University of Waterloo
CS 234 - Data Types and Structures
Spring 2014
Assignment 2
Due: Wednesday June 4th at 4PM

Assignment Guidelines

- Clarifications about the assignment will be posted on Piazza in the thread "Assignment 2 [Official]."
- Instructions on how to submit your assignments to MarkUs will also be posted in Piazza. Your programs must work in the linux.student.cs.uwaterloo.ca environment using Python 2.7.3.
- You may lose up to 20% on a question because it is difficult to read or difficult to understand.

Question 1: Testing Assumptions: Stock.py [10 marks]

a) Download the script Stock.py from Supplementary Material section of Learn. Modify the constructor function for the class Stock so that it checks the arguments and raises ValueError and TypeError exceptions where appropriate. Also include a meaningful error message with the exception that narrows down the source of the exception. Submit your answer via MarkUs [7 marks].

b) In the a file called StockTest.py create a function *testStock*() that tests the constructor of the Stock class to ensure that it raises the appropriate exceptions for the appropriate errors. Submit your answer via MarkUs [3 marks].

Question 2: Using the StockExchange ADT [20 marks]

a) In a file called StockPortfolio.py create a class called StockPortfolio that tracks the profits or losses of buying or selling stock using the StackExchangeADT interface. It consists of the following functions.

CurrentPrices(aStockListingsFile)	- Given the name of a file that contains TSX listings, read in the file and update the prices of each stock. If the file does not exists, raise an IOError exception.
BuyStock(aSymbol, numShares)	- Buy numShares of the stock with symbol aSymbol. Increase the value of the expenditures and the value of the holdings by the total amount of money spent buying the stock. If the stock does not exist, raise a ValueError exception.
SellStock(aSymbol, numShares)	- Sell numShares of the stock with symbol aSymbol. Decrease the value of the holdings and the value of the expenditures by the total amount of money received selling the stock. If the stock is no longer listed in the exchange, assume the stock has value $0.
Expenditures()	- Return the total amount of money you have spent buying stock minus the total amount of money you have received selling stock. It may be a negative number.
ValueOfHoldings()	- Return the total value of the stock you own at its current price.
Profit():	- Return the total value of stock you own minus the expenditures you have made. Return losses as a negative number.

For example you may call CurrentPrices() to set the prices one day, buy some stock. On another date call CurrentPrices again to get the prices for that day and then sell some stock and calculate the profit you realized.

Document all your functions using Preconditions and Postconditions and state any additional assumptions you have to make. Require your constructor for this class to take aStockListingsFile as a parameter so that you can start buying stock right away. You may assume that you always have enough money to buy stock.

Raise ValueError and TypeError exceptions when the user of these functions violates a precondition.

Submit your answer via MarkUs [15 marks].

b) In a file StockPortfolioTest.py create a script that tests your class StockPortfolio. Try to organize your tests in a meaningful way. You may use the StockExchangeList.py module to implement the StockExchange ADT however we will be testing with another implementation of the StockExchange ADT so do not use anything in StockExchangeList.py other than the functions mentioned in StockExchangeADT.py.

   Submit your answer via MarkUs [5 marks].

Question 3: Course Project [10 marks]

For this part of the course project you will take the three methods you created in Assignment 1 (*messageInHTML*(), *runWebFilter*() from webServer.py and *sendMessage*() from webClient.py) and put them all in a single file called WebFilter.py.

Two of the functions, *messageInHTML*() and *sendMessage*(), will largely remain the same but the third function, *runWebFilter*(), will be modified for this assignment.

The new version of *runWebFilter*() should use the *socket, bind, listen*, and *accept* functions just like in webServer.py to receive a message from the browser. Next it will use the *sendMessage* method from webClient.py to forward the message to the website and print the website's response out to the screen. Then like Assignment 1, *runWebFilter*() will echo back to the browser what the browser sent to it using the *messageInHTML*() method to format the message. There will be a slight modification to this process if an error occurs, which will be explained below.

For testing purposes you may want to print out status messages to the screen, but for the final version of WebFilter.py that you submit for marking, only print out the response from the website to the screen.

In order for your script to work you are going to have to be able to get the host name (i.e. name of website) from the message the browser sends to your script and use that host name as the first parameter in your call to *sendMessage(aWebsiteName, aMessage)*.

A HTTP message from your browser requesting the webpage `http://www.lib.uwaterloo.ca/hours/` may look something like the following…

```
GET http://www.lib.uwaterloo.ca/hours/ HTTP/1.1
Host: www.lib.uwaterloo.ca
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:29.0) Gecko/20100101 Firefox/29.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

The first word in the first line, GET, is the name of the function and the first word in each other line (Host, User-Agent, Accept, Accept-Language, etc) are the parameters for the GET function. The first parameter, Host, is required and all the rest are optional. For your WebFilter to work, you must be able to isolate the line that starts with "Host:" and get the rest of the line, which is the host name, and use that string as the first parameter in your call to *sendMessage*(). The string methods *splitlines*() and *split*() may prove useful for this task.

In order to complete this task create a function called *getHost(anHTTPcmd)* which takes as a parameter the HTTP command you received from your browser and returns the name of the host as a string (if it exists in the HTTP command) or raises a ValueError exception if the message does not contain the parameter "Host:" followed by the host name. Do not assume that host appears as the second line of the message. This is likely but not guaranteed.

If the *getHost*() command is successful, *runWebFilter*() should forward the message to the website and print the response on the screen. If *getHost*() raises an exception, *runWebFilter*() will catch the except and sends an error message back to the browser, using the *messageHTML*(*aMessageTitle, aMessage*) function, where *aMessageTitle* is "Error: Host not found" and *aMessage* is the message that the browser sent to the python script.

In summary your new script will contain four functions. Three of them are from Assignment 1 and one is a new function.

> *messageInHTML*(*aMessageTitle, aMessage*) - no modification from Assignment 1
>
> *sendMessage(aWebsiteName, aMessage)* - the only modification from Assignment 1 is that it only prints out what the website sends back to it, no status messages
>
> *runWebFilter(aPort)* - modified to check the message it receives from the browser to see if it has a "Host:" parameter. If it does, forward the message to the website with the *sendMessage*() function and echo the browser message back to the browser. If it does not, send an error message back to the web browser.
>
> *getHost(anHTTPmessage)* - a new function which returns the name of the host from the browser HTTP message or raises a ValueError exception if the host name cannot be found.

If you would prefer, you may use my version of webClient.py and webServer.py as your basis for answering this question rather than use the version you submitted for Assignment 1. They are called webServerA1Q1b.py and webClientA1Q2b.py in the Assignments section of Learn.

Submit your answer via MarkUs [10 marks].