Overview

The goal of this is to write a simple command-line program that will print out a list of food trucks, given a source of food truck data from the San Francisco government's API.

The Task

Data

The San Francisco government's website has a public data source of food trucks (https://data.sfgov.org/Economy-and-Community/Mobile-Food-Schedule/jjew-r69b). The data can be accessed in a number of forms, including JSON, CSV, and XML. How you access the data is up to you, but you can find some useful information about making an API request to this data source here (https://dev.socrata.com/foundry/data.sfgov.org/bbb8-hzi6).

The Problem

Write a command line program that prints out a list of food trucks that are open at the current date and current time, when the program is being run. So if I run the program at noon on a Friday, I should see a list of all the food trucks that are open then.

Criteria

We will primarily evaluate programs on code quality and output correctness.

For quality, we expect code to be easy to read and maintain, performant, testable and reliable. You should submit code that you are proud to have written. However do not include tests in your submission.

Please display results in pages of 10 trucks. That is: if there are more than 10 food trucks open, the program should display the first 10, then wait for input from the user before displaying the next 10 (or fewer if there are fewer than 10 remaining), and so on until there are no more food trucks to display. Display the name and address of the trucks and sort the output alphabetically by name.

Example

```
$ show-open-food-trucks
 NAME ADDRESS Mang Hang Catering 1 Thomas More
 Way Steve's Mobile Deli 145 King Street
```

You should use the programming language you feel most comfortable in. Use of any external libraries or packages is permitted and encouraged!

We've provided basic stubs in a few languages, feel free to use them to get started:

- Java
- JavaScript/Node
- Python
- Ruby

Important note for .NET developers:

- We will be compiling and running your solution using a latest Mono docker image instead of .NET directly.
- Instructions for downloading and running that image can be found here: https://hub.docker.com/_/mono/.
- If you include a single C# file, we will compile and run your project in the docker image by running:
    - csc YourFile.cs
- If you include a C# project, we will compile and run your project in the docker image by running:
    - nuget restore YourFile.sln
    - msbuild YourFile.sln
    - mono YourDirectory/bin/Debug/YourFile.exe
- You should still include a README telling which of these approaches to use and any modifications you may have.

Things we are looking for

Aside from the guidelines listed in the problem and the criteria, the grader(s) of your program will also be assessing the following:

- Food Trucks are sorted alphabetically
- There are clear comments or self-documenting code
- Clear naming conventions
- Concise and readable code
- Code is broken up into multiple classes and methods based on responsibility
- Clean output from program
- Error cases addressed and properly handled
- Code is testable

- The Socrata API is properly used
- Business Logic is not included in the DTOs

Submitting your work

Please email us a zipped folder containing your work. Your submission should include:

1. Your code.
2. A README file that contains instructions on how to do the following on Linux or Mac OSX (if you need an exception to this, let us know):
    - install dependencies.
    - build your program.
    - run your program, with example commands on how to run it if necessary.
3. A one- or two-paragraph write-up that is no more than 250 words describing, at a high level, what you would do differently if you were asked to build this as a full-scale web application. In your write-up, please focus on the technical differences between the command-line program and the web application, rather than on the product differences.

Your submission should not include any binary files or any executable files other than your code, such as .jar files.

Submission Notes

We recommend you spend roughly two hours on this project.

We don't expect your write-up to take more than thirty minutes to complete. Therefore, keep your response high-level and not more than two paragraphs. Do spend time polishing and packaging up your submission so it is easy to install, run, and review.

Our goal is to evaluate your submission as anonymously as possible, so please try to remove any identifying information from your code (your IDE may automatically add your name, for example).