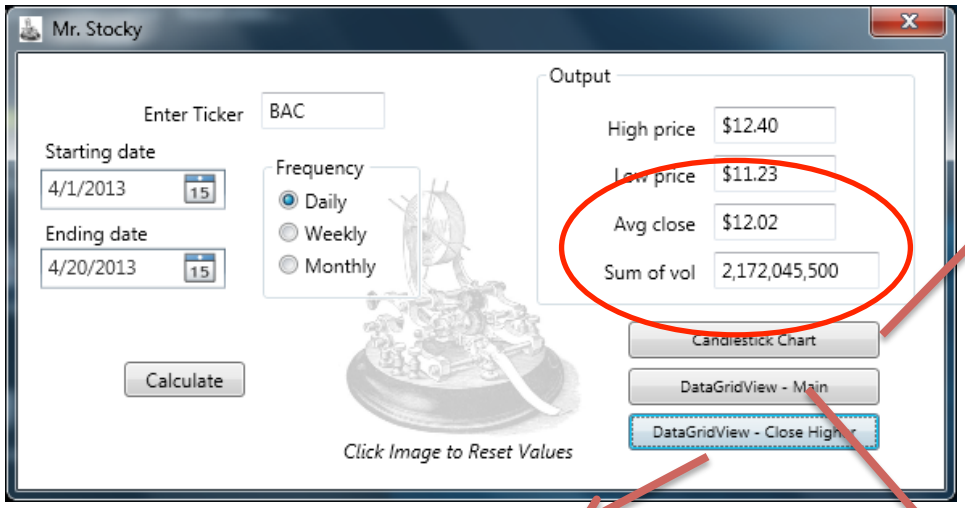
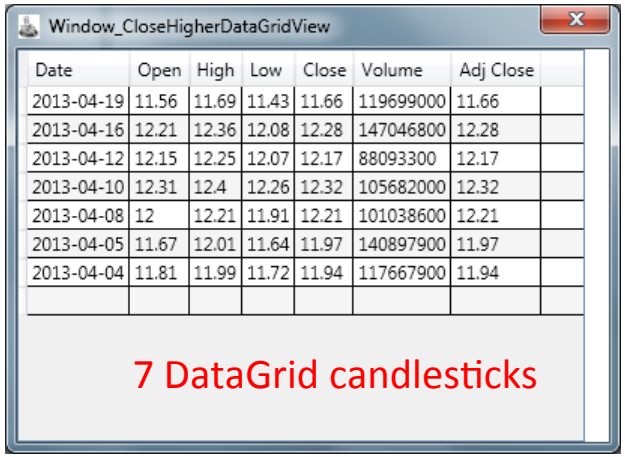
Overview:

- LINQ queries and lambda expressions
- New DataGrid view for green candlesticks (closing price is greater than opening price)
- New output for Avg. Close and Sum of Vol with proper string formatting

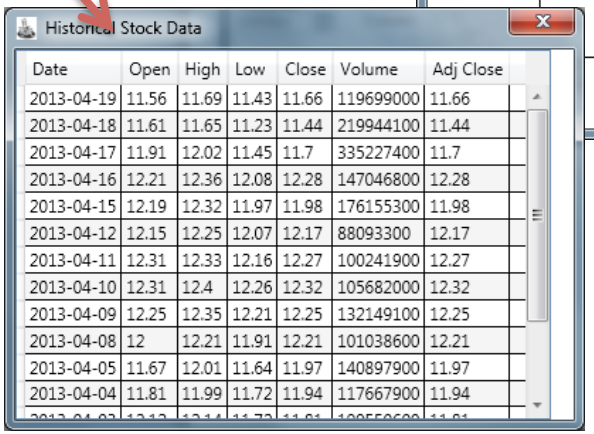


7 green candlesticks



Date	Open	High	Low	Close	Volume	Adj Close
2013-04-19	11.56	11.69	11.43	11.66	119699000	11.66
2013-04-16	12.21	12.36	12.08	12.28	147046800	12.28
2013-04-12	12.15	12.25	12.07	12.17	88093300	12.17
2013-04-10	12.31	12.4	12.26	12.32	105682000	12.32
2013-04-08	12	12.21	11.91	12.21	101038600	12.21
2013-04-05	11.67	12.01	11.64	11.97	140897900	11.97
2013-04-04	11.81	11.99	11.72	11.94	117667900	11.94

7 DataGrid candlesticks



Date	Open	High	Low	Close	Volume	Adj Close
2013-04-19	11.56	11.69	11.43	11.66	119699000	11.66
2013-04-18	11.61	11.65	11.23	11.44	219944100	11.44
2013-04-17	11.91	12.02	11.45	11.7	335227400	11.7
2013-04-16	12.21	12.36	12.08	12.28	147046800	12.28
2013-04-15	12.19	12.32	11.97	11.98	176155300	11.98
2013-04-12	12.15	12.25	12.07	12.17	88093300	12.17
2013-04-11	12.31	12.33	12.16	12.27	100241900	12.27
2013-04-10	12.31	12.4	12.26	12.32	105682000	12.32
2013-04-09	12.25	12.35	12.21	12.25	132149100	12.25
2013-04-08	12	12.21	11.91	12.21	101038600	12.21
2013-04-05	11.67	12.01	11.64	11.97	140897900	11.97
2013-04-04	11.81	11.99	11.72	11.94	117667900	11.94

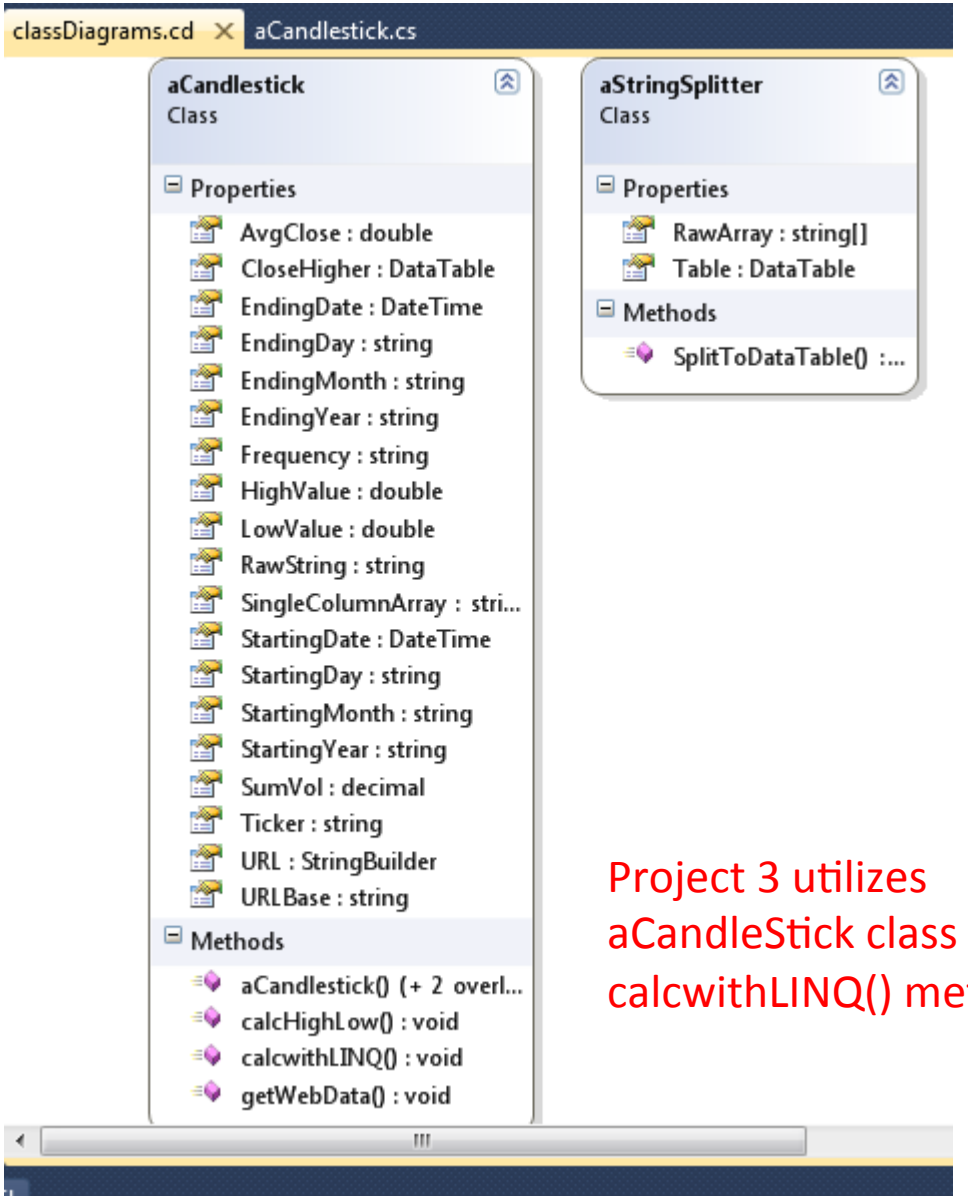
LINQ expressions easy to find in calcWithLINQ() method
Line 228 – 268 “aCandlestick.cs”

aCandlestick.cs

StockAnalyzer.aCandlestick | calcwithLINQ(DataTable table)

```
227
228         /// <summary>
229         /// This method finds stock values with LINQ and lamda expressions
230         /// Min, Max, Average closing price, Sum of volumes
231         /// </summary>
232         /// <param name="table">DataTable</param>
233         public void calcwithLINQ(DataTable table)
234         {
235             try
236             {
237                 // Find min value
238                 List<double> lowLevels = table.AsEnumerable().Select(low => (low.Field<double>("Low"))).ToList();
239                 LowValue = lowLevels.Min();
240
241                 // Find max value
242                 List<double> highLevels = table.AsEnumerable().Select(high => (high.Field<double>("High"))).ToList();
243                 HighValue = highLevels.Max();
244
245                 // Find avg close
246                 List<double> closingLevels = table.AsEnumerable().Select(close => (close.Field<double>("Close"))).ToList(
247                 AvgClose = closingLevels.Average();
248
249                 // Find sum of volumes
250                 List<decimal> sumVol = table.AsEnumerable().Select(sum => (sum.Field<decimal>("Volume"))).ToList();
251                 SumVol = sumVol.Sum();
252
253                 // Find candlesticks where closing is greater than opening
254                 // i.e., green candlesticks
255                 var closeHigherValues = from row in table.AsEnumerable()
256                                         where row.Field<double>("Open") < row.Field<double>("Close")
257                                         select row;
```

Class Diagram



Project 3 utilizes aCandleStick class and calcwithLINQ() method

Program flow

1. Get user input
   - Ticker // On button event
   - StartingDate // On pick event
   - EndingDate// On pick event
   - Frequency // On pick event
2. getWebData
   - Build a URL
   - Use WebClient to get string of data
3. Split the string to DataTable
   - Split string returned by WebClient and Yahoo API. Split on ‘,’ and ‘\n’
   - Construct DataTable from split string array
4. Perform analysis
   - Get high, low, avg, sum of vol, and close higher values with LINQ
5. Display results
   - High, Low, avg, sum on main UI
   - 2 DataGrid Views
   - Candlestick chart

URL String builder

http://ichart.finance.yahoo.com/table.csv?s=RTN&a=00&b=1&c=2013&d=01&e=7&f=2013&g=d&ignore=.csv

String in two pieces:

1. Base "http://ichart.finance.yahoo.com/table.csv?"
2. "s=RTN&a=00&b=1&c=2013&d=01&e=7&f=2013&g=d&ignore=.csv"

Use StringBuilder class to build string based on user input where
s=<ticker>
a=<starting month> //00 – 11 STARTS at ZERO!
b=<starting day> //1-31 depending on month
c=<starting year> //i.e. 2013
d=<ending month>
e=<ending day>
f=<ending year>
g=<bucket> // daily=d, weekly=w, monthly=m

// StringBuilder AppendFormat looks like this…
"s={0}&a={1}&b={2}&c={3}&d={4}&e={5}&f={6}&g={7}&ignore=.csv"