Experiment No: 01

Experiment Name: K-means Clustering Algorithm.

Algorithm:

Step 1: Start.

Step 2: Choose the number of cluster K.

Step 3: Select K random points from the data as centroids.

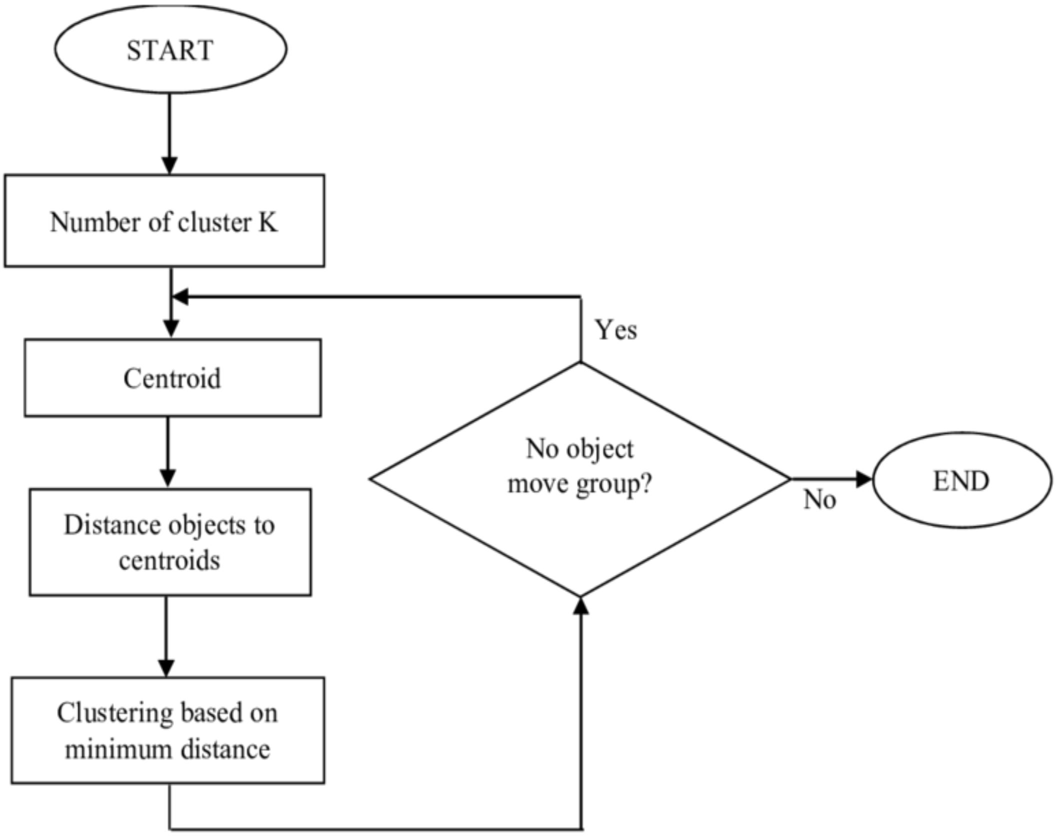
Step 4: Assign all the data points to the closest cluster centroids.

Step 5: Recompute the centroids (New centroids = mean of all points assigned to the cluster) of newly formatted cluster.

Step 6: Repeat step 4 and step 5 until the centroids of newly formed cluster do not change or points remain in the same cluster.

Step 7: Stop.

Flow Chart:


START
Number of cluster K
Centroid
Distance objects to centroids
Clustering based on minimum distance
No object move group?
Yes
No
END

Source Code:

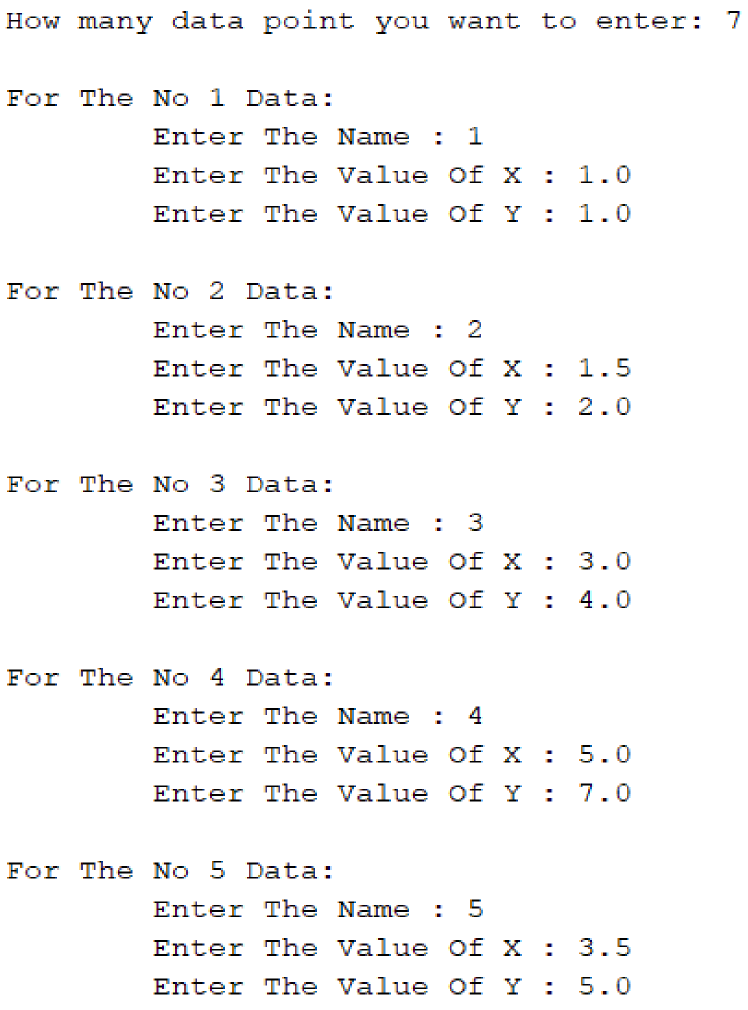
```
package kmeansalgo;

import java.util.Random;

import java.util.Scanner;

public class kmeans {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);


        //Taking input(How many dataset in the question)

        System.out.print("How many data point you want to enter: ");

        int numberOfDataPoint = input.nextInt();


        //Making array of dataset name point x and point y; (0 to
numberOfDataPoint-1) is the range;

        String[] nameOfDataset = new String[numberOfDataPoint];

        double[] pointXOfDataSet = new double[numberOfDataPoint];

        double[] pointYOfDataSet = new double[numberOfDataPoint];


        //Taking Input.

        for (int takeInputOfNameAndPointXAndPointY = 0;
takeInputOfNameAndPointXAndPointY < numberOfDataPoint;
takeInputOfNameAndPointXAndPointY++) {

            System.out.println("\nFor The No
"+(takeInputOfNameAndPointXAndPointY+1)+" Data:");

            System.out.print("\tEnter The Name : ");
            nameOfDataset[takeInputOfNameAndPointXAndPointY] =
input.next();

            System.out.print("\tEnter The Value Of X : ");

            pointXOfDataSet[takeInputOfNameAndPointXAndPointY] =
input.nextDouble();

            System.out.print("\tEnter The Value Of Y : ");

            pointYOfDataSet[takeInputOfNameAndPointXAndPointY] =
input.nextDouble();

        }

        //Taking The Number Of k means how many cluster user want.

        System.out.print("\n\nHow Many Cluster You wanna Create : ");

        int numberOfCluster = input.nextInt();

        //Choosing Centroid;

        double[] centroidX = new double[numberOfCluster];

        double[] centroidY = new double[numberOfCluster];



        //Taking Input(Centroid)

        for (int takeInputCentroid = 0; takeInputCentroid < numberOfCluster;
takeInputCentroid++) {

            System.out.println("\nFor The Centroid No "+(takeInputCentroid+1)+"
:");

            System.out.print("\tEnter The X : ");

            centroidX[takeInputCentroid] = input.nextDouble();

            System.out.print("\tEnter The Y : ");

            centroidY[takeInputCentroid] = input.nextDouble();

        }
```

```
        //How many Time we want iterate at least;
        System.out.print("\n\nEnter The Higest Number Of Iteration: ");
        int numberOfIteration = input.nextInt();

        //Distance Storing
        double[][] distance = new
double[numberOfCluster][numberOfDataPoint];

        //Where the data poing go.
        String[] whereGoing = new String[numberOfDataPoint];

        //For getting new centroid
        double[] totalValueOfX = new double[numberOfCluster];
        double[] totalValueOfY = new double[numberOfCluster];

        //Iteration
        for (int i = 0; i < numberOfIteration; i++) {
            //Calculating Distance from centroid.
            for (int row = 0; row < numberOfCluster; row++) {
                for (int col = 0; col < numberOfDataPoint; col++) {
                    double x1 = pointXOfDataSet[col];
                    double y1 = pointYOfDataSet[col];
                    double x2 = centroidX[row];
                    double y2 = centroidY[row];
                    double diffX = Math.abs(x2-x1);
```

```
          double diffY = Math.abs(y2-y1);

          distance[row][col] = Math.sqrt(Math.pow(diffX, 2) +
Math.pow(diffY, 2));

        }

      }



      //NotDinamic

      int countHowManyGoToK1 = 0;

      int countHowManyGoToK2 = 0;



      //intialize the sum var as 0

      totalValueOfX[0]=0;

      totalValueOfY[0]=0;

      totalValueOfX[1]=0;

      totalValueOfY[1]=0;



      //Grouping

      for (int col = 0; col < numberOfDataPoint; col++) {

        try{

          if(numberOfCluster == 2){  //Not dinamic

            for (int row = 0; row <=0; row++) {

              if(distance[row][col]< distance[row+1][col]){

                whereGoing[col]="K1";

                totalValueOfX[0] =
totalValueOfX[0]+pointXOfDataSet[col];
```

```
                                totalValueOfY[0] =
totalValueOfY[0]+pointYOfDataSet[col];
                                countHowManyGoToK1++;
                            }
                            else{
                                whereGoing[col]="K2";
                                totalValueOfX[1] =
totalValueOfX[1]+pointXOfDataSet[col];
                                totalValueOfY[1] =
totalValueOfY[1]+pointYOfDataSet[col];
                                countHowManyGoToK2++;
                            }
                        }
                    }
                }catch(Exception e){
                    System.out.println("We are not able to design for three or more
cluster now!");
                }
            }
            double tempCentroidXForCluster1 = centroidX[0];
            double tempCentroidYForCluster1 = centroidY[0];
            double tempCentroidXForCluster2 = centroidX[1];
            double tempCentroidYForCluster2 = centroidY[1];
            //New Centroid Point (Not Dinamic);
            centroidX[0] = totalValueOfX[0]/countHowManyGoToK1;
            centroidY[0] = totalValueOfY[0]/countHowManyGoToK1;

            centroidX[1] = totalValueOfX[1]/countHowManyGoToK2;

            centroidY[1] = totalValueOfY[1]/countHowManyGoToK2;

            if(tempCentroidXForCluster1 == centroidX[0] &&
tempCentroidYForCluster1 == centroidY[0] && tempCentroidXForCluster2
== centroidX[1] && tempCentroidYForCluster2 == centroidY[1]){

                  break;

            }

            //Table

            for (int row = 0; row < numberOfCluster; row++) {

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

                for (int col = 0; col < numberOfDataPoint; col++) {

                    System.out.printf("\t%.2f",distance[row][col]);

                }

                System.out.println("");

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

            }

            //Print

            System.out.println("\nThe New Centroid For K1");

            System.out.println("\tx: "+centroidX[0]);

            System.out.println("\ty: "+centroidY[0]);

            System.out.println("The New Centroid For For K2");

            System.out.println("\tx: "+centroidX[1]);
            System.out.println("\ty: "+centroidY[1]);
        }
        System.out.println("\nSo the data set are going to : ");
        for (int i = 0; i < numberOfDataPoint; i++) {
            System.out.println("\t"+nameOfDataset[i]+" is go to the : "+whereGoing[i]);
        }
    }
}
```

Input:

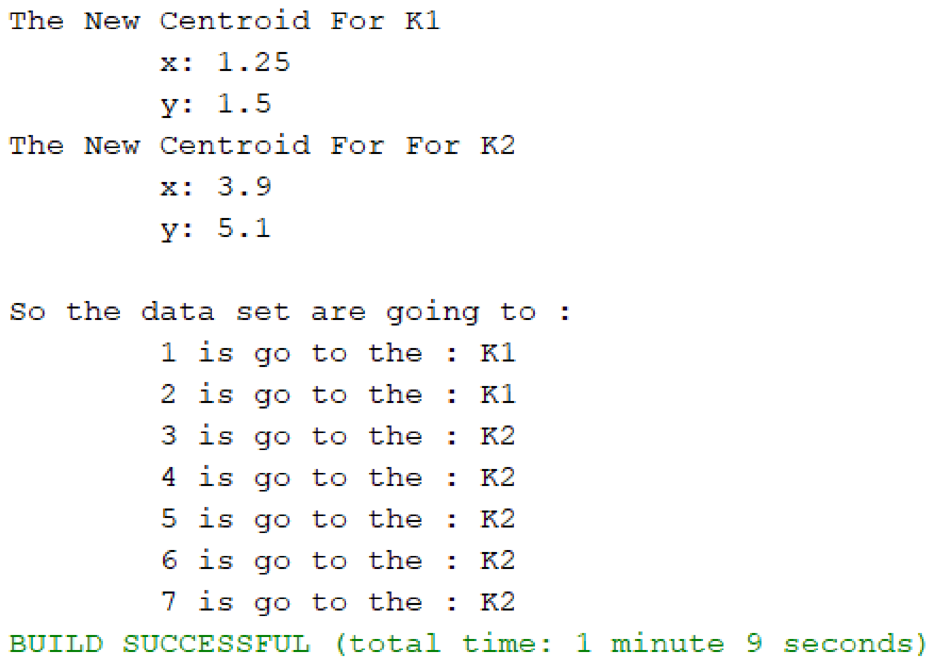


```
How many data point you want to enter: 7

For The No 1 Data:
        Enter The Name : 1
        Enter The Value Of X : 1.0
        Enter The Value Of Y : 1.0

For The No 2 Data:
        Enter The Name : 2
        Enter The Value Of X : 1.5
        Enter The Value Of Y : 2.0

For The No 3 Data:
        Enter The Name : 3
        Enter The Value Of X : 3.0
        Enter The Value Of Y : 4.0

For The No 4 Data:
        Enter The Name : 4
        Enter The Value Of X : 5.0
        Enter The Value Of Y : 7.0

For The No 5 Data:
        Enter The Name : 5
        Enter The Value Of X : 3.5
        Enter The Value Of Y : 5.0
```

```
For The No 6 Data:
        Enter The Name : 6
        Enter The Value Of X : 4.5
        Enter The Value Of Y : 5.0

For The No 7 Data:
        Enter The Name : 7
        Enter The Value Of X : 3.5
        Enter The Value Of Y : 4.5


How Many Cluster You wanna Create : 2

For The Centroid No 1 :
        Enter The X : 1.5
        Enter The Y : 2.0

For The Centroid No 2 :
        Enter The X : 5.0
        Enter The Y : 7.0


Enter The Higest Number Of Iteration: 5
```

Output:

```
-----------------------------------------------------------------------
        1.12    0.00    2.50    6.10    3.61    4.24    3.20
-----------------------------------------------------------------------
-----------------------------------------------------------------------
        7.21    6.10    3.61    0.00    2.50    2.06    2.92
-----------------------------------------------------------------------

The New Centroid For K1
        x: 1.8333333333333333
        y: 2.3333333333333335
The New Centroid For For K2
        x: 4.125
        y: 5.375
-----------------------------------------------------------------------
        1.57    0.47    2.03    5.64    3.14    3.77    2.73
-----------------------------------------------------------------------
-----------------------------------------------------------------------
        5.38    4.28    1.78    1.85    0.73    0.53    1.08
-----------------------------------------------------------------------
```



```
The New Centroid For K1
        x: 1.25
        y: 1.5
The New Centroid For For K2
        x: 3.9
        y: 5.1

So the data set are going to :
        1 is go to the : K1
        2 is go to the : K1
        3 is go to the : K2
        4 is go to the : K2
        5 is go to the : K2
        6 is go to the : K2
        7 is go to the : K2
BUILD SUCCESSFUL (total time: 1 minute 9 seconds)
```

Experiment No: 02

Experiment Name: K-Nearest Neighbor (KNN) Algorithm.

Algorithm:

Step 1: Start.

Step 2: Select the number K of the neighbors.

Step 3: For each point in the test data, calculate the distance between test data and each row of train data with the help of any method namely Euclidean, Manhattan distance etc.

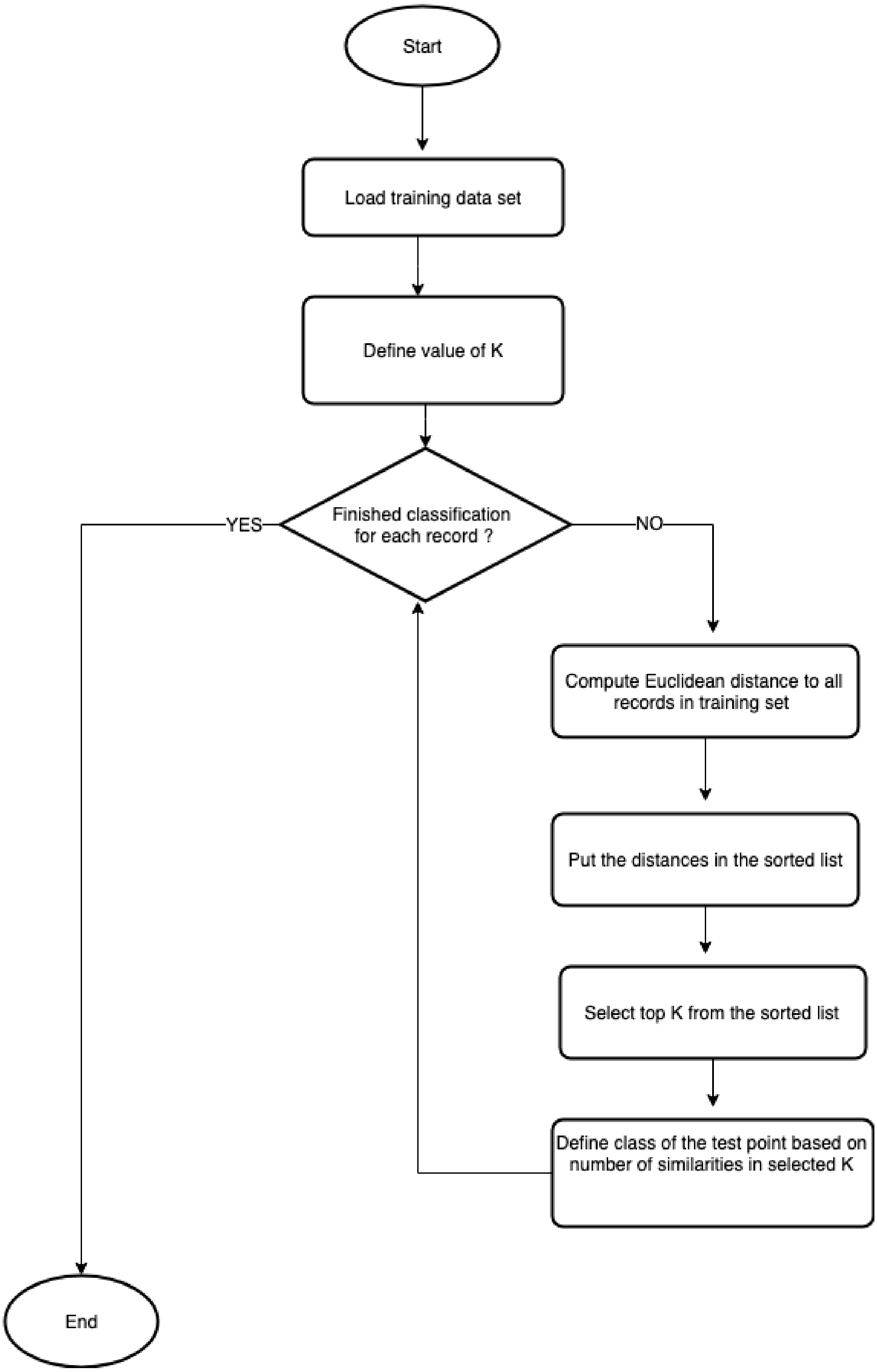
Step 4: Now based on the distance (all) value, sort them into ascending order.

Step 5: It will choose the top K rows from the sorted array or data.

Step 6: It will assign a class to the test point based on most frequent class of these rows.

Step 7: Stop.

Flow Chart:


Start
Load training data set
Define value of K
Finished classification for each record ?
YES
NO
Compute Euclidean distance to all records in training set
Put the distances in the sorted list
Select top K from the sorted list
Define class of the test point based on number of similarities in selected K
End

Source Code:

```
package KNNalgo;
```

```
import java.util.Arrays;
import java.util.Scanner;


public class KNN implements Comparable<KNN>{
    double parameter1, parameter2, distance;
    String label;
    public KNN(double parameter1, double parameter2, String label, double
distance) {
        this.parameter1 = parameter1;
        this.parameter2 = parameter2;
        this.label = label;
        this.distance = distance;
    }
    @Override
    public int compareTo(KNN t) {
        if(this.distance>t.distance){
            return 1;
        }else{
            return -1;
        }
    }
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("----- KNN Algo With Two Attribute Data Point -----
\n\n");
```

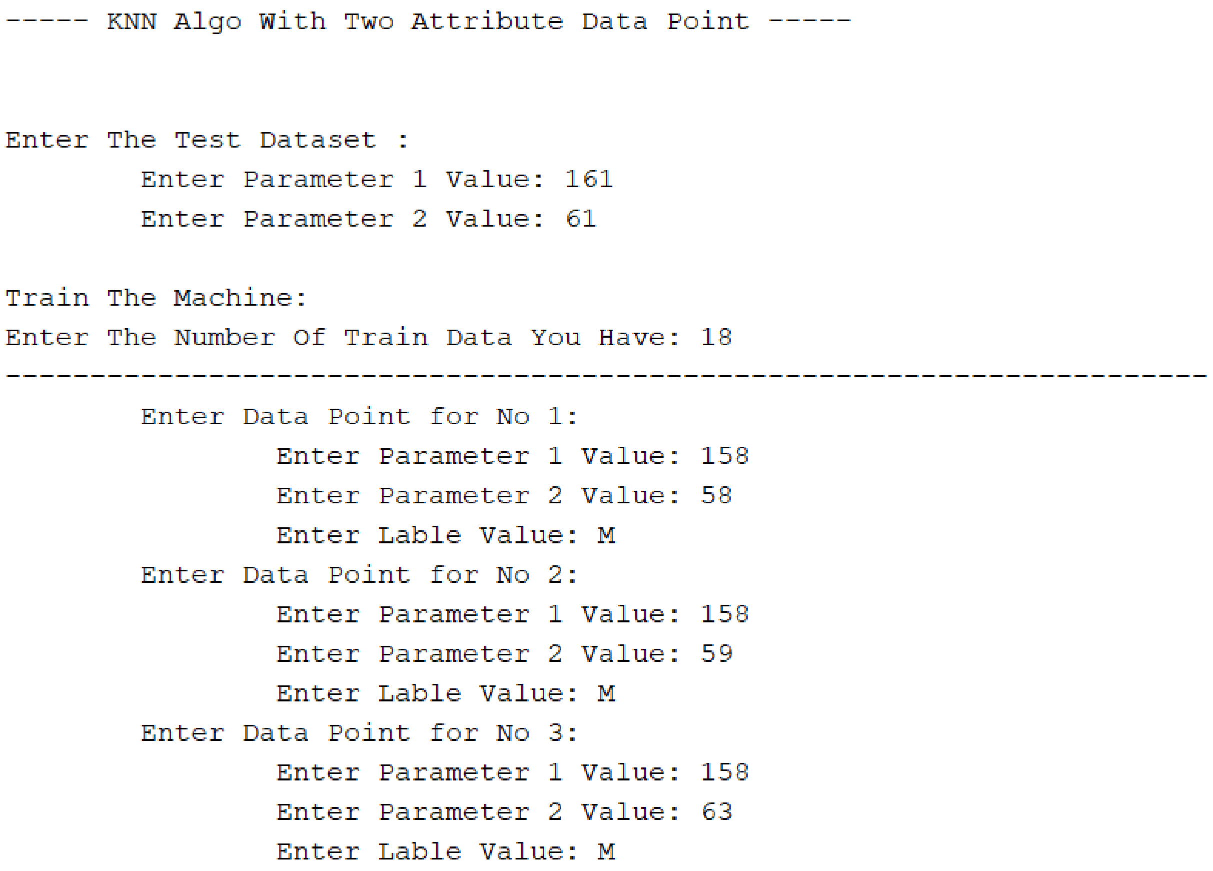
```
//Input test data.
System.out.println("Enter The Test Dataset : ");
System.out.print("\tEnter Parameter 1 Value: ");
double testDataParameter1 = input.nextDouble();
System.out.print("\tEnter Parameter 2 Value: ");
double testDataParameter2 = input.nextDouble();
//Input train data
System.out.println("\nTrain The Machine: ");
System.out.print("Enter The Number Of Train Data You Have: ");
int numberOfTrainData = input.nextInt();

KNN[] arrayOfOb = new KNN[numberOfTrainData];
System.out.println("------------------------------------------------------------------");
for (int i = 0; i < numberOfTrainData; i++) {
    System.out.println("\tEnter Data Point for No "+(i+1)+": ");
    System.out.print("\t\tEnter Parameter 1 Value: ");
    double trainDataParameter1 = input.nextDouble();
    System.out.print("\t\tEnter Parameter 2 Value: ");
    double trainDataParameter2 = input.nextDouble();
    System.out.print("\t\tEnter Lable Value: ");
    String trainDataLabel = input.next();
    //Distance calculation;
    double diffParameter1 = Math.abs(trainDataParameter1-testDataParameter1);
```

```
            double diffParameter2 = Math.abs(trainDataParameter2-
testDataParameter2);
            double distance = Math.sqrt(Math.pow(diffParameter1, 2) +
Math.pow(diffParameter2, 2));


            //Initialization objects by constructor.
            arrayOfOb[i] = new KNN(trainDataParameter1, trainDataParameter2,
trainDataLabel, distance);
        }
        //Sort the array based on distance.
        Arrays.sort(arrayOfOb);
        System.out.println("------------------------------------------------------------
-------\n");
        System.out.print("How Many Neighbour You Want: ");
        int numberOfDataSetWantToCheck = input.nextInt();


        //Printing Table(That much we want).
        System.out.println("\nThe Table: ");
        System.out.println("........................................................................\n");
        System.out.println("Sl No\tAttribute1\tAttribute2\tlabel\tDistance");
        System.out.println("------------------------------------------------------------
-------\n");
        for (int i = 0; i < numberOfDataSetWantToCheck; i++) {
            System.out.print(i+1);
            System.out.print("\t  "+arrayOfOb[i].parameter1);
            System.out.print("\t\t "+arrayOfOb[i].parameter2);
```

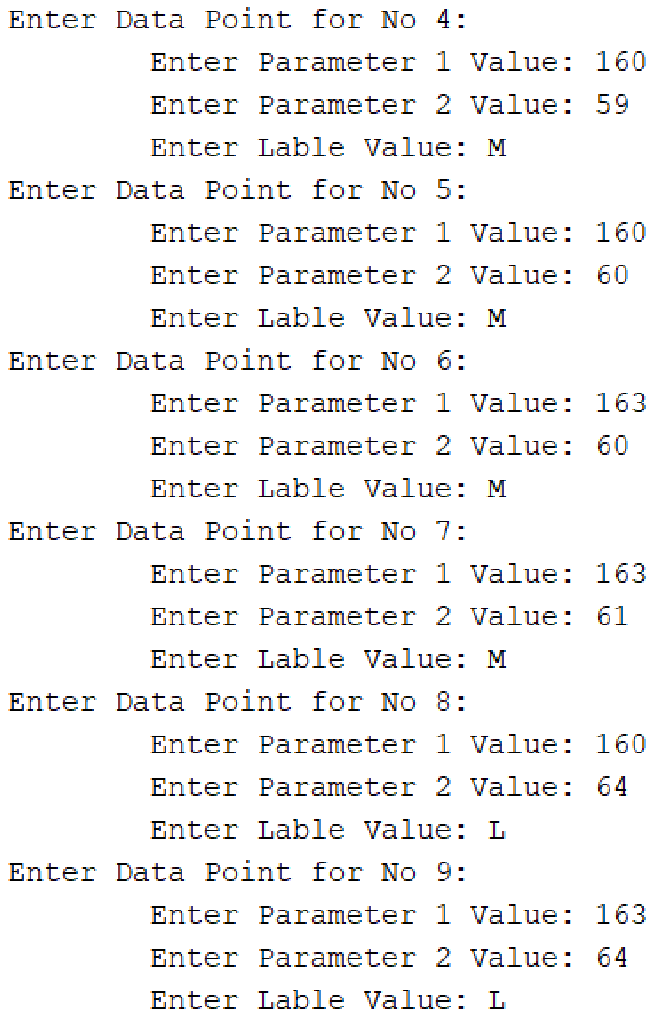
```
    System.out.print("\t\t"+arrayOfOb[i].label);

    System.out.printf("\t %.2f",arrayOfOb[i].distance);

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

}

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


//For two types of label

String firstTypeOfLabel = arrayOfOb[0].label;

String secondTypeOfLabel = null;

int countFirstTypeOfLabel = 0;

int countSecondTypeOfLabel = 0;

for (int i = 0; i < numberOfDataSetWantToCheck; i++) {

    if(firstTypeOfLabel == arrayOfOb[i].label){

        countSecondTypeOfLabel++;

    }

    else{

        secondTypeOfLabel = arrayOfOb[i].label;

        countFirstTypeOfLabel++;

    }

}

String resultOfLabel;

if(countFirstTypeOfLabel>=countSecondTypeOfLabel){

    resultOfLabel = firstTypeOfLabel;

} else{
```

```
            resultOfLabel = secondTypeOfLabel;
        }
        System.out.println("First Type of label is : "+firstTypeOfLabel);
        System.out.println("Second Type of label is : "+secondTypeOfLabel);
        System.out.println("First Type Of Label appearde : "+countFirstTypeOfLabel);
        System.out.println("Second Type Of Label appearde : "+countSecondTypeOfLabel);
        System.out.println("\nHere,\n\tWe can see the desirable label for our input is : "+resultOfLabel);
    }
}
```

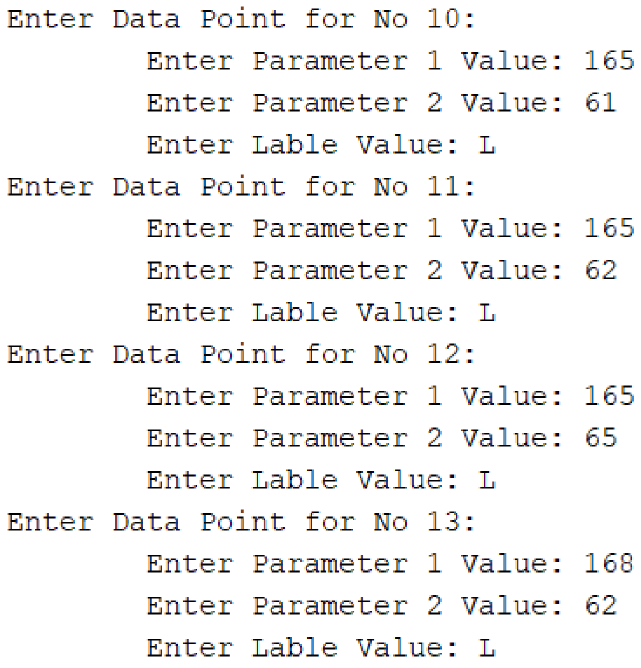
Input:



```
----- KNN Algo With Two Attribute Data Point -----


Enter The Test Dataset :
        Enter Parameter 1 Value: 161
        Enter Parameter 2 Value: 61

Train The Machine:
Enter The Number Of Train Data You Have: 18
----------------------------------------------------------------------
        Enter Data Point for No 1:
                Enter Parameter 1 Value: 158
                Enter Parameter 2 Value: 58
                Enter Lable Value: M
        Enter Data Point for No 2:
                Enter Parameter 1 Value: 158
                Enter Parameter 2 Value: 59
                Enter Lable Value: M
        Enter Data Point for No 3:
                Enter Parameter 1 Value: 158
                Enter Parameter 2 Value: 63
                Enter Lable Value: M
```



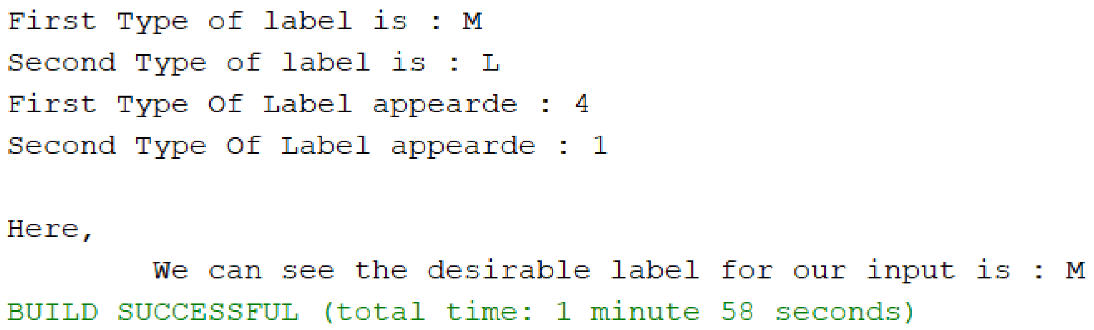
```
Enter Data Point for No 4:
        Enter Parameter 1 Value: 160
        Enter Parameter 2 Value: 59
        Enter Lable Value: M
Enter Data Point for No 5:
        Enter Parameter 1 Value: 160
        Enter Parameter 2 Value: 60
        Enter Lable Value: M
Enter Data Point for No 6:
        Enter Parameter 1 Value: 163
        Enter Parameter 2 Value: 60
        Enter Lable Value: M
Enter Data Point for No 7:
        Enter Parameter 1 Value: 163
        Enter Parameter 2 Value: 61
        Enter Lable Value: M
Enter Data Point for No 8:
        Enter Parameter 1 Value: 160
        Enter Parameter 2 Value: 64
        Enter Lable Value: L
Enter Data Point for No 9:
        Enter Parameter 1 Value: 163
        Enter Parameter 2 Value: 64
        Enter Lable Value: L
```



```
Enter Data Point for No 10:
        Enter Parameter 1 Value: 165
        Enter Parameter 2 Value: 61
        Enter Lable Value: L
Enter Data Point for No 11:
        Enter Parameter 1 Value: 165
        Enter Parameter 2 Value: 62
        Enter Lable Value: L
Enter Data Point for No 12:
        Enter Parameter 1 Value: 165
        Enter Parameter 2 Value: 65
        Enter Lable Value: L
Enter Data Point for No 13:
        Enter Parameter 1 Value: 168
        Enter Parameter 2 Value: 62
        Enter Lable Value: L
        Enter Data Point for No 14:
                Enter Parameter 1 Value: 168
                Enter Parameter 2 Value: 63
                Enter Lable Value: L
        Enter Data Point for No 15:
                Enter Parameter 1 Value: 168
                Enter Parameter 2 Value: 66
                Enter Lable Value: L

        Enter Data Point for No 16:
                Enter Parameter 1 Value: 170
                Enter Parameter 2 Value: 63
                Enter Lable Value: L
        Enter Data Point for No 17:
                Enter Parameter 1 Value: 170
                Enter Parameter 2 Value: 64
                Enter Lable Value: L
        Enter Data Point for No 18:
                Enter Parameter 1 Value: 170
                Enter Parameter 2 Value: 68
                Enter Lable Value: L

----------------------------------------------------------------

How Many Neighbour You Want: 5
```

Output:

```
The Table:
..................................................................

Sl No    Attribute1       Attribute2       label    Distance
------------------------------------------------------------------

1          160.0          60.0             M          1.41
------------------------------------------------------------------
2          163.0          61.0             M          2.00
------------------------------------------------------------------
3          163.0          60.0             M          2.24
------------------------------------------------------------------
4          160.0          59.0             M          2.24
------------------------------------------------------------------
5          160.0          64.0             L          3.16
------------------------------------------------------------------
..................................................................
```



```
First Type of label is : M
Second Type of label is : L
First Type Of Label appearde : 4
Second Type Of Label appearde : 1

Here,
        We can see the desirable label for our input is : M
BUILD SUCCESSFUL (total time: 1 minute 58 seconds)
```