Assignment #1
CSCI 201 Fall 2024
6% of course grade

Title
Weather Conditions

Topics Covered
Java Classes
File I/O
Sorting Algorithms
Basic Java Topics

Introduction
Much of your computer programming experience is likely using C++. In this course, Java is the language we will primarily use. To help transition your knowledge from C++ to Java, you will be creating a program that displays the weather conditions in a specific city. This assignment requires that you parse a file containing a set of weather conditions. To ensure accurate parsing, you will then provide a command line interface to allow a user to query the parsed data.

Data Persistence
Many software projects need to store persistent data, and there are a few ways to do this. Databases, which we will cover later in the class, are primarily used for most web applications, but storing data in files is also very common for many types of applications (i.e. mobile, standalone, web). This assignment will require you to parse a file that contains weather conditions at several locations. The data in the file is going to be stored as a ***JavaScript Object Notation (JSON)*** file, formatted as follows:

```
{
  "data": [
    {
      "location": "Los Angeles",
      "temperature": 68,
      "temperatureMin": 54,
      "temperatureMax": 72,
      "humidity": 35,
      "pressureSeaLevel": 29.80,
      "visibility": 88.3,
      "windSpeed": 8.4,
      "windDirection": 270,
      "condition": "Sunny"
    }
   ]
}
```

There could be multiple items in the *data* array, but there will be at least one. In this example, there is only one object in the *data* array. A sample JSON file, **goodweather.json,** is provided in the D2L Brightspace folder. We recommend that you create your own test files that are longer, since the files we will use for testing will be different from the sample one provided.

Parsing JSON

JSON is a lightweight data-interchange format. In other words, it is a syntax that allows for easy storage and organization of data. It is commonly used to exchange information between client and server, and it is popular because of its language independence and human readability. JSON is built upon two basic data structures that you should already be familiar with: dictionaries (maps) and ordered lists. An object in JSON is represented by an unordered set of name/value pairs (i.e., a dictionary). Objects are contained by braces, { }, inside of which will list the object's attributes (with the syntax name : value), using a comma as the separator.

There are quite a few Java JSON parsing APIs out there. Unfortunately, the JDK does not have built-in support for JSON, but some of the more popular ones include GSON, Jackson, and JSON.simple.

GSON is known for its ease and flexibility in converting Java objects into JSON objects (and vice versa), and it is simple and straightforward to use. It was built by Google, and it has gone through many updates and bug fixes. You may want to start there. No matter which API you choose (and you are not limited to choosing from the ones mentioned in these instructions), you will need to download a JAR file and add it to your Eclipse project. Below are links to the JAR file download for the APIs mentioned above:

GSON source and latest release (2.11.0):
https://github.com/google/gson

GSON MVN Repository:
https://mvnrepository.com/artifact/com.google.code.gson/gson

Our recommended version is GSON 2.9.1, which we have fully tested.

There are a few alternatives to GSON.

Jackson:
https://github.com/FasterXML/jackson-core

JSON.simple:
https://github.com/fangyidong/json-simple

You will need to add the JAR file to your project and then add the JAR file to your Java Build Path in Eclipse to use the library in your code.

First, move the JAR file to the top directory of your Eclipse Project. In other words, if your project is named Assignment 1, put the JAR file into the Assignment 1 directory in your Eclipse workspace directory.

Next, perform the following steps in Eclipse:

1. Right-click on your Eclipse project.
2. Click “Refresh,” which should make your JAR file show in the Package Explorer.
3. Right-click on your Eclipse project again.
4. Choose “Properties”.
5. Select “Java Build Path”.
6. Click the “Libraries” tab.
7. Select the Classpath (do **not** select the Modulepath).
7. Click “Add JARs…”. (do **not** select Add External JARs).
8. Find the JAR in your project directory and add it.
9. Click “OK”.
10. Click “Apply and Close.”

Once you are done, the Java Build Path should look like in **Figure 1**.

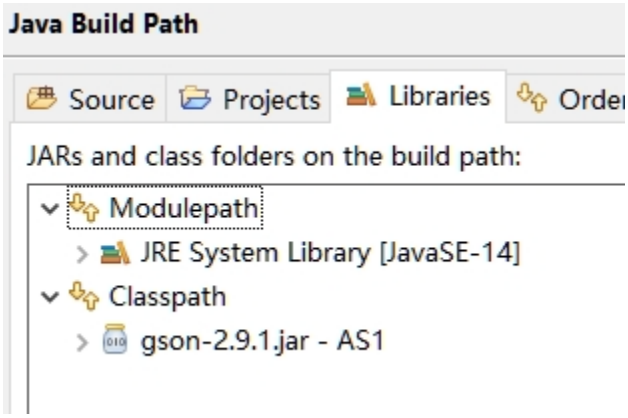

Figure 1. Java Build Path

The Eclipse Package Explorer may like **Figure 2**, when using GSON 2.9.1.

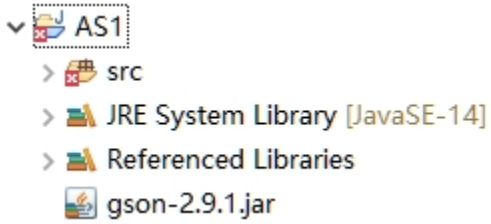

Figure 2. Package Explorer

Assignment

When your program first runs, you will need to prompt the user to enter the name of the JSON file that contains all the weather information.

Your program should validate the existence of the file and whether the file is formatted properly. All the following fields are required. You can assume the following data types for each:

`location – String`
`temperature – int` (measured in degrees Fahrenheit)
`temperatureMin – int` (measured in degrees Fahrenheit)

`temperatureMax – int` (measured in degrees Fahrenheit)
`humidity – int` (measured as a percentage)
`pressureSeaLevel – float` (measured in Inch Hg)
`visibility – float` (measured in miles)
`windSpeed – float` (measured in miles/hour)
`windDirection – int` (measured in degrees)
`condition - String`

Important Note: Keys in each JSON Objects can appear in any order.

If there is any problem parsing the data in the weather file, your program should print out an error message that is as descriptive as possible. The program should then prompt the user to enter another weather file.
Here are some examples of errors in file parsing that you need to catch:

- File not found.
- Data cannot be converted to the proper type as shown above.
- There are too few parameters on one of the lines.
- Two locations have the same "location" field. (Case-insensitive)

Only once a properly formatted weather file is parsed, the program should display a menu as follows:

```
1) Display weather on all locations
2) Search for weather on a location
3) Add a new location
4) Remove a location
5) Sort locations
6) Exit

What would you like to do?
```

Users can enter menu commands by entering the number associated with the command, i.e., entering "1" to display all the locations. The program should discard the invalid option (e.g., 0 or 7 here) and ask the user to enter another input. ***All data entered should be case-insensitive, except for the file names .***

Additionally, when selecting a location to search (option 2 in the main menu), the program should display a sub-menu as follows:

```
1) temperature
2) high and low temperature
3) humidity
```

```
4) pressure
5) visibility
6) wind speed and direction
7) weather conditions
8) Return to main menu
```

Depending on the selection, you will show the temperature, daily low and high temperatures, humidity, pressure, visibility, wind speed and direction, or weather condition. You need to make sure that you handle any error and boundary conditions, such as a city that doesn't exist or entering something that is not valid for the information you want displayed. Your program should also be able to recognize the input "los angeles" when you have a location named "Los Angeles" and reject the location add operation; as it was mentioned previously, inputs are case-insensitive. Option 8 will re-display the original main menu.

The format, requirements, and hints regarding other options can be found in the following sample execution.

Sample Execution

Here is a sample execution of the program with the user input bolded (though the input will not be bolded when you run your program). The file goodweather.,json, provided with the assignment, contains the valid JSON data from the sample provided on page 1

The text inside brackets and with yellow backgrounds, [[XX]], are hints about the components of the program. They should not be displayed in your program output.

Usually, we won't strictly require the number of empty lines in your output to be the same as shown below, but we prefer that you follow the same look ad displayed in the sample output, including formatting and tabulation. IN GENERAL, ouutput like:

"5) Sort locations6) ExitWhat would you do"

all, on a single line, will lead to point deductions. Use `\n` properly.

```
[[Start of sample execution here.]]

[[File reading]]

What is the name of the weather file? noweather.json
The file noweather.json could not be found.

What is the name of the weather file? badweather1.json
The file badweather1.json is not formatted properly.

What is the name of the weather file? badweather2.json
The file badweather2.json is not formatted properly.

What is the name of the weather file? goodweather.json
The file has been properly read.

[[Menu display]]

            1) Display weather on all locations
            2) Search for weather on a location
            3) Add a new location
            4) Remove a location
            5) Sort locations
            6) Exit

[[Invalid input check]]

What would you like to do? 0
```

```
That is not a valid option.

What would you like to do? hello

That is not a valid option.

[[Option 1]]

What would you like to do? 1

Los Angeles
      temperature is 68 degrees Fahrenheit,
      low temperature is 54 degrees Fahrenheit,
      high temperature is 72 degrees Fahrenheit,
      humidity is 35%.
      pressure is 29.80 Inch Hg,
      visibility is 88.3 miles,
      wind speed is 8.4 miles/hour,
      wind direction is 270 degreed,
      weather can be described as sunny

New York
      temperature is 35 degrees Fahrenheit,
      low temperature is 23 degrees Fahrenheit,
      high temperature is 37 degrees Fahrenheit,
      humidity is 21%.
      pressure is 28.70 Inch Hg,
      visibility is 35.3 miles,
      wind speed is 7.4 miles/hour,
      wind direction is 165 degreed,
      weather can be described as snow

[[Option 2]]

           1) Display weather on all locations
           2) Search for weather on a location
           3) Add a new location
           4) Remove a location
           5) Sort locations
           6) Exit

What would you like to do? 2

[[Option 2 – location validation]]

What is the location you would like to search for? USC
```

```
USC could not be found.

What is the location you would like to search for? Los Angeles
[[los angeles should also work]]

[[Option 2 - submenu]]

I have information about the weather in Los Angeles.

           1) temperature
           2) high and low temperature
           3) humidity
           4) pressure
           5) visibility
           6) wind speed and direction
           7) weather conditions
           8) Return to main menu

What weather information do you want to know for Los Angeles? 1
The temperature in Los Angeles is 68 degrees Fahrenheit.

[[Menu omitted for the following sub-options 2-7 to save space]]

What weather information do you want to know for Los Angeles? 2
The high /low temperature in Los Angeles is 72 / 54 degrees Fahrenheit.

What weather information do you want to know for Los Angeles? 3
The humidity in Los Angeles is 35%.

What weather information do you want to know for Los Angeles? 4
The pressure in Los Angeles is 29.8 Inch Hg.

What weather information do you want to know for Los Angeles? 5
The visibility in Los Angeles is 88.3 miles.

What weather information do you want to know for Los Angeles? 6
The wind speed and direction in Los Angeles is 8.4 miles/hour and 270 degreed.

What weather information do you want to know for Los Angeles? 7
The weather in Los Angeles can be described as Sunny.

[[Option 2 - Return to main menu]]

           1) Temperature
```

```
            2) High and low temperature
            3) Humidity
            4) Pressure
            5) Visibility
            6) Wind speed and direction
            7) Weather conditions
            8) Return to main menu

What weather information do you want to know for Los Angeles? 8
```

[[Option 3]]

```
            1) Display weather on all locations
            2) Search for weather on a location
            3) Add a new location
            4) Remove a location
            5) Sort locations
            6) Exit

What would you like to do? 3
```

[[Option 3: Check the case-insensitive uniqueness of the location name.]]

```
What is the location you would like to add weather info? Los Angeles

There is already an entry for Los Angeles.

What is the location you would like to add weather info? Chicago
```

[[Option 3: recording the inputs. Don't forget about the input check.]]

```
What is the temperature?  40
What is the low temperature? 35
What is the high temperature? 45
What is the humidity? 30
What is the pressure? 30.01
What is the visibility? 17.8
What is the wind speed? 30.3
What is the wind direction? 110
What is the weather condition? windy
```

[[Option 3: saving the location. Your location should be able to be seen in option 1 now.]]

```
There is now a new entry for Chicago.
```

[[Option 4]]

```
           1) Display weather on all locations
           2) Search for weather on a location
           3) Add a new location
           4) Remove a location
           5) Sort locations
           6) Exit

What would you like to do? 4
```

[[Display a list of names of existing locations. Hard-coding this list will not work as locations are dynamic, affected by option 3 and this option.]]

```
     1) Los Angeles
     2) New York
     3) Chicago
```

[[Option 4 - removing]]

```
Which location would you like to remove? 2

New York is now removed.
```

[[If you type 4 here again, New York should disappear from your list. It should also disappear from option 1's list.]]

[[Option 5]]

```
           1) Display weather on all locations
           2) Search for weather on a location
           3) Add a new location
           4) Remove a location
           5) Sort locations
           6) Exit

What would you like to do? 5

           1) A to Z
           2) Z to A

How would you like to sort by? 1
```

[[Sort the locations. Two options. Although we do not require output here,

```
location order when calling options 1 and 4 should change accordingly.]]

Your locations are now sorted in alphabetical order (A-Z).

[[Option 6]]

            1) Display weather on all locations
            2) Search for weather on a location
            3) Add a new location
            4) Remove a location
            5) Sort locations
            6) Exit

What would you like to do? 6

[[Saving is not as easy as it looks like! You should record what you have done by
other options: added locations by option 3, removed locations by option 4, and
sorted locations by option 5.]]

            1) Yes
            2) No

Would you like to save your edits? 1

[[Save to the file users given at the beginning.]]

Your edits have been saved to goodweather.json
Thank you for using my program!


[[End of sample execution here.]]
```

Grading Criteria

How you go about implementing the solution is not specifically graded, but the output must match exactly what you see in the execution above. **The maximum number of points earned is 3**.

File I/O (0.5)

0.1 - The filename is read from the user, and the file is appropriately parsed
0.1 - If the file could not be found, an appropriate error message is displayed
0.1 - If the file cannot be parsed, an appropriate error message is displayed
0.1 - If the file has missing parameters, an appropriate error message is displayed
0.1 - The file is properly saved (overwriting the original) upon exiting the program

Reading Inputs (1.3)

0.3 - Invalid user input is handled properly
0.7 - Users can properly navigate the main menu options by using numerical inputs
0.3 - Users can properly navigate the sub menu options by using numerical inputs

Outputs (1.2)

0.2 - “Display weather on all locations” displays all locations from the file, and after adding or removing locations
0.2 - “Search for a weather on a location” displays the correct information for the search query
0.2 - “Add a new location” works properly
0.2 - “Remove a location” works properly
0.4 - Each of the two methods for sorting locations works properly