WeatherApp — Documentation

Vilma Pirilä, Valma Haavisto, Aarni Akkala

December 5, 2023

1 Software Structure

The WeatherApp consists of three main classes: WeatherApp (GUI), Events (event handlin) and API (API calls). Additionally there are classess: LocationWeather (for one locations weather information), Weather (One locations weather foracast or current of a certain timestamp) and Coord (stores latitude and longitude of a place).

1.1 Class Diagram

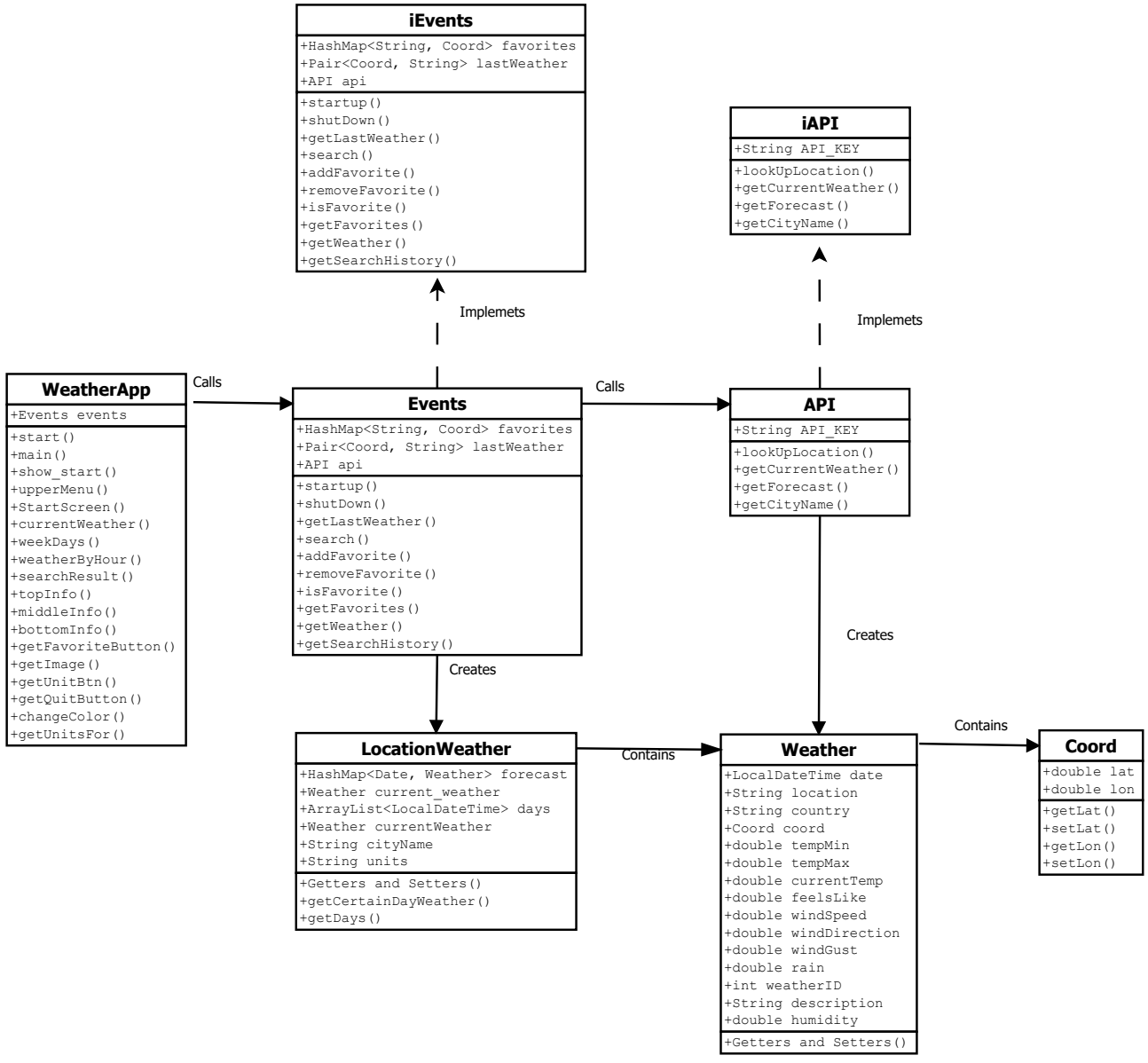

Figure 1: UML Class Diagram of WeatherApp

1.2 Class Responsibilities

`WeatherApp.java` is responsible for handling GUI element placement and event handling. It creates an `Events` object that manages storing favorites, the last searched location, and communicates with the API through an `API.java` object. The `Events.java` class implements the `iEvents.java` interface.

The `API` class, implementing the `iAPI.java` interface, retrieves JSON files containing current weather data and forecasts for given locations from `openweathermap.org`. Additionally, it identifies coordinates and proper names for a given search term (up to 5 results).

These resulting JSONs are parsed to create `Weather` class instances, storing weather conditions (current or forecast) for a specific timestamp. `Events.java` then stores these `Weather.java` objects into `LocationWeather.java` class instances, representing all weather information for a particular location. For example, the current weather and a 4-day forecast for Tampere, Finland. The `Coord.java` class stores the latitude and longitude of a given location.

Refer to the Javadoc documentation for detailed information on the various functions within these classes.

2 Project Functionality

The WeatherApp encompasses a range of essential functionalities, each detailed in the "User Manual" section:

- **Location Search:** Retrieve weather information from openweathermap.org by searching for a specific location.
- **Current Weather Display:** View real-time weather details for a selected location.
- **Forecast Information:** Access a comprehensive forecast, including hourly breakdowns for the current day and daily maximum and minimum temperatures for the next four days. Additionally, receive weather descriptions.
- **Favorites Management:** Easily add or remove locations to and from the favorites list for quick access.
- **Persistent Data:** Save the last accessed location, search history and user favorites to a file, ensuring automatic retrieval and display when the application is relaunched.
- **Unit Customization:** Seamlessly switch between metric and imperial units to suit individual preferences.
- **Search history:** Five last searchwords are saved and displayed under the search bar when typing a searchword.
- **Junit tests:** Junit test classes have been created for Events and API. Junit test compiles without errors.

3 Implemented Classes

Methods

`iAPI` Interface (implemented by API class)

- **Method:** `lookUpLocations`
  - **Preconditions:** `loc` not null or empty, `API_KEY` valid.
  - **Postconditions:** Successful API call returns `Map` of location names and coordinates (not null). `Coord` has matching coordinates. If API data is bad or no connection, `APICallUnsuccessfulException` is thrown.
- **Method:** `getCurrentWeather`
  - **Preconditions:** `coordinates` not null, has `lat` and `lon`, `units` metric or imperial.
  - **Postconditions:** Returned `Weather` object not null. If units not metric or imperial, `InvalidUnitsException` is thrown. If API data is bad or no connection, `APICallUnsuccessfulException` is thrown.
- **Method:** `getForecast`
  - **Preconditions:** `coordinates` not null, `units` metric or imperial.
  - **Postconditions:** Returned `HashMap` not empty. If units not metric or imperial, `InvalidUnitsException` is thrown. If API data is bad or no connection, `APICallUnsuccessfulException` is thrown. Returned `HashMap` may be empty if no forecast information.
- **Method:** `getCityName`
  - **Preconditions:** `latlon.lat` and `latlon.lon` not null.
  - **Postconditions:** Returned string may be empty if no result for coordinates. If API call unsuccessful, `APICallUnsuccessfulException` is thrown.

`iEvents` **Interface**

- **Method:** `startup`
  - **Preconditions:** None
  - **Postconditions:** The API instance (`api`) is initialized. Favorites are loaded from the "favorites.txt" file into the favorites map. Last weather information is loaded from the "lastWeather.txt" file into the lastWeather pair. Search history is loaded from the "searchHistory.txt" file into the searchHistory list.
  - **Exceptions:** `IOException`
- **Method:** `shutDown`
  - **Preconditions:** None
  - **Postconditions:** The current location and favorites are saved to a file. The current weather based on the last weather coordinates and units is returned. The city name and units of the returned `LocationWeather` object are set.
  - **Exceptions:** `IOException`
- **Method:** `getLastWeather`
  - **Preconditions:** None
  - **Postconditions:** Locations based on the input are looked up using the API. The top 5 locations are sorted and returned in a `TreeMap`. The input is added to the search history. If the search history size exceeds 5, the oldest element is removed.
  - **Return:** `LocationWeather` of the place that was shown when the app was closed last time
  - **Exceptions:** `InvalidUnitsException, APICallUnsuccessfulException`
- **Method:** `search`
  - **Preconditions:** None
  - **Postconditions:** Fetch first 5 search results that match the search phrase the best.
  - **Parameters:** `input` - The text in the search box
  - **Return:** Alphabetical list of locations in the form: "location, country_prefix" and Coordinates.
  - **Exceptions:** `APICallUnsuccessfulException`
- **Method:** `addFavorite`
  - **Preconditions:** None
  - **Postconditions:** Updates favorite information of the given location.
  - **Parameters:** `latlong` - Coordinates of the location, `name` - Name of the place
  - **Return:** Container of favorite locations and coords
- **Method:** `removeFavorite`
  - **Preconditions:** None
  - **Postconditions:** Removes location from favorites
  - **Parameters:** `latlong` - Coordinates of the location, `name` - Name of the place
  - **Return:** Container of favorite locations and coords
- **Method:** `isFavorite`
  - **Preconditions:** None
  - **Postconditions:** Searches if coordinates are marked as a favorite
  - **Parameters:** `latlong` - Coordinates of the location
  - **Return:** `true` if it's a favorite, `false` otherwise
- **Method:** `getFavorites`
  - **Preconditions:** None
  - **Postconditions:** Returns a container of favorite locations and coords
  - **Return:** Container of favorite locations and coords
- **Method:** `getSearchHistory`

  - **Preconditions:** None
  - **Postconditions:** Returns a container of the last 5 searched locations
  - **Return:** Container of last 5 searched locations

- **Method:** `getWeather`
  - **Preconditions:** None
  - **Postconditions:** Gets location's weather information and saves it to a `LocationWeather` object.
  - **Parameters:** `latlong` - Coordinates of the place, `units` - Options: "imperial" or "metric"
  - **Return:** Current day weather information
  - **Exceptions:** `InvalidUnitsException, APICallUnsuccessfulException`

OTHER CLASSES WILL NOT FIT INTO THIS DOCUMENTATION FILE AS IT HAS TO BE LESS THAN 5 PAGES. EVERY METHOD IS COMMENTED ON JAVA CODE. INCLUDING PRE- AND POSTCONDITIONS.

4 Division of Work

Below are listed the responsibilities (agreed and actual) of each team member on this project:

Vilma Pirilä:

- `WeatherApp.java:` Implemented the entire GUI class.
- `GUI weather icons:` Weather icons shown on GUI were designed by Vilma.

Valma Haavisto:

- `Events.java:` Implemented the entire class and test class for it.
- `LocationWeather.java:` Implemented part of the functions.
- `iEvents & iAPI interfaces:` Planned interfaces of the three main classes.
- `Organizing team meetings:` Organized team meetings and created a communication platform on Telegram.

Aarni Akkala:

- `API.java:` Implemented the entire class and test classes for it.
- `LocationWeather.java:` Implemented part of the functions.
- `iEvents & iAPI interfaces:` Created interfaces of the three main classes.
- `Documentation.pdf:` Made documentation file and class diagram.

5 User Manual

This section provides a brief introduction to using the WeatherApp program.

5.1 Running the app

The main program is `WeatherApp.java`. Execute this file, and a graphical user interface should appear.

5.2 Main window

When on the main screen of WeatherApp, you can view the current weather conditions for the last accessed location. If no location has been accessed before, the program starts from main window that includes a greeting message instead. Under the current conditions, a forecast for the current day and the next four days is displayed. Clicking on these days reveals a detailed description of the weather conditions for each day (time, description icon, temperature, wind direction, wind speed, and humidity).

5.3 Searching for a location

Utilize the topmost search bar to search for a location by entering its name (e.g., Tampere) and clicking Enter "Search". A list of locations matching the search key is presented (up to five locations). By clicking on a result, the user is presented with the weather information and forecast for the chosen place on the main window, as previously described. Five last search results are shown under the search bar when typing a searchword.

5.4 Favorites

The user can add a location to favorites by pressing "Add to favorites" or remove a location by pressing "Remove from favorites." Favorites are presented at the top in the "Favorites" section. From this list, the user can choose a location and receive its weather information. (Access more favorites by pressing "more.")

5.5 Units

There are two units available: metric or imperial. In metric, the temperature is in Celsius and wind speed in m/s. In imperial, the units are in Fahrenheit, and wind speed is in miles/hour.

5.6 Closing program

When the program is closed, the last viewed location, search history and favorites are saved to memory and retrieved when the program is relaunched.

5.7 Additional features

Search history is shown under the search bar when typing a searchword. Only the matches of the last 5 searches that match the currently typed keyword are shown. Search history is saved to a txt file when app is closed and is read back when app is opened.

Additional unit support is available. Available units are metric and imperial. They can be changed from upper right corner.

6 Known Bugs and Missing Features

1. Openweathermap.org doesn't always retrieve rain information if it should.
2. Weather icons are sometimes different when calling in metric units vs imperial units. weathermap.org gives different IDs.