FeedParser Class before refactoring

```
package com.StockTake;


import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.net.URL;

import java.nio.charset.Charset;

import java.util.Calendar;

import java.util.LinkedList;

import java.util.StringTokenizer;

import android.util.Log;


public class FeedParser {


	public void getFeed(Finance toPopulate, String currentStock) {

		BufferedReader reader;

		String csvData[] = null;


		reader = null;

		csvData = null;

		try {
```

```
                reader = getCsvRealtime(currentStock);
                csvData = parseCsvRealtime(reader);
        } catch (IOException e) {
        }
        toPopulate.setLast((Float.parseFloat(csvData[1]))/100f);
        toPopulate.setName(currentStock);
        toPopulate.setInstantVolume(Integer.parseInt(csvData[2]));

        try {
                reader = getCsvHistoric(currentStock, "Weekly");
                csvData = parseHistoricVolume(reader);
        } catch (Exception e) {
        }
        toPopulate.setClose((Float.parseFloat(csvData[0]) / 100f));
        toPopulate.setVolume(Integer.parseInt(csvData[1]));

}

public LinkedList<Float> getHistoricFeed(String currentStock, String time) {
        BufferedReader reader;
        LinkedList<Float> csvHistoricList = new LinkedList<Float>();

        try {
                reader = getCsvHistoric(currentStock, time);
                csvHistoricList = parseCsvHistoric(reader);
```

```
        } catch (IOException e) {
        }
        return csvHistoricList;
}

public BufferedReader getCsvHistoric(String stockSymbol, String timeFrame) {
        URL feedUrl = null;
        InputStream is = null;

        Calendar cal = Calendar.getInstance();
        int day = 0, month = 0, year = 0;

        if (timeFrame.equals("Weekly")) {
                day = cal.get(Calendar.DAY_OF_MONTH) - 8;
                month = cal.get(Calendar.MONTH);
                year = cal.get(Calendar.YEAR);
        } else if (timeFrame.equals("Monthly")) {
                day = cal.get(Calendar.DAY_OF_MONTH);
                month = cal.get(Calendar.MONTH) - 1;
                year = cal.get(Calendar.YEAR);
        } else if (timeFrame.equals("Yearly")) {
                day = cal.get(Calendar.DAY_OF_MONTH);
                month = cal.get(Calendar.MONTH);
                year = cal.get(Calendar.YEAR) - 1;
        }
```

```
                try {
                        feedUrl = new URL("http://ichart.yahoo.com/table.csv?s="
                                        + stockSymbol + ".L&a=" + month + "&b=" + day + "&c="
                                        + year);
                } catch (IOException e) {
                }
                try {
                        is = feedUrl.openStream();
                } catch (IOException e) {
                        Log.e("error", e.toString());
                }

                return new BufferedReader(new InputStreamReader(is,
                                Charset.forName("UTF-8")));
        }

        public BufferedReader getCsvRealtime(String stockSymbol) throws IOException {
                // Generate URL
                URL feedUrl = new URL("http://finance.yahoo.com/d/quotes.csv?s="+
stockSymbol + ".L&f=nb2b3va");

                InputStream is = feedUrl.openStream();

                return new BufferedReader(new InputStreamReader(is,
                                Charset.forName("UTF-8")));
```

```
}


private LinkedList<Float> parseCsvHistoric(BufferedReader csvToParse)
                throws IOException {
        String strLine = "";
        StringTokenizer st = null;
        int lineNumber = 0, tokenNumber = 0;
        LinkedList<Float> historicList = new LinkedList<Float>();
        while (((strLine = csvToParse.readLine()) != null)) {
                lineNumber++;
                if (lineNumber != 1) {
                        st = new StringTokenizer(strLine, ",");
                        String token;
                        while (st.hasMoreTokens()) {
                                tokenNumber++;
                                token = st.nextToken();
                                if (tokenNumber == 5) {
                                        historicList.addFirst(Float.parseFloat(token));
                                }
                        }
                        tokenNumber = 0;
                }
        }
        return historicList;
}
```

```
private String[] parseHistoricVolume(BufferedReader csvToParse)
                throws IOException {
        String strLine = "";
        StringTokenizer st = null;
        int lineNumber = 0, tokenNumber = 0;
        String[] csvData = new String[2];
        while (((strLine = csvToParse.readLine()) != null)) {
                lineNumber++;
                if (lineNumber == 2) {
                        st = new StringTokenizer(strLine, ",");
                        String token;
                        while (st.hasMoreTokens()) {
                                tokenNumber++;
                                token = st.nextToken();
                                if (tokenNumber == 5) {
                                        csvData[0] = token;
                                }
                                if (tokenNumber == 6) {
                                        csvData[1] = token;
                                }
                        }
                        tokenNumber = 0;
                }
        }
```

```
        return csvData;
}

private String[] parseCsvRealtime(BufferedReader csvToParse) {
        String strLine = "";
        StringTokenizer st = null;
        int tokenNumber = 0;
        String csvdata[] = new String[4];

        try {
                strLine = csvToParse.readLine();
        } catch (IOException e) {}
        strLine = strLine.replace("\"", "");
        st = new StringTokenizer(strLine, ",");
        String token;
        float ask = 0f;
        float bid = 0f;

        while (st.hasMoreTokens()) {
                token = st.nextToken();
                if (tokenNumber == 0) {
                        csvdata[0] = token; // name in first field
                }
                if (tokenNumber == 1) {
                        ask = Float.parseFloat(token);
```

```
                }
                if (tokenNumber == 2) {
                        bid = Float.parseFloat(token);
                        csvdata[1] = Float.toString((ask + bid) / 2); // price in second
field

                }
                if (tokenNumber == 3) {
                        csvdata[2] = token; // volume in third field
                }
                tokenNumber++;
        }
        return csvdata;
    }
}
```