De La Salle University - Manila
Gokongwei College of Engineering

Weather Tracker App

LBYCPEI - EQ1
Term 2, A.Y. 2022 -2023

Submitted by:
Llopis, Ferndale T.
Molina, Marcus Cristan S.
Pelagio, Dana Ysabelle A.

Submitted to:
RUIZ, RAMON STEPHEN L.

Submitted on:

August 3, 2023

I. INTRODUCTION

Weather forecasting is an essential aspect of our daily lives, influencing our plans, activities, and decisions. With the increasing reliance on technology and smartphones, weather-tracking applications have become indispensable tools for users to access up-to-date weather information. Our final paper presents the development of the Weather Tracker App, a graphical user interface (GUI) application implemented in Java. The primary objective of this app is to provide users with up-to-date weather forecast information in a user-friendly manner. This paper will delve into the architecture and key features of the Weather Tracker App, explaining how it operates and ensuring the accuracy of the weather data.

The Weather Tracker App is structured into three packages: Forecast, Location, and UIApp. The Forecast package handles forecast data and related operations. It includes classes like FeedEntry, ForecastValues, WeatherHttpRest, and WeatherImp. FeedEntry represents the data structure for storing weather forecast information, while the other classes handle fetching weather data from the external OpenWeatherMap API. The Location package is responsible for geolocation-related functionality. The GetCurrentRegion class utilizes the MaxMind GeoIP database to obtain the user's geographic location based on their IP address. By retrieving the user's city and country code, the app can tailor the weather forecast for the user's specific location. Finally, the UIApp package contains the graphical user interface implementation of the Weather Tracker App. The main class, MainWeatherFrame, serves as the entry point for the

application and is responsible for creating and displaying the GUI components for weather information.

The Weather Tracker App functions by first identifying the user's geographic location using their IP address. The GetCurrentRegion class leverages the MaxMind GeoIP database to accurately determine the user's city and country code. This geolocation data is then utilized to fetch region-specific weather forecasts. Once the user's location is established, the app uses the OpenWeatherMap API to fetch real-time weather data for that particular region. The WeatherHttpRest and WeatherImp classes handle the API communication and data retrieval processes. The app displays various weather parameters, such as the current temperature, weather description, humidity, pressure, and wind speed.

Ensuring the accuracy of weather data is crucial for the app's reliability. By fetching data from the reputable OpenWeatherMap API, the app leverages a reliable source of weather information. OpenWeatherMap is known for providing up-to-date and accurate forecasts for various locations worldwide. In the case of an unstable network connection, the Weather Tracker App is designed to inform users about the issue. When the app detects a problem with the network connection, it will display a notification, alerting the user that there might be an issue with their connection. This prompt allows users to take appropriate actions, such as checking their network settings or connecting to a more stable network, to ensure they receive reliable weather data.

The Weather Tracker App provides a valuable solution for users to access accurate and real-time weather information. Its simple and intuitive design, combined with the robustness of the OpenWeatherMap API and the MaxMind GeoIP database, makes it a reliable tool for staying informed about weather conditions and making well-informed decisions based on accurate weather forecasts.

II. RELATED WORK

The section may examine further programs or academic investigations that use geolocation services based on IP addresses to pinpoint the location of users. They might talk about how these services work with the Weather Tracker App, as well as their dependability and accuracy.

Existing Weather Tracking Applications To compare features, user interface design, and functionality, the authors may investigate and review different weather tracking applications on the market. This would demonstrate the distinctiveness and benefits of their weather tracking app. Use of JavaFX and Swing Because these two technologies are used by the program to generate the user interface, the authors may make references to previous work or academic articles that examine how to use them to build graphical user interfaces. SwingWorker is used in the project's BackgroundWorker class's implementation of background tasks. The writers may talk about further research projects or applications that use background employees for related objectives. The AbsoluteLayout class is a unique layout manager that is used to position UI elements precisely. The related work section may compare the advantages and disadvantages of

utilizing AbsoluteLayout with other layout management strategies. Weather APIs and Data Sources Different weather APIs and data sources that give weather information could be covered in the section on related work. To support their selection of the OpenWeatherMap API, they might contrast the accuracy, coverage, and update frequency of several APIs.

Other Weather Data Visualization Studies Research studies or projects that concentrate on weather data visualization methodologies could be included in the related work area. This could involve talking about how people are successfully shown weather parameters. The authors may use studies or other weather applications with appealing user interfaces when discussing UI design for weather applications. They might talk about UI design fundamentals including data visualization, icon use, color scheme, and layout to make an intuitive and user-friendly interface.

III. PROPOSED APPLICATION

The application is a weather tracking app that would utilize an API key from OpenWeatherMap to gather accurate weather data to show the users. Upon opening the app the users would be greeted by a UI that would contain any relevant information that the user would need in a weather tracking application.

Overall the goal of the weather tracking app is to provide users with accurate weather information by getting this data from a reliable source such as OpenWeatherMap. Whether users need to plan their daily activities, prepare for a trip, or stay informed about

severe weather events, this app will be their go-to tool for accurate weather data and timely updates.

IV. IMPLEMENTATION/OOP ASPECTS

Classes such as "ForecastValues," "FeedEntry," "TemperatureRoundSplit," and "WeatherImp" are defined in this application to represent several aspects of weather forecasting. Each class isolates its properties and functionality, making it easier to maintain and modify each part of the program individually.

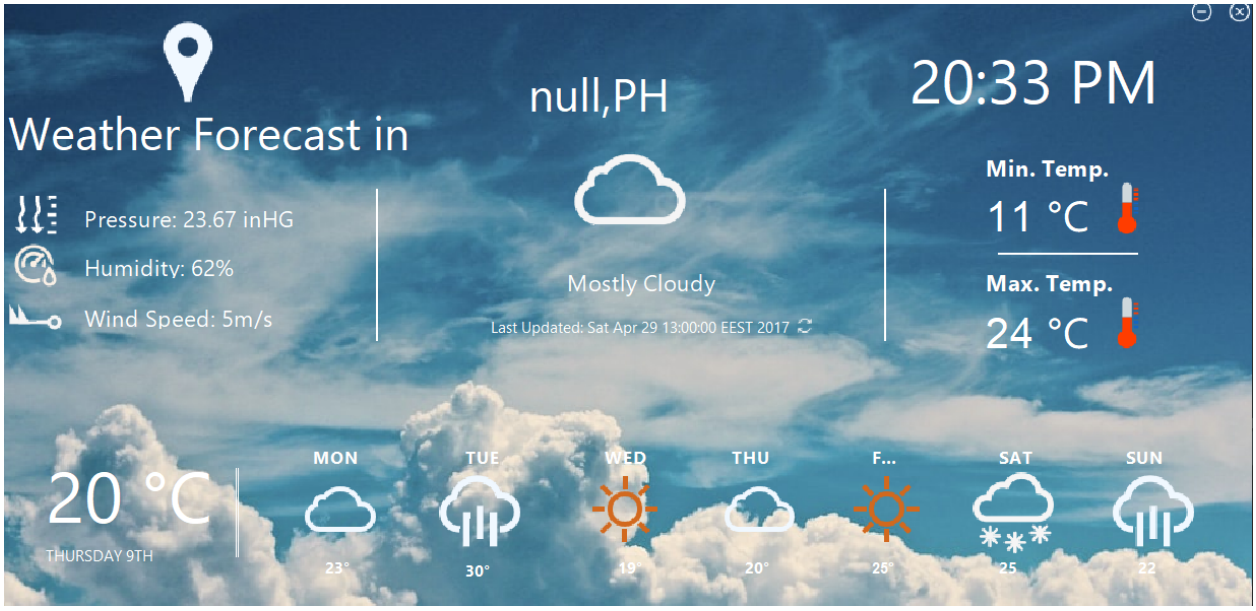
The program utilizes encapsulation of public and private classes with getters and setters to securely access and modify the properties. Encapsulation conceals the internal implementation details, allowing the application to manage access to its data and secure it from unauthorized modifications.

Overall, using OOP in the weather forecasting program promotes code reusability, flexibility, and maintainability by providing a systematic and organized approach to development. Using OOP principles, the application may manage weather data more efficiently, improve user experience, and ensure the application remains adaptive and durable in the face of future issues.

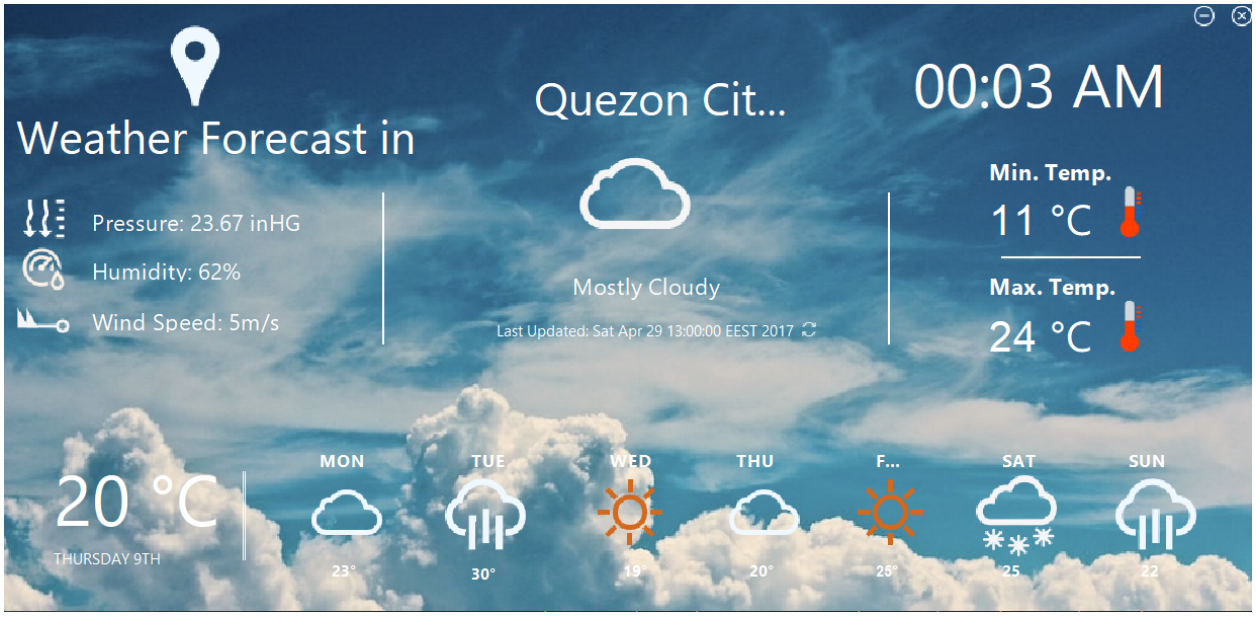
V. WALKTHROUGH/DATA/RESULTS

- **User Interface**

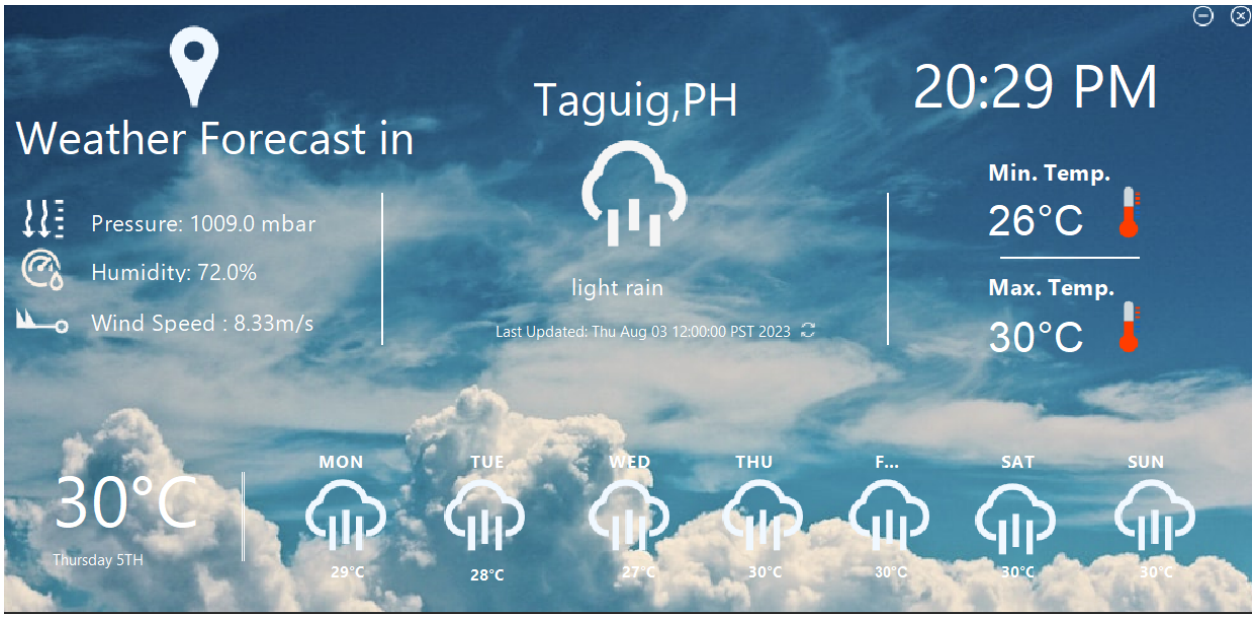
- The JFxBuilder class is responsible for building the JavaFX user interface. It uses JavaFX to create a UI window that will display weather information and any error dialogs if needed. The UI is built using JavaFX controls such as labels, buttons, and text areas.
- The DynamicJLabelList and DynJLabelObject classes are used to manage lists of dynamic JLabels (Swing's label) that will be used to display weather information for different days. These dynamic labels are used to show the temperature and weather icons for each day in the forecast.
- The BlockUI class is used to control the visibility of the UI components based on whether weather information is available. It fetches weather data using the WeatherHttpRest class and updates the UI accordingly.
- The BackgroundWorker class extends SwingWorker and performs background tasks related to fetching weather data from the API and updating the UI with the received data.
- The AbsoluteLayout class is a custom layout manager that allows absolute positioning of components within the container. It is used to precisely position UI components in the window.



- **Weather API**
    - The weather API integration is done using the WeatherHttpRest class. It is responsible for making HTTP REST requests to the weather API, passing the user's location coordinates to retrieve weather information.
    - The FeedEntry class and other related classes from the com.plugin.awesomejava.Forecast package likely represent the data models used to parse the JSON response received from the weather API. These classes hold the relevant weather information, such as temperature, humidity, weather descriptions, etc.



- **Display Weather information**
    - The BackgroundWorker class fetches weather data in the background using the WeatherHttpRest class. It processes the received weather data and updates the UI components accordingly.
    - The UI displays the current weather information, including the temperature, weather icon, description, humidity, wind speed, etc. It also shows the weather forecast for multiple days using dynamic JLabels.
    - The AbsoluteLayout is used to position the UI components in the window precisely. The AbsoluteConstraints class is used to specify the exact position and size of each component.
    - If there is an error while fetching weather data, such as an API request failure or no data available, the application will show an error dialog using the JFxBuilder class.



- **UIApp.plantuml**

  This UML diagram represents the class structure of the UIApp application. The application appears to be related to weather information and has several classes that handle different aspects of the user interface and background tasks.

  The main classes in the diagram include:

- AbsoluteConstraints: This class seems to define constraints for UI components in the application.

- AbsoluteLayout: It is a layout manager class that handles the positioning of UI components.

- BackgroundWorker: This class seems to handle background tasks related to weather updates and UI updates.

- BlockUI: A class related to blocking or unblocking user interface interaction.

- DialogObject: This class represents dialog objects used to display messages or alerts.

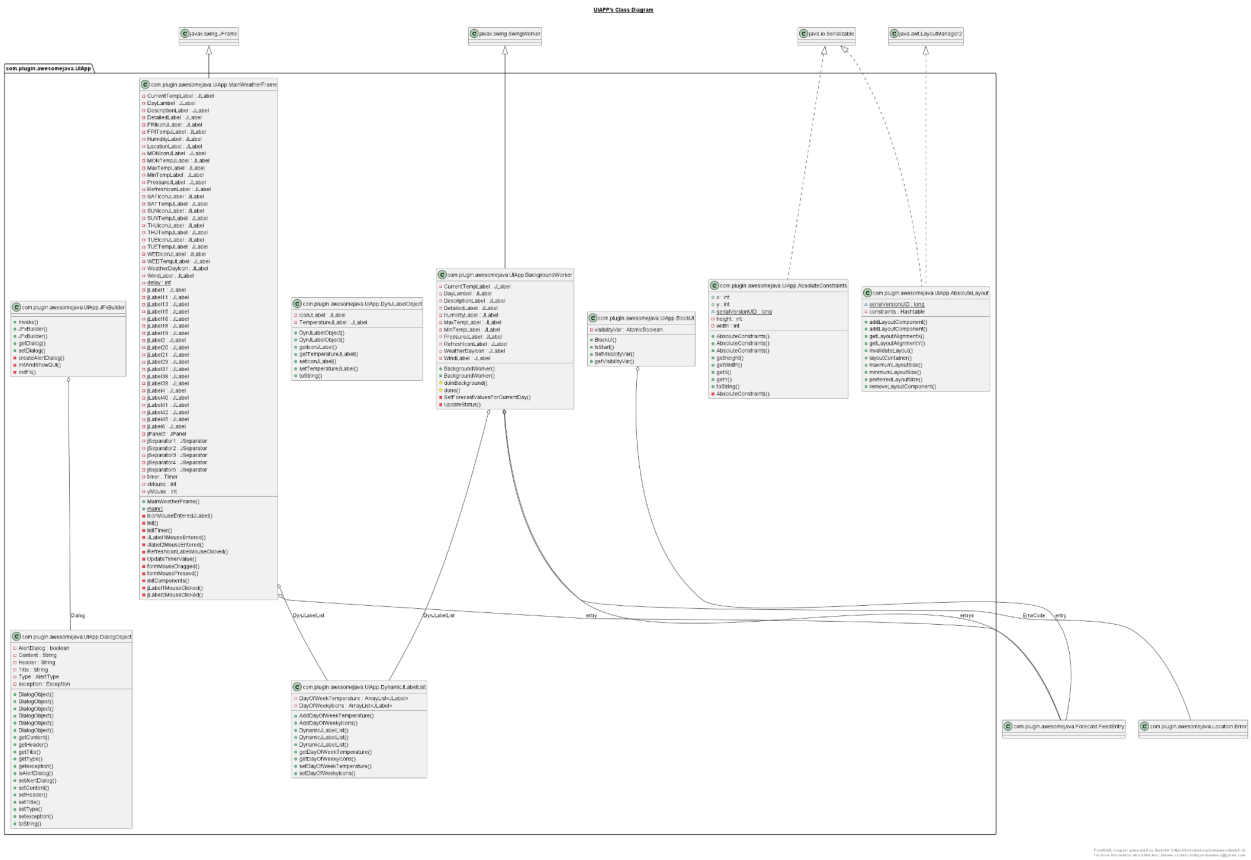
- DynJLabelObject: It appears to be related to dynamic JLabel objects.

- DynamicJLabelList: This class manages lists of dynamic JLabel objects.

- JFxBuilder: A class responsible for building UI elements using JavaFX.

- MainWeatherFrame: This class represents the main weather frame in the application, containing various weather-related labels and components.

The associations between classes are also represented in the diagram, showing the relationships between various classes. For example, the BackgroundWorker class is associated with DynamicJLabelList, Error, and FeedEntry classes, indicating its involvement in handling dynamic labels, error codes, and weather feed entries.



VI. CONCLUSION AND FUTURE WORK

Overall, The Weather Tracker App developed in Java is a graphical user interface (GUI) application that provides weather forecast information to the user. It retrieves weather data from the OpenWeatherMap API and displays it in a user-friendly manner. The app includes features like displaying current temperature, weather description, humidity, pressure, wind speed, and weather forecast for the upcoming days of the week.

The app is divided into three packages such as Forecast, Location, and UIAPP

where Forecast: handles the forecast data and related operations. It includes classes like FeedEntry, ForecastValues, WeatherHttpRest, and WeatherImp. The FeedEntry class likely represents the data structure for storing weather forecast information, while the

other classes handle fetching weather data from an external API (e.g., OpenWeatherMap). For the Location: it handles geolocation-related functionality. It includes classes like GetCurrentRegion, which is responsible for obtaining the user's geographic location based on their IP address using the MaxMind GeoIP database. UIApp: This package contains the graphical user interface (GUI) implementation of the weather tracking application. The main class MainWeatherFrame represents the main entry point for the application and is responsible for creating and displaying the GUI components for weather information.

The application provides a simple and intuitive way for users to view weather forecasts for different days of the week, along with detailed weather information like temperature, humidity, and pressure.

Future Work:

These are the several areas where the Weather Tracker App could be further improved and expanded:

1. Improved UI Design: Enhance the app's user interface to make it more visually appealing and user-friendly. Consider adding more graphical elements, interactive animations, or dynamic backgrounds to improve the overall user experience.
2. Geolocation Services: Implement geolocation services to allow users to enter their location manually or use GPS services to get real-time weather data for their current location.

3. Weather Alerts: Introduce a feature to display weather alerts and warnings for severe weather conditions, such as storms, hurricanes, or extreme temperatures.

4. Weather Maps: Integrate weather maps and radar data to provide users with visual representations of weather patterns and forecasts on a map.
5. Historical Weather Data: Add the capability to view historical weather data, allowing users to analyze past weather patterns and trends.

The Weather Tracker App may be made into a more capable and feature-rich tool for users to stay updated on weather conditions in their neighborhood and beyond by addressing these areas for improvement and continuing to develop and expand the app's capabilities.

VII. CONTRIBUTIONS (Individual contributions)

For the contribution of each member:

Research weather data sources	All
Design the user interface	Marcus
Implement location class	Dana
Implement weather tracker class	Dale
Implement daily climate change updates class	All
Integrate weather data retrieval using APIs	All
Testing and debugging	All
Finalize user manual and technical documentation	All

Prepare project presentation	All

VIII. REFERENCES

- Weather tracker - brycepedroza.com. (n.d.). https://brycepedroza.com/tweather_bryce_pedroza.pdf
- Oracle (2013, March 14). What Is JavaFX? | JavaFX 2 Tutorials and Documentation. https://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm
- YouTube. (2017, May 6). Beautiful Java Gui Weather app with swing in intellij IDEA-netbeans. YouTube. https://www.youtube.com/watch?v=H5PB91liSao
- Climate News. (n.d.). ScienceDaily. https://www.sciencedaily.com/news/earth_climate/climate/
- YouTube. (2023, June 28). Building a weather app with Java. YouTube. https://www.youtube.com/watch?v=8befqDW1jb4
- Martin, R. C. (2008). Clean code: A handbook of agile software craftsmanship. Prentice Hall.
- Freeman, E., & Robson, E. (2004). Head first design patterns. O'Reilly Media.
- Dea, C., Grunwald, G., Pereda, J., & Phillips, S. (2017). JavaFX 9 by example. Packt Publishing.
- Kleppmann, M. (2017). Designing data-intensive applications: The big ideas behind reliable, scalable, and maintainable systems. O'Reilly Media.
- Bloch, J. (2017). Effective Java. Addison-Wesley.