Project name: A Simple Stock Data Analyzer

In this project, we would create the tool to conduct simple analysis on the real-time stock market data. With the symbol of New York Stock Exchange stock, one can get its current trading price by extracting the information from a web page. For example, the current stock price of Google (NYSE symbol = GOOG) can be extracted from page http://finance.yahoo.com/quote/GOOG. Here, a Java API developed by the third party would be utilized to obtain the stock price. The API is called `YahooFinance` and one can use the following statements to easily return the current price of GOOG: (More information about the API can be found here https://financequotes-api.com/ and the library file is provided in the starter code).

```
Stock stock = YahooFinance.get("GOOG");
BigDecimal price = stock.getQuote().getPrice();
```

The tool works like an in-memory database which caches the real-time market data in the memory to allow fast information retrieval from a large amount of data. Each record stored in the database includes the stock symbol, company name and current price. Stock symbol is one record's key, and company name and current price are one record's value, which forms an key-value pair in one database record.

You would create an myStock class to implement the database and provide following functionalities such that others can use it as a tool for simple stock data analyzing. First, your database must be able to quickly retrieve the current price and company name from the database using the primary key stock symbol.

Second, frequent record update and insertion are expected, which requires that the data record should be added and updated quickly in the database. Third, the database should quickly retrieve the stock record with top k prices, where k's value is from 0 to the number of total records in the database.

The starter code is provided in `myStock.java` file. To achieve the three fast information retrievals mentioned before, your task is to implement the following operations:

```
myStock() // initialize the in-memory database with proper data structures
stockInfo get(String symbol) // get the record using primary key symbol,
should be O(1)
void insertOrUpdate(String symbol, stockInfo stock) // insert or update one
record to the data structures, and should be at least O(log(n))
List<Map.Entry<String, stockInfo>> top(int k) // return a list of records
with top k prices, and should be O(k)
```

To initialize your database, the file `US-Tech-Symbols.txt` is provided. Each line in the file contains one record of stock symbol and its company name separated by the colon. The real-time price for each stock could be obtained on-the-fly using the `YahooFinance` API. According to the test code in the starter code `myStock.java`, the following output is expected (actual price of the stock may vary).

```
==========Top 10 stocks==========
[1]CABO Cable One, Inc. 1328.01
[2]GOOG Alphabet Inc. 1291.37
[3]GOOGL Alphabet Inc. 1289.61
[4]EQIX Equinix, Inc. (REIT) 543.2
[5]TDY Teledyne Technologies Incorporated 336.16
[6]SHOP Shopify Inc. 307.91
[7]NTES NetEase, Inc. 307.72
[8]AVGO Broadcom Inc. 305.0
[9]FICO Fair Isaac Corporation 300.82
[10]LRCX Lam Research Corporation 278.1
==========Stock info retrieval==========
VMW VMware, Inc. 163.55
CHL China Mobile Limited 40.78
```

Implement the tools following the comments and hints about how they are supposed to work in the `myStock.java`. In the starter code, beside the input file `US-Tech-Symbols.txt` and `myStock.java`, all necessary library files are provided and make sure you include them in your project's build path to allow the successful code execution. To do that, right click your project and choose "Build Path -> Add External Archives", and select all library files (i.e., the jar files). Finally, please submit your completed myStock.java and a screenshot of your output to Blackboard. You may not be able to receive full credits if you fail to submit the screenshot.