, not all source are the same format.
But the processes are the same.

The computer execute process is

1. give a URL (we click the link);
2. open the HTML corresponding URL (browser do it);
3. save data (browser do it).

That is, we need to define three function:

- `getURL`
- `getData`
- `writeDownloadFile`
Syntax for URL

Every HTTP URL conforms to the syntax of a generic URI. The URI generic syntax consists of a hierarchical sequence of five components:

\[
\text{URI} = \text{scheme}:[//\text{authority}]\text{path}[,?\text{query}]][#\text{fragment}]
\]

So, we rename `getURL` as `getURI` and define the input and the output are `str`.

That is, the following functions

- `str: uri=getURI(int: yyyyymmdd)`
- `getdata`
- `writeDownloadFile`
Comma-Separated Values

In computing, a comma-separated values (CSV) file is a delimited text file that uses a comma to separate values. A CSV file stores tabular data (numbers and text) in plain text. Each line of the file is a data record. Each record consists of one or more fields, separated by commas.

https://en.wikipedia.org/wiki/Comma-separated_values

Listing 1: example.csv

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a, b, c</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>d, e, f</td>
<td></td>
</tr>
</tbody>
</table>

Table: example.csv

OK, we know the csv is str and define the intup and the the output are str.
That is, the following functions

1. str: uri=getURI(int: yyyymmdd)
2. str: data=getData(str: uri)
3. writeDownloadFile

The Web browser save daily quotes data with default file name MI_INDEX.csv.
If all file has same file name, then we only has the last file.
So, we need to define file name for each data.

1. str: uri=getURI(int: yyyymmdd)
2. str: data=getData(str: uri)
3. str: filename=getFilename(str: yyyymmdd)
4. NoneType: None=writeDownloadFile(str: data,str: filename)
Download Function API (Daily Quotes) (7/7)

Since user maybe want to search a single day or multiple days, the download API is defined as:

NoneType: download(int: yyyymdd[1+])

Read Function API (Daily Quotes) (1/5)

Clearly, we need to read a file and return user requirement data:

- getFilename
- readFile
- infoSearch
Since user maybe wants to search a single ticker or multiple tickers (single info or multiple info), the input of function is defined as

\[
\text{str: } \text{vlist}[1+][0+] = \text{infoSearch(str: } \text{tickers}[1+],
\text{str: } \text{infoNames}[1+], \text{list: } \text{rows})
\]
e.g.

- `infoSearch('0050', 'Opening Price')`  
  `'77.7'`
- `infoSearch('0050', ['Opening Price', 'Closing Price'])`  
  `[77.7, 77.65]`
- `infoSearch(['0050', '0051'], 'Opening Price')`  
  `[['77.70'], ['30.74']]`

Since user maybe want to search a single day or multiple days, the download API is defined as:

```python
str: info[1+][0+]=read(str: tickerIDs[1+], int: infoNames[1+], int: yyyymmdd[1+])
```
Listing 2: getURI

```python
def getURI(yyyymmdd):
    scheme="http"
    authority="www.twse.com.tw"
    path="/en/exchangeReport/MI_INDEX"
    query=[["response","csv"],
           ["date",str(yyyymmdd)], \
           ["type","ALL"]  \
           ]
    uri=scheme+"://"+authority+path+"?
    for n in range(len(query)):
        uri=uri+query[n][0]"="+query[n][1]"&"
    uri=uri[0:-1]
    return uri
```

We can use Web browser to check this URL.
Now, we define a function getData to get data from this URL.

Listing 3: getURI

```python
response=csv&date=20190227&type=ALL
```
Listing 4: `urllib.request.urlopen`

```python
Help on function urlopen in module urllib.request:

urlopen(url, data=None, timeout=<object object at 0
x0000017A0CB4E250>, *, cafile=None, capath=None,
cadefault=False, context=None)
Open the URL url, which can be either a string or
a Request object.

*data* must be an object specifying additional
data to be sent to the server, or None if no such
data is needed. See Request for details.

urllib.request module uses HTTP/1.1 and includes a
"Connection:close" header in its HTTP requests.
```

Listing 5: `getData`

```python
from urllib.request import urlopen as _urlOpen
def getData(url):
    data=_urlOpen(url)
    return data
```
The following return is a object.

Listing 6: return of getData

```python
1
2
<http.client.HTTPResponse object at 0x000001CBA7D726A0>
```

We need to read this memory.

Listing 7: getData

```python
1
2
3
4
5
def getData(url):
    data=_getResponse(url)
    return data
6
7
def _getResponse(url):
    with _urlOpen(url) as response:
        data=response.read()
    return data
```
The following return is ‘bytes’ class.

Listing 8: return of getData

```
'b"2019/02/27 Market Summary\n2019/02/27..."
```

We need to convert it to str.

Listing 9: getData

```
from urllib.request import urlopen as _urlOpen

def getData(uri):
    data=_getResponse(uri)
    return data

def _getResponse(uri):
    with _urlOpen(uri) as response:
        data=response.read()
        data=data.decode("utf-8",errors="ignore")
    return data
```
The following return has carriage return and newline.

**Listing 10:** return of `getData`

```python
'"2019/02/27 Market Summary"\r\n"2019/02/27 ..."
```

What’s this?

---

Carriage Return and Newline

- \r is carriage return, which is what moves where you are typing on the page back to the left (or right if that is your culture)
- \n is new line, which moves your paper up a line.

[https://softwareengineering.stackexchange.com/questions/29075/difference-between-n-and-r-n](https://softwareengineering.stackexchange.com/questions/29075/difference-between-n-and-r-n)
Listing 11: getData

```python
from urllib.request import urlopen as _urlOpen

def getData(uri):
    data=_getResponse(uri)
    data=fixData(data)
    return data

def fixData(data):
    data=data.replace("\r\n","\n")
    return data

def _getResponse(uri):
    with _urlOpen(uri) as response:
        data=response.read()
        data=data.decode("utf-8",errors="ignore")
    return data
```

Listing 12: getFilename

```python
def getFilename(yyyymmdd):
    filename='{filename}.{extension}'.format(filename=yyyymmdd,extension='csv')
    return filename
```
Now, we can crawl data.

Listing 13: writeDownloadFile

```python
def writeDownloadFile(data, filename):
    with open(filename, encoding='utf-8', mode="w") as f:
        f.write(data)
```

Listing 14: folder and file

```python
download.py
_download
|- getURI.py
|- getData.py
|- getFilename.py
|- writeDownloadFile.py
```
Listing 15: download

```python
from _download.getURI import getURI as getURI
from _download.getData import getData as _getData
from _download.getFilename import getFilename as _getFilename
from _download.writeDownloadFile import writeDownloadFile as _writeDownloadFile

def download(yyyymmdd):
    uri = _getURI(yyyymmdd)
    data = _getData(uri)
    filename = _getFilename(yyyymmdd)
    _writeDownloadFile(data, filename)
```

Sometimes, download is crash since the user is banned.
If the crash happens, then we should try it again.

Listing 16: time.sleep

```python
Help on built-in function sleep in module time:

sleep(...)  
sleep(seconds)  
Delay execution for a given number of seconds.
The argument may be a floating point number for subsecond precision.
```
Listing 17: download/getData

```python
def getData(uri, tryAgainTime):
    try:
        data = _getData(uri)
    except:
        _pause(tryAgainTime)
        data = getData(uri)
    return data
```

Listing 18: download

```python
from time import sleep as _pause

def download(yyyymmdd, tryAgainTime=60):
    uri = _getURI(yyyymmdd)
    data = _getData(uri)
    filename = _getFilename(yyyymmdd)
    _writeDownloadFile(data, filename)
```
If the file exists, then we not need to download again.

Listing 19: os.path.isfile

```python
Help on function isfile in module genericpath:

isfile(path)
    Test whether a path is a regular file
```

Listing 20: download

```python
from os.path import isfile as _isfile

def download(yyyymmdd, tryAgainTime=60):
    filename = _getFilename(yyyymmdd)
    if _isfile(filename):
        return None
    uri = _getURI(yyyymmdd)
    data = _getData(uri)
    _writeDownloadFile(data, filename)
```
If we fast crawl two files, then we could be banned.

Listing 21: download

```python
def download(yyyymmdd, pauseTime=5, tryAgainTime=60):
    filename = _getFilename(yyyymmdd)
    if not _isfile(filename):
        uri = _getURI(yyyymmdd)
        data = _getData(uri)
        _writeDownloadFile(data, filename)
    _pause(pauseTime)
```

We rename download as _singleDownload_.
We use a while-loop to download from yyyymmddStart to yyyymmddEnd.

Listing 22: download

```python
def download(yyyymmddStart, yyyymmddEnd, \n    pauseTime=5, tryAgainTime=60\n):
    yyyymmdd = yyyymmddStart
    onoff = True
    while onoff:
        _singleDownload(yyyymmdd, \n            pauseTime, tryAgainTime\n        )
        yyyymmdd = getNextDay(yyyymmdd)
        if yyyymmdd > yyyymmddEnd:
            onoff = False
```
Listing 23: Help of datetime.date

Help on class date in module datetime:

class date(builtins.object)
    | date(year, month, day) --> date object
    | Methods defined here:
    | __add__(self, value, /)
    | Return self+value.
    ...
    strftime(...)
    format -> strftime() style string.

-- More --

Listing 24: Help of datetime.timedelta

Help on class timedelta in module datetime:

class timedelta(builtins.object)
    | Difference between two datetime values.
    | timedelta(days=0, seconds=0, microseconds=0,
    | milliseconds=0, minutes=0, hours=0, weeks=0)
    | All arguments are optional and default to 0.
    | Arguments may be integers or floats, and may be
    | positive or negative.

-- More --
Listing 25: getNextDay

```python
import datetime as _datetime

def getNextDay(yyyymmdd):
    yyyy, mm, dd = splitYYYYMMDD(yyyymmdd)
    today = _datetime.date(yyyy, mm, dd)
    nextDay = today + _datetime.timedelta(days=1)
    nextDay = int(nextDay.strftime("%Y%m%d"))
    return nextDay
```

But the user maybe wants to crawl only one-day data.

Listing 26: download

```python
def _getYYYYMMDDStartAndEnd(yyyymmddInterval):
    if type(yyyymmddInterval) == list:
        yyyymmddStart = yyyymmddInterval[0]
        yyyymmddEnd = yyyymmddInterval[1]
    else:
        yyyymmddStart = yyyymmddInterval
        yyyymmddEnd = yyyymmddStart
    return yyyymmddStart, yyyymmddEnd
```
Listing 27: download

def download(yyyymmddInterval, 
   .pauseTime=5, tryAgainTime=60 \ 
):
    yyyymmddStart, yyyymmddEnd = _getYYYYMMDDStartAndEnd( 
        yyyymmddInterval)
    yyyymmdd = yyyymmddStart
    onoff = True
    while onoff:
        _singleDownload(yyyymmdd, 
            pauseTime, tryAgainTime \ 
        )
        yyyymmdd = _getNextDay(yyyymmdd)
        if yyyymmdd > yyyymmddEnd:
            onoff = False

Listing 28: Help of getFilename

def getFilename(yyyymmdd):
    filename = '{filename}.{extension}'.format(filename= 
        yyyymmdd, extension='csv')
    return filename
Listing 29: Help of csv.reader

Help on built-in function reader in module _csv:

\[
\text{reader(...)}
\]

\[
\text{csv_reader = reader(iterable [, dialect='excel']}
\]

\[
[\text{optional keyword args}]}
\]

\[
\text{for row in csv_reader:}
\]

\[
\text{process(row)}
\]

The "iterable" argument can be any object that returns a line of input for each iteration, such as a file object or a list. The optional "dialect" parameter is discussed below. The function also accepts optional keyword arguments which override -- More --

Listing 30: fileRead

\[
\text{from csv import reader as _csvRead}
\]

\[
\text{def fileRead(filename):
}\]

\[
\text{with open(filename) as csvFile:
}\]

\[
\text{table=[rowText for rowText in _csvRead(csvFile}
\]

\[
\text{delimiter=",",}]
\]

\[
\text{return table}
\]
Search row includ tickerID
  * tickerIDsRowSearch

Search value in corresponding row
  * rowInfosValueSearch

Since tickerID must in first column, we collect all elements in the column.

Listing 31: fileRead

```python
def _getFirstColumn(rows):
    if len(rows) == 0:
        return None
    column = []
    for row in rows:
        if len(row) == 0:
            c0 = None
        elif len(row) > 0:
            c0 = row[0]
        column.append(c0)
    return column
```
If the first word in `tickerID` is '0', then the `tickerID` must be convert to '="tickerID"'.

Listing 32: `tickerIDsRowSearch`

```python
def tickerIDsRowSearch(tickerIDs, table):
    if type(tickerIDs) == list:
        ...
    else:
        column = _getFirstColumn(table)
        if column is None:
            return None
        if tickerIDs[0] == '0':
            tickerIDs = '"{}"'.format(s=tickerIDs)
        index = column.index(tickerIDs)
        rows = table[index]
        return rows
```

Listing 33: `tickerIDsRowSearch`

```python
def tickerIDsRowSearch(tickerIDs, table):
    if type(tickerIDs) == list:
        rows = []
        for tickerID in tickerIDs:
            row0 = tickerIDsRowSearch(tickerID, table)
            rows.append(row0)
    else:
        ...
    return rows
```
Listing 34: tickerIDsRowSearch

```python
def infoSearch(tickerIDs, infoNames, table):
    rows = _tickerIDsRowSearch(tickerIDs, table)
    if type(tickerIDs) == list:
        vlist = []
        for n in range(len(tickerIDs)):
            vlist0 = _rowInfosValueSearch(infoNames, rows[n])
            if type(vlist0) != list:
                vlist0 = [vlist0]
            vlist.append(vlist0)
    else:
        vlist = _rowInfosValueSearch(infoNames, rows)
    return vlist
```

Listing 35: tickerIDsRowSearch

```python
def read(yyyymmdd, tickerIDs, infoNames):
    filename = _getFilename(filename)
    table = _fileRead(filename)
    info = _infoSearch(tickerIDs, infoNames, table)
    return info
```
We rename read as _singleRead.
We use a while-loop to read from yyyyymmddStart to yyyyymmddEnd.
In this while-loop, _singleRead is used multiple times.
However, we maybe have some miss since this function has multiple inputs.

Listing 36: functools.partial
```python
class partial(builtins.object):
    partial(func, *args, **keywords) - new function
    with partial application of the given arguments
    and keywords.

    Methods defined here:
    __call__(self, /, *args, **kwargs)
    Call self as a function.
    __delattr__(self, name, /)
    Implement delattr(self, name).
    __getattr__(self, name, /)
    Implement getattr.
    __getattribute__(self, name, /)
    Implement __getattribute__.
    __hash__(self, /)
    Return hash(self).
    __init__(self, /, *args, **kwargs)
    Initialize self.
    __new__(self, /, *args, **kwargs)
    Create a new instance of type self.
    __reduce__(self, /)
    Return state information for pickling.
    __reduce_ex__(self, /, _excludelst)
    Return state information for pickling.
    __str__(self, /)
    Return a string representa-
Listing 37: read

```python
def read(tickerIDs, infoNames, yyyymmddInterval):
    singleRead = _partial(_singleRead, \
        tickerIDs=tickerIDs, \n        infoNames=infoNames)
    ... 
```

Listing 38: read

```python
def read(tickerIDs, infoNames, yyyymmddInterval):
    ... 
    if type(yyyymmddInterval) != list:
        yyyymmdd = yyyymmddInterval
        vlist = singleRead(yyyymmdd)
    else:
        yyyymmdd = yyyymmddInterval[0]
        yyyymmddEnd = yyyymmddInterval[1]
        vlist = []
        while yyyymmdd <= yyyymmddEnd:
            vlist0 = singleRead(yyyymmdd)
            yyyymmdd = _getNextDay(yyyymmdd)
            vlist.append(vlist0)
        return vlist
```
We want the download file save in a nice folder (not anywhere). Moreover, the user can select the folder path.

Listing 39: Help of os.path.join

```python
Help on function join in module ntpath:
join(path, *paths)
    # Join two (or more) paths.
```

Listing 40: Help of os.path.isdir

```python
Help on built-in function _isdir in module nt:
_isdir(path, /)
    Return true if the pathname refers to an existing directory.
```
Listing 41: Help of os.path.isdir

```
Help on function makedirs in module os:
makedirs(name, mode=511, exist_ok=False)
    makedirs(name [, mode=0o777][, exist_ok=False])

Super-mkdir; create a leaf directory and all intermediate ones. Works like mkdir, except that any intermediate path segment (not just the rightmost) will be created if it does not exist. If the target directory already exists, raise an OSError if exist_ok is False. Otherwise no exception is raised. This is recursive.
```

Listing 42: getFolderPath

```
from os.path import join as _pathJoin
from os.path import isdir as _isdir
from os import makedirs as _mkdir

def getFolderPath(databasePath):
    folderpath=_pathJoin(databasePath,'TWSE')
    if not _isdir(folderpath):
        _mkdir(folderpath)
    folderpath=_pathJoin(folderpath,'Daily_Quotes')
    if not _isdir(folderpath):
        _mkdir(folderpath)
    return folderpath
```
Listing 43: download/_singleDownload

```python
def _singleDownload(yyyymmdd, databasePath, pauseTime, tryAgainTime):
    yyyymmdd = str(yyyymmdd)
    filename = _getFilename(yyyymmdd)
    folderpath = _getFolderPath(databasePath)
    filename = _pathJoin(folderpath, filename)
    ...
```

Listing 44: download

```python
def download(yyyymmddInterval, databasePath,\
             pauseTime, tryAgainTime,\
             ):\n    ...
```

Listing 45: read/_singleRead

```python
def _singleRead(yyyymmdd, tickerIDs, infoNames):
    filename = _getFilename(yyyymmdd)
    folderpath = _getFolderPath(databasePath)
    filename = _pathJoin(folderpath, filename)
    table = _fileRead(filename)
    info = _infoSearch(tickerIDs, infoNames, table)
    return info
```

Listing 46: read

```python
def read(tickerIDs, infoNames, yyyymmddInterval,\
         databasePath,\n         ):
    ...
```
Listing 47: dailyQuotes

class dailyQuotes():
    def __init__(self, databasePath='.':
        self.databasePath=databasePath
    def download(self, 
        yyyymmddInterval, 
        pauseTime=5, tryAgainTime=60
    ):
        _download(yyyymmddInterval, 
            self.databasePath, 
            pauseTime, 
            tryAgainTime 
        )
    ...

Listing 48: dailyQuotes

class dailyQuotes():
    def __init__(self, databasePath='.':
        self.databasePath=databasePath
    ...
    def read(self, 
        tickerIDs, infoNames, yyyymmddInterval
    ):
        vlist=_read(tickerIDs, 
            infoNames, 
            yyyymmddInterval,
            self.databasePath
        )
    
    return vlist
Listing 49: dailyQuotes

```java
public class dailyQuotes {
    |- public attribute databasePath
    |- public operation download
    |- public operation read
```

Assume we done security code crawler.

Listing 50: dailyQuotes

```java
public class dailyQuotes {
    |- public attribute databasePath
    |- public operation download
    |- public operation read
```

Listing 51: securityCode

```java
public class securityCode {
    |- public attribute databasePath
    |- public operation download
    |- public operation read
```

That shows that we can define a framework and contract out they.
Listing 52: `twseCrawlerStandard`

class `twseCrawlerStandard`:
    def __init__(self, databasePath="."):
        self.databasePath = databasePath
    def download(self):
        pass
    def read(self):
        pass

Listing 53: `dailyQuotes`

class `dailyQuotes`(`twseCrawlerStandard`):
    ...

Listing 54: `securityCode`

class `securityCode`(`twseCrawlerStandard`):
    ...

Listing 55: abstract `twseCrawlerStandard`

```python
import abc
class twseCrawlerStandard(metaclass=abc.ABCMeta):
    def __init__(self, databasePath="."):
        self.databasePath = databasePath
    @abc.abstractmethod
def download(self):
        pass
    @abc.abstractmethod
def read(self):
        pass
```
Moreover, we can prevent potential problems by format and divides all operations.

e.g.

- The URL is replaced as a new URL;
- The provided data format change;
Listing 59: dailyQuotes

```java
public class dailyQuotes extends twseCrawlerStandard
    {
    public attribute databasePath
    public operation download
        use operations
            getURI writeDownloadFile
            getData getFolderPath
            getFilename
    public operation read
        use operation
            fileRead getFilename
            infoSearch getFolderPath
    }
```

They use so many same operations.

Listing 60: securityCode

```java
public class securityCode extends twseCrawlerStandard
    {
    public attribute databasePath
    public operation download
        use operations
            getURI writeDownloadFile
            getData getFolderPath
            getFilename
    public operation read
        use operation
            fileRead getFilename
            infoSearch getFolderPath
    }
```
Listing 61: downloadInterface
1 interface class downloadInterface
2 |- public operation getURI
3 |- public operation getData
4 |- public operation writeDownloadFile

Listing 62: readInterface
1 interface class readInterface
2 |- public operation fileRead
3 |- public operation infoSearch

Listing 63: NameInterface
1 interface class NameInterface
2 |- public operation getFolderPath
3 |- public operation getFilename

Listing 64: dailyQuotesDownload
1 protected class dailyQuotesDownload <
   downloadInterface

Listing 65: dailyQuotesRead
1 protected class dailyQuotesRead < readInterface

Listing 66: dailyQuotesNamePath
1 protected class dailyQuotesNamePath < NameInterface
Listing 67: dailyQuotes

```java
1 public class dailyQuotes < twseCrawlerStandard
2 | - public attribute databasePath
3 | - public operation download
4 | | - use protected class dailyQuotesDownload
5 | | - use protected class dailyQuotesNamePath
6 | - public operation read
7 | - use protected class dailyQuotesRead
8 | - use protected class dailyQuotesNamePath
```

Listing 68: securityCodeDownload

```java
1 protected class securityCodeDownload < downloadInterface
```

Listing 69: securityCodeRead

```java
1 protected class securityCodeRead < readInterface
```

Listing 70: securityCodeNamePath

```java
1 protected class securityCodeNamePath < NameInterface
```
Listing 71: securityCode

```
1 public class securityCode < twseCrawlerStandard
2 | - public attribute databasePath
3 | - public operation download
4 | | - use protected class securityCodeDownload
5 | | - use protected class securityCodeNamePath
6 | - public operation read
7 | - use protected class securityCodeRead
8 | - use protected class securityCodeNamePath
```

All the class has the same attribute databasePath.

Listing 72: dailyQuotes

```
1 public class dailyQuotes < twseCrawlerStandard
2 | - public attribute databasePath
3 | - public operation download
4 | - public operation read
```

Listing 73: securityCode

```
1 public class securityCode < twseCrawlerStandard
2 | - public attribute databasePath
3 | - public operation download
4 | - public operation read
```
Listing 74: crawlerStandard

```python
import abc

class crawlerStandard(metaclass=abc.ABCMeta):
    def __init__(self, databasePath='.'):
        self.databasePath = databasePath
    @abc.abstractmethod
    def create(self, crawlerName):
        pass
```

Listing 75: twseCrawler

```python
class twseCrawler(crawlerStandard):
    def create(self, crawlerName):
        returnClass = {
            "dailyQuotes".lower():
                _dailyQuotes(self.databasePath),
            "dailyQuotesCHE".lower():
                _dailyQuotesCHE(self.databasePath),
            "securityCode".lower():
                _securityCode(self.databasePath)
        }
        returnClass = returnClass[crawlerName.lower()]
        return returnClass
```
Listing 76: crawlerStandard

```java
1 public abstract class crawlerStandard
2       public attribute databasePath
3       public abstract operation create
```

Listing 77: twseCrawler

```java
1 public class twseCrawler < crawlerStandard
2       public attribute databasePath
3       public operation create
```

Figure: UML of TWSE (click figure 🌱)
Now, we expand the content, securityCode, based on our design.

- **Realize interfaces**
  - securityCodeDownload
  - securityCodeRead
  - securityCodeNamePath

- **Realize operations**
  - download
  - read

### Listing 78: securityCodeDownloadOperations/getURI

```python
def getURI():
    scheme = "http"
    authority = "isin.twse.com.tw"
    path = "/isin/e_C_public.jsp"
    query = [["strMode","2"]]
    uri = scheme + "://" + authority + path + "?"
    for n in range(len(query)):
        uri = uri + query[n][0] + "=" + query[n][1] + "&"
    uri = uri[0:-1]
    return uri
```
Listing 79: securityCodeDownloadOperations/getData

```python
1 def getData(uri):
2     data = _getResponse(uri)
3     return data
```

Listing 80: HTML Code of securityCode

```html
<!-- saved from url=(0053)http://isin.twse.com.tw/isin
<e_C_public.jsp?strMode=2 -->
<html><head><meta http-equiv="Content-Type" content="
<text/html; charset=Big5"<link rel="stylesheet"
<href="/e_C_public_files/style1.css" type="text/
<css">
</head><body><h2><strong><font class="h1">List
<of ISIN Code for Listed Equities</font></h2>
</center>Date Stock Updated:2019/03/05 </center>
</font></h2><h2><font
</font>
</center><font>
<font color="red">The date of listing is
```
Select we need.

Listing 81: securityCodeDownloadOperations/getData

```python
1  def _fixData(data):
2      stocksBeginKeyword='<tr><td bgcolor=#FAFAD2 colspan=7><B>Stocks</B></td></tr>'
3      stocksBeginIndex=data.find(stocksBeginKeyword)+len(stocksBeginKeyword)
4      stocksEndKeyword='<tr><td bgcolor=#FAFAD2 colspan=7><B>Listing Warrants</B></td></tr>'
5      stocksEndIndex=data.find(stocksEndKeyword)
6      data=data[stocksBeginIndex:stocksEndIndex]
7      ...
```

Listing 82: securityCodeDownloadOperations/getData/stringPatternSearch

```python
1  def stringPatternSearch(string, begin, end):
2      stringPartition=[]
3      onoff=True
4      while onoff:
5          string=string[string.find(begin):]
6          endIndex=string.find(end)
7          if endIndex==-1:
8              onoff=False
9          else:
10             ...
11      stringPartition=[s[len(begin):-len(end)] for s in stringPartition]
12      return stringPartition
```
Listing 83: `securityCodeDownloadOperations/getData/stringPatternSearch`

```python
def stringPatternSearch(string, begin, end):
    ...
    else:
        endIndex = endIndex + len(end)
        if endIndex + 1 != len(string):
            stringPartition.append(string[0:endIndex])
            string = string[endIndex:]
        else:
            stringPartition.append(string[0:])
    string = ''
    stringPartition = [s[len(begin):-len(end)] for s in stringPartition]
    return stringPartition
```

Listing 84: `securityCodeDownloadOperations/getData/fixData`

```python
def fixData(data):
    ...
    data = data[stocksBeginIndex:stocksEndIndex]
    data = _stringPatternSearch(data, '<tr>', '</tr>')
    data = [_stringPatternSearch(r, '<td bgcolor=#FAFAD2>', '</td>') for r in data]
    ...
```
SecurityCode&SecurityName save as SecurityCode@SecurityName

Listing 85:
```
securityCodeDownloadOperations/getData/fixData/fitAt

1 def fitAt(singleData):
2     if len(singleData)==0:
3         return singleData
4     r=singleData[0].split('@')[0:2]
5     [r.append(rr) for rr in singleData[1:]]
6     return r
```

Listing 86:
```
securityCodeDownloadOperations/getData/fixData

1 def fixData(data):
2     ...
3     data=data[stocksBeginIndex:stocksEndIndex]
4     data=_stringPatternSearch(data,'<tr>','</tr>')
5     data=[_stringPatternSearch(r,'<td bgcolor=#FAFAD2>','</td>') for r in data]
6     data=[fitAt(r) for r in data]
7     ...
```
Listing 87: `securityCodeDownloadOperations/getData/fixData/deleteSpace`

```python
def deleteSpace(string):
    onoff=True
    while onoff:
        string0=string.replace(' ','
        onoff=(string!=string0)
        string=string0
        if len(string)!=0:
            if string[0]=='
                string=string[1:]
        if len(string)!=0:
            if string[-1]=='
                string=string[0:-1]
    return string
```

Listing 88: `securityCodeDownloadOperations/getData/fixData`

```python
def fixData(string):
    ...
    data=data[stocksBeginIndex:stocksEndIndex]
    data=_stringPatternSearch(data,'<tr>','</tr>')</tdbgcolor=#FAFAD2>→' for r in data]
    data=[[_fitAt(r) for r in data]
    data=[_deleteSpace(rr) for rr in r] for r in data]
    ...
```
Listing 89: securityCodeDownloadOperations/getData/fixData/appendHeader

```python
def appendHeader(data):
    r=[['Security Code', \
        'Security Name', \
        'ISIN Code', \
        'Date Listed', \
        'Market', \
        'Industrial Group',\n        'CFICode', \
        'Remarks'
    ]]
    [r.append(rr) for rr in data]
    return r
```

Listing 90: securityCodeDownloadOperations/getData/fixData/fixData

```python
def fixData(data):
    data=data[stocksBeginIndex:stocksEndIndex]
    data=_stringPatternSearch(data,'<tr>','</tr>')
    data=[_stringPatternSearch(r,'<td bgcolor=#FAFAD2>','</td>') for r in data]
    data=_fitAt(r) for r in data]
    data=[_deleteSpace(rr) for rr in r]
    data=_appendHeader(data)
    return data
```
def writeDownloadFile(data, filename):
    data = _list2csv(data)
    with open(filename, encoding='big5', mode='w') as f:
        f.write(data)

def list2csv(vlist):
    string = ''
    for v in vlist:
        for vv in v:
            string = string + '{vv}'.format(vv=vv) + ','
    string = string[0:-1] + '
    return string

def fileRead(filename):
    with open(filename) as csvFile:
        csvFileText = [rowText for rowText in _csvRead(csvFile, delimiter=',')]
    return csvFileText
### Listing 94: `securityCodeReadOperations/infoSearch`

```python
def infoSearch(infoNames, data):
    if type(infoNames) == list:
        vlist = [singleInfoSearch(infoName, data) for infoName in infoNames]
        r = [[] for n in range(len(vlist[0]))]
        for v in vlist:
            for n, vv in enumerate(v):
                r[n].append(vv)
    else:
        infolist = data[0]
        index = infolist.index(infoNames)
        data = data[1:]
        r = [d[index] for d in data]
    return r
```

### Listing 95: `securityCodeNamePathOperations/getFolderPath`

```python
def getFolderPath(databasePath):
    folderpath = _pathJoin(databasePath, 'TWSE')
    if not _isdir(folderpath):
        _mkdir(folderpath)
    folderpath = _pathJoin(folderpath, 'Listed_Equities_ISIN_Code_List')
    if not _isdir(folderpath):
        _mkdir(folderpath)
    return folderpath
```
Listing 96: securityCodeNamePathOperations/getFilename

```python
1 def getFilename():
2     filename='stocks.csv'
3     return filename
```

Listing 97: securityCode/download

```python
1 def download(databasePath, pauseTime, tryAgainTime):
2     filename=_securityCodeNamePathOperations.getFilename()
3     folderpath=_securityCodeNamePathOperations.getFolderPath(databasePath)
4     filename=_pathJoin(folderpath, filename)
5     uri=_securityCodeDownloadOperations.getURI()
6     data=_getData(uri, tryAgainTime)
7     _securityCodeDownloadOperations.writeDownloadFile(data, filename)
8     _pause(pauseTime)
```
Listing 98: `securityCode/download/getData`

```python
def getData(uri, tryAgainTime):
    try:
        data = _securityCodeDownloadOperations.getData(uri)
    except:
        _pause(tryAgainTime)
        data = _getData(uri, tryAgainTime)
    return data
```

Listing 99: `securityCode/read`

```python
def read(infoNames, databasePath, pauseTime, tryAgainTime):
    filename = _securityCodeNamePathOperations.getFilename()
    folderpath = _securityCodeNamePathOperations.getFolderPath(databasePath)
    filename = _pathJoin(folderpath, filename)
    if not _isfile(filename):
        _download(databasePath, pauseTime, tryAgainTime)
    data = _securityCodeReadOperations.fileRead(filename)
    vlist = _securityCodeReadOperations.infoSearch(infoNames, data)
    return vlist
```