# CS1083 Assignment #6 - Winter 2021

**Due: Friday, February 26th before 4:00pm (Atlantic), submitted in the Assignment 4 dropbox in Desire2Learn. (Read the submission instructions at the end of this document carefully).**

The purpose of this assignment is to get experience with parsing text files and handling I/O exceptions

**This assignment is to be done individually. If you have questions, direct them to a tutor/assistant during a help session in the "Faculty of Computer Science Student Success Centre" team or to your course instructor.**

---

## Weather Data

Write a Java program to parse climate data from Environment Canada and report maximum and minimum temperatures for a given month.

Two csv (comma-separated values) files for the weather station at the Fredericton Airport in January and February (partial data) 2021 are included in D2L. These files were obtained from http://climate.weather.gc.ca/index_e.html. If you want to see data for any month, click on Historical Data, put Fredericton as the location, and select Fredericton Airport. You can download csv files for any month from the Hourly Data page.

Each csv file starts with a header:

```
"Longitude (x)","Latitude (y)","Station Name","Climate ID","Date/Time (LST)","Year","Month","Day","Time (LST)","Temp (°C)","Temp Flag","Dew Point Temp (°C)","Dew Point Temp Flag","Rel Hum (%)","Rel Hum Flag","Precip. Amount (mm)","Precip. Amount Flag","Wind Dir (10s deg)","Wind Dir Flag","Wind Spd (km/h)","Wind Spd Flag","Visibility (km)","Visibility Flag","Stn Press (kPa)","Stn Press Flag","Hmdx","Hmdx Flag","Wind Chill","Wind Chill Flag","Weather"
```

The columns we are interested in are the Date/Time, and the temperature in degrees Celcius. Lines after the header contain weather data for each hour of the month. For example, for the first reading in January a temperature of -2.5 is recorded:

```
"-66.54","45.87","FREDERICTON INTL A","8101505","2021-01-01
00:00","2021","01","01","00:00","-2.5","","-8.0","","66","","0.0","","29","",
"21","","16.1","","101.94","","","","-9","","NA"
```

Here is the output of the program for the first file:

```
$ java TemperatureStats
en_climate_hourly_NB_8101505_01-2021_P1H.csv
Maximum: 2021-01-15 15:00: 3.1
Minimum: 2021-01-31 08:00: -19.3
```

Note that the file name is provided as a command line argument, and that the maximum and minimum temperatures are printed along with the date and time they occurred.

Data may be incomplete. Your program must handle two possibilities. First, a line from the csv file may be incomplete, such as in the file for February 2021, where the weather data ends on "2021-02-15 23:00", but the remaining hours to the end of the month appear in the file with the weather data missing. Here's how your program should handle this:

```
$ java TemperatureStats
en_climate_hourly_NB_8101505_02-2021_P1H.csv
invalid entry at 2021-02-16 00:00
invalid entry at 2021-02-16 01:00
…
invalid entry at 2021-02-28 22:00
invalid entry at 2021-02-28 23:00
Maximum: 2021-02-03 16:00: 2.6
Minimum: 2021-02-10 08:00: -23.0
```

Note that date and time for each line with missing data is identified (not all data is shown to keep the output short!).

The second way data may be incomplete is that a value may be missing, e.g. "" shows up in the temperature column, instead a value (such as "14.3").

In the following example, the first temperature in the January file was removed and replaced with "" (and saved in a file called `test.csv`), which is handled in the same way as the first issue by identifying the line with an invalid entry:

```
$ java TemperatureStats test.csv
invalid entry at 2021-01-01 00:00
```

```
Maximum: 2021-01-15 15:00: 3.1
Minimum: 2021-01-31 08:00: -19.3
```

Your program must be designed as follows, with two classes:

`TemperatureParser.java`:

- Constructor: passed the filename, it creates a scanner object, reads the header, and finds which column contains the "Date/Time (LST)" string and which column contains the "Temp (°C)" string. This will allow extraction of the date/time string and temperature values for each line. The scanner and the two column numbers are stored as instance variables.
- `parseLine()`: parses a line of a csv weather file, using the scanner object created by the constructor. It finds the date/time string and the temperature and stores them in instance variables. This method finds these values using the columns identified by the constructor.
- Accessor methods to retrieve the date/time string and the temperature, and determine whether there are any more lines in the file.
- You also may want to create private helper methods.

`TemperatureStats.java`, with a main method that does the following:

- Creates an instance of `TemperatureParser` using the filename passed in as a command line argument. If the filename is missing from the command line then the program terminates with the message `Usage: java TemperatureStats file.csv`
- Parses each line of the file using the `parseLine()` method (and the accessor methods) of `TemperatureParser` and records and prints the minimum and maximum temperatures, along with the associated date/time string.
- Handles the two kinds of exceptions described above: incomplete line or missing value (empty string instead), by printing a message identifying the line where the problem occurs (OK to assume that the date/time string will always be there), as in the examples above.
- Handles any IOExceptions (ie: incorrect file names) by printing an appropriate message and ending the program.

## Testing

**Test your program** with the following 6 cases:

- 1,2: The two csv files provided
- 3,4: Two files resulting from replacing the temperature in one of the weather files with an empty string in the first data line (i.e., not counting the header) and in another data line.
- 5: Running `TemperatureParser` without a command line argument

• 6: Running `TemperatureParser` with an incorrect file name

**Your electronic submission (submitted via Desire2Learn) will consist of <u>two files</u>:**

i. a written report. This should begin with a title page that includes: the course (CS 1083), your section (FR01B, FR02B, FR03B), the assignment number, your full name, and your UNB student number. The report must include:

   a. all the source code (`TemperatureParser` and `TemperatureStats` classes) for your solution (with Javadoc comments included)

   b. the sample input and output from running `TemperatureStats` (for all 6 cases).

   This written report should be prepared using a word processor; we recommend using Microsoft Word (i.e. create a .docx file for your report). Copy & paste your java source code & required output into the report document. Add appropriate headings for each part. Fix up the formatting where necessary, adjusting line breaks & page breaks to ensure that your document is easy to read. Use a monospaced font for your code to maintain proper indentation.) Once the report is complete and you've checked it all over, save the .docx file for your own records, and then **save a second copy in pdf format for submission**. (Note: Be sure to open that file in a pdf viewer to verify that the pdf was generated correctly.) **The SINGLE pdf file containing your report will be submitted to the appropriate assignment drop box on Desire2Learn**. (It is important that you submit a pdf file and NOT the original Word document. This pdf will allow the marker to write comments directly on your work to give you better feedback.)

   Note: name this report as follows: **YourName_CS1083_As6_Report.pdf**

ii. an archive file (**.zip**) that contains all your Java source code and the extra two input files (with the missing values) for this assignment. Make sure that your archive includes all .java files (in case the marker wishes to compile & run your code to test it). You should not include the report document or the .class files in your archive. This archive should be submitted as a **single file** to the appropriate drop box on Desire2Learn.

   Note: name this archive file as follows: **YourName_CS1083_As6_Archive.zip**