```
package edu.gmu.classifier.em;

import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;

import javax.swing.JFrame;

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.statistics.HistogramDataset;
import org.jfree.data.xy.DefaultXYDataset;

public class Homework8
{
        public static void main( String args[] ) throws IOException
        {
                //load 2000 examples from the data file
                double[] data = loadDataFile( "/home/ulman/CSI873/lec9/Geoffrey.txt" );

                // get the number of data elements loaded from the file
                int size = data.length;

                // create an array to store the estimated zj values for each data element
                double[] z1 = new double[size];
                double[] z2 = new double[size];

                // initialize the distribution means to 0
                double u1 = Math.random( ) * 6 - 3;
                double u2 = Math.random( ) * 6 - 3;

                // the assumed sigma for both distributions is 1;
                double sigma = 1;

                // run for a fixed number of iterations
                int iterations = 30;

                // allocate memory for storing u1 and u2 values at each iteration (for plotting)
                double[] u1l = new double[iterations];
                double[] u2l = new double[iterations];

                // run the EM algorithm
                for ( int i = 0; i < iterations; i++ )
                {
                        // Step 1 (calculate hidden variable values E[z_ij])
                        z1 = calculate_z_ij( data, u1, u2, sigma, z1 );
                        z2 = calculate_z_ij( data, u2, u1, sigma, z2 );

                        // Step 2 (calculate uj values)
                        u1 = calculate_u_j( data, z1 );
                        u2 = calculate_u_j( data, z2 );

                        // store and print results at each iteration
                        u1l[i] = u1;
                        u2l[i] = u2;
                        System.out.printf( "Iteration: %d Mean 1: %.2f Mean 2: %.2f%n", i, u1, u2 );
                }

                // create jfreechart dataset for plotting purposes
                DefaultXYDataset dataset = new DefaultXYDataset( );

                double[][] seriesData1 = new double[2][iterations];
                for ( int i = 0; i < iterations; i++ )
                {
                        seriesData1[0][i] = i;
                        seriesData1[1][i] = u1l[i];
                }
                dataset.addSeries( "u1", seriesData1 );

                double[][] seriesData2 = new double[2][iterations];
                for ( int i = 0; i < iterations; i++ )
                {
                        seriesData2[0][i] = i;
                        seriesData2[1][i] = u2l[i];
                }
                dataset.addSeries( "u2", seriesData2 );

                JFreeChart chart2 = ChartFactory.createXYLineChart( "EM Algorithm Means", "Iteration", "Mean", dataset,
PlotOrientation.VERTICAL, true, false, false );
                ChartPanel chartPanel2 = new ChartPanel( chart2 );
                JFrame frame2 = new JFrame( );
                frame2.setSize( 1000, 1000 );
                frame2.add( chartPanel2 );
                frame2.setVisible( true );
```

```
                HistogramDataset dataset2 = new HistogramDataset( );

                double[] seriesData3 = new double[size];
                for ( int i = 0; i < size; i++ )
                {
                        seriesData3[i] = data[i];
                }
                dataset2.addSeries( "data", seriesData3, 60 );

//              XYLineAnnotation a1 = new XYLineAnnotation( u1, 0.0, u1, 1.0 );
//              XYLineAnnotation a2 = new XYLineAnnotation( u2, 0.0, u2, 1.0 );

                //( title, xAxisLabel, yAxisLabel, dataset2, orientation, legend, tooltips, urls )
                JFreeChart chart3 = ChartFactory.createHistogram( "Data Histogram", "Value", "Relative Frequency", dataset2,
PlotOrientation.VERTICAL, true, false, false );
                ChartPanel chartPanel3 = new ChartPanel( chart3 );
                JFrame frame3 = new JFrame( );
                frame3.setSize( 1000, 1000 );
                frame3.add( chartPanel3 );
                frame3.setVisible( true );
        }

        // a helper function for loading data from the provided data file
        public static double[] loadDataFile( String file ) throws IOException
        {
                BufferedReader in = new BufferedReader( new InputStreamReader( new FileInputStream( file ) ) );

                List<Double> dataList = new ArrayList<Double>( );
                String line = null;

                while ( ( line = in.readLine( ) ) != null )
                {
                        String[] tokens = line.split( "[\\s]+" );
                        for ( String token : tokens )
                        {
                                dataList.add( Double.parseDouble( token ) );
                        }
                }

                double[] dataArray = new double[dataList.size( )];
                for ( int i = 0; i < dataList.size( ); i++ )
                {
                        dataArray[i] = dataList.get( i );
                }

                return dataArray;
        }

        // EM Algorithm step 1
        public static double[] calculate_z_ij( double[] data, double u1, double u2, double sigma, double[] z )
        {
                for ( int i = 0; i < data.length; i++ )
                {
                        double n1 = norm( data[i], u1, sigma );
                        double n2 = norm( data[i], u2, sigma );

                        z[i] = n1 / ( n1 + n2 );
                }

                return z;
        }

        // EM Algorithm step 2
        public static double calculate_u_j( double[] data, double[] z )
        {
                double numerator = 0.0;
                for ( int i = 0; i < data.length; i++ )
                {
                        numerator += z[i] * data[i];
                }

                double denominator = 0.0;
                for ( int i = 0; i < data.length; i++ )
                {
                        denominator += z[i];
                }

                return numerator / denominator;
        }

        // calculate the normal cdf given a point x, a mean and a variance
        public static double norm( double x, double u, double sigma )
        {
                double diff = x - u;
                return Math.exp( - ( 1.0 / ( 2.0 * sigma * sigma ) ) * diff * diff );
        }
}
```