**CMP3004-Formal Languages And Automata Theory Project Report**

***Asaf Emre Öztürk-1601462***

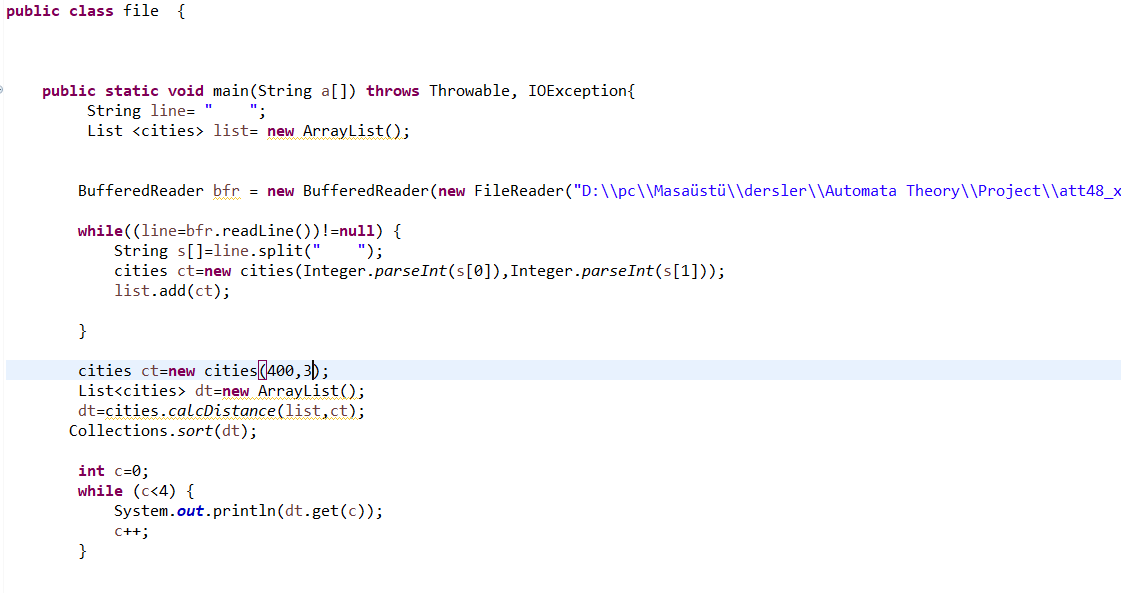
***Sıtkı Berk Onur-1613570***

***Deniz Bulmuş-1613528***

**KNN-Algorithm**

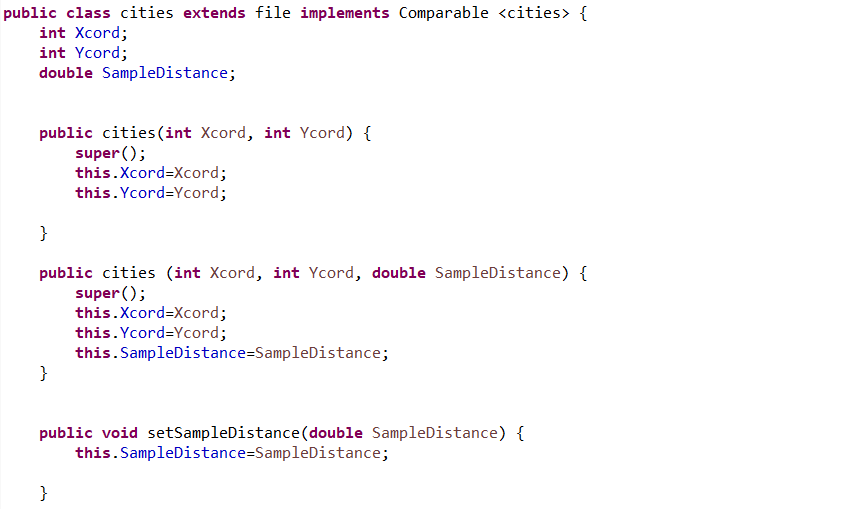
**File.java**

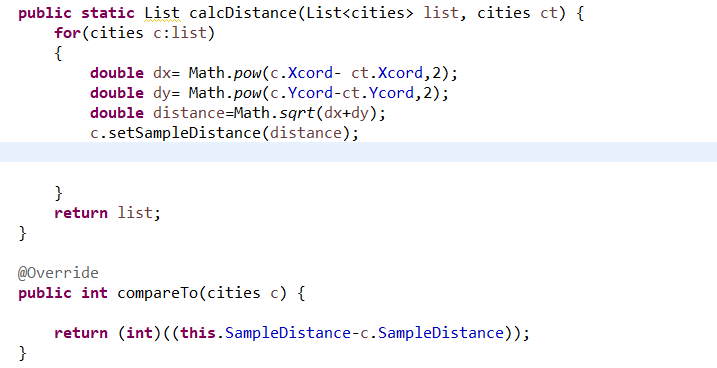
We have main function in file. List cities creates new arraylist. Bufferedreader buffers and reads txt file. While line isn’t null String s splits text, new cities parse x and y coordinates and add to list. After that takes reference point, creates new list to calculate distance between reference after the while loop shows the first given in number of nearest cities.



**Cities.java**

Citites class have Xcord, Ycord and sampledistance variable, there are two constructors with super function.Setsampledistance gets sampledistance. calcDistance list takes list cities and cities ct. İn for loop cities c traverse through list. dx and dy calculates distance between ct and c. And sets distance as sample distance. Lastly calcDistance function returns list. At last, we add comparable interface to compare possible neighbors of cities. CompareTo function takes reference distance and substract it from given city object.





**D&C Algorithm**

**splitNodes function**

this function splits nodes in to two equal- sets. It splits on the axis which has bigger span. It first finds the coordinate, then finds the median and just assign each point in the set according to the value of its coordinates. If else statement was used for splitting the x and y coordinates.

**mergeSols function**

This function is for merging the two solutions. It is searching for the variables for first and second tour (the cities) then it merges them together at minimum cost. It will try all possible edges in the first and in the second one. There will be calculation of delta in the code.

**solveTSP function**

This solves the TSP using a divide and conquer method. It splits the nodes in part1 and part2 and calculate the solutions for each part then merge the solutions into one whole.

Then in the main function will get the functions working and show the output to us.

**Greedy Solution**

**calcDistFromLine():** calculates the distance of each connected 2 nodes. it compares all combinations while remove used nodes.

**nearestNode():** takes reference for cities and adds to nearest node. Also removes node from list as a parameter.

**displayList():** Gets constant list of cities by their reference and diplays each elements from the list.

**getCoords():** Reads file and get x and y coordinates and fills a cities vectors with values. Returns these vectors later on.

**calcDist():** Uses 2 cities structures by reference then calculates their distance between them, then returns double value.