COMP.CS.140 Ohjelmointi 3: Rajapinnat ja tekniikat

Project: WeatherApp

Mikhail Silaev

Fumika Matsuda

Biswa Upreti

Table of Contents

Structure Of The Application 1

Responsibilities of the key classes 2

Project Functionality 2

Classes containing pre and post conditions 3

The division of work 4

Short user manual 4

Missing features 4

Image Credits 5

AI use 5

Instructions on running the project 5

Structure Of The Application

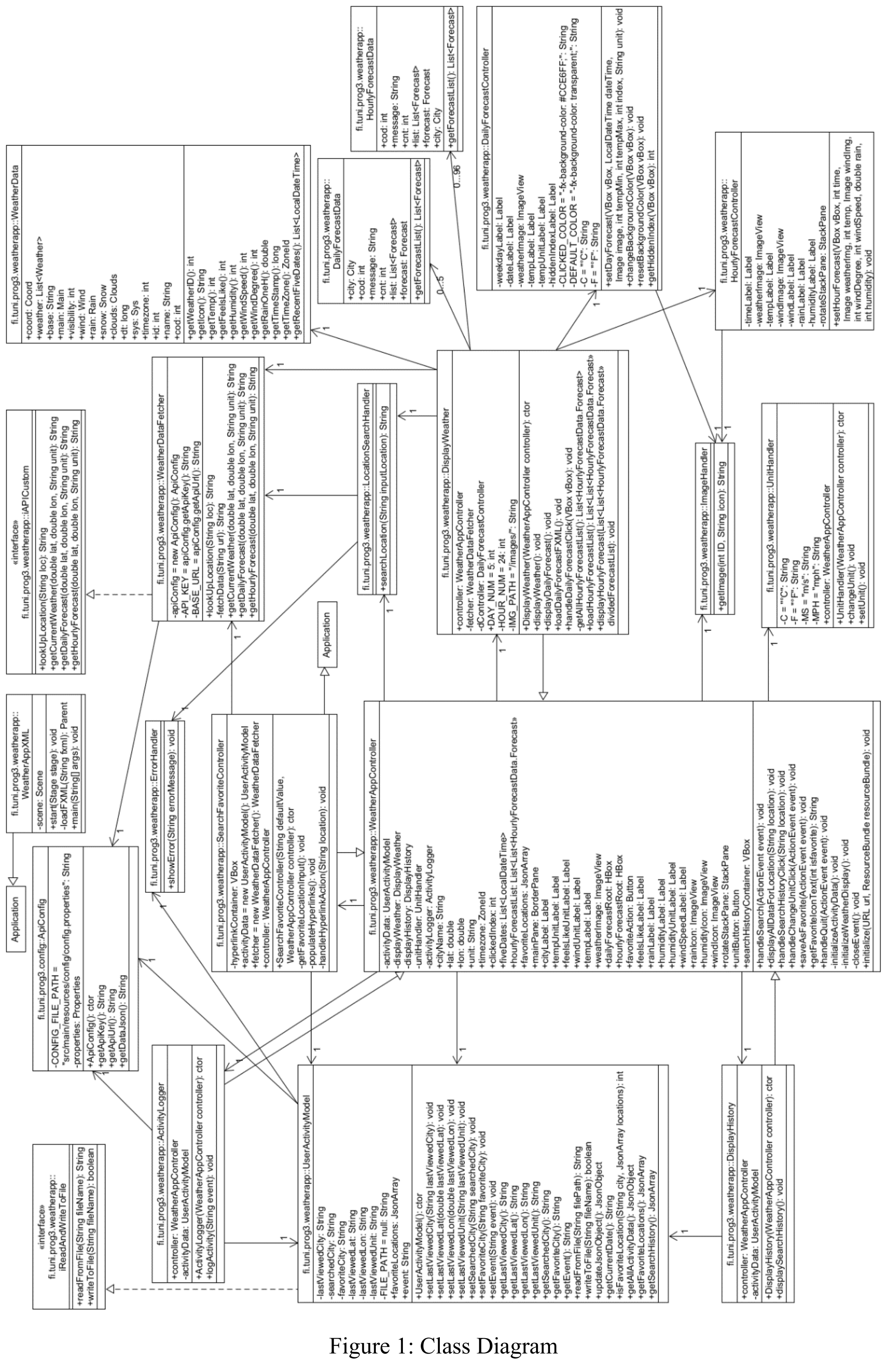
fi.tuni.prog3.weatherapp::WeatherData
fi.tuni.prog3.weatherapp::DailyForecastData
fi.tuni.prog3.weatherapp::HourlyForecastData
fi.tuni.prog3.weatherapp::DailyForecastController
fi.tuni.prog3.weatherapp::HourlyForecastController
«interface» fi.tuni.prog3.weatherapp::iAPICustom
fi.tuni.prog3.weatherapp::WeatherDataFetcher
fi.tuni.prog3.weatherapp::LocationSearchHandler
fi.tuni.prog3.weatherapp::DisplayWeather
fi.tuni.prog3.weatherapp::ImageHandler
fi.tuni.prog3.weatherapp::UnitHandler
fi.tuni.prog3.weatherapp::WeatherAppXML
fi.tuni.prog3.weatherapp::ErrorHandler
fi.tuni.prog3.weatherapp::SearchFavoriteController
Application
fi.tuni.prog3.weatherapp::WeatherAppController
fi.tuni.prog3.config::ApiConfig
fi.tuni.prog3.weatherapp::ActivityLogger
fi.tuni.prog3.weatherapp::UserActivityModel
«interface» fi.tuni.prog3.weatherapp::iReadAndWriteToFile
fi.tuni.prog3.weatherapp::DisplayHistory

Figure 1: Class Diagram

Responsibilities of the key classes

Class Name	Responsibility
ApiConfig	Provides the API configurations.
WeatherAppXML	An app starter. Loads weather.fxml.
WeatherAppController	A controller tied to weather.fxml. Controls the operation of the entire application.
WeatherDataFetcher	Implementation class of iAPICustom class (interface). A fetcher to get data from OpenWeatherMap.
WeatherData	A type-class for acquired weather data. Getter classes have also been implemented.
DailyForecastData	A type-class for acquired daily forecast data. Getter classes have also been implemented.
DailyForecastController	A controller tied to dailyForecast.fxml. Controls the operation of the daily forecast display.
HourelyForecastData	A type-class for acquired hourly forecast data. Getter classes have also been implemented.
HourelyForecastController	A controller tied to hourlyForecast.fxml. Controls the operation of the hourly forecast display.
DisplayWeather	Displays the current weather and forecast.
LocationSearchHandler	Handler to retrieve the name, latitude, and longitude of the location according to the user input.
SearchFavoriteController	A controller tied to searchFavoriteInputDialog.fxml. Controls the operation of the favorite list in the search window.
DisplayHistory	Displays the search history.
UserActivityModel	Implementation class of iReadAndWriteToFile class (interface). Reads/writes JSON files at program start/end.
ActivityLogger	Logger to record user activity to a JSON file.
ImageHandler	Handler to retrieve images according to weather type.
ErrorHandler	Handler to display an error message to the user.
ErrorLogger	Logger to record when exceptions or errors occur.

Project Functionality

Upon opening the program, users will see the current weather for the last seen location, the simple forecast for the next 5 days including today, and a detailed forecast for the day selected.

The current weather displays an image representing the weather type, the temperature, the temperature sensation, the precipitation for the last hour, the wind direction and speed, and the humidity. The simple forecast shows the date, an image representing the weather type, and minimum and maximum temperatures. The detailed forecast displays today's hourly forecast by default when opening the program or viewing another city. The contents, from top to bottom, are time, temperature, weather icon, wind direction, wind speed, probability of precipitation (in decimal format), and humidity (%). The units for temperature and wind speed can be switched between metric and imperial.

Users can search for the weather of the location they want to know by opening a new window from the search button in the upper right corner and typing the name of the location. For example, a search for tokyo/ Tokyo/ TOKYO/ tokyo,japan will return unique display results. If a location name for which no search result exists is retrieved, an error message will be displayed in a pop-up window.

Locations can be saved as favorites by clicking the ☆ button in the upper right corner. The star mark will turn black for locations that are registered as favorites. If you want to remove a location from your favorites, you can do so by clicking on the ★ button. 5 is the upper limit, and if you try to add more locations to your favorites, an error message will appear in a pop-up window. The locations you have added to your favorites will be listed below the search box, and you can click on them to view the weather for that location.

By switching the tabs in the middle section, users can view the search history. The history shows the date, time, and location of the last 10 searches. As the favorites, clicking on the name of a location will display the weather for that location.

The program can be properly exited either with the Quit button or with the Close button in the upper right corner. When you exit the program, the currently displayed location, location search history, favorite locations, and units that were displayed are saved in a JSON file. This is used to achieve the default display the next time the program is started.

Classes containing pre and post conditions

Class name	Pre-condition	Post-condition
WeatherAppXML	weather.fxml exists in the specified directory.	The application starts and the screen appears.
WeatherAppController	All parameters are valid, the required fxml files are present and loaded, and the controls are set.	All expected display with information and all expected functions are given.
WeatherDataFetcher	API key and URL are valid, and the specified location is searchable.	Data obtained from the API is returned as a string.
DailyForecastController	All parameters are valid.	Data and click behavior are set in the Daily Forecast area.

HourelyForecastController	All parameters are valid.	Data are set in the Hourly Forecast area.
SearchLocationHandler	The parameter exists, and the string returned by the fetcher class method can be divided into three parts.	The location name and its latitude and longitude are returned as string arrays from a search with the input text.
SearchFavoriteController	searchFavoriteDialog.fxml exists in the specified directory and favoriteLocations exists in the JSON file.	All data about the location name clicked by the user is displayed and the dialog is closed.
UserActivityModel	A formatted JSON file to be read/written exists in the specified path.	A JSON file is read/written.
ErrorLogger	The specified path must exist.	Log is recorded.

The division of work

Tasks were managed using a Trello board. An online meeting was held to discuss the GUI design. The following is a list of the tasks that each person has taken the lead on.

- **Mikhail Silaev**: The base format for the app/ Current weather display/ Obtaining and handling weather-type images/ Unit testing
- **Biswa Upreti**: The base format for the app/ Reading and writing JSON files/ Favorite functionality/ Search history functionality/ Implementing logger/ Security Control
- **Fumika Matsuda**: Fetching data from OpenWeatherMap/ Daily and hourly forecast displays/ Unit conversion functionality/ Documentation

Short user manual

- Open the program to see the current weather and forecast. Click on any day in the middle section, and a detailed forecast for that day will appear in the lower section.
- To find weather for any location, click on the search button and input the city name in the search box. You can click on favorite lists to quickly access saved locations.
- To save the currently displayed location as a favorite, click on the ☆ button in the upper right corner of the main window. Up to five favorites can be saved. To remove a city from favorites, display the weather for that city and click on the ★ button.
- To change unit, click on the Imperial (or Metric) button in the upper left corner of the main window.
- To exit, click on the Quit or close button. Your settings and locations are saved automatically.

Missing features

The following is a list of improvements that were not implemented this time due to time constraints.

- **The hourly forecast display**: If the number of elements to be displayed on days 1 and 5 is extremely small, it will not look good. The format should be changed to an appropriate one or the display should be such that it is not a bug if nothing is displayed for Day 1. Also, It is hard to know which information represents what. We tried to add headings, but it was difficult to make them look good.
- **The favorite functionality**: It is not possible to remove a favorite location directly from the list of favorites in the search window. Also, There is a lag between clicking on a favorite location and the weather showing up, and users might be tempted to click the OK button during that time. It does not crash, but the UI should be improved if the default is that it takes a long time to retrieve data.
- **The whole app**: It is not designed for use in a variety of environments (e.g., display sizes).

Image Credits

Freepik: arrow.png, icon-pit: rain.png, humid.png, Dorova: all other images

AI use

Look up syntaxis here and there in ChatGPT. Most frequently I write some version of the code, copy it to ChatGPT which then produces a corrected version. Also, When the cause of the error could not be pinpointed (especially in fxml files), questions to ChatGPT were sometimes helpful. It is useful to ask for suggestions for improvement or for help when we do not understand the cause of an error, based on the common understanding that we write our code ourselves.

Instructions on running the project

Please follow the instructions below to set up the project. The reason for having example properties or data files is to manage changes in future with version control.

Updating Properties File:

- Make a copy of *WeatherApp/src/main/resources/config/example.properties* to/as *WeatherApp/src/main/resources/config/config.properties*
- Replace *your_api_key* with your API Token from openweatherdata

Structured data storage as JSON:

- Make a copy of *WeatherApp/src/main/resources/data/example.json* to/as *WeatherApp/src/main/resources/data/data.json*
-

After the previous steps are done, run the program as for example: “mvn javafx:run”