**CSC 3020 – Java Programming**

**Homework 4 – [your name]**

**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

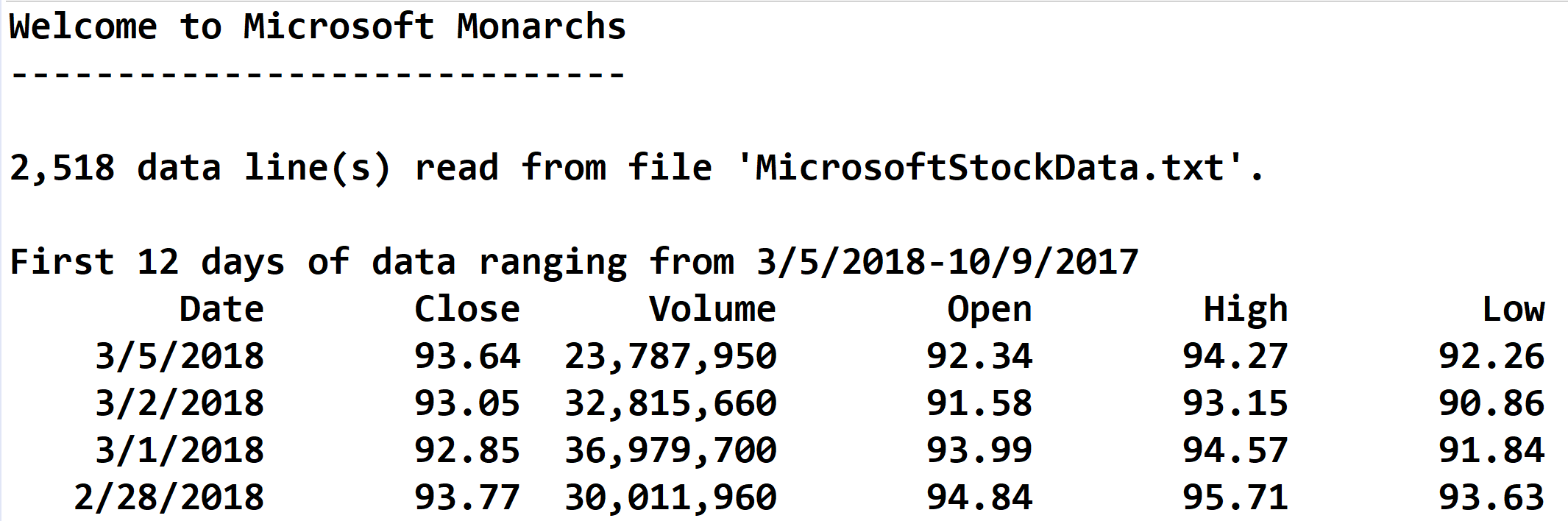
2/28/2018,93.77,30011960,94.84,95.705,93.63

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.

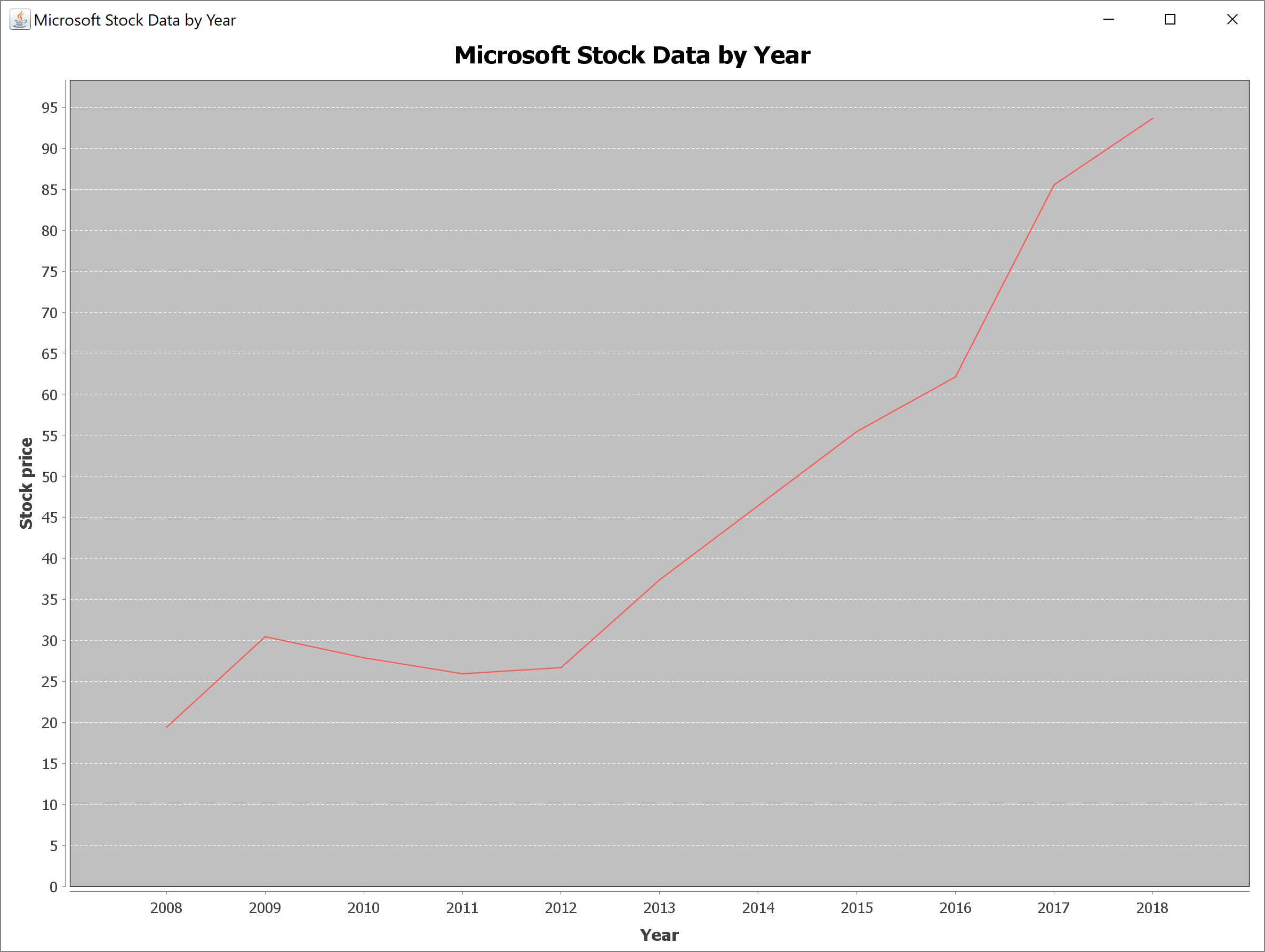
**chartData** – this function charts the closing value of the stock on the latest day of each year. The chart is a line chart with the year along the x-axis and the stock value along the y-axis. It calls function **createPanel**, places the panel in a frame, and shows the frame.

**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:



**//======================================================================**

**//**

**// Title: Microsoft Monarchs**

**// Course: CSC 3020**

**// Homework: 4-1**

**// Author: Dan Ouellette**

**// Date: 21 March 2018**

**// Description:**

**// This Java application analyzes stock data from Microsoft over the**

**// past eleven years. It reads the following fields from an input file:**

**// date, close, volume, open, high, and low. It then determines and**

**// prints:**

**// 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.**

**// The application then charts the year-end closing values of the stock.**

**//**

**//======================================================================**

**package HW04\_01;**

**// Import classes**

**import java.io.FileInputStream;**

**import java.io.FileNotFoundException;**

**import java.util.Scanner;**

**import java.util.StringTokenizer;**

**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.chart.plot.PlotOrientation;**

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

**import org.jfree.data.general.DefaultPieDataset;**

**import org.jfree.data.general.PieDataset;**

**//======================================================================**

**// class HW04\_01**

**//======================================================================**

**public class HW04\_01**

**{**

**//------------------------------------------------------------------**

**// Constants**

**//------------------------------------------------------------------**

**// Declare constants**

**public static final int ARRAY\_SIZE\_1 = 3000;**

**public static final int ARRAY\_SIZE\_2 = 11;**

**public static final int PRINT\_MAX = 12;**

**public static final String FILE\_NAME = "MicrosoftStockData.txt";**

**public static final String COLFMTS1 = "%12s";**

**public static final String COLFMTF1 = "%,12.2f";**

**public static final String COLFMTD1 = "%,12d";**

**public static final String COLFMTD2 = "%,1d";**

**public static final String COLFMTS2 = "%-12s";**

**// Declare variables**

**private static int lines;**

**private static int years;**

**private static String[] chartYears = new String[ARRAY\_SIZE\_2];**

**private static double[] chartValues = new double[ARRAY\_SIZE\_2];**

**//------------------------------------------------------------------**

**// readTextFile**

**//------------------------------------------------------------------**

**public static void readTextFile(String[] dates, double[] closes,**

**int[] volumes, double[] opens, double[] highs, double[] lows)**

**{**

**// Declare constants**

**final String DELIMITERS = ",";**

**// Declare variables**

**Scanner fileIn = null;**

**String line;**

**StringTokenizer tokens;**

**// Attempt to open input file**

**try**

**{**

**// Assign external file to file handle**

**fileIn = new Scanner(new FileInputStream(FILE\_NAME));**

**// Scan past header line**

**line = fileIn.nextLine();**

**// Loop to read data**

**lines = 0;**

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

**{**

**// Read line and store columns**

**line = fileIn.nextLine();**

**tokens = new StringTokenizer(line, DELIMITERS);**

**dates[lines] = tokens.nextToken();**

**closes[lines] = Double.parseDouble(tokens.nextToken());**

**volumes[lines] = Integer.parseInt(tokens.nextToken());**

**opens[lines] = Double.parseDouble(tokens.nextToken());**

**highs[lines] = Double.parseDouble(tokens.nextToken());**

**lows[lines] = Double.parseDouble(tokens.nextToken());**

**// Increment line counter**

**lines = lines + 1;**

**}**

**// Show number of lines read**

**System.out.println("\n" + String.format(COLFMTD2, lines) +**

**" data line(s) read from file '" + FILE\_NAME + "'.");**

**// Close input file**

**fileIn.close();**

**}**

**// Handle file error**

**catch (FileNotFoundException e)**

**{**

**System.out.println("Error: file '" + FILE\_NAME +**

**"' not found.");**

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

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

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

**}**

**}**

**//----------------------------------------------------------------**

**// printData**

**//----------------------------------------------------------------**

**public static void printData(String[] dates, double[] closes,**

**int[] volumes, double[] opens, double[] highs, double[] lows)**

**{**

**// Print heading and column headers**

**System.out.println("\nFirst " + PRINT\_MAX +**

**(PRINT\_MAX == 1 ? " day" : " days") +**

**" of data ranging from " + dates[0] + "-" + dates[11]);**

**System.out.printf(COLFMTS1 + COLFMTS1 + COLFMTS1 + COLFMTS1 +**

**COLFMTS1 + COLFMTS1 + "%n", "Date", "Close", "Volume",**

**"Open", "High", "Low");**

**// Loop to print data**

**for (int i = 0; i < lines; i++)**

**if (i < PRINT\_MAX)**

**System.out.printf(COLFMTS1 + COLFMTF1 + COLFMTD1 +**

**COLFMTF1 + COLFMTF1 + COLFMTF1 + "%n", dates[i],**

**closes[i], volumes[i], opens[i], highs[i], lows[i]);**

**}**

**//----------------------------------------------------------------**

**// analyzeData**

**//----------------------------------------------------------------**

**public static void analyzeData(String[] dates, double[] closes,**

**int[] volumes, double[] opens, double[] highs, double[] lows)**

**{**

**// Declare variables**

**double stockHigh = -Double.MAX\_VALUE;**

**int stockHighIndex = 0;**

**double stockLow = Double.MAX\_VALUE;**

**int stockLowIndex = 0;**

**double stockVolHighByDay = -Double.MAX\_VALUE;**

**int stockVolHighByDayIndex = 0;**

**double diff;**

**String prevYear = "";**

**String currYear;**

**// Loop to analyze data**

**years = 0;**

**for (int d = 0; d < lines; d++)**

**{**

**// Test for high close**

**if (stockHigh < closes[d])**

**{**

**stockHigh = closes[d];**

**stockHighIndex = d;**

**}**

**// Test for low close**

**if (stockLow > closes[d])**

**{**

**stockLow = closes[d];**

**stockLowIndex = d;**

**}**

**// Test for high close**

**diff = highs[d] - lows[d];**

**if (stockVolHighByDay < diff)**

**{**

**stockVolHighByDay = diff;**

**stockVolHighByDayIndex = d;**

**}**

**// Test if new year**

**currYear = dates[d].substring(dates[d].length() - 4);**

**if (!currYear.equals(prevYear))**

**{**

**chartYears[years] = currYear;**

**chartValues[years] = closes[d];**

**prevYear = currYear;**

**years = years + 1;**

**}**

**}**

**// Print year results**

**System.out.println("\nYear closes");**

**System.out.printf(COLFMTS2 + COLFMTS1 + "%n", "Year", "Close");**

**for (int y = 0; y < years; y++)**

**System.out.printf(COLFMTS2 + COLFMTF1 + "%n",**

**chartYears[y], chartValues[y]);**

**// Print analysis results**

**System.out.println("\nData Analysis");**

**System.out.printf(COLFMTS2 + COLFMTS1 + COLFMTS1 + "%n",**

**"Category", "Date", "Close");**

**System.out.printf(COLFMTS2 + COLFMTS1 + COLFMTF1 + "%n",**

**"High", dates[stockHighIndex], stockHigh);**

**System.out.printf(COLFMTS2 + COLFMTS1 + COLFMTF1 + "%n",**

**"Low", dates[stockLowIndex], stockLow);**

**System.out.printf(COLFMTS2 + COLFMTS1 + COLFMTF1 + "%n",**

**"Volatile day", dates[stockVolHighByDayIndex],**

**stockVolHighByDay);**

**}**

**//----------------------------------------------------------------**

**// chartData**

**//----------------------------------------------------------------**

**public static void chartData()**

**{**

**// Create panel and frame**

**JPanel panel = createPanel();**

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

**frame.setSize(2400, 1800);**

**frame.setTitle("Microsoft Stock Data by Year");**

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

**// Add panel to frame and show frame**

**frame.add(panel);**

**frame.setVisible(true);**

**}**

**//------------------------------------------------------------------**

**// createPanel**

**//------------------------------------------------------------------**

**public static JPanel createPanel()**

**{**

**DefaultCategoryDataset dataset = createDataset();**

**JFreeChart chart = createChart(dataset);**

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

**return panel;**

**}**

**//------------------------------------------------------------------**

**// createDataset**

**//------------------------------------------------------------------**

**private static DefaultCategoryDataset createDataset()**

**{**

**// Create dataset object**

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

**// Loop to set dataset pairs (label, value)**

**for (int i = chartYears.length - 1; i >= 0; i--)**

**dataset.addValue(chartValues[i], "stock", chartYears[i]);**

**return dataset;**

**}**

**//------------------------------------------------------------------**

**// createChart**

**//------------------------------------------------------------------**

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

**{**

**JFreeChart chart = ChartFactory.createLineChart(**

**"Microsoft Stock Data by Year", // Chart title**

**"Year", // x-axis label**

**"Stock price", // y-axis label**

**dataset,**

**PlotOrientation.VERTICAL,**

**false, // Show legend**

**true, // Show tooltips**

**false); // Show urls**

**return chart;**

**}**

**//----------------------------------------------------------------**

**// main**

**//----------------------------------------------------------------**

**public static void main (String[] args)**

**{**

**// Declare variables**

**String[] dates = new String[ARRAY\_SIZE\_1];**

**double[] closes = new double[ARRAY\_SIZE\_1];**

**int[] volumes = new int[ARRAY\_SIZE\_1];**

**double[] opens = new double[ARRAY\_SIZE\_1];**

**double[] highs = new double[ARRAY\_SIZE\_1];**

**double[] lows = new double[ARRAY\_SIZE\_1];**

**// Show application header**

**System.out.println("Welcome to Microsoft Monarchs");**

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

**// Read and print data**

**readTextFile(dates, closes, volumes, opens, highs, lows);**

**printData(dates, closes, volumes, opens, highs, lows);**

**// Analyze data and print and chart results**

**analyzeData(dates, closes, volumes, opens, highs, lows);**

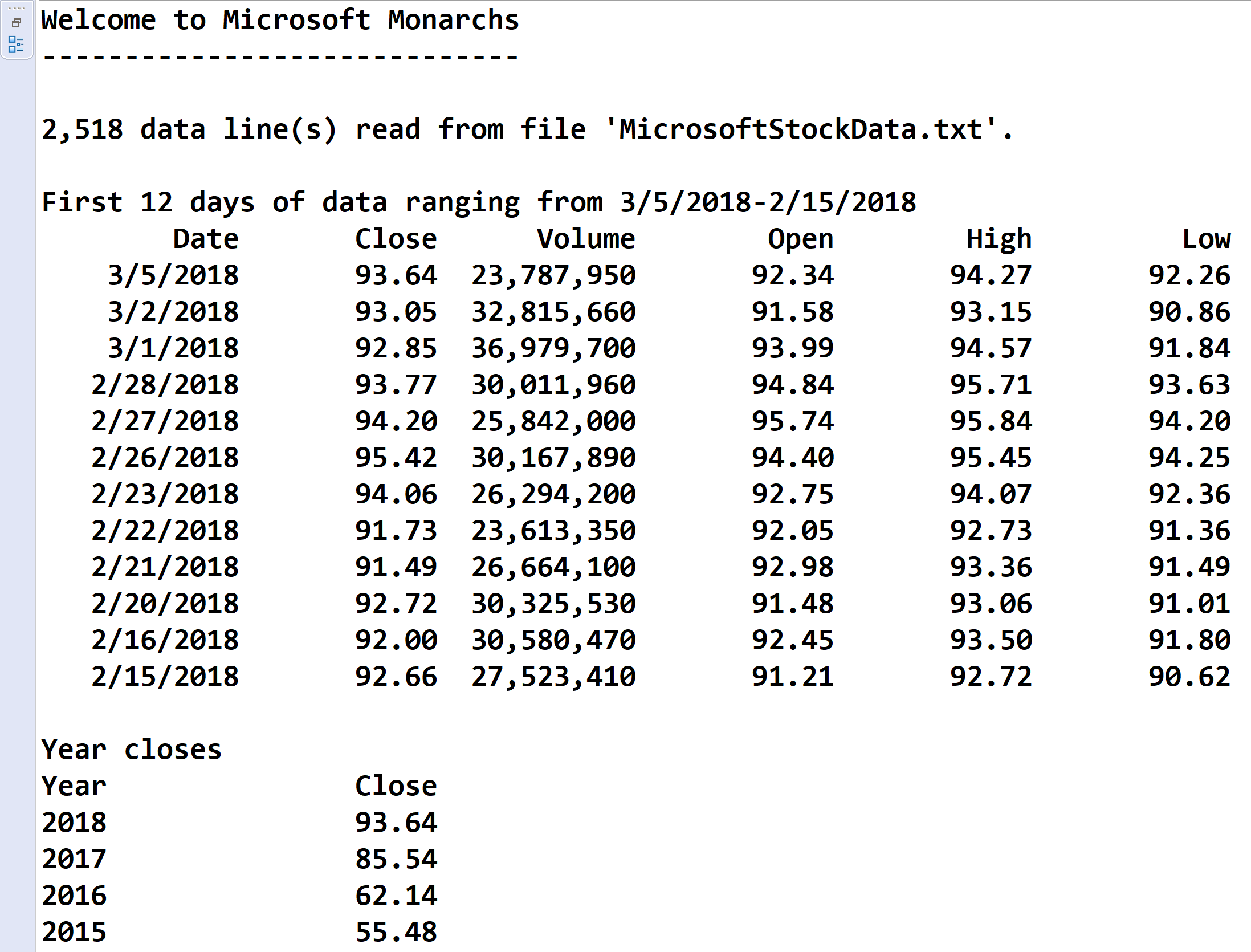
**chartData();**

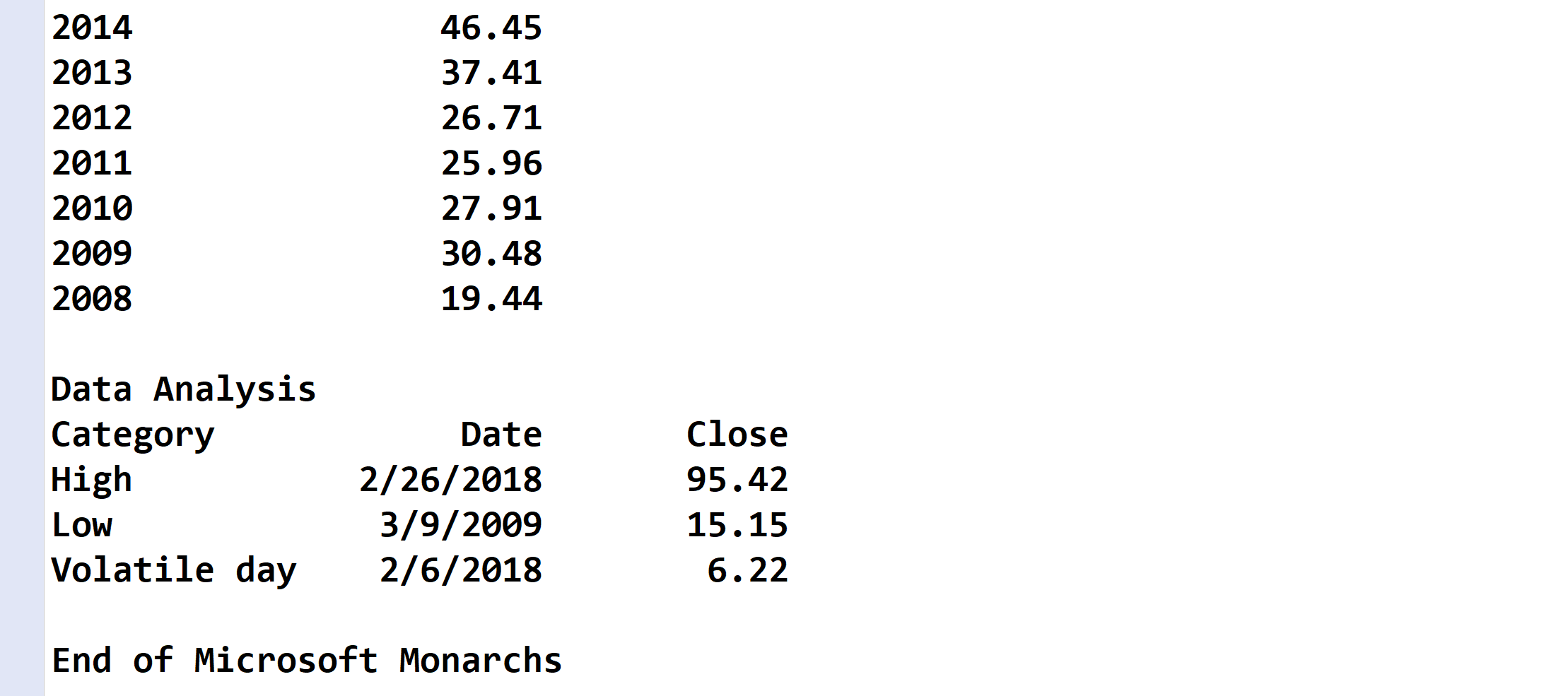
**// Show application close**

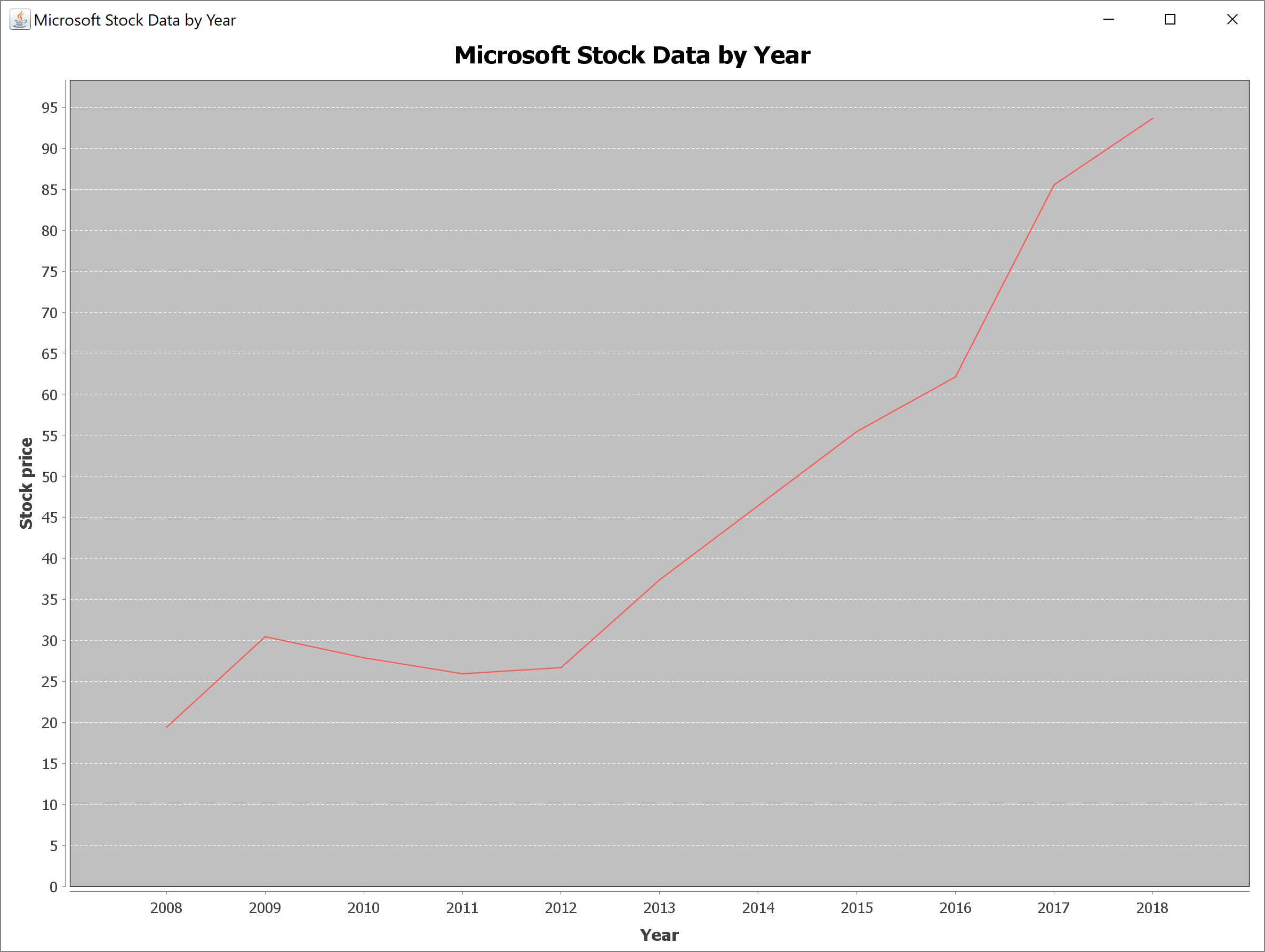
**System.out.println("\nEnd of Microsoft Monarchs");**

**}**

**}**







\* **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**.