1. Well-commented Java Source code:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.io.*;
import java.text.SimpleDateFormat;
import java.util.*;
import org.json.*;

public class WeatherApp extends JFrame {
    private JTextField cityInput;
    private JButton fetchButton;
    private JComboBox<String> unitSelector;
    private JTextArea weatherDisplay;
    private JLabel weatherIcon;
    private DefaultListModel<String> historyModel;
    private JList<String> historyList;
    private JPanel mainPanel;
    private JLabel backgroundLabel;

    private String apiKey = "YOUR_API_KEY"; // Replace with your
OpenWeatherMap API key
    private String units = "metric"; // Default to Celsius

    public WeatherApp() {
        setTitle("Weather Information App");
        setSize(800, 600);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLocationRelativeTo(null);

        // Main panel with background
        mainPanel = new JPanel() {
            Image backgroundImage = null;

            @Override
            protected void paintComponent(Graphics g) {
                super.paintComponent(g);
                if (backgroundImage != null) {
                    g.drawImage(backgroundImage, 0, 0, getWidth(),
getHeight(), this);
                }
            }

            public void setBackgroundImage(Image image) {
                this.backgroundImage = image;
                repaint();
            }
        };
        mainPanel.setLayout(new BorderLayout());
        setContentPane(mainPanel);

        // Top panel for input
        JPanel topPanel = new JPanel();
        topPanel.setOpaque(false);
        cityInput = new JTextField(15);
        fetchButton = new JButton("Get Weather");
        unitSelector = new JComboBox<>(new String[]{"Celsius",
"Fahrenheit"});
        topPanel.add(new JLabel("Enter City:"));
        topPanel.add(cityInput);
        topPanel.add(fetchButton);
        topPanel.add(new JLabel("Units:"));
        topPanel.add(unitSelector);
        mainPanel.add(topPanel, BorderLayout.NORTH);

        // Center panel for weather display
        JPanel centerPanel = new JPanel(new BorderLayout());
        centerPanel.setOpaque(false);
        weatherDisplay = new JTextArea();
        weatherDisplay.setEditable(false);
        weatherDisplay.setFont(new Font("Monospaced", Font.PLAIN,
14));
        JScrollPane scrollPane = new JScrollPane(weatherDisplay);
        centerPanel.add(scrollPane, BorderLayout.CENTER);

        // Weather icon
        weatherIcon = new JLabel();
        centerPanel.add(weatherIcon, BorderLayout.EAST);

        mainPanel.add(centerPanel, BorderLayout.CENTER);

        // History panel
        historyModel = new DefaultListModel<>();
        historyList = new JList<>(historyModel);
        JScrollPane historyScrollPane = new
JScrollPane(historyList);
        historyScrollPane.setPreferredSize(new Dimension(200, 0));
        mainPanel.add(historyScrollPane, BorderLayout.WEST);

        // Action listeners
        fetchButton.addActionListener(e -> fetchWeather());
        unitSelector.addActionListener(e -> {
            String selectedUnit = (String)
unitSelector.getSelectedItem();
            units = selectedUnit.equals("Celsius") ? "metric" :
"imperial";
            if (!cityInput.getText().trim().isEmpty()) {
                fetchWeather();
            }
        });
    }

    private void fetchWeather() {
        String city = cityInput.getText().trim();
        if (city.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Please enter a city
name.", "Input Error", JOptionPane.ERROR_MESSAGE);
            return;
        }

        try {
            // Fetch current weather
            String weatherUrl =
"https://api.openweathermap.org/data/2.5/weather?q=" +
URLEncoder.encode(city, "UTF-8")
                    + "&appid=" + apiKey + "&units=" + units;
            String weatherResponse = getHttpResponse(weatherUrl);
            JSONObject weatherJson = new
JSONObject(weatherResponse);

            // Extract weather data
            String weatherMain =
weatherJson.getJSONArray("weather").getJSONObject(0).getString("main
");
            String weatherDescription =
weatherJson.getJSONArray("weather").getJSONObject(0).getString("desc
ription");
            String iconCode =
weatherJson.getJSONArray("weather").getJSONObject(0).getString("icon
");
            double temperature =
weatherJson.getJSONObject("main").getDouble("temp");
            int humidity =
weatherJson.getJSONObject("main").getInt("humidity");
            double windSpeed =
weatherJson.getJSONObject("wind").getDouble("speed");
            long timestamp = weatherJson.getLong("dt");

            // Display weather data
            StringBuilder sb = new StringBuilder();
            sb.append("City: ").append(city).append("\n");
            sb.append("Weather: ").append(weatherMain).append("
(").append(weatherDescription).append(")\n");
            sb.append("Temperature:
").append(temperature).append(units.equals("metric") ? " °C" : "
°F").append("\n");
            sb.append("Humidity: ").append(humidity).append(" %\n");
            sb.append("Wind Speed:
").append(windSpeed).append(units.equals("metric") ? " m/s" : "
mph").append("\n\n");

            // Fetch forecast
            String forecastUrl =
"https://api.openweathermap.org/data/2.5/forecast?q=" +
URLEncoder.encode(city, "UTF-8")
                    + "&appid=" + apiKey + "&units=" + units;
            String forecastResponse = getHttpResponse(forecastUrl);
            JSONObject forecastJson = new
JSONObject(forecastResponse);
            JSONArray forecastList =
forecastJson.getJSONArray("list");

            sb.append("Short-term Forecast:\n");
            for (int i = 0; i < 5; i++) {
                JSONObject forecast = forecastList.getJSONObject(i);
                String dateTime = forecast.getString("dt_txt");
                double temp =
forecast.getJSONObject("main").getDouble("temp");
                String desc =
forecast.getJSONArray("weather").getJSONObject(0).getString("descrip
tion");
                sb.append(dateTime).append(" -
").append(temp).append(units.equals("metric") ? " °C" : " °F")
                        .append(" - ").append(desc).append("\n");
            }

            weatherDisplay.setText(sb.toString());

            // Set weather icon
            ImageIcon icon = new ImageIcon(new
URL("https://openweathermap.org/img/wn/" + iconCode + "@2x.png"));
            weatherIcon.setIcon(icon);

            // Set dynamic background
            setDynamicBackground(timestamp);

            // Add to history
            String timeStamp = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss").format(new Date());
            historyModel.addElement(city + " @ " + timeStamp);

        } catch (Exception ex) {
            JOptionPane.showMessageDialog(this, "Error fetching
weather data: " + ex.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
        }
    }
    private String getHttpResponse(String urlStr) throws IOException
{
        StringBuilder result = new StringBuilder();
        URL url = new URL(urlStr);
        HttpURLConnection conn = (HttpURLConnection)
url.openConnection();
        conn.setRequestMethod("GET");
        try (BufferedReader reader = new BufferedReader(new
InputStreamReader(conn.getInputStream()))) {
            String line;
            while ((line = reader.readLine()) != null) {
                result.append(line);
            }
        }
        return result.toString();
    }

    private void setDynamicBackground(long unixTime) {
        Calendar calendar = Calendar.getInstance();
        calendar.setTimeInMillis(unixTime * 1000L);
        int hour = calendar.get(Calendar.HOUR_OF_DAY);

        String imageName;
        if (hour >= 6 && hour < 12) {
            imageName = "morning.jpg";
        } else if (hour >= 12 && hour < 18) {
            imageName = "afternoon.jpg";
        } else if (hour >= 18 && hour < 20) {
            imageName = "sunset.jpg";
        } else {
            imageName = "night.jpg";
        }

        try {
            Image image = new
ImageIcon(getClass().getResource(imageName)).getImage();
            ((JPanel) getContentPane()).setBackgroundImage(image);
        } catch (Exception e) {
```

```java
            System.out.println("Background image not found: " +
imageName);
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            new WeatherApp().setVisible(true);
        });
    }
}
```

2. **README File:**

Java Swing Weather Application

A full-featured Java Swing desktop application that displays real-time weather data using the [OpenWeatherMap API](https://openweathermap.org/api). This application includes a modern, user-friendly GUI, live weather icons, short-term forecasts, unit conversions, dynamic backgrounds based on time of day, and a search history log.

Features

API Integration
- Connects to OpenWeatherMap API to fetch:
  - Current weather data
  - 5-timepoint short-term forecast

GUI Design
- Built entirely using Java Swing.
- Clean, organized layout with:
  - City name input
  - Unit selector (Celsius / Fahrenheit)
  - Weather info display area
  - Weather condition icon
  - Search history panel
  - Dynamic background

Weather Information Displayed
- City name
- Weather condition (e.g., Clear, Clouds)
- Description (e.g., scattered clouds)
- Temperature (in selected units)
- Humidity (%)
- Wind speed (in selected units)
- 5 short-term forecast timepoints (temperature + condition)

Icon Representation
- Uses OpenWeatherMap icon codes to show weather images like:
  - Sun for clear sky
  - Clouds for cloudy weather
  - Rain for rainy conditions

Unit Conversion
- Switch between:
  - Celsius (metric system)
  - Fahrenheit (imperial system)
  - Wind: m/s or mph

Error Handling
- Alerts user when:
  - City name is invalid or not found
  - API call fails due to connection issues or invalid key

Search History Tracking
- Displays a history list of all searched cities
- Includes timestamp for each search

Dynamic Backgrounds
- Background image changes based on local time of the city:
  - Morning (6 AM – 12 PM)
  - Afternoon (12 PM – 6 PM)
  - Sunset (6 PM – 8 PM)
  - Night (8 PM – 6 AM)

Prerequisites

- Java JDK 8 or above
- Internet connection (for API calls)
- [OpenWeatherMap API Key](https://openweathermap.org/appid)
- [org.json library](https://github.com/stleary/JSON-java) (for parsing JSON)