```
// This program is copyright VUW.
// You are granted permission to use it to construct your answer to a COMP102 assignment.
// You may not distribute it in any other way without permission.

/* Code for COMP102 - 2024T3, Assignment 3
 * Name:
 * Username:
 * ID:
 */

import ecs100.*;
import java.util.*;
import java.nio.file.*;
import java.io.*;
import java.awt.Color;

/**
 * WeatherReporter
 * Analyses weather data from files of weather-station measurements.
 *
 * The weather data files consist of a set of measurements from weather stations around
 * New Zealand at a series of date/time stamps.
 * For each date/time, the file has:
 *  A line with the date and time (four integers for day, month, year, and time)
 *   eg "24 01 2021 1900"  for 24 Jan 2021 at 19:00
 *  A line with the number of weather-stations for that date/time
 *  Followed by a line of data for each weather station:
 *   - name: one token, eg "Cape-Reinga"
 *   - (x, y) coordinates on the map: two numbers, eg   186 38
 *   - four numbers for temperature, dew-point, suface-pressure, and sea-level-pressure
 * Some of the data files (eg weather1-hot.txt, and weather1-cold.txt) have data for just one
date/time.
 * The weather-all.txt has data for lots of times. The date/times are all in order.
 * You should look at the files before trying to complete the methods below.
 *
 * Note, the data files were extracted from MetOffice weather data from 24-26 January 2021
 */

public class WeatherReporter{

    public static final double DIAM = 10;       // The diameter of the temperature circles.
    public static final double LEFT_TEXT = 10;  // The left of the date text
    public static final double TOP_TEXT = 50;   // The top of the date text

    /**   CORE
     * Plots the temperatures for one date/time from a file on a map of NZ
     * Asks for the name of the file and opens a Scanner
     * It is good design to call plotSnapshot, passing the Scanner as an argument.
     */
    public void plotTemperatures(){
        /*# YOUR CODE HERE */

        try {Scanner sc = new Scanner(Path.of(UIFileChooser.open("Choose data file")));
            this.plotSnapshot( sc);
            sc.close();
        }
        catch(IOException e){UI.printf("File failure %s\n",e);}

    }

    /**
     * CORE:
     *  Plot the temperatures for the next snapshot in the file by drawing
     *   a filled coloured circle (size DIAM) at each weather-station location.
     *  The colour of the circle should indicate the temperature.
     *
     *  The method should
     *   - read the date/time and draw the date/time at the top-left of the map.
     *   - read the number of stations, then
```

```
     *   - for each station,
     *     - read the name, coordinates, and data, and
     *     - plot the temperature for that station.
     *   (Hint: You will find the getTemperatureColor(...) method useful.)
     *
     *  COMPLETION:
     *  Also finds the highest and lowest temperatures at that time, and
     *  plots them with a larger circle.
     *  (Hint: If more than one station has the highest (or coolest) temperature,
     *         you only need to draw a larger circle for one of them.
     */
    public void plotSnapshot(Scanner sc){
        UI.drawImage("map-new-zealand.gif", 0, 0);
        /*# YOUR CODE HERE */

        int nextDate = sc.nextInt();
        int nextMonth = sc.nextInt();
        int nextYear = sc.nextInt();
        int nextTime = sc.nextInt();
        int nextStation = sc.nextInt();

        String nextTotal = nextDate +"/" + nextMonth+"/" + nextYear + " at " + nextTime;

        UI.drawString(nextTotal,LEFT_TEXT,TOP_TEXT);

        double highest = 0;
        double highestx = 0;
        double highesty = 0;
        Color highestc =null;

        double lowest = 1000;
        double lowestx = 0;
        double lowesty =0;
        Color lowestc = null;
        for(int i = 0; i<nextStation;i++){
            sc.next();
            double x = sc.nextDouble();
            double y = sc.nextDouble();

            double temp = sc.nextDouble();


            Color c = this.getTemperatureColor(temp);
            UI.setColor(c);
            if(temp > highest){
                highest = temp;
                highestx = x;
                highesty =y;
                highestc = c;}
            if(temp < lowest){
                lowest = temp;
                lowestx = x;
                lowesty = y;
                lowestc = c;}
            UI.fillOval(x-DIAM/2,y-DIAM/2,DIAM,DIAM);
            sc.nextLine();

        }
        UI.setColor(highestc);
        UI.fillOval(highestx,highesty,DIAM * 2,DIAM * 2);
        UI.setColor(lowestc);
        UI.fillOval(lowestx,lowesty,DIAM * 2,DIAM * 2);

    }
    /**   COMPLETION
     * Displays an animated view of the temperatures over all
     * the times in a weather data files, plotting the temperatures
     * for the first date/time, as in the core, pausing for half a second,
     * then plotting the temperatures for the second date/time, and
```

```
     * repeating until all the data in the file has been shown.
     *
     * (Hint, use the plotSnapshot(...) method that you used in the core)
     */
    public void animateTemperatures(){
        /*# YOUR CODE HERE */
        int progress = 0;

        try {Scanner scA = new Scanner( Path.of("weather-all.txt") );

            while(scA.hasNext()){
                this.plotSnapshot(scA);
                UI.sleep(500);


            }
            scA.close();

    }
    catch(IOException e){UI.printf("File failure %s\n",e);}



    }


/**   COMPLETION
 * Prints a table of all the weather data from a single station, one line for each day/time.
 * Asks for the name of the station.
 * Prints a header line
 * Then for each line of data for that station in the weather-all.txt file, it prints
 * a line with the date/time, temperature, dew-point, surface-pressure, and  sealevel-pressure
 * If there are no entries for that station, it will print a message saying "Station not found".
 * Hint, the \t in a String is the tab character, which helps to make the table line up.
 */
public void reportStation(){
String stationName = UI.askString("Name of a station: ");
UI.printf("Report for %s: \n", stationName);
UI.println("Date       \tTime \ttemp \tdew \tkPa \t\tsea kPa");

/*# YOUR CODE HERE */

try {Scanner scR = new Scanner( Path.of("weather-all.txt") );
            boolean found = false;
            while(scR.hasNext()){
                int nextDateR = scR.nextInt();
                int nextMonthR = scR.nextInt();
                int nextYearR = scR.nextInt();
                int nextTimeR = scR.nextInt();
                int nextStationR = scR.nextInt();

                for(int i = 0; i<nextStationR;i++){
                String nextName = scR.next();
                scR.nextDouble();
                scR.nextDouble();
                double nextTemp = scR.nextDouble();
                double nextDew = scR.nextDouble();
                double nextKpa = scR.nextDouble();
                double nextSea = scR.nextDouble();

                if(nextName.equalsIgnoreCase(stationName)){
                found = true;
                UI.printf("%d/%d/%d\t%d\t%s\t%s\t%s\t%s
\n",nextDateR,nextMonthR,nextYearR,nextTimeR,
String.valueOf(nextTemp),String.valueOf(nextDew),String.valueOf(nextKpa),String.valueOf(nextSea));
//nextTemp,nextDew,nextKpa,nextSea);

                }
```

```
                }
            }
            scR.close();
            if(!found){UI.println("Station not found");}



    }
    catch(IOException e){UI.printf("File failure %s\n",e);}




}

/**
 * Returns a color representing that temperature
 * The colors are increasingly blue below 15 degrees, and
 * increasingly red above 15 degrees.
 */
public Color getTemperatureColor(double temp){
double max = 37, min = -5, mid = (max+min)/2;
if (temp < min || temp > max){
return Color.white;
}
else if (temp <= mid){ //blue range: hues from .7 to .5
double tempFracOfRange = (temp-min)/(mid-min);
double hue = 0.7 -  tempFracOfRange*(0.7-0.5);
return Color.getHSBColor((float)hue, 1.0F, 1.0F);
}
else { //red range: .15 to 0.0
double tempFracOfRange = (temp-mid)/(max-mid);
double hue = 0.15 -  tempFracOfRange*(0.15-0.0);
return Color.getHSBColor((float)hue, 1.0F, 1.0F);
}
}

/**
 * Setup the interface with buttons
 */
public void setupGUI(){
UI.initialise();
UI.addButton("Plot temperature", this::plotTemperatures);
UI.addButton("Animate temperature", this::animateTemperatures);
UI.addButton("Report",  this::reportStation);
UI.addButton("Quit", UI::quit);
UI.setWindowSize(800,750);
UI.setFontSize(18);
}

/**
 *  Main: Create object and call setupGUI on it
 */
public static void main(String[] arguments){
WeatherReporter obj = new WeatherReporter();
obj.setupGUI();
}

}
```