Introduction

The Weather App is a dynamic Java Swing-based application that allows users to retrieve real-time weather data by entering a city name. The app features an interactive graphical user interface (GUI) with custom images and intuitive controls. It integrates external APIs—including Weather Forecast and Geolocation APIs—to fetch up-to-date weather information, which is then parsed using the Simple JSON library.

Project Overview

The application consists of:

- A user-friendly GUI where users can enter a city name.
- A search functionality that retrieves weather data via API calls.
- Visual display elements (images and text) to present temperature, weather conditions, humidity, and wind speed.
- Backend logic that handles API calls, JSON parsing, and error checking.

Code Structure

Main Class

Purpose:

Sets up the GUI components using Java Swing.

• Key Responsibilities:

- Configuring the main JFrame and JPanel.
- o Adding components such as text fields, labels, and buttons.
- Implementing an action listener on the search button to trigger the weather data retrieval and update the GUI elements accordingly.

AppLauncher Class

Purpose:

Contains the main method to launch the application.

Key Responsibilities:

- Initializes the Swing GUI in the Event Dispatch Thread using SwingUtilities.invokeLater.
- Creates an instance of the Main class to display the application window.

BackEnd Class

• Purpose:

Handles external API calls and data processing.

Key Responsibilities:

getWeatherData(String locationName):

Retrieves weather data for the specified city by first fetching geolocation details and then making a weather forecast API call.

getLocationData(String locationName):

Makes an API call to the Geolocation API to convert a city name into geographical coordinates (latitude and longitude).

fetchApiResponse(String urlString):

Opens a connection to the provided URL and returns an HttpURLConnection object after verifying the connection.

findIndexOfCurrentTime(JSONArray timeList):

Iterates through the time entries from the API response to find the index corresponding to the current hour.

convertWeatherCode(long weatherCode):

Converts numeric weather codes from the API into human-readable weather conditions (e.g., "Clear", "Cloudy", "Rain", "Snow").

Features and Functionality

Interactive GUI:

Built with Java Swing; includes a search field, buttons, and dynamic image displays.

Real-Time Weather Data:

Integrates with external APIs to fetch current weather details based on user input.

Custom Image Handling:

Loads and displays images for various weather conditions (e.g., clear, cloudy, rain, snow).

Data Parsing:

Uses the Simple JSON library to parse API responses efficiently.

User Feedback:

Updates GUI elements (temperature, weather condition, humidity, and wind speed) dynamically based on API responses.

Installation and Build Instructions

Prerequisites

- Java Development Kit (JDK): Version 8 or later.
- **Git:** For cloning the repository.
- **Simple JSON Library:** Ensure that the JSON library is added to the project's classpath.
- **Image Assets:** Verify that image files (e.g., clear.png, cloudy.png) are available in the specified directories.

User Instructions

Starting the Application

1. Launch the App:

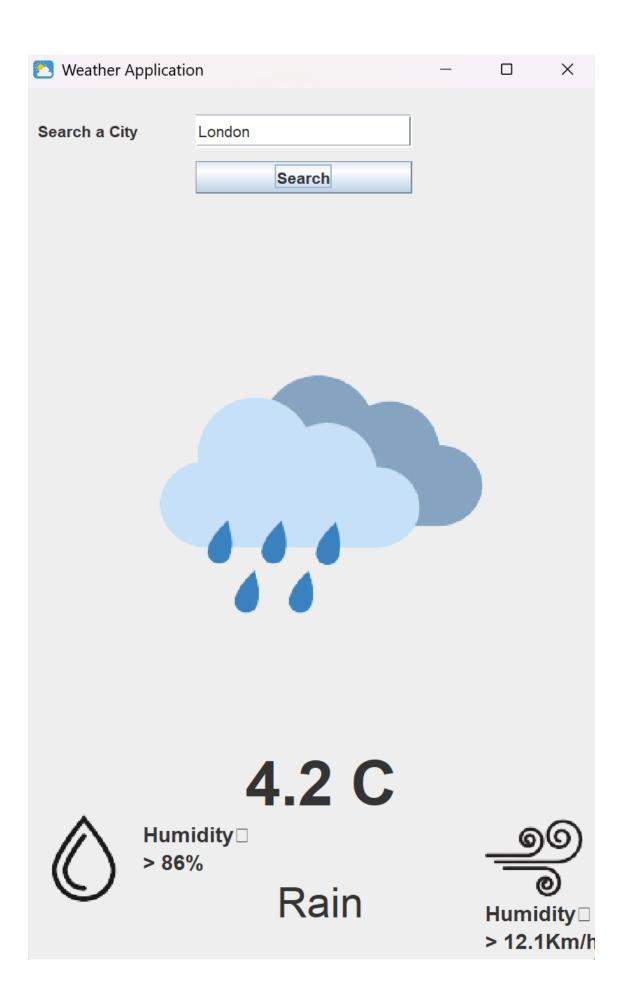
Run the application using the AppLauncher class

2. Search for a City:

- Upon launch, the application displays a GUI with a search field labeled "Search a City".
- o Enter the name of the city for which you want to retrieve weather data.
- Click the Search button.

3. Viewing Weather Information:

- Once you click **Search**, the application will fetch weather data using the integrated APIs.
- o The GUI will update to display:
 - Weather Image: Represents the current weather condition (e.g., clear, cloudy, rain, snow).
 - Temperature: Shown in degrees Celsius.
 - Weather Description: A textual representation of the weather condition.
 - Humidity and Wind Speed: Displayed alongside respective icons.



Dependencies and External Resources

- Java Swing: For building the GUI.
- **Simple JSON:** For parsing JSON responses.
- External APIs:
 - o Weather Forecast API: Open Meteo API
 - o **Geolocation API:** Open Meteo Geocoding API
- Image Assets:

Provided via local file paths (ensure images are in the correct directories).