CMPT 213 Assignment 2

Due: Oct 10, 11:59:59pm.

Overview

A Tokimon handler is a person who trains and cares for Tokimons. Their job is to coordinate a team of Tokimons and gather their strengths. Each team's members and their relative compatibility is kept in a JSON file much like the one that is given as part of this assignment. Each JSON file corresponds to one Tokimon in the database and gives information about their compatibility to the team as well as to each member of their team. Note that compatibility scores are not symmetric. For example, Toki1's compatibility score towards Toki2 may not necessarily be the same as Toki2's compatibility score to Toki1.

Your task in this assignment is to create a program which finds all JSON files in a folder and processes them to create the teams. It then generates a .CSV file which a handler can look at and evaluate each team's strengths and weaknesses.

Technical Requirements:

- You must create an object-oriented system of your own design. You must have at least 3 non-trivial classes by the time you are done. Each class should demonstrate strong cohesion: the class has one guiding purpose, and all methods and fields of the class fit well in the class.
- Put all your classes in the package ca.cmpt213.as2 (this helps with marking).
- Your code must conform to the programming style guide (available in the resources section of the course website). All classes must have class-level JavaDoc comments describing the purpose of the class.

Collecting Inputs:

- Your main class (the class that contains your `main()` function) must be called `TokimonProcessor`.
- The main class accepts two arguments:
    1. Path to the directory containing the input .JSON files.
    2. Path to the output directory where the generated .csv will be placed.

  If an invalid number of command line arguments is supplied (too many or too few), then print out what arguments are expected and exit the program. If the input folder (for .JSON files) or the output folder (for the generated .csv file) do not exist, then print an error message and exit. You may assume that each argument contains no spaces.
- Search the input folder (recursively) for any JSON files and read them into your program. You must use a `FileFilter` object for filtering the files found by `File.listFiles()`. You may use any approach you like for reading JSON files. I recommend using the GSON library. JSON files end in the .json extension, case insensitive. Other files must be ignored. You may assume that JSON input files have the same format as the ones supplied with this assignment. Spacing in a JSON file is arbitrary but it must conform to some basic rules: https://www.w3schools.com/js/js_json_syntax.asp.
- The first Tokimon mentioned in the JSON file refers to the Tokimon that this file is about. The rest of the Tokimons (if any), are its teammates.
- Generate an error if any of the following errors occur:
    1. Incorrect number of arguments provided to program.
    2. Input folder does not exist.
    3. No JSON files are found. Bad JSON file format or missing required fields. It is OK to not check if file has extra fields.
    4. Any score is less than 0.

- If there are any errors with the input data then: Display a message stating the error, Display the filename and path of the problematic JSON file (if applicable), and Exit the program with exit code -1, without creating any output files. It is OK to display multiple errors before exiting (if multiple problems with input data detected).

Organizing the data:

- Analyze the data from each Tokimon's JSON file to recreate the teams (i.e., infer who is teamed up with whom based on who is in their JSON file). Note that each Tokimon's JSON input file has an entry for not only themselves, but also each other teammate.
- **Teaming algorithm**: Tokimon S belongs to team T if team T already contains another Tokimon R such that R has an association to S. If a Tokimon S belongs to no existing teams, place that Tokimon in a new team.
- Errors:
    - It is an error if Tokimon S is mentioned by Tokimons who are in two different teams. (i.e., S can only be a member of one team).
    - It is an error if any Tokimon who is mentioned in the JSON file of another Tokimon in the team fails to submit a JSON file.
    - It is an error if any Tokimon in a team does not mention all other Tokimons in that team.
    - If Tokimon S is already in a team, it is an error if S's property does not match its existing recorded properties. For example, if S was defined to have name "Toki1", it cannot later have name "Toki2" in another JSON file; this is an error.
    - In the event of an error: Display a message stating the error, Display the filename and path of the problematic JSON file (when possible), and Exit the program with exit code -1, without creating any output files. It is OK to display multiple errors before exiting if multiple problems are found.
    - **Note**: Two Tokimons are the same if they have the same `id` field. Ignore spaces at the start of end of an `id` (hint String's `trim()` function) Comparison is case insensitive (hint String's `equalsIgnoreCase()` function).

Outputting the data**:**

Generate a comma separated value file (.csv) which contains the following columns.

- For each team, generate a one row header that just says "Team 1" or "Team 2" (indexed accordingly).
- For each Tokimon S in the team:
    - Generate one row for each other Tokimon R in the team. This row should list the id of S, the id of R, the compatibility score from S to R, and any comment from S to R. Generate one row showing the compatibility score S to the team, the comment from S to the team, and showing S's extra comments.
- Output Format Constraints
    - Use the following for the column headings in your CSV file:
      `Team#, From Toki, To Toki, Score, Comment, , Extra`
    - Sorting: Inside the teams, output Tokimons in alphabetical order (sorted by lower-case id).
    - **Please see sample outputs for more detail**

Column description:

- **Team Number** - This column only has data for the "header" row for each team.
- **From Toki** – the `id` of Tokimon S. The source Tokimon.
- **To Toki** – the `id` of "other" Tokimon R. For the rows that display S's own compatibility score, display "-".
- **Score** - Compatibility Score to the from Tokimon S to Tokimon R. Display as a floating-point number with 1 digit after the decimal point.
- **Comment** - Display the comment associated with this row (from Tokimon S to Tokimon R).
- **Extra** – The extra comment property of each Tokimon.

Sample Run:

- A sample run of your program may look something like:

```
> java TokimonProcessor "./InputTestDataSets/1-OneTeamOneToki"
"./AnalysisResults_v2/1-OneTeamOneToki"
```

This would generate the `team_info.csv` file located in the `1-OneTeamOneToki` folder. Please note that the path may also be an absolute path.

Marking Scheme:

[5 marks] Single Group - generates correct and well-formatted .CSV file.
[10] Multiple Groups – correct number of Tokimons per team, correlating Tokimons to teams, generate correct and well-formatted .CSV file.
[6] Error Handling - print reasonable error message in each case.
[3] Object Oriented Design

Submission

Submit a zip file of your project (according to the directions outlined in the assignments link of the course website) to the coursys server. https://courses.cs.sfu.ca/.

Your project must:

- Import external dependencies (such as GSON) via Maven, or else included them in your project.
- Generate a JAR file as part of the build process.

Please note: all submissions are automatically checked for similarities of all other submissions on the server.

THE END