CSCI-201: Principles of Software Development

Spring 2022

PA1: Stranded

Concepts Covered:

- Java Collections Framework
- File I/O
- JSON parsing

Introduction:

You are a delivery man in a post-apocalyptic world where a phenomenon known as “timefall” has wrecked the entire world. Timefall is no different from regular rainfall aside from it rapidly aging everything it touches. You have just finished your last delivery of the day and are heading home on foot when all of a sudden dark cloud begin to sweep in. Timefall is imminent but your protective gear is damaged from a fight with some bandits earlier on. Spending any amount of time in a timefall unprotected is surely a death sentence but there are no shelters nearby. You must act quickly.

Luckily, your wrist cuff is connected to the Chiral Network which enables instant teleportation to any timefall shelter on the network. However, because you are a cheap person, your wrist cuff is a mid-tier model that only supports limited Chiral frequencies. Furthermore, if a shelter is already experiencing timefall, its Chiral signal has a high probability of becoming corrupted and you will not survive the teleporation process. In summary, you can only teleport to a shelter with a supported frequency and not currently experiencing timefall.

What To Do:

- Write a Java program that accepts a list of available timefall shelters and choose the first available shelter to teleport to. Chiral frequencies supported by your wrist cuff will be provided by comma separated user input. The list of available timefall shelters will be provided in JSON form.

- Your program must make use of at least one collection in the java.util.Collection framework to search for the first available shelter (for example ArrayList, LinkedList, PriorityQueue, etc.). Failure to do so will result in significant deductions regardless of performance.

- Prune unavailable shelters from the file if one is encountered during the search process and save the modified JSON file (overwrite original).

Object Notation:

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 timefall shelters and various information about them. The data in the file is going to be stored as a JSON file, formatted as follows:

```
[
  {
    "chiralFrequency": 0,
    "timefall": true,
    "guid": "4c73d893-4afe-48b9-95b3-b7034b990f7f",
    "name": "West Knot Shelter",
    "phone": "+1 (833) 500-2122",
    "address": "534 Lorimer Street, Waiohinu, United Cities of America, 3061"
  },
]
```

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. I recommend that you create your own test files that are longer, since the file 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. 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. The JSON format can be confusing when you first use it, so feel free to use sites like JSONLint (https://jsonlint.com/) to check if JSON files you create contain valid JSON or not.

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 of converting Java objects into JSON objects (and vice versa), and it is simple and straightforward to use. Therefore, in this assignment we will be using GSON. Here are a couple of blogs discussing the merits of choosing one of these APIs over another if you are interested:

https://www.overops.com/blog/the-ultimate-json-library-json-simple-vs-gson-vs-jackson-vs-json/

https://javarevisited.blogspot.com/2016/09/top-5-json-library-in-java-JEE.html#axzz7HnYL8NBl

You need to add the JAR file (we prefer that you use the gson-2.8.6.jar version since it is the most updated) manually to your Java Build Path in Eclipse. First, move the JAR file to the top directory of your Eclipse project. In other words, if your project is named `username_CSCI201_Assignment1`, put the JAR file into the `username_CSCI201_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. Ensure “Classpath” is selected.
8. Click “Add JARs”.
9. Find the JAR in your project directory and select it.
10. Click “OK”.
11. Click “Apply and Close”.

JSON Data Format:

When your program first runs, you will need to prompt the user to enter the name of the JSON file that contains the data on the different restaurants. Your program should validate that the file is formatted properly. You can assume the following data types for each:

chiralFrequency: int

timefall: bool

guid: String

name: String

phone: String

address: String

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

- File not found
- Data cannot be converted to the proper type as shown above
- Missing data parameters

Extra data types are allowed. For example, if there was another line like “email”: “adamchik@usc.edu”, this would be a valid file we could test your code against, so you should NOT print out an error message in this case.

Sample Input:

```
[
  {
    "chiralFrequency": 0,
    "timefall": true,
    "guid": "6b3a247e-eaa1-42fd-8b54-9a8dee59e3f1",
    "name": "Isotronic Shelter",
    "phone": "+1 (823) 537-3279",
    "address": "918 Vine Street, Siglerville, United Cities of America, 4460"
  },
  {
    "chiralFrequency": 1,
    "timefall": false,
    "guid": "c4c62388-98a2-4240-8a6b-5a352e05b5fb",
    "name": "Geekular Shelter",
    "phone": "+1 (988) 523-2394",
    "address": "462 Temple Court, Calverton, United Cities of America, 8728"
  },
  {
    "chiralFrequency": 2,
    "timefall": true,
    "guid": "a2e9f68c-c164-4ca1-a13a-7166ee0fa9cf",
    "name": "Rooforia Shelter",
    "phone": "+1 (890) 488-2489",
    "address": "426 Clinton Avenue, Ogema, United Cities of America, 9082"
  },
  {
    "chiralFrequency": 3,
    "timefall": false,
    "guid": "f2ac81a3-dca4-4c3f-a7c4-193e32fc7b9e",
    "name": "Maineland Shelter",
    "phone": "+1 (804) 504-2189",
    "address": "977 Bay Street, Duryea, United Cities of America, 3695"
  },
  {
    "chiralFrequency": 4,
    "timefall": true,
    "guid": "9125ab11-db16-4c2e-b84f-fdd504cb6d64",
    "name": "Dreamia Shelter",
    "phone": "+1 (827) 597-2957",
    "address": "527 Polhemus Place, Temperanceville, United Cities of America, 6841"
  },
  {
    "chiralFrequency": 5,
    "timefall": true,
    "guid": "885b4a72-a20f-4690-af66-772e21a7264c",
    "name": "Ozean Shelter",
    "phone": "+1 (812) 517-3382",
    "address": "514 Verona Street, Ellerslie, United Cities of America, 9544"
  }
]
```

Sample Output:

```
Welcome to Bridges Link.

Please provide timefall shelter data source: sample.json
=== Data accepted ===

Please provide supported Chiral frequencies: 1, 7, 9, 3

1) List all available shelters within the min and max of supported Chiral
frequencies
2) Search for a shelter by Chiral frequency
```

```
3) Search for a shelter by name
4) Sort shelters by Chiral frequency
5) Jump to a shelter with the lowest supported Chiral frequency
Choose an option: 1

2 results

Shelter information:
- Chiral frequency: 1
- Timefall: None
- GUID: c4c62388-98a2-4240-8a6b-5a352e05b5fb
- Name: Geekular Shelter
- Phone: +1 (988) 523-2394
- Address: 462 Temple Court, Calverton, United Cities of America, 8728

Shelter information:
- Chiral frequency: 3
- Timefall: None
- GUID: f2ac81a3-dca4-4c3f-a7c4-193e32fc7b9e
- Name: Maineland Shelter
- Phone: +1 (804) 504-2189
- Address: 977 Bay Street, Duryea, United Cities of America, 3695

1) List all available shelters within the min and max of supported Chiral
frequencies
2) Search for a shelter by Chiral frequency
3) Search for a shelter by name
4) Sort shelters by Chiral frequency
5) Jump to a shelter with the lowest supported Chiral frequency
Choose an option: 3

What shelter’s name are you looking for? GeKuLar SheLter.

No such shelter…

What shelter’s name are you looking for? GeeKuLar SheLter

Found!

Shelter information:
- Chiral frequency: 1
- Timefall: None
- GUID: c4c62388-98a2-4240-8a6b-5a352e05b5fb
- Name: Geekular Shelter
- Phone: +1 (988) 523-2394
- Address: 462 Temple Court, Calverton, United Cities of America, 8728
```

```
1) List all available shelters within the min and max of supported Chiral
frequencies
2) Search for a shelter by Chiral frequency
3) Search for a shelter by name
4) Sort shelters by Chiral frequency
5) Jump to a shelter with the lowest supported Chiral frequency
Choose an option: 2

What Chiral frequency are you looking for? 20

That Chiral frequency does not exist.

What Chiral frequency are you looking for? 5

Shelter information:
- Chiral frequency: 5
- Timefall: Current
- GUID: 885b4a72-a20f-4690-af66-772e21a7264c
- Name: Ozean Shelter
- Phone: +1 (812) 517-3382
- Address: 514 Verona Street, Ellerslie, United Cities of America, 9544

1) List all available shelters within the min and max of supported Chiral
frequencies
2) Search for a shelter by Chiral frequency
3) Search for a shelter by name
4) Sort shelters by Chiral frequency
5) Jump to a shelter with the lowest supported Chiral frequency
Choose an option: 4

Shelters successfully sorted by Chiral frequency.

1) List all available shelters within the min and max of supported Chiral
frequencies
2) Search for a shelter by Chiral frequency
3) Search for a shelter by name
4) Sort shelters by Chiral frequency
5) Jump to a shelter with the lowest supported Chiral frequency
Choose an option: 5

=== Commencing timefall shelter search ===
=== Matching timefall shelter found! ===
Shelter information:
- Chiral frequency: 1
- Timefall: None
- GUID: c4c62388-98a2-4240-8a6b-5a352e05b5fb
- Name: Geekular Shelter
- Phone: +1 (988) 523-2394
- Address: 462 Temple Court, Calverton, United Cities of America, 8728

=== Commencing Chiral jump, see you in safety. ===
\\\\\\END OF EXAMPLE 1, BEGIN EXAMPLE 2 (THIS IS NOT PART OF OUTPUT RUN)\\\\\

Welcome to Bridges Link.

Please provide timefall shelter data source: badinput.json
The file badinput.json could not be found.

Please provide timefall shelter data source: sample.json
=== Data accepted ===

Please provide supported Chiral frequencies: 4, 5

1) List all available shelters within the min and max of supported Chiral
frequencies
2) Search for a shelter by Chiral frequency
3) Search for a shelter by name
4) Sort shelters by Chiral frequency
5) Jump to a shelter with the lowest supported Chiral frequency
Choose an option: 5

=== Commencing timefall shelter search ===
=== Chiral frequency 4 unstable, Chiral jump unavailable. ===
=== Removing target shelter from the list of shelters and saving updated list
to disk. ===
=== Chiral frequency 5 unstable, Chiral jump unavailable. ===
=== Removing target shelter from the list of shelters and saving updated list
to disk. ===
=== No shelter available. You are DOOMED. ===
\\\\\\END OF EXAMPLE 2, BEGIN EXAMPLE 3 (THIS IS NOT PART OF OUTPUT RUN)\\\\\

Welcome to Bridges Link.
Please provide timefall shelter data source: sample.json

=== Data accepted ===
Please provide supported Chiral frequencies: 2, 3

1) List all available shelters within the min and max of supported Chiral
frequencies
2) Search for a shelter by Chiral frequency
3) Search for a shelter by name
4) Sort shelters by Chiral frequency
5) Jump to a shelter with the lowest supported Chiral frequency
Choose an option: 5
```

```
=== Commencing timefall shelter search ===
=== Chiral frequency 2 unstable, Chiral jump unavailable. ===
=== Removing target shelter from the list of shelters and saving updated list
to disk. ===

=== Matching timefall shelter found! ===
Shelter information:
- Chiral frequency: 3
- Timefall: None
- GUID: f2ac81a3-dca4-4c3f-a7c4-193e32fc7b9e
- Name: Maineland Shelter
- Phone: +1 (804) 504-2189
- Address: 977 Bay Street, Duryea, United Cities of America, 3695

=== Commencing Chiral jump, see you in safety. ===

\\\\\\\\\\\\\\\END OF EXAMPLE 3 (THIS IS NOT PART OF OUTPUT RUN)\\\\\\\\\\\\\
```

Output Clarification:

1) Numbers given as supported Chiral frequencies do not have to be in sorted order. “9, 3, 1, 7” would have been a perfectly valid input in our first example.
2) It is valid if no frequencies are entered as supported Chiral frequencies.
3) All Chiral frequencies will be unique within a file. All names will be unique within a file.
4) When jumping to a shelter with a supported Chiral frequency, all shelters probed to see if timefall is true should be removed from your collection. This action should be reflected in the resulting JSON file.
5) A valid phone number must have some country code with the ‘+’ in front of it and follow the same format as sample.json. This means parenthesis around the area code and a dash in the appropriate place as shown above. Any amount of whitespace in the phone number String is valid.
6) You may assume addresses are always valid addresses.

Grading Criteria:

Reading User Input (30%)

· 5% - The program prompts the user for filename & Chiral frequency input

· 5% - The program keeps prompting the user to choose options until the user chooses to jump to shelter (option 5)

· 10% - Invalid user input is caught and dealt with appropriately (ask again)

· 10% - User input is parsed correctly

File I/O (30%)

· 2.5% - The filename is read from the user and the file is appropriately parsed

· 2.5% - If the file could not be found, an appropriate error message is displayed

· 2.5% - If the file cannot be parsed, an appropriate error message is displayed

· 2.5% - If the file has missing parameters, an appropriate error message is displayed

· 10% - If the file is malformatted in any other way, an appropriate error message is displayed (Ensure the program does not crash)

· 5% - Any pruned shelters should be saved to the input file

· 5% - Sorted shelters should be saved to the input file (option 4)

Collections (40%)

· 10% - A built-in collection from java.util is used for searching

· 5% - A matching shelter can be found when both Chiral frequency and timefall conditions are met

· 5% - When listing shelters for option 1, only the available shelters with the supported chiral frequencies within the range

· 5% - A shelter can be found by searching by chiral frequency

· 5% - A shelter can be found by searching by name

· 5% - No shelter can be found when the conditions are not met

· 5% - Any unavailable shelter found during the Chiral jump should be removed from the built-in java collection

Submission:

Export your Eclipse project and submit it on DEN. It is of paramount importance that you export correctly.

If we cannot import your project or if it contains errors after importing, you will get a zero.

Some notes:

- You MUST include the Gson JAR file as part of your Eclipse project. Without this, we will not be able to run your submission and you will get a zero.
- It is up to you if you wish to include JSON files you create as part of your Eclipse project. We have our own files we will test you against, so we will ignore any files of yours.

Q & A:

1) Do we have to use the function signatures that were provided in the starter files? For example, readFile and setSupportedFrequencies in the main file?
   No, you're free to change it up if you want.

2) Do we allow leading and tailing whitespace on GUID?
   Yes, GUID like "   57eeadfa-e423-4127-a3e8-7043bb5c85f8   " should be considered valid

3) When "List all available shelters within the min and max of supported Chiral Frequencies" are we filtering out the shelters where timefall is true?
   Yes

4) In case user input an empty line, what should we do?
   Should prompt the user again

5) Are capital letters allowed in guid?
   Yes

6) Option 5 pruning?
   Only prune the shelters that have been searched for timefall. This means that only shelters that are listed in the supported frequencies have the possibility of being removed when the user chooses option 5.

7) Is it always ensured that chiral frequency, time fall, guid, name, and phone are provided in that order?
   No

8) On number 1, when it says, "list available shelters within the min and max of supported chiral frequencies", is it saying list in ascending order of chiral frequencies regardless of whether or not timefall is true?
   Order doesn't matter

9) On number 4, when it says sort shelters by chiral frequency, is it by ascending order? If it is by ascending order, then what is the difference between the above bullet point and the current?

Ascending order (number 4 refers to how the file's shelters should be sorted when saving to the file at the end of the program)