Assignment

You will write a program to find the coldest day of the year and other interesting facts about the temperature and humidity in a day. To test your program, you will use the **nc_weather** data folder that has a folder for each year; you can download a .zip folder with these files by clicking here. In the **year** folder there is a CSV file for every day of the year; each file has the following information. For example, in the 2014 folder, we show parts of the file **weather-2014-01-08.csv**, the weather data from January 8, 2014.

	A	B	C	D	E	M	N	O
1	TimeEST	TemperatureF	Dew PointF	Humidity	Sea Leve	DirDegrees	DateUTC	
2	12:51 AM	16	6.1	65	3	240	1/8/2014 5:51	
3	1:51 AM	17.1	6.1	62	3	240	1/8/2014 6:51	
4	2:51 AM	19	5	54	3(	0	1/8/2014 7:51	
5	3:51 AM	18	6.1	60	3(	0	1/8/2014 8:51	
6	4:51 AM	18	6.1	60	3(	0	1/8/2014 9:51	
7	5:51 AM	17.1	6.1	62	3(	0	1/8/2014 10:51	
8	6:51 AM	16	7	68	3(	0	1/8/2014 11:51	
9	7:15 AM	15.8	6.8	68	3(	0	1/8/2014 12:15	
10	7:51 AM	15.1	8.1	74	3(	0	1/8/2014 12:51	
11	8:51 AM	21.9	9	58	3	180	1/8/2014 13:51	
12	9:51 AM	26.1	7	44	3(	0	1/8/2014 14:51	
13	10:51 AM	30	8.1	40	3(	230	1/8/2014 15:51	
14	11:51 AM	34	7	32	3(	210	1/8/2014 16:51	
15	12:51 PM	37	7	29	3(	170	1/8/2014 17:51	
16	1:51 PM	41	8.1	26	3(	180	1/8/2014 18:51	
17	2:51 PM	42.1	7	24	3(	220	1/8/2014 19:51	
18	3:51 PM	42.1	7	24	3(	220	1/8/2014 20:51	
19	4:51 PM	41	8.1	26	3(	190	1/8/2014 21:51	
20	5:51 PM	36	8.1	31	3(	150	1/8/2014 22:51	
21	6:51 PM	34	8.1	34	3(	160	1/8/2014 23:51	
22	7:51 PM	33.1	9	37	3(	160	1/9/2014 0:51	
23	8:51 PM	32	9	38	3(	180	1/9/2014 1:51	
24	9:51 PM	30	10.9	45	3(	0	1/9/2014 2:51	
25	10:51 PM	30.9	10	42	3(	0	1/9/2014 3:51	
26	11:51 PM	28	10	47	3(	0	1/9/2014 4:51	

You will write a program with several methods and tester methods to test each method you write. You should start with the methods from the lesson to find the hottest temperature in a day (and thus in a file) and the hottest temperature in many files and their tester methods. You can use these to write similar methods to find the coldest temperatures.

Specifically you should write the following methods.

1. Write a method named **coldestHourInFile** that has one parameter, a CSVParser named **parser**. This method returns the **CSVRecord** with the coldest temperature in the file and thus all the information about the coldest temperature, such as the hour of the coldest temperature. You should also write a void method named **testColdestHourInFile()** to test this method and print out information about that coldest temperature, such as the time of its occurrence.

NOTE: Sometimes there was not a valid reading at a specific hour, so the temperature field says -9999. You should ignore these bogus temperature values when calculating the lowest temperature.

2. Write the method **fileWithColdestTemperature** that has no parameters. This method should return a string that is the name of the file from selected files that has the coldest temperature. You should also write a void method named **testFileWithColdestTemperature()** to test this method. Note that after determining the filename, you could call the method **coldestHourInFile** to determine the coldest temperature on that day. When **fileWithColdestTemperature** runs and selects the files for January 1–3 in 2014, the method should print out

```
Coldest day was in file weather-2014-01-03.csv
Coldest temperature on that day was 21.9
All the Temperatures on the coldest day were:
2014-01-03 05:51:00: 41.0
2014-01-03 06:51:00: 39.0
2014-01-03 07:51:00: 35.1
2014-01-03 08:51:00: 30.9
2014-01-03 09:51:00: 28.0
2014-01-03 10:51:00: 25.0
2014-01-03 11:51:00: 24.1
2014-01-03 12:51:00: 23.0
2014-01-03 13:51:00: 25.0
2014-01-03 14:51:00: 26.1
2014-01-03 15:51:00: 28.0
2014-01-03 16:51:00: 30.0
2014-01-03 17:51:00: 30.9
2014-01-03 18:51:00: 33.1
2014-01-03 19:51:00: 33.1
2014-01-03 20:51:00: 33.1
2014-01-03 21:51:00: 30.9
2014-01-03 22:51:00: 28.9
2014-01-03 23:51:00: 28.9
2014-01-04 00:51:00: 26.1
2014-01-04 01:51:00: 24.1
2014-01-04 02:51:00: 24.1
2014-01-04 03:51:00: 23.0
2014-01-04 04:51:00: 21.9
```

3. Write a method named **lowestHumidityInFile** that has one parameter, a CSVParser named **parser**. This method returns the CSVRecord that has the lowest humidity. If there is a tie, then return the first such record that was found.

Note that sometimes there is not a number in the Humidity column but instead there is the string "N/A". This only happens very rarely. You should check to make sure the value you get is not "N/A" before converting it to a number.

Also note that the header for the time is either TimeEST or TimeEDT, depending on the time of year. You will instead use the DateUTC field at the right end of the data file to get both the date and time of a temperature reading.

You should also write a void method named **testLowestHumidityInFile()** to test this method that starts with these lines:

```
FileResource fr = new FileResource();
CSVParser parser = fr.getCSVParser();
CSVRecord csv = lowestHumidityInFile(parser);
```

and then prints the lowest humidity AND the time the lowest humidity occurred. For example, for the file **weather-2014-01-20.csv**, the output should be:

```
Lowest Humidity was 24 at 2014-01-20 19:51:00
```

NOTE: If you look at the data for January 20, 2014, you will note that the Humidity was also 24 at 3:51pm, but you are supposed to return the first such record that was found.

4. Write the method **lowestHumidityInManyFiles** that has no parameters. This method returns a CSVRecord that has the lowest humidity over all the files. If there is a tie, then return the first such record that was found. You should also write a void method named **testLowestHumidityInManyFiles()** to test this method and to print the lowest humidity AND the time the lowest humidity occurred. Be sure to test this method on two files so you can check if it is working correctly. If you run this program and select the files for January 19, 2014 and January 20, 2014, you should get

```
Lowest Humidity was 24 at 2014-01-20 19:51:00
```

5. Write the method **averageTemperatureInFile** that has one parameter, a CSVParser named **parser**. This method returns a double that represents the average temperature in the file. You should also write a void method named **testAverageTemperatureInFile()** to test this method. When this method runs and selects the file for January 20, 2014, the method should print out

```
Average temperature in file is 44.93333333333334
```

6. Write the method **averageTemperatureWithHighHumidityInFile** that has two parameters, a CSVParser named **parser** and an integer named **value**. This method returns a double that represents the average temperature of only those temperatures when the humidity was greater than or equal to value. You should also write a void method named **testAverageTemperatureWithHighHumidityInFile()** to test this method. When this method runs checking for humidity greater than or equal to 80 and selects the file for January 20, 2014, the method should print out

```
No temperatures with that humidity
```

If you run the method checking for humidity greater than or equal to 80 and select the file March 20, 2014, a wetter day, the method should print out

```
Average Temp when high Humidity is 41.78666666666667
```

Link to FAQ page for this course: http://www.dukelearntoprogram.com/course2/faq.php

ProgrammingExercise-ParsingWeatherDat…

Complete