Hello Breanna,

The first thing that should happen do before you start writing any code is to make sure you understand what the problem (requirements) is you are trying to solve. Then you design your solution. A UML is very helpful in the design process, and it is a good idea to make it part of your best practices. A UML helps you identify what classes you will need and what their relationships are with each other. All this needs to happen **before** one line of code is written. If a programmer starts writing code before the design is solidified and written down, then the product suffers. A design with a UML is first before you worry about parsing.

As far as the design goes, it needs to include an Observer pattern. Once you get this laid out, it will be clear that the LocalStocks class is the Observable (not the Observer). Another name for this is Subject. It will extend the Subject interface. It is this class that has the data the other classes (Observers) want. It is this class that holds the list of registered observers. This is the class that will receive data from an outside source (In our case the driver program) or it could read and parse the data itself. This is a personal design decision, and ether way is correct. Once the data has been updated, then LocalStocks will notify its observers of a change and get them the data.

The observers (in my solution, they are named averageobserver, changeobserver, selectionsobserver) will receive the data. They extend the Observer interface that has the update method. LocalStocks calls each of the observes update method to give them (push) the data. Each observer will parse what data then need. In this case, the data comes in the form of an array. In my solution the observers also extend a display interface. Go through the Observer Pattern in the recorded lecture, there is a very similar code example, which is also available as a download from lesson 2.

As far as parsing goes, this can be done using the string class methods of the particular language you are using. If you are more comfortable with C++ or C##, use can it. There are several ways to part the .dat. What I did was read a line in at a time into an array. Skip the first line. The formatting (fields) of the ticker.dat is in the assignment instructions. The Company and Ticker Symbol are the fields that are variable length. This is where most of the parsing will need to take place. What I did (and there are other ways to do this) is read the line from left to right until I hit a number. That will be the first char of the current price. All fields right of it are delimited by a space. From the first char, I read backwards (right to left) until I hit a space, which would be the starting char of the ticker symbol. Now I have the symbol. Anything left of this char is the Company name. You can then use an array to hold your parsed data. This would be what you send to the observers.

I hope this is helpful.