```
package HW_1_Kmeans;
import java.io.*;
import java.util.Random;
public class Ren_YC_HW1 {
        public static void main(String[] args) {
                /**
                 *  Read file and store data into array.
                 */
                double[][] data = ReadFile_StoreDataIntoArray(args, 200, 3);
                /**
                 *  int variable way is for doing the randomly choosing and manually visual choosing the centroids.
                 */
                int way = 1;
                while (way <= 2) {
                        if (way == 1)
                                System.out.println("This is the way of Randomly choosing the centroids at the beginning."
                                                + "************************");
                        else if (way == 2)
                                System.out.println("\n\n\n\n\n\nThis is the way of visualizing the centroids at the beginning."
                                                + "************************");
                        /**
                         *  Step 5 - Repeat steps 2 - 4 for 5 times.
                         */
                        int counter = 1;
                        while (counter <= 5) {
                                System.out.println("Repeating the " + counter + " time.==========================\n");
                                /**
                                 *         Step 1 - Set K = 3; pass 3 to the method random_Centroids.
                                 *  Step 2 - Randomly generates 3 centroids.
                                 */
                                double [][] random_Centroids = RandomDouble_StoreDataIntoArray(3, 2, 2);
                                /**
                                 *  Step 6 - try to select the centroids manually.
                                 */
                                if (way == 2) {
                                        random_Centroids[0][0] = 42.5;
                                        random_Centroids[0][1] = 42.5;
                                        random_Centroids[1][0] = 26.5;
                                        random_Centroids[1][1] = 44.0;
                                        random_Centroids[2][0] = 40.5;
                                        random_Centroids[2][1] = 26.5;
                                        System.out.println("Manually picked centroids are: ");
                                }
                                if (way == 1)
                                        System.out.println("Randomly generated centroids are: ");
                                System.out.print("\t");
                                for (int i = 0; i < 3; i++) {
                                        for (int j = 0; j < 2; j++) {
                                                System.out.print(random_Centroids[i][j] + " ");
                                        }
                                        System.out.println();
                                        System.out.print("\t");
                                }
                                ClusterAssignment(random_Centroids, data);
                                boolean Centroid_is_not_fixed = true;
                                int LoopTimes = 0;
                                /**
                                 * Step 3 - Test algorithm on the centroids until convergence.
                                 */
                                while(Centroid_is_not_fixed) {
                                        /**
```

```
                     *  Assign cluster regarding to the centroids.
                     */
                    double [][] newCentoids = FindClusterCentroid(data);
                    /**
                     * printing the starting centroids.
                     */
                    System.out.println("\nStarting Centroids: ");
                    System.out.print("\t");
                    for (int i = 0; i < newCentoids.length; i++) {
                        for (int j = 0; j < newCentoids[0].length; j++) {
                            System.out.print( newCentoids[i][j] + "\t");
                        }
                        System.out.println();
                        System.out.print("\t");
                    }
                    Centroid_is_not_fixed = false;
                    for (int i = 0; i < 3; i++) {
                        for (int j = 0; j < 2; j++) {
                            if ( random_Centroids[i][j] != newCentoids[i][j] ) {
                                Centroid_is_not_fixed = true;
                                break;
                            }
                        }
                    }
                    if (Centroid_is_not_fixed == true) {
                        for (int i = 0; i < newCentoids.length; i++) {
                            for (int j = 0; j < newCentoids[0].length; j++) {
                                random_Centroids[i][j] = newCentoids[i][j];
                            }
                        }
                        ClusterAssignment(random_Centroids, data);
                    }
                    LoopTimes++;
                }//while
                System.out.println("\nLoop " + LoopTimes + " times for finding the final centroids.");
                System.out.println("\nFinal Centroids are: ");
                System.out.print("\t");
                for (int i = 0; i < random_Centroids.length; i++) {
                    for (int j = 0; j < random_Centroids[0].length; j++) {
                        System.out.print(random_Centroids[i][j] + "\t");
                    }
                    System.out.println();
                    System.out.print("\t");
                }
                /**
                 *  Step 4 - Calculate the IV and EV.
                 */
                double IV, EV, ie;
                IV = IV(random_Centroids, data);
                System.out.println("\nIV is: " + IV);
                EV = EV( data );
                System.out.println("EV is: " + EV);
                ie = IV / EV;
                System.out.println("IV / EV is: " + ie  + "\n");
                counter++;
            }//while (counter)
            way++;
        }//while (way <= 2)
    }//main
    @SuppressWarnings("resource")
    public static double [][] ReadFile_StoreDataIntoArray(String[] args, int row, int col) {
        if(args.length == 0) {
            System.out.println("No file specified");
            return null;
        }
        else {
            FileReader theFile;
            BufferedReader inFile;
            String oneLine;
            double [][] data = new double [row][col];
            try {
                theFile = new FileReader(args[0]);
                inFile = new BufferedReader(theFile);
                oneLine = inFile.readLine();
                int i = 0;
                while( !(oneLine.isEmpty()) ) {
                    String numbers[] = oneLine.split("\\s");
                    data[i][0] = Double.parseDouble(numbers[0]);
                    data[i][1] = Double.parseDouble(numbers[1]);
                    oneLine = inFile.readLine();
                    i++;
                }
            } catch (Exception e) {
                // TODO: handle exception
            }//catch
            return data;
        }//else
    }//ReadFile
    public static double [][] RandomDouble_StoreDataIntoArray(int row, int col, int Integer_digits) {
        double [][] array_with_RandomNums = new double [row][col];
        Random x = new Random();
        for (int i = 0; i < row; i++) {
            for (int j = 0; j < col; j++) {
                array_with_RandomNums[i][j] = (x.nextDouble() * Math.pow(10, Integer_digits) ) % 40 + 15;
            }
        }
        return array_with_RandomNums;
    }
    public static void ClusterAssignment( double [][] centroids, double [][] point) {
        double min;
        for (int i = 0; i < point.length; i++) {
            min = EuclideanDistance(centroids[0][0], centroids[0][1], point[i][0], point[i][1]);
            point[i][2] = 0;
            for (int j = 1; j < centroids.length; j++) {
                if (min > EuclideanDistance(centroids[j][0], centroids[j][1], point[i][0], point[i][1])) {
                    min = EuclideanDistance(centroids[j][0], centroids[j][1], point[i][0], point[i][1]);
                    point[i][2] = j;
                }
            }
        }
    }
    public static double EuclideanDistance( double x1,  double y1, double x2, double y2) {
        return Math.sqrt( Math.pow(x2 - x1, 2) + Math.pow(y2 - y1, 2) );
    }
    public static double [][] FindClusterCentroid( double [][] pointSet ){
        double [][] sum = new double [3][2];
        for (int i = 0; i < sum.length; i++) {
            for (int j = 0; j < sum[0].length; j++) {
                sum[i][j] = 0.0;
            }
        }
        int num[] = {0,0,0};
        for (int j = 0; j < pointSet.length; j++) {
            if (pointSet[j][2] == 0.0) {
                sum[0][0] += pointSet[j][0];
                sum[0][1] += pointSet[j][1];
                num[0]++;
            }
            else if (pointSet[j][2] == 1.0) {
                sum[1][0] += pointSet[j][0];
                sum[1][1] += pointSet[j][1];
                num[1]++;
            }
            else if (pointSet[j][2] == 2.0) {
                sum[2][0] += pointSet[j][0];
                sum[2][1] += pointSet[j][1];
                num[2]++;
            }
        }
        for (int i = 0; i < sum.length; i++) {
            sum[i][0] = sum[i][0] / num[i];
            sum[i][1] = sum[i][1] / num[i];
        }
        return sum;
    }
    public static double IV (double [][] centroids, double [][] points) {
        double iv = 0.0;
        for (int i = 0; i < centroids.length; i++) {
            for (int j = 0; j < points.length; j++) {
                if ( points[j][2] == (double) i) {
                    iv += EuclideanDistance(centroids[i][0], centroids[i][1], points[j][0], points[j][1]);
                }
            }
        }
        return iv;
    }
    public static double EV (double [][] points) {
        double ev = 0.0;
        for (int k = 0; k < 3; k++) {
            for (int i = 0; i < points.length; i++) {
                if(points[i][2] == k) {
                    for (int j = 0; j < points.length; j++) {
                        if (points[j][2] != k) {
                            ev += EuclideanDistance(points[i][0], points[i][1], points[j][0], points[j][1]);
                        }
                    }
                }
            }
        }
        return (ev / points.length);
    }
    public static int Cluster_volumn(double [][] points, double type) {
        int cv = 0;
        for (int i = 0; i < points.length; i++) {
            if (points[i][2] == type) cv++;
        }
        return cv;
    }
}
```