

Assignment #1
CSCI 201 Fall 2018
2.5% of course grade

Title

Calendar Database

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 grading tool with simple functionalities.

This assignment requires that you parse a file containing a set of users with a set of calendar events for each. 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 data about academic courses.

The data in the file is going to be stored as a JavaScript Object Notation (JSON) file. 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. 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 currently have built-in support for JSON, but some of the more popular ones include GSON,

Jackson, and JSON.simple. Here are a couple of blogs discussing the merits of choosing one of these APIs over another if you are interested:

<http://blog.takipi.com/the-ultimate-json-library-json-simple-vs-gson-vs-jackson-vs-json/>
<http://javarevisited.blogspot.com/2016/09/top-5-json-library-in-java-JEE.html>

In short, it seems GSON is known for its ease and flexibility of converting Java objects into JSON objects (and vice versa), and it is simple and straightforward to use. You may want to start there. No matter which API you choose (and you are not limited to choosing from the ones mentioned on 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: <https://mvnrepository.com/artifact/com.google.code.gson/gson/2.8.0> (under 'Files', click download JAR)

Jackson: <https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-core/2.2.3> (under 'Files', click download JAR)

JSON.simple: <https://mvnrepository.com/artifact/com.googlecode.json-simple/json-simple/1.1.1> (under 'Files', click download BUNDLE)

You will need to add the JAR file to your Java Build Path in Eclipse in order to use the library. First, move the JAR file to the top directory of your Eclipse project. In other words, if your project is named Assignment1, put the JAR file into the Assignment1 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. Click "Add JARs".
8. Find the JAR in your project directory and add it.
9. Click "Okay".

Assignment

When your program first runs, you will need to prompt the user to enter the name of a JSON file. After reading the filename from the user, you will attempt to parse the file. If there are errors in the file, you will display an error message with as much description as you can as to the problem. This will most likely be provided to you by your chosen parser, so don't worry about trying to

make it more descriptive than that. After displaying the error message, you will loop back to allow the user to enter another JSON file.

If the file is a valid JSON file, you will parse it and store it in variables in your program. I recommend that you create classes to store all of the data, and use good object-oriented principles. After all the data is stored in your Java objects, you will give the user a menu to allow them to query the data.

Here is a sample JSON file. A JSON file that can be used for testing purposes is posted on the course website. I recommend that you create your own test files, since the file we will use for testing will be different from the sample one provided.

```
{
  "Users": [
    {
      "Name": {
        "Fname": "Tommy",
        "Lname": "Trojan"
      },
      "Events": [
        {
          "Title": "Kristy's Birthday Party",
          "Time": "2:00 pm",
          "Date": {
            "Month": "July",
            "Day": 31,
            "Year": 2018
          }
        },
        {
          "Title": "Doctor's Appointment",
          "Time": "10:00 am",
          "Date": {
            "Month": "September",
            "Day": 13,
            "Year": 2018
          }
        }
      ]
    }
  ],
  {
```

```

    "Name": {
        "Fname": "Jeffrey",
        "Lname": "Miller"
    },
    "Events": [
        {
            "Title": "Lecture",
            "Time": "8:00 am",
            "Date": {
                "Month": "August",
                "Day": 25,
                "Year": 2018
            }
        }
    ]
}
]
}

```

The data contained with each user includes first name, last name, and a list of events. Each event has a title, the time, and the date. The time will be on a 12-hour clock, so there is an “am” or “pm” associated with each time.

Once the data is correctly parsed by the program, the menu of options will give the user the following functionalities: Display User’s Calendar, Add User, Remove User, Add Event, Delete Event, Sort Users, Write File, and Exit.

The “Display User’s Calendar” option should display the names of all the users in the calendar’s database as well as each user’s appointments (title, time, and date).

The “Add User” and “Remove User” options add and delete users from the calendar’s database, along with their appointments.

The “Add Event” and “Delete Event” options delete a user’s appointment.

The “Sort Users” option sort the users alphabetically ascending (A-Z) or descending (Z-A).

The “Write File” option gives the user the option to write the current data out to a file to save it. Save the results in the same file that was opened by the user.

The “Exit” option displays a message indicating the end of the program before terminating. If a change has been made to the file since the last time the file was saved, prompt the user to ask if he would like to save the file before exiting.

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).

What is the name of the input file? **invalid.json**

That file is not a well-formed JSON file.

What is the name of the input file? **notfound.json**

That file could not be found.

What is the name of the input file? **assignment1.json**

- 1) Display User's Calendar
- 2) Add User
- 3) Remove User
- 4) Add Event
- 5) Delete Event
- 6) Sort Users
- 7) Write File
- 8) Exit

What would you like to do? **9**

That is not a valid option

- 1) Display User's Calendar
- 2) Add User
- 3) Remove User
- 4) Add Event
- 5) Delete Event
- 6) Sort Users
- 7) Write File
- 8) Exit

What would you like to do? **1**

- 1) Trojan, Tommy
 - a. Kristy's Birthday Party, 2:00 pm, July 31, 2018
 - b. Doctor's Appointment, 10:00 am, September 13, 2018
- 2) Miller, Jeffrey
 - a. Lecture, 8:00 am, August 25, 2018

- 1) Display User's Calendar
- 2) Add User
- 3) Remove User
- 4) Add Event
- 5) Delete Event
- 6) Sort Users
- 7) Write File
- 8) Exit

What would you like to do? **4**

- 1) Trojan, Tommy
- 2) Miller, Jeffrey

To which user would you like to add an event?

1

What is the title of the event? **Lunch with Jeffrey Miller**

What time is the event? **12:00 pm**

What month? **12**

What day? **15**

What year? **2018**

Added: Lunch with Jeffrey Miller, 12:00 pm, December 15, 2018 to Tommy Trojan's calendar.

- 1) Display User's Calendar
- 2) Add User
- 3) Remove User
- 4) Add Event
- 5) Delete Event
- 6) Sort Users
- 7) Write File
- 8) Exit

What would you like to do? **2**

What is the user's name? **Chad**

Invalid, must have first and last name.

What is the user's name? **Chad Blake**

- 1) Display User's Calendar
- 2) Add User
- 3) Remove User
- 4) Add Event
- 5) Delete Event
- 6) Sort Users
- 7) Write File
- 8) Exit

What would you like to do? **5**

- 1) Trojan, Tommy
- 2) Miller, Jeffrey
- 3) Blake, Chad

From which user would you like to delete an event? **3**

Calendar is empty.

- 1) Display User's Calendar
- 2) Add User
- 3) Remove User
- 4) Add Event
- 5) Delete Event
- 6) Sort Users
- 7) Write File
- 8) Exit

What would you like to do? **5**

- 1) Trojan, Tommy
- 2) Miller, Jeffrey
- 3) Blake, Chad

From which user would you like to delete an event? **1**

- 1) Kristy's Birthday Party, 2:00 pm, July 31, 2018

- 2) Doctor's Appointment, 10:00 am, September
- 3) Lunch with Jeffrey Miller, 12 pm, December 15, 2018

Which event would you like to delete? **1**

Kristy's Birthday Party has been deleted.

- 1) Display User's Calendar
- 2) Add User
- 3) Remove User
- 4) Add Event
- 5) Delete Event
- 6) Sort Users
- 7) Write File
- 8) Exit

What would you like to do? **7**

File has been saved.

- 1) Display User's Calendar
- 2) Add User
- 3) Remove User
- 4) Add Event
- 5) Delete Event
- 6) Sort Users
- 7) Write File
- 8) Exit

What would you like to do? **5**

- 1) Ascending (A-Z)
- 2) Descending (Z-A)

How would you like to sort? **1**

- 1) Blake, Chad
- 2) Miller, Jeffrey
- 3) Trojan, Tommy

- 1) Display User's Calendar
- 2) Add User
- 3) Remove User
- 4) Add Event
- 5) Delete Event
- 6) Sort Users
- 7) Write File
- 8) Exit

What would you like to do? 8

Changes have been made since the file was last saved.

- 1) Yes
- 2) No

Would you like to save the file before exiting? 2

File was not saved.

Thank you for using my program!

Grading Criteria

The manner by which you go about implementing the solution is not specifically graded, but make sure you parse the JSON file properly and produce output as similar as possible to what is displayed above

File Parsing (0.8%)

- 0.2% - the filename is read from the user
- 0.3% - if the file cannot be parsed, an appropriate error message is displayed
- 0.3% - if the file cannot be found, an appropriate error message is displayed

Output (1.7%)

- 0.2% - The calendar is displayed properly
- 0.1% - An invalid menu entry gives an error message and returns to the same menu
- 0.2% - Adding a user requires a first and last name and saves new user to the calendar database
- 0.2% - Removing a user properly removes them from the calendar database
- 0.2% - Adding an event to a user's calendar works correctly
- 0.1% - Delete an event to a user's calendar works correctly
- 0.2% - Sorting the user correctly sorts based on option chosen by the user
- 0.2% - Write File option correctly outputs roster and all changes to the same file that is currently open
- 0.1% - Exiting prints a message before terminating the program
- 0.2% - Exit prompts the user to save the file if a change has been made before it was saved and saves properly if necessary