**CSC 3020 – Java Programming**

**Homework 4 – [your name]**

**25 points – Due March 21, 10am**

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

Mustafa Chowdhury

Ge3306

**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

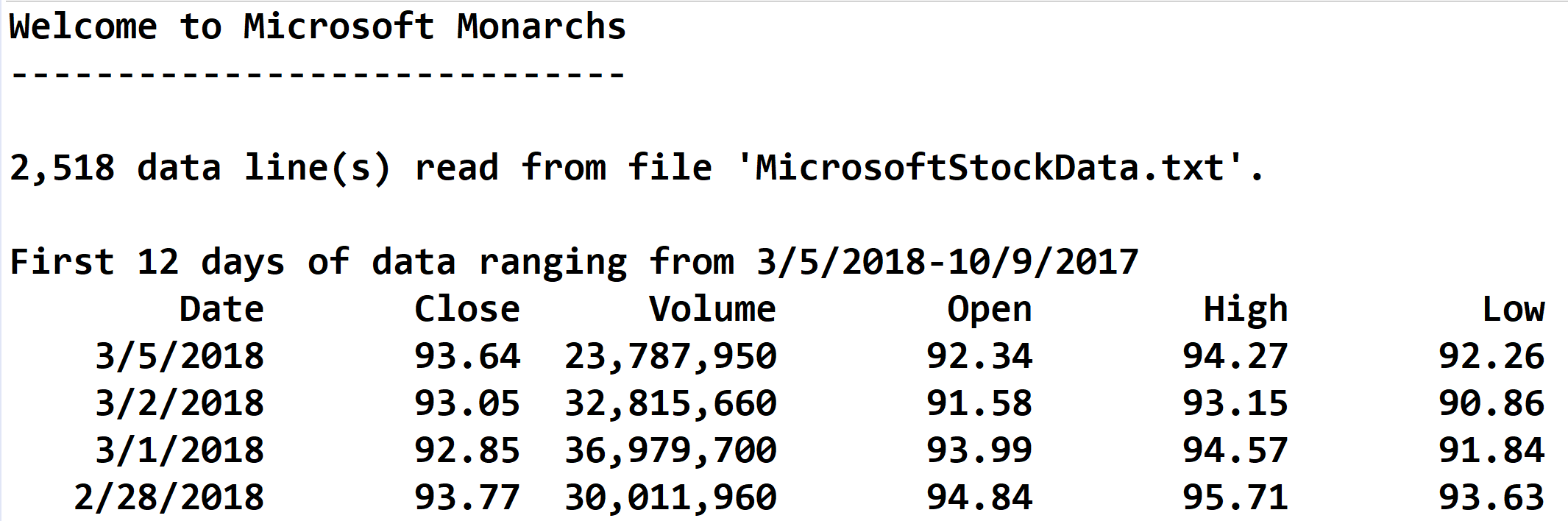
2/28/2018,93.77,30011960,94.84,95.705,93.63

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.

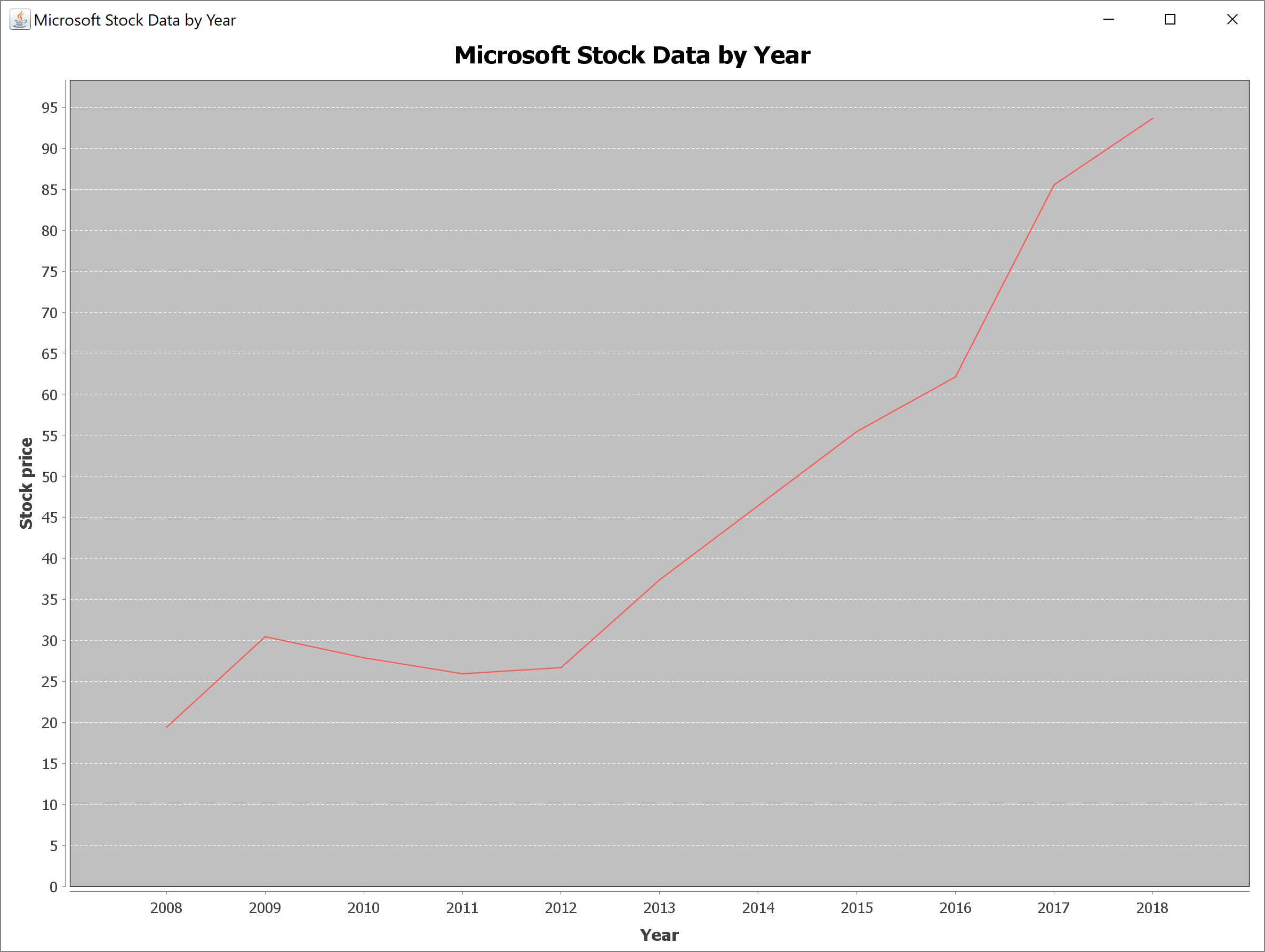
**chartData** – this function charts the closing value of the stock on the latest day of each year. The chart is a line chart with the year along the x-axis and the stock value along the y-axis. It calls function **createPanel**, places the panel in a frame, and shows the frame.

**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: Microsoft Monarchs**

**// Course: CSC 3020**

**// Homework: Homework 4**

**// Author: Mustafa Chowdhury, ge3306**

**// Date: 03/20/2018**

**// Description:**

**// 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.**

**// 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**

**//**

**// analyzeData – finding highest lowest closing value and higest difference between high and low values**

**// than draw a line chart**

**//**

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

**// package**

**package project\_4;**

**//importing package**

**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**

**public class project\_4 {**

**// declare arrayList and file name and data count**

**// make it static so it can be access by any function**

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

**// date**

**public static ArrayList<String> d = new ArrayList<String>();**

**//close**

**public static ArrayList<Double> c = new ArrayList<Double>();**

**// volume**

**public static ArrayList<Integer> v = new ArrayList<Integer>();**

**// open**

**public static ArrayList<Double> o = new ArrayList<Double>();**

**//high**

**public static ArrayList<Double> h = new ArrayList<Double>();**

**// low**

**public static ArrayList<Double> l = new ArrayList<Double>();**

**// year corresponding with closing value**

**public static ArrayList<Integer> y = new ArrayList<Integer>();**

**// end year closing value**

**public static ArrayList<Double> endC = new ArrayList<Double>();**

**// static variable to count**

**// how many lines in the text file**

**static int dataCount =0;**

**// Declare formatting constants**

**public static final String COLFMT1 = "%-12s";**

**public static final String COLFMT2 = "%13.2f";**

**public static final String COLFMT3 = "%,15d";**

**public static final String COLFMT4 = "%13.2f";**

**public static final String COLFMT5 = "%13.2f";**

**public static final String COLFMT6 = "%13.2f";**

**private static String header;**

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

**// read data from text file**

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

**public static void readTextFile()**

**{**

**// Declare variables**

**Scanner fileIn = null;**

**boolean headerRead;**

**String line;**

**// Attempt to open input file**

**try**

**{**

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

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

**// Loop to read data**

**headerRead = false;**

**//dataCount = 0;**

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

**{**

**// Read line and test if header row**

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

**if (!headerRead)**

**{**

**header = line;**

**headerRead = true;**

**}**

**else**

**{**

**// splitting each line into a substring and**

**// convert it into desired data type**

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

**d.add(split[0]);**

**c.add(Double.parseDouble(split[1]));**

**v.add(Integer.parseInt(split[2]));**

**o.add(Double.parseDouble(split[3]));**

**h.add(Double.parseDouble(split[4]));**

**l.add(Double.parseDouble(split[5]));**

**// Increment line counter**

**dataCount = dataCount + 1;**

**}**

**}**

**// Show number of lines read**

**System.out.println(dataCount + " 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());**

**}**

**}**

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

**// print data from text file**

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

**@SuppressWarnings("resource")**

**public static void printData() throws IOException**

**{**

**// printing dates of the range**

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

**+ d.get(0)+" - " + d.get(11) + "\n");**

**// printing heading of the column**

**System.out.printf(COLFMT1+ "\t " +COLFMT1+COLFMT1+"\t "+COLFMT1+" "+COLFMT1 +" "+ COLFMT1 +"%n",**

**"Date","close", "Volume", "Open", "High", "Low");**

**// print first 12 lines from text file**

**for (int i = 0; i<12; i++)**

**{**

**System.out.printf(**

**COLFMT1 + COLFMT2 + COLFMT3 + COLFMT4 + COLFMT5 + COLFMT6 + "%n",**

**d.get(i),**

**c.get(i),**

**v.get(i),**

**o.get(i),**

**h.get(i),**

**l.get(i));**

**}**

**}**

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

**// analyze data**

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

**public static void analyzeData()**

**{**

**// declaring double variable**

**//get the value of the first index**

**//from closing value and save it into variables**

**Double highestData = c.get(0);**

**Double lowestData = c.get(0);**

**// declaring integer variable**

**int idxH = 0;**

**int idxL = 0;**

**// finding highest and lowest value from closing value array list**

**for (int k =1; k<c.size(); k++)**

**{**

**// comparing if the index value less or**

**// greater than the highest data**

**if (c.get(k) > highestData)**

**{**

**highestData = c.get(k);**

**idxH = k;**

**}**

**else if (c.get(k) < lowestData)**

**{**

**lowestData = c.get(k);**

**idxL = k;**

**}**

**}**

**// getting the highest differences by subtracting high and low**

**Double diff = h.get(0) - l.get(0);**

**int diff\_idx = 0;**

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

**{**

**if ((h.get(idx) -l.get(idx))>diff)**

**{**

**diff = h.get(idx) -l.get(idx);**

**diff\_idx = idx;**

**}**

**}**

**// Spiting the dates into tokens**

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

**// add the first value into year array and end closing value**

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

**endC.add(c.get(0));**

**// getting the year and its end date closing value**

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

**{**

**String s1 = d.get(index);**

**String s2 = d.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 = d.get(index).split("/");**

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

**endC.add(c.get(index));**

**}**

**}**

**// Now printing all the finding**

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

**+ "\n\nYearly Closing Value: ");**

**System.out.printf("%n%12s %22s", "Year", " Closing Value");**

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

**{**

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

**}**

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

**// print**

**System.out.printf("\n\nAnalyzed Data: \n\n\t %10s %23s %15s %n", "Labels", "Date", "Value");**

**System.out.printf("\t%-25s %10s %14.2f", "Highest Close Data",d.get(idxH), c.get(idxH));**

**System.out.printf("\n\t%-25s %10s %14.2f", "Lowest Close Data",d.get(idxL), c.get(idxL));**

**System.out.printf("\n\t%-25s %10s %14.2f", "High/Low Difference",d.get(diff\_idx), diff);**

**}**

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

**// All Chart Drawing method in below:**

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

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

**// chartData**

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

**private static void chartData()**

**{**

**// Create panel and frame**

**JPanel panel = createPanel();**

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

**frame.setSize(3000, 2000);**

**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;**

**}**

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

**// createChart**

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

**private static DefaultCategoryDataset createDataset()**

**{**

**// Create data set object**

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

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

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

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

**return dataset;**

**}**

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

**// createChart**

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

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

**{**

**JFreeChart chart = ChartFactory.createLineChart(**

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

**return chart;**

**}**

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

**// main**

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

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

**{**

**// Show application header**

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

**System.out.println("--------------------------------------" +**

**"-----------------------\n");**

**// Load data from text file**

**// calling method**

**readTextFile();**

**//print data**

**//calling printData method**

**printData();**

**// calculate and analyze data**

**// calling method**

**analyzeData();**

**// calling chart data**

**chartData();**

**// Show application close**

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

**+ "-----------------------------\nEnd of Microst Monachars Program");**

**}**

**}**

*[your program output here]*

