```java
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.*;
import java.util.stream.Collectors;
import java.util.stream.Stream;

public class Main {

    static double correctAnswers = 0;
    static int option;

    public static void main(String[] args) {

        int k = 30;

        List<String> trainingStrings = readFile("trainingFile.txt");
        List<Attribute> trainingList = new ArrayList<>();

        trainingStrings.forEach(s -> trainingList.add(new Attribute(s)));
        Scanner scanner = new Scanner(System.in);

        System.out.println("==========================");
        System.out.println("== Klasyfikator k-NN  ==");
        System.out.println("==========================");
        System.out.println("\n==========================");
        System.out.print("Podaj parametr K:");
        k = scanner.nextInt();
        System.out.println("==========================");
        scanner.nextLine();
        System.out.println("\n==========================");
        System.out.println("Wczytaj z pliku ----> 1");
        System.out.println("Podaj ręcznie   ----> 2");
        System.out.println("==========================");
        System.out.print("Opcja: ");
        option = scanner.nextInt();

        scanner.nextLine();

        switch (option) {
            case 1 -> {
                List<String> testStrings = readFile("testFile.txt");
                List<Attribute> testList = new ArrayList<>();
                testStrings.forEach(s -> testList.add(new Attribute(s)));
                doKnnAlg(k, trainingList, testList);
            }
            case 2 -> {


                System.out.println("\n==========================");
                System.out.println("Podaj swój wektor [x;x;x;x]");
                System.out.println("Koniec programu ----> exit ");
                System.out.println("==========================");
                Scanner scanner1 = new Scanner(System.in);
                while (true) {
                    List<Attribute> testInput = new ArrayList<>();
                    System.out.print("Wektor: ");
                    String line = scanner1.nextLine();
                    if (line.equals("exit")) break;
                    testInput.add(new Attribute(line));
                    doKnnAlg(k, trainingList, testInput);
                    System.out.println("\n==========================");

                }

            }

        }
    }

    private static void doKnnAlg(int k, List<Attribute> trainingList, List<Attribute> testList) {
```

```
73          for (Attribute testAtt : testList) {
74
75              List<Distance> distances = new ArrayList<>();
76
77              for (Attribute treningAtt : trainingList) {
78                  distances.add(new Distance(testAtt, treningAtt));
79              }
80              sort(distances);
81              Map<String, Integer> hashMap = new LinkedHashMap<>();
82
83              for (int i = 0; i < k; i++) {
84                  String attributeName = distances.get(i).getTraning().getName();
85                  if (!hashMap.containsKey(attributeName)) {
86                      hashMap.put(attributeName, 1);
87                  } else {
88                      int value = hashMap.get(attributeName) + 1;
89                      hashMap.replace(attributeName, value);
90                  }
91              }
92              Map.Entry<String, Integer> maxEntry = null;
93              for (Map.Entry<String, Integer> entry : hashMap.entrySet()) {
94                  if (maxEntry == null || entry.getValue().compareTo(maxEntry.getValue()) > 0) {
95                      maxEntry = entry;
96                  }
97              }
98              assert maxEntry != null;
99              if (option == 2) {
100                 hashMap.forEach((key, value) -> System.out.println(key + " " + value));
101             } else if (testAtt.getName().equals(maxEntry.getKey())) {
102                 correctAnswers++;
103             }
104             //hashMap.forEach((key, value) -> System.out.println(key + " " + value));
105         }
106         if (option == 1) {
107             System.out.println("Accuracy: " + correctAnswers / testList.size() * 100 + "%");
108         }
109
110     }
111
112     public static List<String> readFile(String fileName) {
113         List<String> result = null;
114         try (Stream<String> lines = Files.lines(Paths.get(fileName))) {
115             result = lines.collect(Collectors.toList());
116         } catch (IOException e) {
117             e.printStackTrace();
118         }
119         return result;
120     }
121
122     public static void sort(List<Distance> distances) {
123         distances.sort((o1, o2) -> {
124             if (o1.getDistance() < o2.getDistance()) {
125                 return -1;
126             } else if (o1.getDistance().equals(o2.getDistance())) {
127                 return 0;
128             } else {
129                 return 1;
130             }
131         });
132     }
133
134
135 }
136
```

```java
public class Distance {

    private final Attribute test;
    private final Attribute traning;
    private double distance;

    public Distance(Attribute test, Attribute traning) {
        this.test = test;
        this.traning = traning;
        calcDistance();
    }

    public Attribute getTest() {
        return test;
    }

    public Attribute getTraning() {
        return traning;
    }

    public Double getDistance() {
        return distance;
    }

    public void calcDistance() {

        for (int i = 0; i < test.getCoords().size(); i++) {
            distance += Math.pow(test.get(i) - traning.get(i), 2);
        }
        this.distance = Math.sqrt(distance);

    }

    @Override
    public String toString() {
        return distance+" " + test + " " + traning;
    }
}

```

```java
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;

public class Attribute {

    private List<Double> coords;
    private String name;

    public Attribute(String line) {
        String[] splitedLine = line.split(";");
        List<String> lineAsList = new ArrayList<>(Arrays.asList(splitedLine).subList(0, splitedLine.length - 1));
        this.name = splitedLine[splitedLine.length - 1];
        coords = lineAsList.stream().map(Double::parseDouble).collect(Collectors.toList());

    }

    public List<Double> getCoords() {
        return coords;
    }

    public String getName() {
        return name;
    }


    public Double get(int i) {
        return coords.get(i);
    }


    @Override
    public String toString() {
        return coords.toString() + " " + name;
    }
}

```