Firstly this code opens the input file and reads it. To store and classify the data, code firstly split the spaces and gets the information about station names, station coordinates, RGB values for the colors of metro lines, breakpoints and the metro line names that breakpoints are belong to. While reading, the code stores the information in arraylists by making use of indexing. A station's index in the station list is the same as its coordinates' index in the coordinate list. After classifying all the data, the code takes input from the user, firstly checks whether the stations are valid. If not, gives the error message and returns the program. Afterwards, the program finds the neighbors of the breakpoints so that the findWay method can be called to check there is a way or not. If not, which means method returns false, code gives the error message and returns the program. If there is a way, program prints the path to follow to reach the target station.

StdDraw Part: The code firstly draws the lines by iterating over the station list. For each metro line, each neighbor-pair stations are connected to each other by a line. (Since we have the coordinates of stations, it is just drawing a line from a point to the another.) Then the stations are drawn as white points. Afterwards, to reach the animation format, current station is drawn by colored big point and each iteration, it is updated. Additionally, visited stations are kept in a list and they are updated as well, so that we can draw them as colored but smaller points. Finally, station names are written.

The code consists of two methods: getLineOfStation and findWay. These methods are written to find the index of metro line of a given station and to find a way from a starting station to a target.

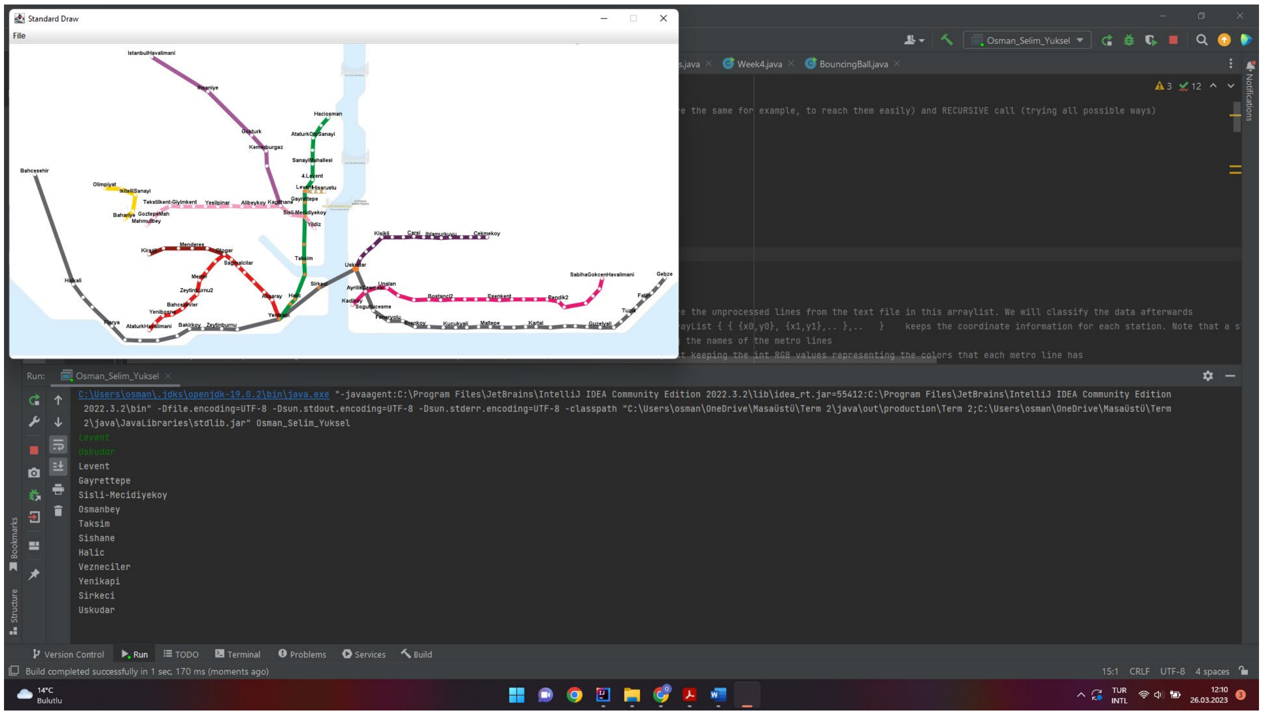
Method 1: getLineOfStation This method takes two parameters; an ArrayList<ArrayList<String>> stations2D (Master station arraylist) and a String. It returns an integer value that represents the index of the metro line that the given station stays on. The method uses a for loop to iterate over the master 2D station ArrayList. Then, for each metro line, it checks whether the station exists in that line. If the station exists in the metro line, the method stores the index of the metro line and sets the flag variable to true, then it breaks the loop. If the flag variable is still false after the loop, it returns -1 which means that the station is not valid. Otherwise, it returns the index of the metro line.

Method 2: findWay This method takes six parameters; ArrayList<ArrayList<String>> breakPointNeighborList, ArrayList<String> brkPoints, ArrayList<ArrayList<String>> stations2D, ArrayList<String> roadMap, String sourceStation, and String targetStation. It also returns a boolean value that indicates whether there is a way from the starting station to the target or not. If there is a way, it updates the roadmap list which contains the path to go. The method first finds the metro line of the source station using the getLineOfStation method. Then, it finds the indices of the source and target stations in their respective metro line lists. If the source station is equal to the target station, the method adds the target station to the roadmap and returns true.

If the source station is a breakpoint, the method checks the neighbors of the source station and calls the findWay method recursively for each unvisited neighbor. In other words, function checks all possibilities. if the way is found, it adds the source station to the roadmap and returns true.

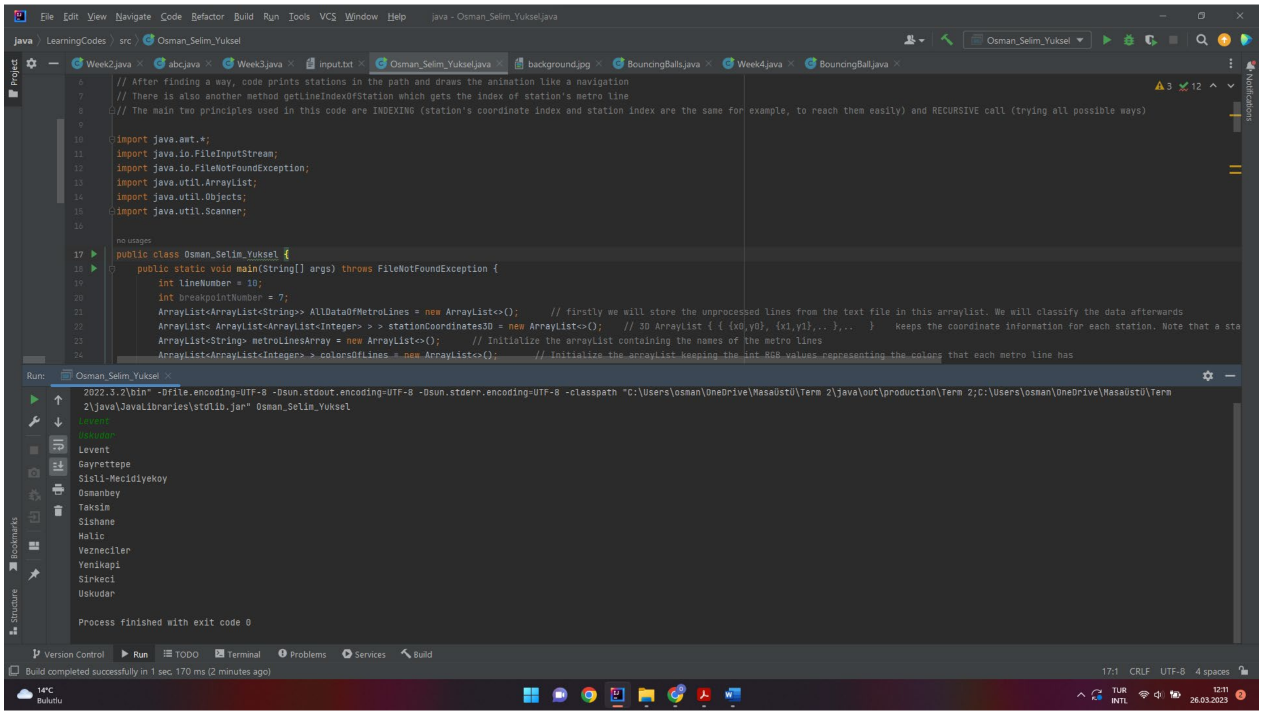
If the source station is not breakpoint, that means it can only have neighbor from the right or left, the method checks whether the neighbor station is visited before or not. If it is not visited, the method calls the findWay method recursively for the neighbor station. If the way is found, it adds the source station to the roadmap and returns true. If no way is found from the source station to the target station, the method returns false.

Testcase 1:



```
C:\Users\osman\.jdks\openjdk-19.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.3.2\lib\idea_rt.jar=55412:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition
 2022.3.2\bin" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath "C:\Users\osman\OneDrive\MasaÜstü\Term 2\java\out\production\Term 2;C:\Users\osman\OneDrive\MasaÜstü\Term
 2\java\JavaLibraries\stdlib.jar" Osman_Selim_Yuksel
Levent
Uskudar
Levent
Gayrettepe
Sisli-Mecidiyekoy
Osmanbey
Taksim
Sishane
Halic
Vezneciler
Yenikapi
Sirkeci
Uskudar
```

After closing StdDraw window:

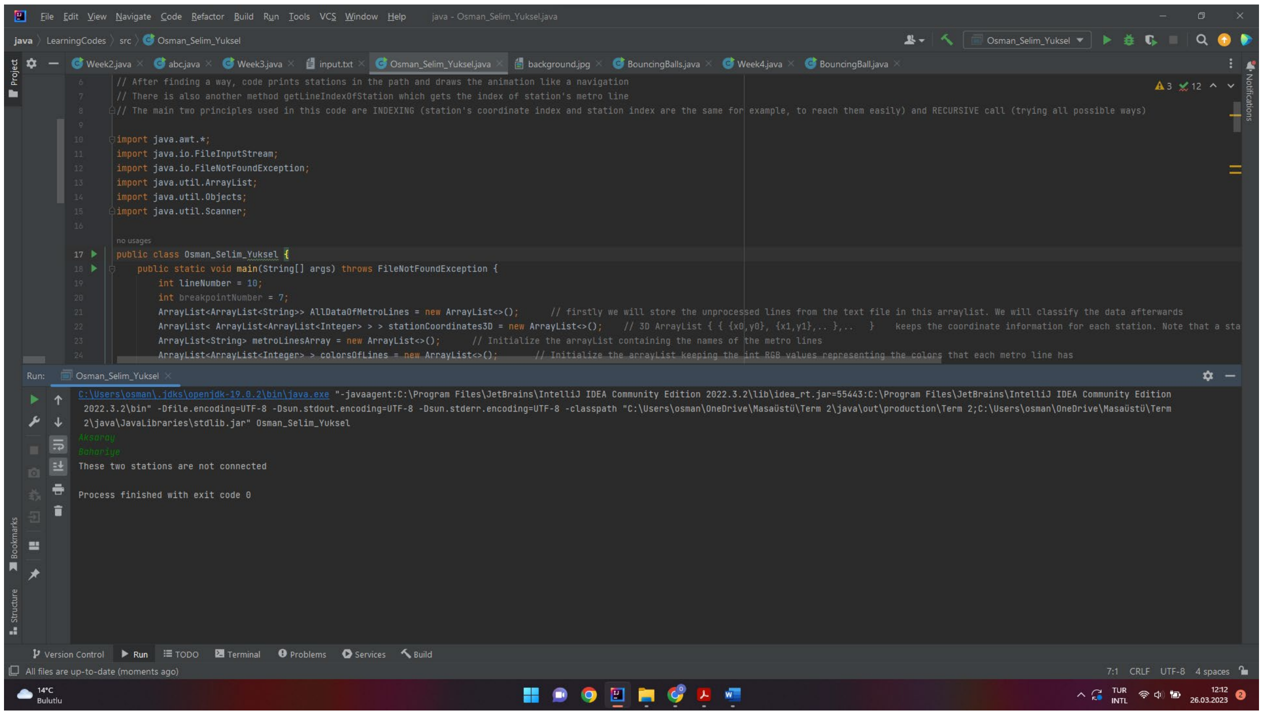


```java
// After finding a way, code prints stations in the path and draws the animation like a navigation
// There is also another method getLineIndexOfStation which gets the index of station's metro line
// The main two principles used in this code are INDEXING (station's coordinate index and station index are the same for example, to reach them easily) and RECURSIVE call (trying all possible ways)

import java.awt.*;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Objects;
import java.util.Scanner;

public class Osman_Selim_Yuksel {
    public static void main(String[] args) throws FileNotFoundException {
        int lineNumber = 10;
        int breakpointNumber = 7;
        ArrayList<ArrayList<String>> AllDataOfMetroLines = new ArrayList<>();      // firstly we will store the unprocessed lines from the text file in this arraylist. We will classify the data afterwards
        ArrayList< ArrayList<ArrayList<Integer> > > stationCoordinates3D = new ArrayList<>();    // 3D ArrayList { { {x0,y0}, {x1,y1},.. },..   }    keeps the coordinate information for each station. Note that a sta
        ArrayList<String> metroLinesArray = new ArrayList<>();      // Initialize the arrayList containing the names of the metro lines
        ArrayList<ArrayList<Integer> > colorsOfLines = new ArrayList<>();      // Initialize the arrayList keeping the int RGB values representing the colors that each metro line has
```

```
 2022.3.2\bin" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath "C:\Users\osman\OneDrive\MasaÜstü\Term 2\java\out\production\Term 2;C:\Users\osman\OneDrive\MasaÜstü\Term
 2\java\JavaLibraries\stdlib.jar" Osman_Selim_Yuksel
Levent
Uskudar
Levent
Gayrettepe
Sisli-Mecidiyekoy
Osmanbey
Taksim
Sishane
Halic
Vezneciler
Yenikapi
Sirkeci
Uskudar

Process finished with exit code 0
```

Testcase2:

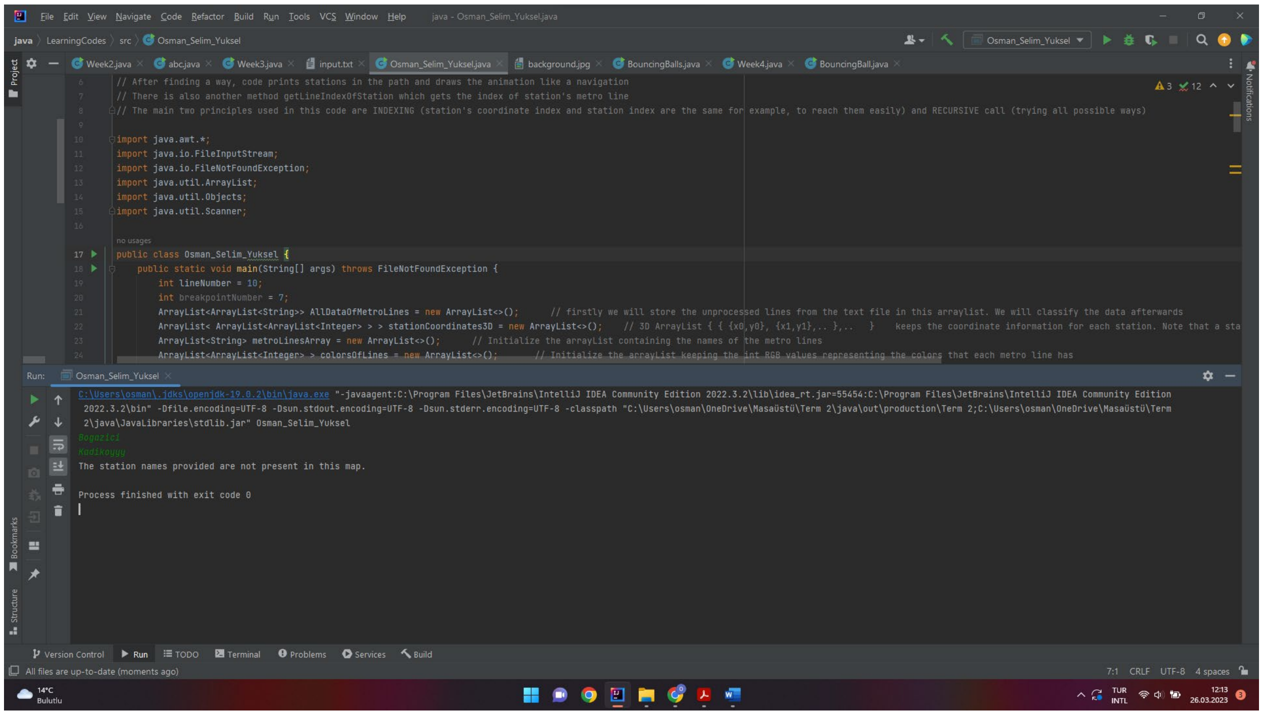
Canvas did not pop up



```java
// After finding a way, code prints stations in the path and draws the animation like a navigation
// There is also another method getLineIndexOfStation which gets the index of station's metro line
// The main two principles used in this code are INDEXING (station's coordinate index and station index are the same for example, to reach them easily) and RECURSIVE call (trying all possible ways)

import java.awt.*;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Objects;
import java.util.Scanner;

public class Osman_Selim_Yuksel {
    public static void main(String[] args) throws FileNotFoundException {
        int lineNumber = 10;
        int breakpointNumber = 7;
        ArrayList<ArrayList<String>> AllDataOfMetroLines = new ArrayList<>();      // firstly we will store the unprocessed lines from the text file in this arraylist. We will classify the data afterwards
        ArrayList< ArrayList<ArrayList<Integer> > > stationCoordinates3D = new ArrayList<>();    // 3D ArrayList { { {x0,y0}, {x1,y1},.. },..   }    keeps the coordinate information for each station. Note that a sta
        ArrayList<String> metroLinesArray = new ArrayList<>();      // Initialize the arrayList containing the names of the metro lines
        ArrayList<ArrayList<Integer> > colorsOfLines = new ArrayList<>();      // Initialize the arrayList keeping the int RGB values representing the colors that each metro line has
```

```
C:\Users\osman\.jdks\openjdk-19.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.3.2\lib\idea_rt.jar=55443:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition
 2022.3.2\bin" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath "C:\Users\osman\OneDrive\Masaüstü\Term 2\java\out\production\Term 2;C:\Users\osman\OneDrive\Masaüstü\Term
 2\java\JavaLibraries\stdlib.jar" Osman_Selim_Yuksel
Aksaray
Bahariye
These two stations are not connected

Process finished with exit code 0
```

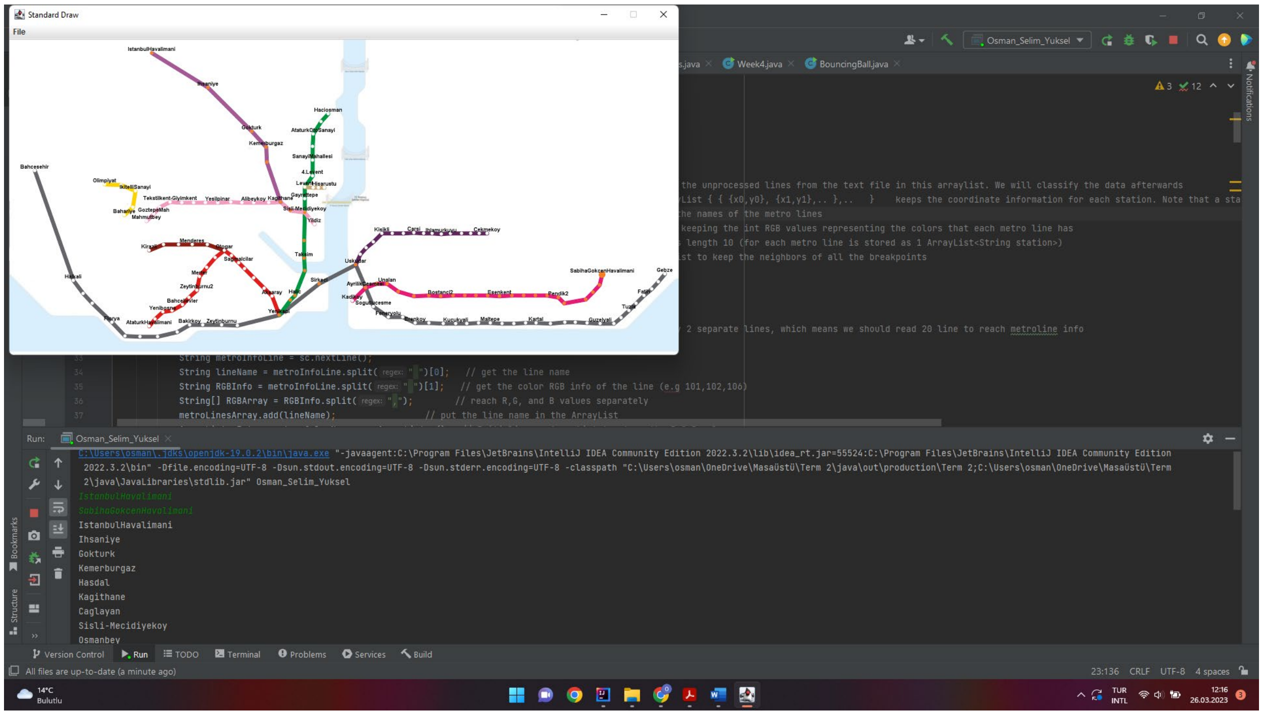
Testcase3:   Canvas did not pop up



```java
// After finding a way, code prints stations in the path and draws the animation like a navigation
// There is also another method getLineIndexOfStation which gets the index of station's metro line
// The main two principles used in this code are INDEXING (station's coordinate index and station index are the same for example, to reach them easily) and RECURSIVE call (trying all possible ways)

import java.awt.*;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Objects;
import java.util.Scanner;

public class Osman_Selim_Yuksel {
    public static void main(String[] args) throws FileNotFoundException {
        int lineNumber = 10;
        int breakpointNumber = 7;
        ArrayList<ArrayList<String>> AllDataOfMetroLines = new ArrayList<>();      // firstly we will store the unprocessed lines from the text file in this arraylist. We will classify the data afterwards
        ArrayList< ArrayList<ArrayList<Integer> > > stationCoordinates3D = new ArrayList<>();    // 3D ArrayList { { {x0,y0}, {x1,y1},.. },..   }    keeps the coordinate information for each station. Note that a sta
        ArrayList<String> metroLinesArray = new ArrayList<>();      // Initialize the arrayList containing the names of the metro lines
        ArrayList<ArrayList<Integer> > colorsOfLines = new ArrayList<>();      // Initialize the arrayList keeping the int RGB values representing the colors that each metro line has
```

```
C:\Users\osman\.jdks\openjdk-19.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.3.2\lib\idea_rt.jar=55454:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition
 2022.3.2\bin" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath "C:\Users\osman\OneDrive\Masaüstü\Term 2\java\out\production\Term 2;C:\Users\osman\OneDrive\Masaüstü\Term
 2\java\JavaLibraries\stdlib.jar" Osman_Selim_Yuksel
Bogazici
Kadikoyyy
The station names provided are not present in this map.

Process finished with exit code 0
```

Testcase4:

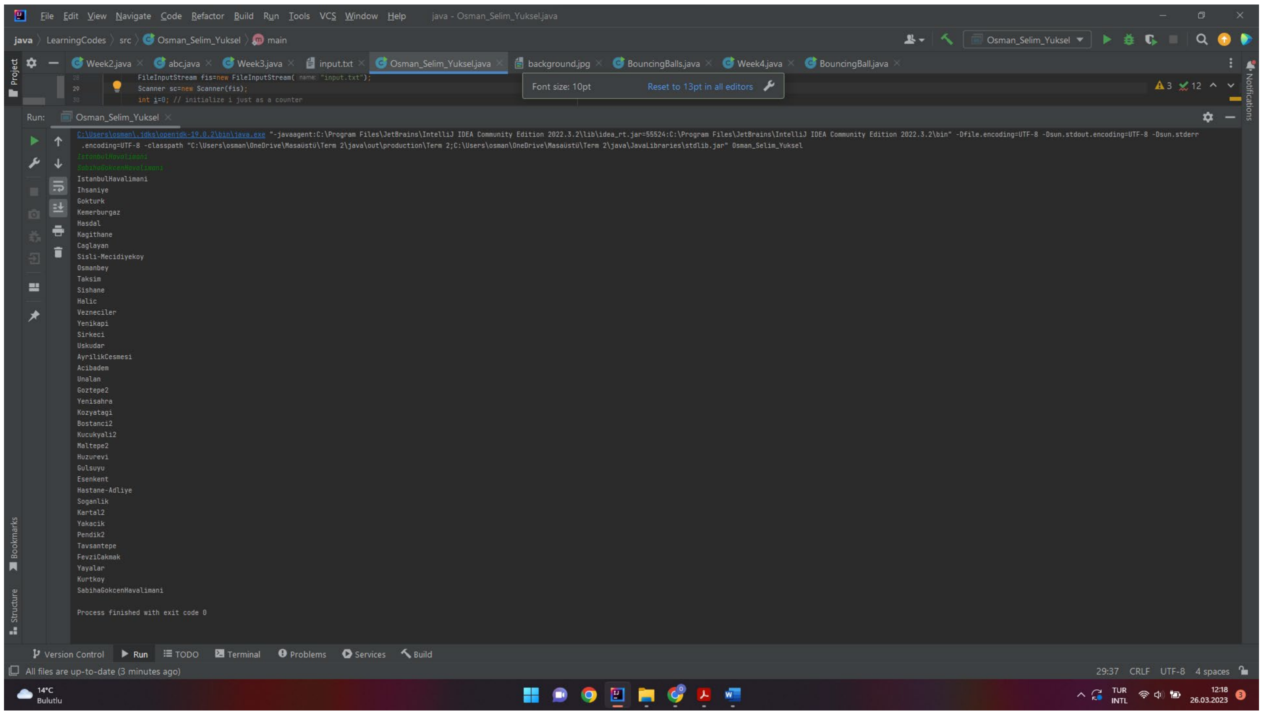


```
the unprocessed lines from the text file in this arraylist. We will classify the data afterwards
List { { {x0,y0}, {x1,y1},.. },..   }    keeps the coordinate information for each station. Note that a sta
he names of the metro lines
keeping the int RGB values representing the colors that each metro line has
 length 10 (for each metro line is stored as 1 ArrayList<String station>)
st to keep the neighbors of all the breakpoints

 2 separate lines, which means we should read 20 line to reach metroline info

String metroInfoLine = sc.nextLine();
String lineName = metroInfoLine.split( regex: " ")[0];   // get the line name
String RGBInfo = metroInfoLine.split( regex: " ")[1];   // get the color RGB info of the line (e.g 101,102,106)
String[] RGBArray = RGBInfo.split( regex: ",");         // reach R,G, and B values separately
metroLinesArray.add(lineName);                         // put the line name in the ArrayList
```

```
C:\Users\osman\.jdks\openjdk-19.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.3.2\lib\idea_rt.jar=55524:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition
 2022.3.2\bin" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath "C:\Users\osman\OneDrive\Masaüstü\Term 2\java\out\production\Term 2;C:\Users\osman\OneDrive\Masaüstü\Term
 2\java\JavaLibraries\stdlib.jar" Osman_Selim_Yuksel
IstanbulHavalimani
SabihaGokcenHavalimani
IstanbulHavalimani
Ihsaniye
Gokturk
Kemerburgaz
Hasdal
Kagithane
Caglayan
Sisli-Mecidiyekoy
Osmanbey
```

All files are up-to-date (a minute ago)   23:136 CRLF UTF-8 4 spaces

Console output:



```
FileInputStream fis=new FileInputStream( name: "input.txt");
Scanner sc=new Scanner(fis);
int i=0; // initialize i just as a counter
```

```
C:\Users\osman\.jdks\openjdk-19.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.3.2\lib\idea_rt.jar=55524:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.3.2\bin" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr
 .encoding=UTF-8 -classpath "C:\Users\osman\OneDrive\Masaüstü\Term 2\java\out\production\Term 2;C:\Users\osman\OneDrive\Masaüstü\Term 2\java\JavaLibraries\stdlib.jar" Osman_Selim_Yuksel
IstanbulHavalimani
SabihaGokcenHavalimani
IstanbulHavalimani
Ihsaniye
Gokturk
Kemerburgaz
Hasdal
Kagithane
Caglayan
Sisli-Mecidiyekoy
Osmanbey
Taksim
Sishane
Halic
Vezneciler
Yenikapi
Sirkeci
Uskudar
AyrilikCesmesi
Acibadem
Unalan
Goztepe2
Yenisahra
Kozyatagi
Bostanci2
Kucukyali2
Maltepe2
Huzurevi
Gulsuyu
Esenkent
Hastane-Adliye
Soganlik
Kartal2
Yakacik
Pendik2
Tavsantepe
FevziCakmak
Yayalar
Kurtkoy
SabihaGokcenHavalimani

Process finished with exit code 0
```

All files are up-to-date (3 minutes ago)   29:37 CRLF UTF-8 4 spaces