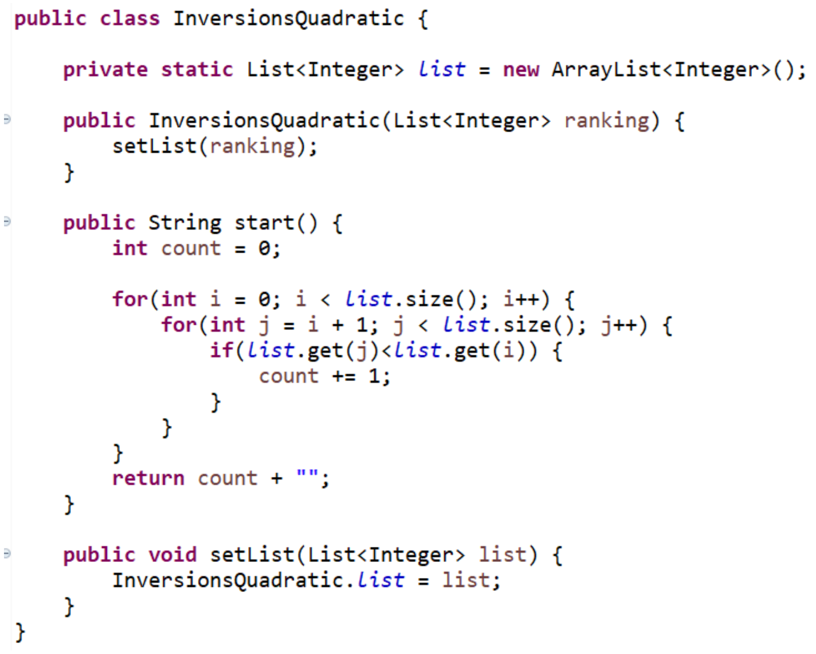
Algorithmics	Student information	Date	Number of session
	UO:275688	02/03/21	3
	Surname: Fernández Esparta		
	Name: Mikel		

Activity 1. [Counting inversions]

For this activity, we are given three clases: InversionsTimes, Inversions, and InversionsQuadratic. The last two are yet to be implemented, in which we have to obtain the number of inversions for a given list or ranking. An inversion (I,j) occurs when in a list, the value for i is lesser than j (i < j), given that both values are different from one another. In order to solve this problem, we need to use divide and conquer given the complexities O(n^2) and O(nlogn).

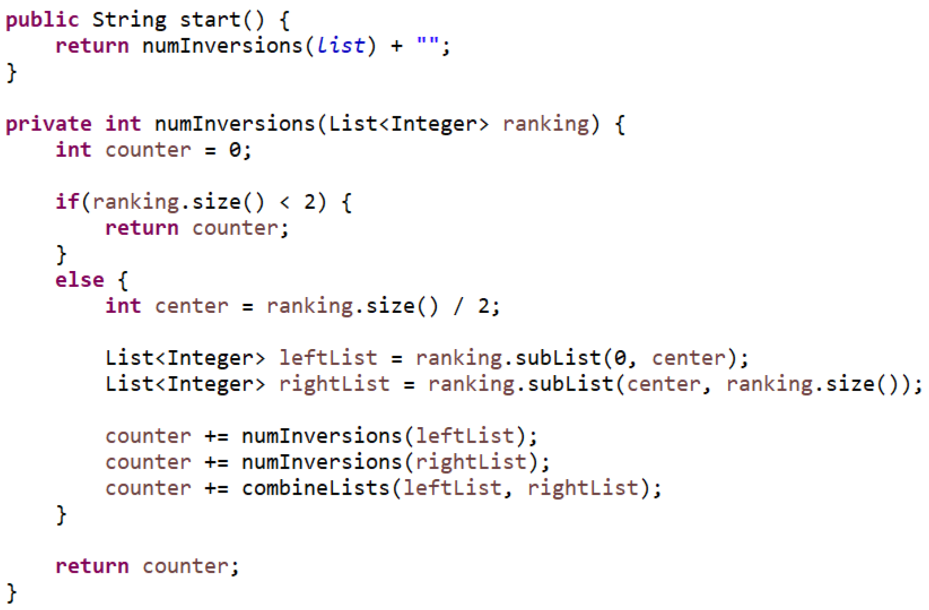
Once implemented, we will run the InversionsTimes to obtain the measurements of time, which are going to be stored in a table comparing both of them.

This is the implementation for the InversionsQuadratic class, which contains two for loops that makes the complexity O(n^2):

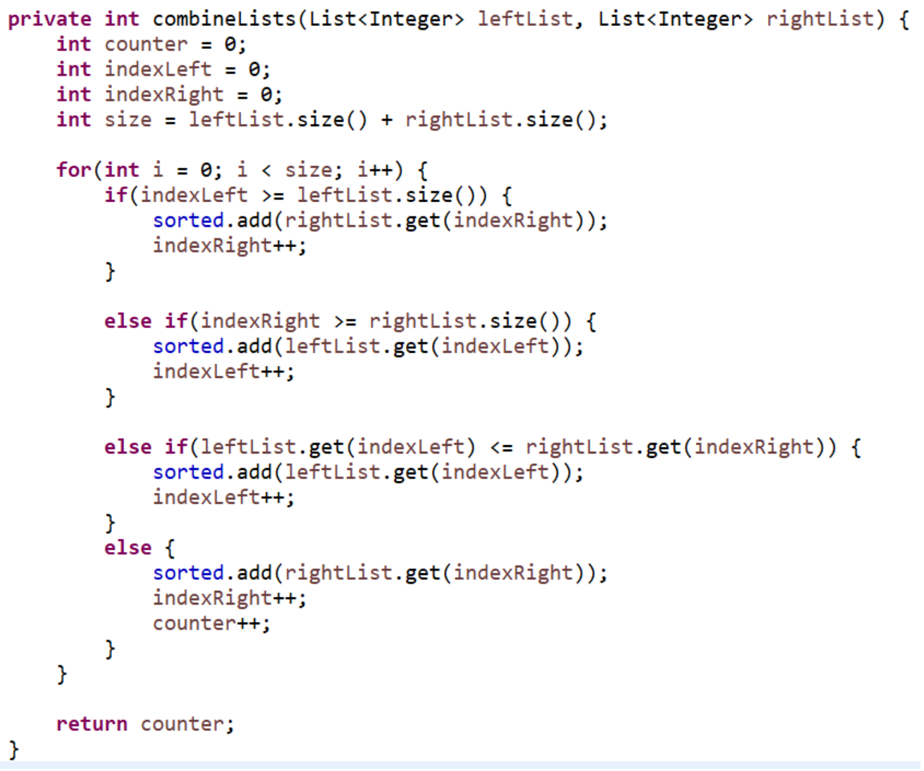


```java
public class InversionsQuadratic {

    private static List<Integer> list = new ArrayList<Integer>();

    public InversionsQuadratic(List<Integer> ranking) {
        setList(ranking);
    }

    public String start() {
        int count = 0;

        for(int i = 0; i < list.size(); i++) {
            for(int j = i + 1; j < list.size(); j++) {
                if(list.get(j)<list.get(i)) {
                    count += 1;
                }
            }
        }
        return count + "";
    }

    public void setList(List<Integer> list) {
        InversionsQuadratic.list = list;
    }
}
```

And this, is the code for the O(nlogn) version, dividing the list in two halves and iterating them using recursion while also counting the number of inversions produced, in case the first value is greater than the following one.



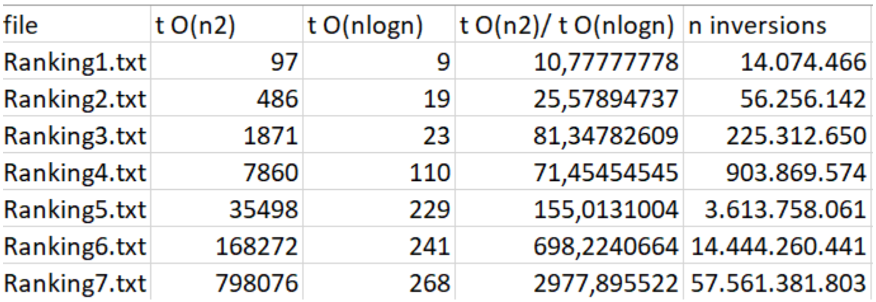
```
public String start() {
    return numInversions(list) + "";
}

private int numInversions(List<Integer> ranking) {
    int counter = 0;

    if(ranking.size() < 2) {
        return counter;
    }
    else {
        int center = ranking.size() / 2;

        List<Integer> leftList = ranking.subList(0, center);
        List<Integer> rightList = ranking.subList(center, ranking.size());

        counter += numInversions(leftList);
        counter += numInversions(rightList);
        counter += combineLists(leftList, rightList);
    }

    return counter;
}
```



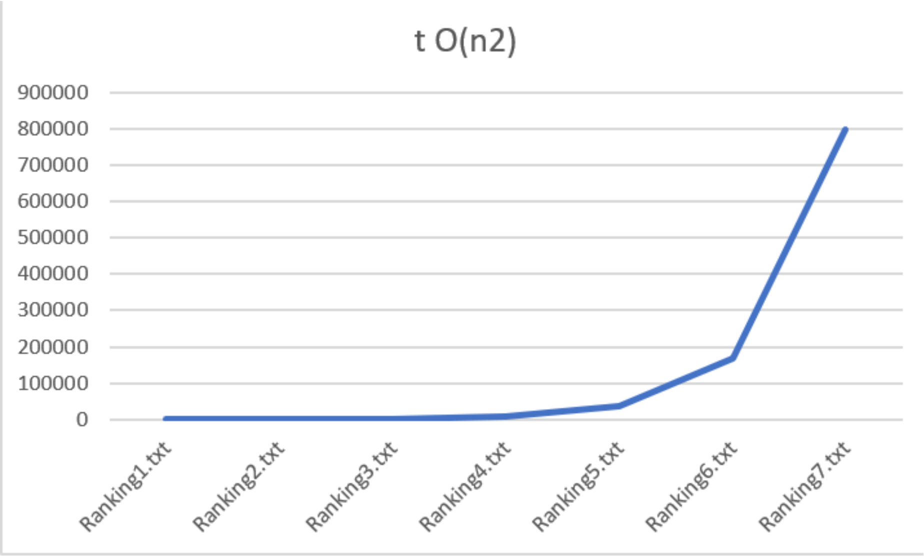
```
private int combineLists(List<Integer> leftList, List<Integer> rightList) {
    int counter = 0;
    int indexLeft = 0;
    int indexRight = 0;
    int size = leftList.size() + rightList.size();

    for(int i = 0; i < size; i++) {
        if(indexLeft >= leftList.size()) {
            sorted.add(rightList.get(indexRight));
            indexRight++;
        }

        else if(indexRight >= rightList.size()) {
            sorted.add(leftList.get(indexLeft));
            indexLeft++;
        }

        else if(leftList.get(indexLeft) <= rightList.get(indexRight)) {
            sorted.add(leftList.get(indexLeft));
            indexLeft++;
        }
        else {
            sorted.add(rightList.get(indexRight));
            indexRight++;
            counter++;
        }
    }

    return counter;
}
```

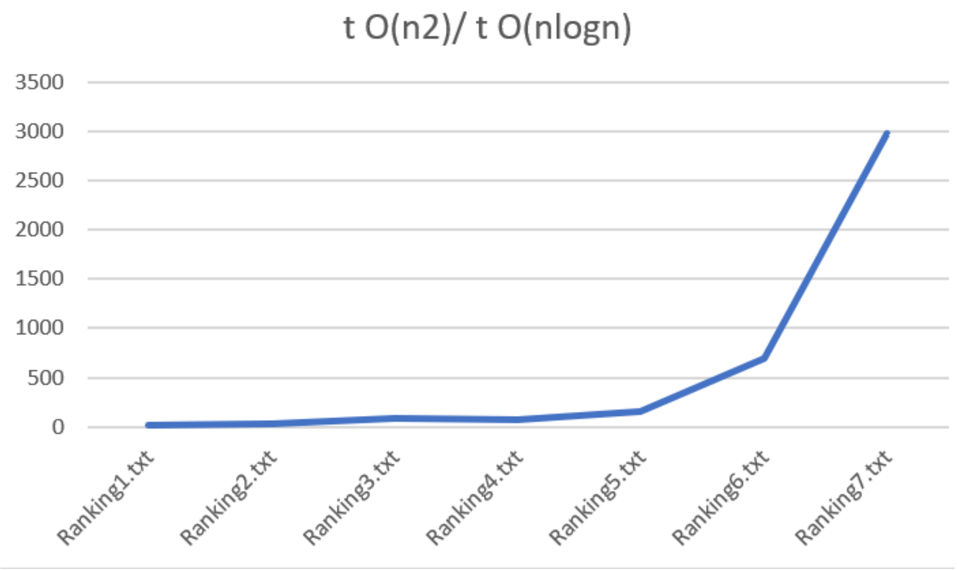
Algorithmics	Student information	Date	Number of session
	UO:275688	02/03/21	3
	Surname: Fernández Esparta		
	Name: Mikel		

These are the values obtained, that correspond to their time complexity:



file	t O(n2)	t O(nlogn)	t O(n2)/ t O(nlogn)	n inversions
Ranking1.txt	97	9	10,77777778	14.074.466
Ranking2.txt	486	19	25,57894737	56.256.142
Ranking3.txt	1871	23	81,34782609	225.312.650
Ranking4.txt	7860	110	71,45454545	903.869.574
Ranking5.txt	35498	229	155,0131004	3.613.758.061
Ranking6.txt	168272	241	698,2240664	14.444.260.441
Ranking7.txt	798076	268	2977,895522	57.561.381.803


t O(n2)
Ranking1.txt
Ranking2.txt
Ranking3.txt
Ranking4.txt
Ranking5.txt
Ranking6.txt
Ranking7.txt


t O(n2)/ t O(nlogn)
Ranking1.txt
Ranking2.txt
Ranking3.txt
Ranking4.txt
Ranking5.txt
Ranking6.txt
Ranking7.txt