P1 Proposal: Meteorite Catcher

Brief Project Description:

The Meteorite Catcher app helps users to quickly search through and list known meteorites by their mass. The app makes use of a Red-Black Tree to efficiently insert the meteorite data and retrieve it in a sorted manner. The data is loaded from a local data file that the user specifies.

Representative Tasks Performed Using this Application:

1. Find the meteorite with the highest mass in the data set.
2. Search for and list all meteorites between 503-504 grams of weight.

Backend Developer (BD) Role

The backend developer writes code that reads in meteorite data from a CSV file, inserts it into a Red-Black Tree, and accesses it based on commands from the frontend. To read in the data, the backend developer develops a java class that defines a single meteorite object and that can be inserted into the Red-Black Tree using each meteorite's mass (in grams) as the key, sorted in reverse order. The backend supports data sets formatted in the same way as this NASA dataset from kaggle in CSV format. The frontend accesses the backend's functionality through method calls on the backend object.

Interface Design Responsibilities:

- An interface for a class that defines a single meteorite and exposes properties required by the frontend: name, latitude, longitude, fall, and mass (in grams).
- An interface for the backend that exposes the required functionality to the frontend: read data from a file, get a list of meteorites with the maximum mass in the data set, get a list of meteorites with a mass between two specified thresholds.

Presentation Responsibilities:

After integration, the backend developer demonstrates a search with the app to find the meteorites with the highest mass in the data set.

Frontend Developer (FD) Role

The frontend developer writes code that drives an interactive loop of prompting the user to select a command, then requests any required details about that command from the user, and then displays the results of the command. The commands to include are: a command to specify (and load) a data file, a command to list the meteorites with the highest mass in the data set, a command to list all meteorites with a mass between two specified thresholds, and a command to exit the app. The results are computed with the help of the BD's code. When the user enters invalid input, instructive feedback about what they should enter is displayed.

Interface Design Responsibilities:

- An interface for a class that implements the functionality of the frontend for the app. The class has a constructor that accepts a reference to the backend and a java.util.Scanner instance to read user input. It also has a method that starts the main command loop for the user. To allow for easier testing of the frontend, the command loop is broken down into a separate method for the main menu and each of the sub menus.

Presentation Responsibilities:

After integration, the frontend developer will record a video demonstration of the task: Search for and list all meteorites between 503-504 grams of weight.