2023 State Farm Coding Competition - Round 1

Welcome to the 2023 State Farm Coding Competition!

Background

Your team works for an insurance company and have been sent disaster data and their related claims, insurance agents, and claims handlers.

Your task is to analyze and fufill various business requirements using test-driven development. You have been given a skeleton project to get you started.

Disclaimer: All people, disasters, and other entities appearing in the datasets are fictitious and randomly generated. Any resemblance to real persons, dead or alive, disasters, real or otherwise, or other real-life entities and events, past or present, is purely coincidental. Values in the datasets are purely fictitious and can be exacerbated.

The Problem

For this year's challenge, you will:

- ☑ Be provided four data sets in JSON format.
- ☑ Be given three skeleton projects (java, nodejs, python) to choose from (pick one).
- ☑ Implement methods that will operate on the datasets to produce some expected outputs.
  - Stub methods are provided to get you started.
  - Corresponding unit tests will leverage the methods above.

The objective: complete as many method stubs as you can such that they pass the given unit tests, using the four sets of data provided. Please read the description of each method stub carefully before attempting to complete it.

Do NOT change anything in the unit tests.

About the Data Sets

The four data sets are as follows:

1. `sfcc_2023_agents.json` - Representational JSON array of insurance agents across the United States including region, state, and spoken languages.
2. `sfcc_2023_claim_handlers.json` - Representational JSON array of claim handlers including first and last names.
3. `sfcc_2023_claims.json` - Representational JSON array of insurance claims filed including status, which disaster they relate to, which agent and claim handler are assigned, estimated cost, and more.
4. `sfcc_2023_disasters.json` - Representational JSON array of disasters including their name, location (state, latitude, longitude), impact radius in miles, and various dates (start, end, declared disaster).

Bonus *Nice to Have* Features

Once you've completed the above functions and unit tests, consider implementing your version of the following extra features:

- Create a simple REST API to interact with the JSON data.
- Consider other important or interesting data correlations, and demonstrate these correlations either functionally, visually, or both.
- Have a better idea for bonus features than we've indicated here? Go for it!

First Actions

- **Make sure you have all the required software already installed. Check the main `README.md` for the editor we recommend and the skeleton project's README for software that must be installed**
- Clone this repository and open your choosen skeleton project (java, python, or nodejs) in your IDE
- Read through the skeleton project's README to make sure the right dependencies are installed and that the project runs
- If you leverage any additional libraries/packages, please be sure to add them to the relevant package file (see skeleton project README for instructions)
- Run your unit tests, code, and repeat

When You Are Done

- Update the `FEEDBACK.md` and include the following information:
    - Your team name
    - Name(s) of each individual participating
    - How many unit tests you were able to execute successfully
    - Document and describe any additional *Nice to Have* features you've included
        - This will help the judges properly grade your submission
        - Explain how to properly execute your new enhancements
- Push your changes to a single code branch for your team
- Open a single pull request against the main State Farm Coding Competition repository before 11:59PM CDT on October 14th, 2023
    - If you make any commits after midnight without prior approval from codingcompetition@statefarm.com, your submission will be disqualified
    - If you so choose, you can open your pull request at any time during the competition and continue to update your branch as long as you do not make any commits after Round 1 has concluded

Rules

- Contestants cannot seek help from individuals outside their team.
- Teams should have the necessary software and tools installed *prior* to the competition
- If you believe this document and the unit tests conflict, the unit tests are the highest authority
- Projects must be written in the specified languages and their versions or later
- You may create helper methods/functions, but they must be stored in a separate file

How You Will Be Graded

- 100% core requirements met, including:

    - Code must compile and execute
    - Number of unit tests that pass using correct functionality in the program
    - Uses as many of the following best programming practices:
        - Well-named variables, classes, and functions
        - Clear and concise comments as needed
        - Language-specific style guide followed
            - Java Style Guide - Google
            - Python Style Guide - PEP 8
            - JavaScript Style Guide - Google
        - Simplicity and maintainability
        - Reusability and scalability
- Do not complete any *Nice to Have* features unless you have all the unit tests completed
    - Bonus credit awarded for any extra features added (up to 10%)
    - *Nice to Have* features will not be graded if base unit tests do not pass