k-Nearest Neighbour exercise session

In this exercise session, you will read an external file with Iris flowers and create an internal database in Java as it was done in **previous exercise session**. A new file contains list of 150 observations of *iris* flowers from 3 different species – *iris-setosa, iris-versicolor* and *iris-virginica*. There are 4 measurements of given flowers: *sepal length*, *sepal width*, *petal length* and *petal width*, all in the same unit of centimetres. In this session you need to implement **k-Nearest Neighbour method** to define specie of iris flower based on dimensional measurements.

Exercise instructions:

1. Use the code developed in **Java exercise session VI** to create an internal database FlowerList (ArrayList of Flower objects) from *csv file*.
2. Divide the extracted database into 2 sets – learning set (120 measurements -40 of each type) and testing set (30 measurements)
3. Use the learning set to develop **kNN** algorithm to define a specie of an *iris* flower based on dimensional parameters:
   - Calculate distance to a set of measurements.
   - Locate distances in increasing order.
4. Test a developed **kNN** algorithm with a testing set.

Core structure of the code:

```
import java.util.*;
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;


public class kNNmethod {


      public static void main(String[] args){

        String dataFile = "iris_learn.csv";
        BufferedReader br = null;
        String line = "";
        String SplitBy = ",";

        ArrayList<Flower> flowerList = new ArrayList<Flower>();

        try {
        // use code from Java exercise VI to create internal database of iris flowers
             } catch (FileNotFoundException e) {
                 e.printStackTrace();
             } catch (IOException e) {
                 e.printStackTrace();
             }

        // a test value
        double[] query = {5.2, 3.3, 2.5, 0.4};

        // create object Result that contains distance to a specific flower and flower's type
        ArrayList<Result> resultList = new ArrayList<Result>();

        for(Flower flower : flowerList){
            double dist = 0.0;
```

```
            // calculate variable dist - square of Euclidean distance from query measurements
            to flower's dimensional parameters

            double distance = Math.sqrt(dist);

            resultList.add(new Result(distance, flower.type));
        }

        Collections.sort(resultList, new DistanceComparator());

        for(int x = 0; x < resultList.size(); x++){
            System.out.println(resultList.get(x).type + "…" + resultList.get(x).distance);
        }
    }
}
```

```
import java.util.Comparator;

public class DistanceComparator implements Comparator<Result> {
        @Override
        public int compare(Result a, Result b) {
           return a.distance < b.distance ? -1 : a.distance == b.distance ? 0 : 1;
        }
    }
```