**CSC 3020 – Java Programming**

**Homework 4 – [Sayem Chowdhury]**

**25 points – Due March 21, 10am**

**Late deadline is March 23, 11:59pm, but 20% off**

**a)** Save this document with your name and the homework number somewhere in the file name.

**b)** Type/paste your answers into the document.

**c)** Submit this document and your .java file(s) to the Blackboard item where you downloaded this document. Do not submit a zip file but individually attach your files.

You’ve been hired by *Microsoft Monarchs* to write a Java console application that analyzes their stock data over the past eleven years. Use text file **MicrosoftStockData.txt** as input to the application. The first line of the file contains the column headers. The rest of the lines contain daily Microsoft stock data. Values are separated by commas into the following columns:

|  |  |  |
| --- | --- | --- |
| Column | Data type | Purpose |
| Date | String | Trading date |
| Close | double | Ending share value on that day. |
| Volume | int | Number of shares traded during that day. |
| Open | double | Starting share value on that day. |
| High | double | Highest share value on that day. |
| Low | double | Lowest share value on that day. |

Here are the first five lines of the file:

date,close,volume,open,high,low

3/5/2018,93.64,23787950,92.34,94.27,92.26

3/2/2018,93.05,32815660,91.58,93.15,90.86

3/1/2018,92.85,36979700,93.99,94.57,91.84

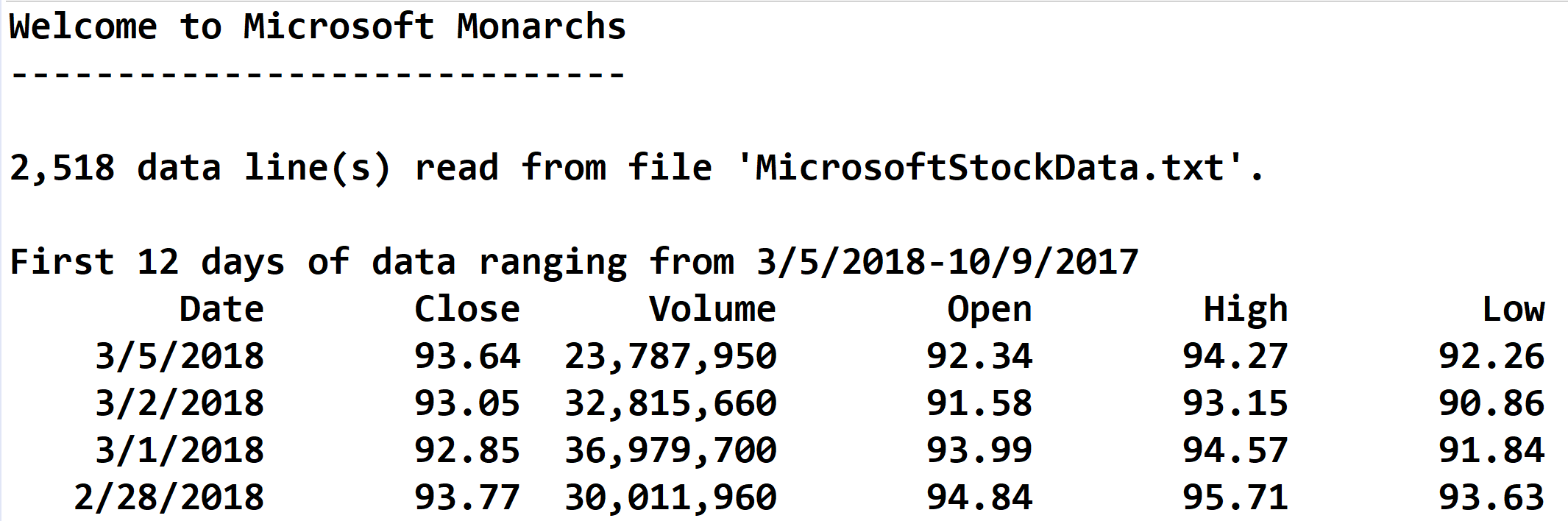
2/28/2018,93.77,30011960,94.84,95.705,93.63

The application reads the data into arrays, prints the first several rows of the data, and then analyzes and charts the data. Create the following functions:

**●** **main** – this function defines one array for each column of data in the input file such that there are six parallel arrays. It then calls these functions in sequence: **readTextFile**, **printData**, **analyzeData**, and **chartData**.

**●** **readTextFile** – this function reads each line of the file, parses it into six columns, and stores each token in its respective array. It skips past the first line which contains the column headers.

**●** **printData** – this function prints two header lines. The first indicates the range of dates in the data. The second shows the column headers. It then prints the first twelve lines of the data. Here is sample output:



**●** **analyzeData** – this function determines the following:

(1) The date and value of the highest daily stock closing value.

(2) The date and value of the lowest daily stock closing value.

(3)The date and highest difference between the daily high and low stock values.

(4) The closing value of the stock on the latest day of each year. For 2018, this is March 5. For each of the other ten years, this is December 31. Store the years and closing values in two parallel arrays. These arrays may then be used to print and chart this data.

It then prints the yearly closing values (4) formatted in two columns:

✓ The year.

✓ The closing value.

It then prints the highest (1), lowest (2), and differences (3) formatted in three columns:

✓ A label.

✓ A date.

✓ A closing value or difference.

Format all real numbers to two decimal places. For the following functions, use the JFreeChart library. This was discussed in the **Session 12 Notes**. See the **Chart drawing** sample application on Blackboard.

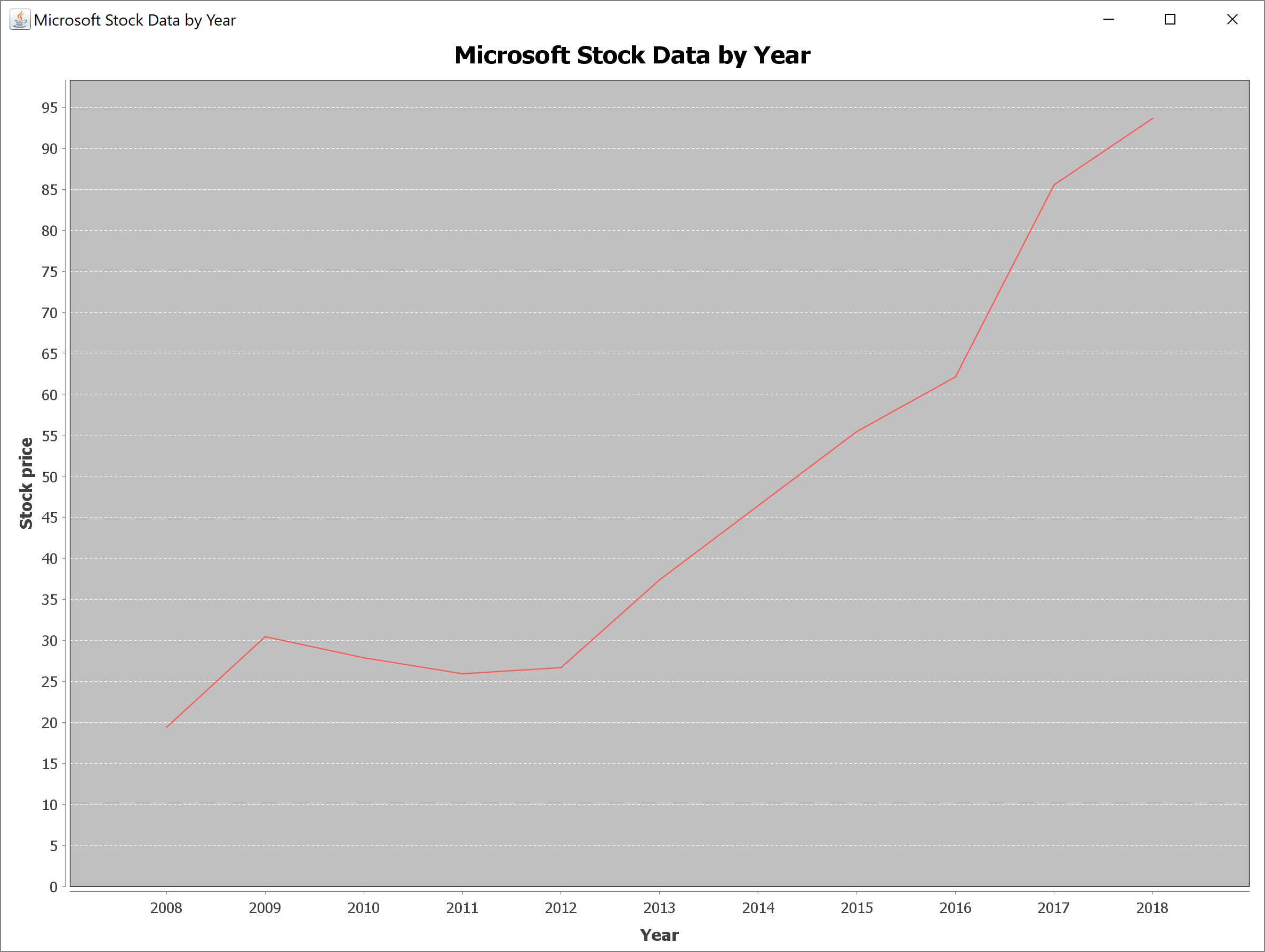
**chartData** – this function charts the closing value of the stock on the latest day of each year. The chart is a line chart with the year along the x-axis and the stock value along the y-axis. It calls function **createPanel**, places the panel in a frame, and shows the frame.

**createPanel** – this function call functions **createDataset** and **createChart**. It uses class **DefaultCategoryDataset** instead of class **PieDataset**.

**createDataset** – this function uses class **DefaultCategoryDataset** instead of class **PieDataset**. It also uses the arrays created when determining the closing value of the stock on the latest day of each year.

**createChart** – this function uses class **createLineChart** instead of class **createPieChart**. See web pages [www.jfree.org/jfreechart/api/gjdoc/org/jfree/chart/ChartFactory.html#createLineChart:String:String:String:CategoryDataset:PlotOrientation:boolean:boolean:boolean](http://www.jfree.org/jfreechart/api/gjdoc/org/jfree/chart/ChartFactory.html#createLineChart:String:String:String:CategoryDataset:PlotOrientation:boolean:boolean:boolean) and [www.tutorialspoint.com/jfreechart/jfreechart\_line\_chart.htm](https://www.tutorialspoint.com/jfreechart/jfreechart_line_chart.htm) for more information.

Here is what the chart should look like:



*[your program code here*

//======================================================================

// Title: <MS\_Stock\_Market\_2018>

// Course: CSC 3020

// Homework: <#4>

// Author: <Sayem Chowdhury>

// Date: <date>

// Description:

// <aAJava console application...

// ...that analyzes their stock data over the past eleven years.>

//======================================================================

//Package Name

**package** MS\_Stock;

//Import Classes

//-------------------------------------------------------------------------

**import** java.io.FileInputStream;

**import** java.io.FileNotFoundException;

**import** java.io.IOException;

**import** java.util.ArrayList;

**import** java.util.Scanner;

**import** javax.swing.JFrame;

**import** javax.swing.JPanel;

**import** org.jfree.chart.ChartFactory;

**import** org.jfree.chart.ChartPanel;

**import** org.jfree.chart.JFreeChart;

**import** org.jfree.data.category.DefaultCategoryDataset;

//------------------------------------------------------------------

//Class Name

**public** **class** MS\_Stock\_Market\_2018 {

**static** **int** *dataCount* =0;

//Declare formatting constants

**private** **static** String *header*;

//Function that read file and save data into specified ArrayList

**public** **static** **void** readTextFile(ArrayList<String> date, ArrayList<Double> close,

ArrayList<Integer> volume,ArrayList<Double> open, ArrayList<Double> high, ArrayList<Double> low)

{

// Declaring variables

Scanner fileIn = **null**;

**boolean** headerRead;

String line;

// Attempt to open input file

//try catch Block

**try**//Exception handler

{

// Assign external file to file handle

fileIn = **new** Scanner(**new** FileInputStream("MicrosoftStockData.txt"));

// Loop to read data

headerRead = **false**;

//dataCount = 0;

**while** (fileIn.hasNextLine())

{

// Read each line and test if header row

line = fileIn.nextLine();

**if** (!headerRead)

{

*header* = line;

headerRead = **true**;

}

**else**

{

String[] Token = line.split(",");

date.add(Token[0]);

close.add(Double.*parseDouble*(Token[1]));

volume.add(Integer.*parseInt*(Token[2]));

open.add(Double.*parseDouble*(Token[3]));

high.add(Double.*parseDouble*(Token[4]));

low.add(Double.*parseDouble*(Token[5]));

// Incrementing line counter

*dataCount* = *dataCount* + 1;

}

}

// Show number of lines read

System.***out***.println(*dataCount* + " data line(s) " +

"read from file '" + "MicrosoftStockData.txt" + "'.");

// Close input file

fileIn.close();

}

// Handle file error

**catch** (FileNotFoundException e) //catch Block

{

System.***out***.println("Error: file '" + "MicrosoftStockData.txt" +

"' not found.");

System.***out***.println("Default folder: " +

System.*getProperty*("user.dir"));

System.***out***.println("Error message:\n" + e.getMessage());

}

}

//---------------------------Print Function----------------------------------------------

**public** **static** **void** printData(ArrayList<String> date, ArrayList<Double> close, ArrayList<Integer> volume,

ArrayList<Double> open, ArrayList<Double> high, ArrayList<Double> low) **throws** IOException

{

System.***out***.println("\nFirst 12 days of data rangaing from "

+ "3/5/2018 - 2/15/2018");

System.***out***.print("Date "+"close " +"Volume "+"Open "+"High "+"Low \n");

//System.out.printf("%13s %13s %,15s %13s %13s %13s%n", "Date","Close", "Volume", "Open", "High", "low");

System.***out***.println("-------------------------------------------------------------------------------------------------");

**int** i=0;

**while**(i<12)

{

System.***out***.printf("%-13s %13.2f %,15d %13.2f %13.2f %13.2f%n", date.get(i), close.get(i),

volume.get(i), open.get(i),high.get(i), low.get(i));

i++;

}

}

//-------------------------------------------Function analyzeData-------------------------------------------------------------------

**public** **static** **void** analyzeData(ArrayList<String> date,

ArrayList<Double> close, ArrayList<Integer> volume,

ArrayList<Double> open, ArrayList<Double> high, ArrayList<Double> low, ArrayList<Integer> y, ArrayList<Double> endC)

{

**int** i=0, maxIndex=-1;

Double max= **null**;

**for** (Double x : close)

{

**if** ((x!=**null**) && ((max==**null**) || (x>max)))

{

max = close.get(i);

maxIndex = i;

}

i++;

}

**int** j =0, minIndex=-1;

Double min= **null**;

**for** (Double x : close)

{

**if** ((x!=**null**) && ((min==**null**) || (x<min)))

{

min = close.get(j);

minIndex = j;

}

j++;

}

Double diff = high.get(0) - low.get(0);

**int** diff\_idx = 0;

**for** (**int** idx = 0; idx<high.size(); idx++)

{

**if** ((high.get(idx) -low.get(idx))>diff)

{

diff = high.get(idx) -low.get(idx);

diff\_idx = idx;

}

}

String[] only\_year = date.get(0).split("/");

y.add(Integer.*parseInt*(only\_year[2]));

endC.add(close.get(0));

**for** (**int** index= 1; index<date.size(); index++)

{

String s1 = date.get(index);

String s2 = date.get(index -1);

**int** i1 = s1.length()-1;

**int** i2 = s2.length()-1;

**if** (s1.charAt(i1) != s2.charAt(i2))

{

only\_year = **new** String[3];

only\_year = date.get(index).split("/");

y.add(Integer.*parseInt*(only\_year[2]));

endC.add(close.get(index));

}

}

System.***out***.printf("%n%13s %20s", "Year", "Closing Value");

**for** (**int** x = 0; x<y.size(); x++)

{

System.***out***.printf("\n%13s %13.2f",y.get(x), endC.get(x));

}

// print Labels Data And Value

System.***out***.printf("\n\n %10s %40s %18s %n", "Labels", "Date", "Value");

System.***out***.printf("%-45s %10s %14.2f", "Highest Closing Data",date.get(maxIndex), close.get(maxIndex));

System.***out***.printf("\n%-45s %10s %14.2f", "Lowest Closing Data",date.get(minIndex), close.get(minIndex));

System.***out***.printf("\n%-45s %10s %14.2f", "Difference of Highest and Lowest Data ",date.get(diff\_idx), diff);

}

//----------------------------------------Function CharData------------------------------------------------------------

//=========================================================================

**private** **static** **void** chartData(ArrayList<Double> yearEndClosingV, ArrayList<Integer> endY)

{

// Create panel and frame

JPanel panel = *createPanel*(yearEndClosingV,endY);

JFrame frame = **new** JFrame();

frame.setSize(2400, 1800);

frame.setTitle("Microsoft Stock Data");

frame.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

// Add panel to frame and show frame

frame.add(panel);

frame.setVisible(**true**);

}

//==========================Function creating Panel=================================================

**public** **static** JPanel createPanel(ArrayList<Double> endC, ArrayList<Integer> y)

{

DefaultCategoryDataset dataset = *createDataset*(endC, y);

JFreeChart chart = *createChart*(dataset);

JPanel panel = **new** ChartPanel(chart);

**return** panel;

}

//==========================Function DefaultCategoryDataset========================================================================

//Function DefaultCategoryDataset

**private** **static** DefaultCategoryDataset createDataset(ArrayList<Double> endC, ArrayList<Integer> y)

{

// Creating Data set object

DefaultCategoryDataset dataset = **new** DefaultCategoryDataset();

// for loop

**for** (**int** i = endC.size()-1; i>= 0; i--)

dataset.setValue(endC.get(i), "Closing Value", y.get(i));

**return** dataset;

}

//=========================== Creating\*\*Jfree\*\*\*Chart ==============================================================================================

**private** **static** JFreeChart createChart(DefaultCategoryDataset dataset)

{

JFreeChart chart = ChartFactory.*createLineChart*(

"Microsoft Stock Data by year", "Year", "Stock Price" ,dataset);

**return** chart;

}

//============================== Function Main =================================================================================================

**public** **static** **void** main(String[] args) **throws** IOException {

// **TODO** Auto-generated method stub

ArrayList<String> date = **new** ArrayList<String>();

ArrayList<Double> close = **new** ArrayList<Double>();

ArrayList<Integer> vol = **new** ArrayList<Integer>();

ArrayList<Double> open = **new** ArrayList<Double>();

ArrayList<Double> high = **new** ArrayList<Double>();

ArrayList<Double> low = **new** ArrayList<Double>();

ArrayList<Integer> year = **new** ArrayList<Integer>();

ArrayList<Double> closing\_val = **new** ArrayList<Double>();

// Header of the application

System.***out***.println("Welcome to Text File Input And Output " +

"With File Specification");

System.***out***.println("-------------------------------------------------------------------------\n");

// Loading data from text file

*readTextFile*(date, close, vol, open, high, low);

*printData*(date, close, vol, open, high, low);

*analyzeData*(date, close, vol, open, high, low, year, closing\_val);

*chartData*(closing\_val, year);

// close Application

System.***out***.println("\n\nEnd of Text File Input And Output With File Specification");

System.***out***.println("End of Application");

}

}

*]\**

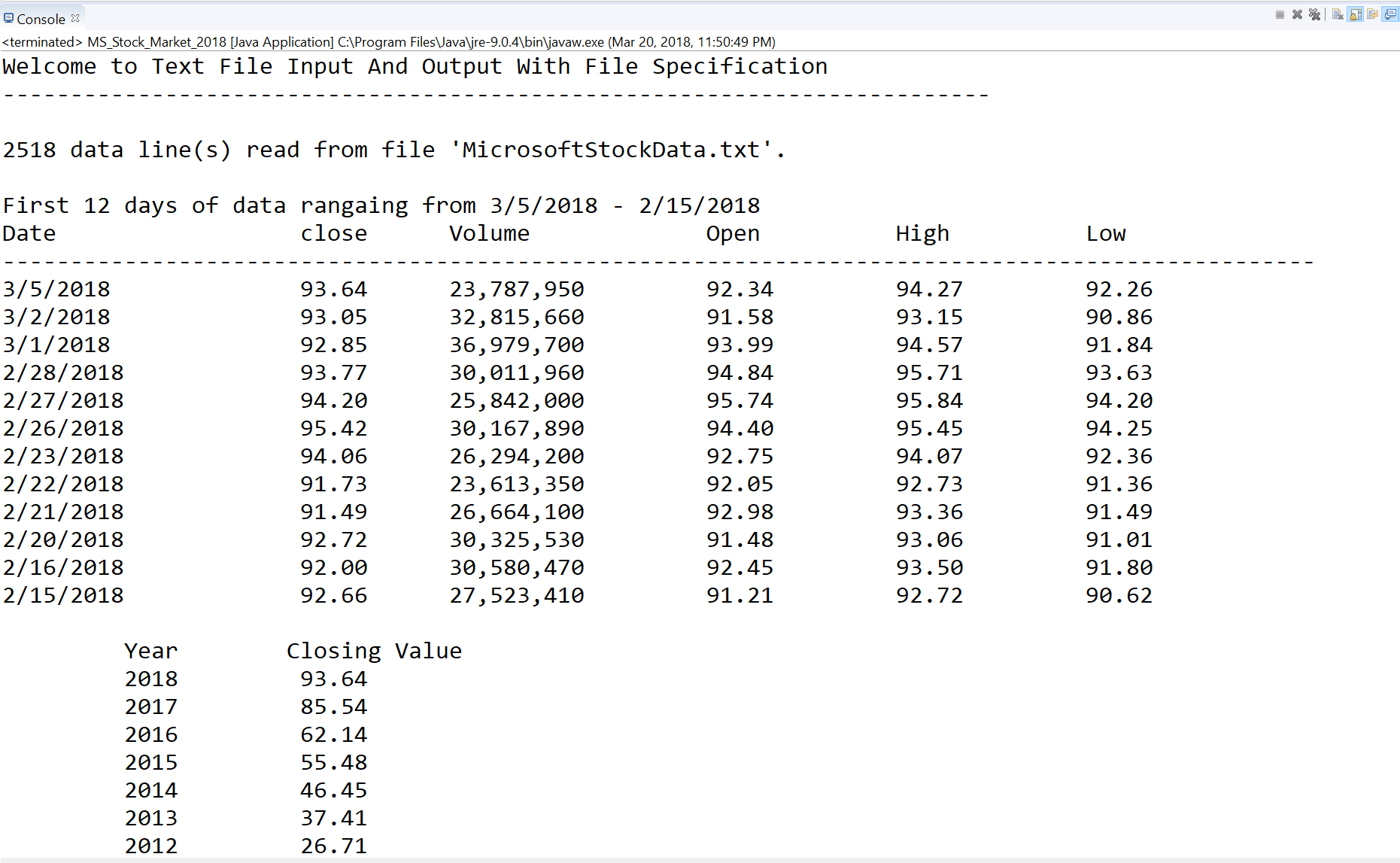
**If possible, format your code like this:**

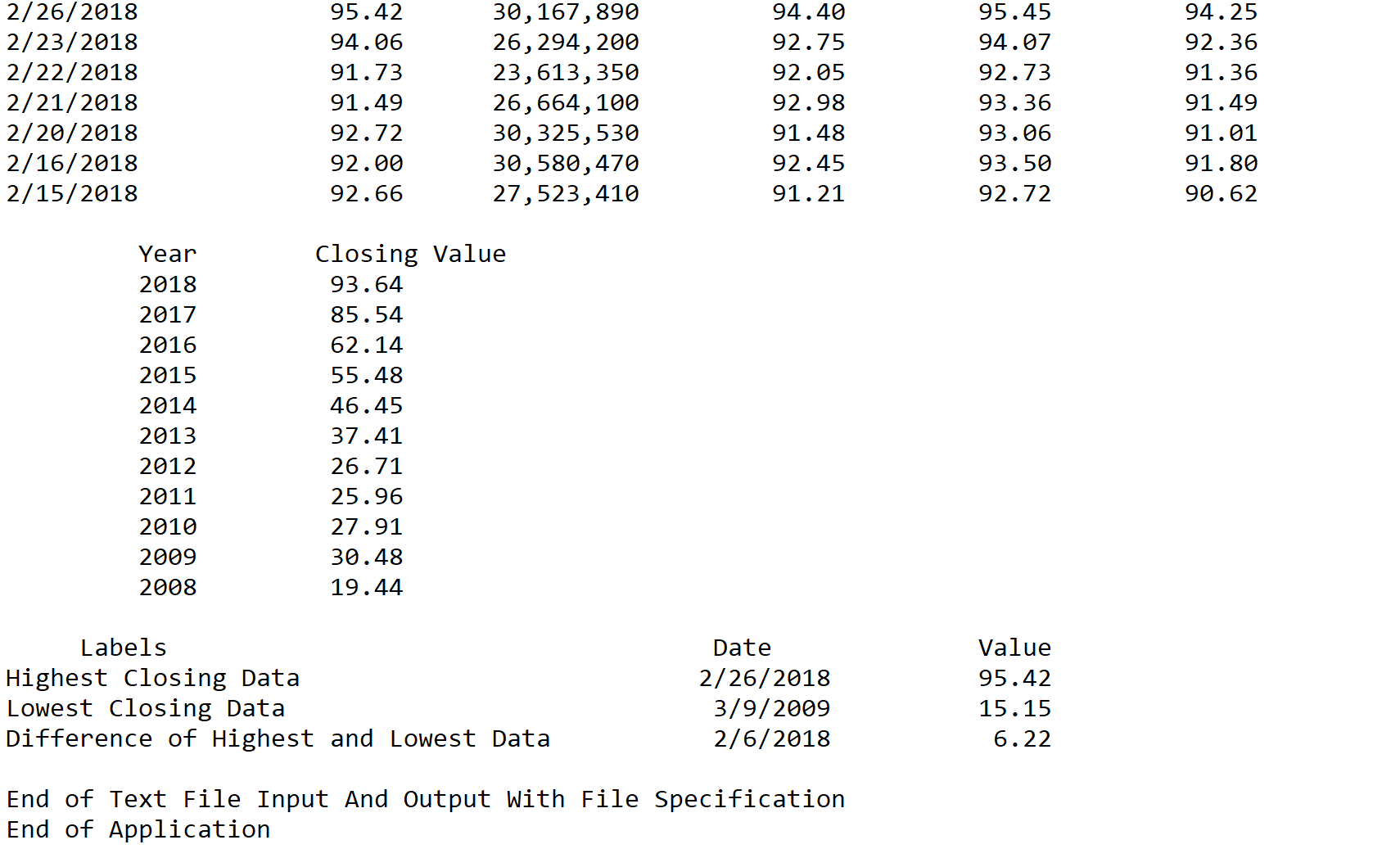
**Font “Courier New”**

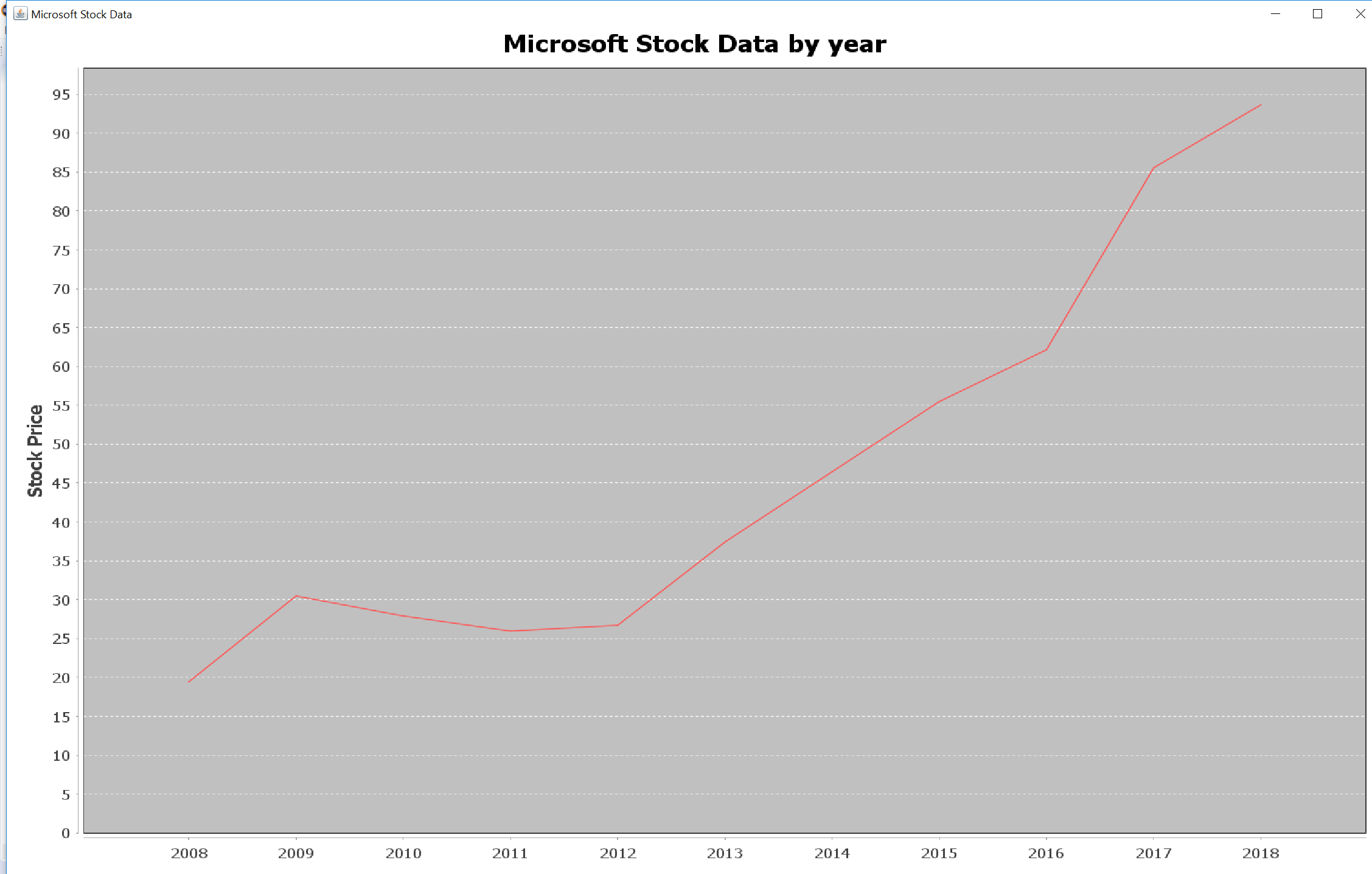
**Font size “9”**

**Bold**

*[your program output here*







*]\*\**

\* **Copying-and-pasting application code to a Word document**

1) From the program editor window, press **CTRL-A** and press **CTRL-C**.

2) From within the Word document, press **CTRL-V**.

\*\* **Copying-and-pasting application output to a Word document**

1) From the Eclipse main screen, maximize the Console window.

2) From the Console window, press **ALT-PrintScreen**.

3) From within the Word document, press **CTRL-V**.