KONGU ENGINEERING COLLEGE

(Autonomous)

Perundurai, Erode – 638 060

DEPARTMENT OF ELECTRONICS AND INSTRUMENTATION ENGINEERING

WEATHER FORCASTING SYSTEM

A MICROPROJECT REPORT

for

JAVA PROGRAMMING (22ITC31)

Submitted by

DEVISRI P(23EIR019)

DHAKSHINYA Y(23EIR020)

DHANRAJ T(23EIR021)

KONGU ENGINEERING COLLEGE

(Autonomous)

Perundurai, Erode – 638 060

DEPARTMENT OF ELECTRONICS AND INSTRUMENTATION ENGINEERING

BONAFIDE CERTIFICATE

Name & Roll No.

DEVISRI P(23EIR019)

DHAKSHINYA Y(23EIR020)

DHNRAJ T(23EIR021)

Course Code	: **22ITC31**
Course Name	: **JAVA PROGRAMMING**
Semester	**III**

Certified that this is a bonafide record of work for application project done by the above students for **22ITC31-JAVA PROGRAMMING** during the academic year **2024-2025.**

Submitted for the Viva Voce Examination held on ______________________

Faculty In-Charge **Year In-charge**

<u>**TABLE OF CONTENTS**</u>

Weather Forecasting System: A Menu-Driven Application for City Temperature Management

Abstract

The **Weather Forecasting System** is a Java-based application designed to manage and display temperature records for multiple cities. The system provides functionalities to add, update, delete, and view city temperature data, with temperature conversion between Celsius and Fahrenheit. Utilizing a menu-driven interface, the program demonstrates key object-oriented programming principles such as encapsulation and modularity. This report outlines the problem statement, methodology, implementation, and results achieved using the application.

Problem Statement

In modern weather applications, managing temperature records for various cities dynamically and efficiently is critical. There is a need for a lightweight system that enables the user to interactively add, update, or delete temperature records and display them in different formats. This project addresses this need by creating a simple console-based weather forecasting system to manage city-wise temperatures.

Methodology

1. **Requirements Analysis**:

- Identify operations required for city temperature management: Add, update, delete, and display.
- Include functionality for temperature conversion from Celsius to Fahrenheit.

2. **Design**:

- Create a modular system with separate classes for functionality (`WeatherForecastingSystem`) and user interaction (`Main` class).
- Use `ArrayList` for dynamic storage of city and temperature data.

3. **Implementation Steps**:

- Design methods for adding, updating, deleting, and displaying city temperatures.
- Implement a utility function for Celsius-to-Fahrenheit conversion.
- Develop a menu-driven console-based interface.

4. **Testing**:

- Test each functionality independently and validate the results for different input cases.

Implementation

Class Design

- **Weather Forecasting System**: Manages temperature data and provides operations like add, update, delete, and display.
- **Main Class**: Provides a menu-driven interface to interact with the user.

Key Features

1. Add a city and its temperature in Celsius.
2. Update the temperature of an existing city.
3. Delete a city from the records.
4. Insert a city and temperature at a specific position in the list.

5. Convert temperatures from Celsius to Fahrenheit.
6. Display records in both Celsius and Fahrenheit formats.

Results and Discussion

1. **Functionality Verification**:
- Successfully added, updated, and deleted city-temperature pairs dynamically.
- Displayed city temperatures accurately in both Celsius and Fahrenheit.

2. **Efficiency**:
- The use of `arraylist` allowed dynamic handling of records without predetermined limits.

3. **Limitations**:
- No data persistence: All records are lost when the program exits.
- Limited input validation: Users can input invalid data, causing errors.

4. **Future Enhancements**:
- Implement database or file-based storage for persistence.
- Improve error handling and validation for robust input processing.

Conclusion

The **Weather Forecasting System** successfully demonstrates the implementation of a dynamic temperature management application using Java. It showcases the application of object-oriented programming and a menu-driven interface for user interaction. While functional, the system can be further enhanced with persistent storage and additional features for real-world applications.

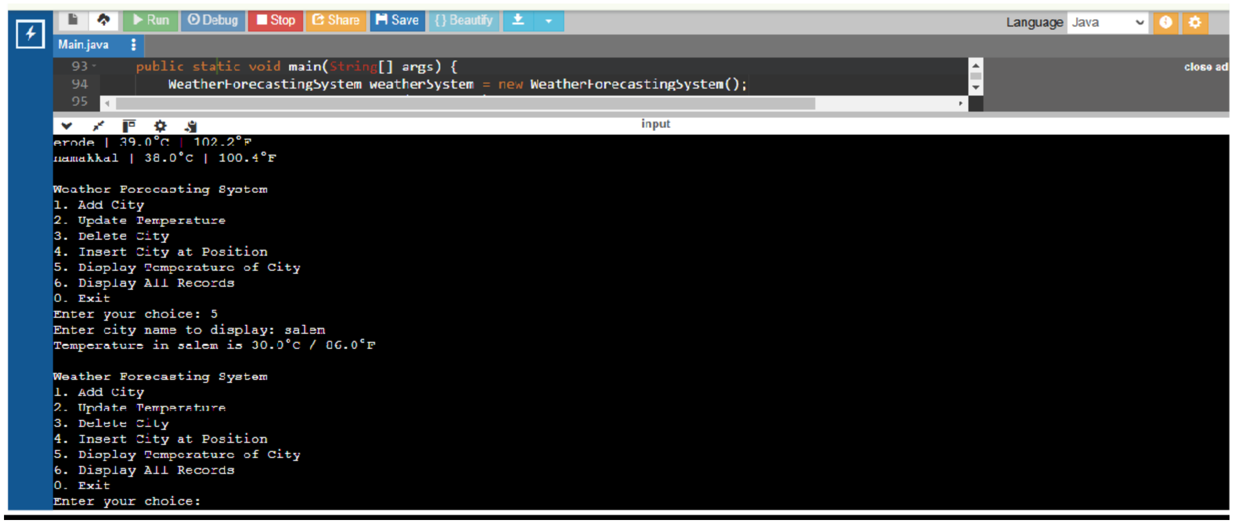
Sample coding

```
import java.util.Scanner;

public class WeatherForecastingSystem {

    private String[] cities;
    private double[] temperatures;
    private double[] previousDayTemperatures;

    public WeatherForecastingSystem() {
        cities = new String[0];
        temperatures = new double[0];
        previousDayTemperatures = new double[0];
    }

    public double convertToFahrenheit(double celsius) {
        return (celsius * 9 / 5) + 32;
    }

    public void updateTemperature(String city, double newTemperature) {
        int index = findCityIndex(city);
        if (index != -1) {
            temperatures[index] = newTemperature;
            System.out.println("Temperature of " + city + " updated to " + newTemperature + "°C.");
        } else {
            System.out.println("City not found in the list.");
        }
    }

    public void addCity(String city, double temperature) {
        String[] newCities = new String[cities.length + 1];
        double[] newTemperatures = new double[temperatures.length + 1];
        double[] newPreviousDayTemperatures = new double[previousDayTemperatures.length + 1];

        for (int i = 0; i < cities.length; i++) {
            newCities[i] = cities[i];
            newTemperatures[i] = temperatures[i];
            newPreviousDayTemperatures[i] = previousDayTemperatures[i];
        }

        newCities[cities.length] = city;
        newTemperatures[temperatures.length] = temperature;
        newPreviousDayTemperatures[previousDayTemperatures.length] = temperature;

        cities = newCities;
        temperatures = newTemperatures;
        previousDayTemperatures = newPreviousDayTemperatures;
        System.out.println("City " + city + " added with temperature " + temperature + "°C.");
    }

    public void deleteCity(String city) {
        int index = findCityIndex(city);
        if (index != -1) {
            String[] newCities = new String[cities.length - 1];
            double[] newTemperatures = new double[temperatures.length - 1];
            double[] newPreviousDayTemperatures = new double[previousDayTemperatures.length -
1];

            for (int i = 0; i < index; i++) {
                newCities[i] = cities[i];
                newTemperatures[i] = temperatures[i];
                newPreviousDayTemperatures[i] = previousDayTemperatures[i];
            }
            for (int i = index + 1; i < cities.length; i++) {
                newCities[i - 1] = cities[i];
                newTemperatures[i - 1] = temperatures[i];
                newPreviousDayTemperatures[i - 1] = previousDayTemperatures[i];
            }

            cities = newCities;
            temperatures = newTemperatures;
            previousDayTemperatures = newPreviousDayTemperatures;

            System.out.println("City " + city + " has been deleted.");
        } else {
            System.out.println("City not found in the list.");
        }
    }

    public void insertCityAtPosition(String city, double temperature, int position) {
        if (position < 0 || position > cities.length) {
            System.out.println("Invalid position.");
            return;
        }

        String[] newCities = new String[cities.length + 1];
        double[] newTemperatures = new double[temperatures.length + 1];
        double[] newPreviousDayTemperatures = new double[previousDayTemperatures.length + 1];

        for (int i = 0; i < position; i++) {
            newCities[i] = cities[i];
            newTemperatures[i] = temperatures[i];
            newPreviousDayTemperatures[i] = previousDayTemperatures[i];
        }

        newCities[position] = city;
        newTemperatures[position] = temperature;
        newPreviousDayTemperatures[position] = temperature;

        for (int i = position; i < cities.length; i++) {
            newCities[i + 1] = cities[i];
            newTemperatures[i + 1] = temperatures[i];
            newPreviousDayTemperatures[i + 1] = previousDayTemperatures[i];
        }

        cities = newCities;
        temperatures = newTemperatures;
        previousDayTemperatures = newPreviousDayTemperatures;

        System.out.println("City " + city + " inserted at position " + position + ".");
    }

    public void displayPreviousDayRecord(String city) {
        int index = findCityIndex(city);
        if (index != -1) {
            System.out.println("Previous day's temperature for " + city + ": " +
previousDayTemperatures[index] + "°C.");
        } else {
            System.out.println("City not found in the list.");
        }
    }

    public void displayAllCitiesAndTemperatures() {
        if (cities.length == 0) {
            System.out.println("No cities are available in the system.");
        } else {
            System.out.println("Cities and their temperatures:");
            for (int i = 0; i < cities.length; i++) {
                System.out.println(cities[i] + ": " + temperatures[i] + "°C");
            }
        }
    }

    private int findCityIndex(String city) {
        for (int i = 0; i < cities.length; i++) {
            if (cities[i].equalsIgnoreCase(city)) {
                return i;
            }
        }
        return -1;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        WeatherForecastingSystem system = new WeatherForecastingSystem();
```

```
while (true) {
    System.out.println("\nWeather Forecasting System");
    System.out.println("1. Add City");
    System.out.println("2. Delete City");
    System.out.println("3. Update Temperature");
    System.out.println("4. Convert Temperature to Fahrenheit");
    System.out.println("5. Display Previous Day's Record");
    System.out.println("6. Insert City at Specific Position");
    System.out.println("7. Show All Cities and Temperatures");
    System.out.println("8. Exit"); // New option
    System.out.print("Enter your choice: ");
    int choice = scanner.nextInt();
    scanner.nextLine(); // Consume the newline character

    switch (choice) {
        case 1:
            System.out.print("Enter city name: ");
            String cityToAdd = scanner.nextLine();
            System.out.print("Enter temperature (Celsius): ");
            double tempToAdd = scanner.nextDouble();
            system.addCity(cityToAdd, tempToAdd);
            break;
        case 2:
            System.out.print("Enter city name to delete: ");
            String cityToDelete = scanner.nextLine();
            system.deleteCity(cityToDelete);
            break;
        case 3:
            System.out.print("Enter city name to update: ");
            String cityToUpdate = scanner.nextLine();
            System.out.print("Enter new temperature (Celsius): ");
            double newTemp = scanner.nextDouble();
            system.updateTemperature(cityToUpdate, newTemp);
            break;
        case 4:
            System.out.print("Enter temperature in Celsius to convert: ");
            double celsius = scanner.nextDouble();
            double fahrenheit = system.convertToFahrenheit(celsius);
            System.out.println("Temperature in Fahrenheit: " + fahrenheit);
            break;
        case 5:
            System.out.print("Enter city name to get previous day's record: ");
            String cityToCheck = scanner.nextLine();
            system.displayPreviousDayRecord(cityToCheck);
            break;
        case 6:
            System.out.print("Enter city name to insert: ");
            String cityToInsert = scanner.nextLine();
```

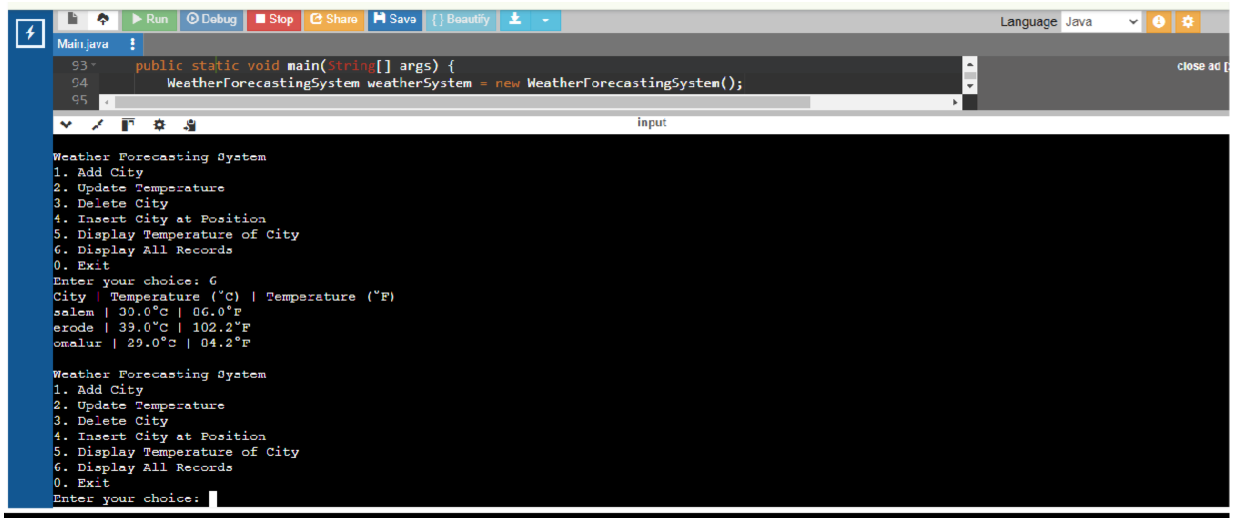
```
                System.out.print("Enter temperature (Celsius): ");
                double tempToInsert = scanner.nextDouble();
                System.out.print("Enter position to insert city at: ");
                int position = scanner.nextInt();
                system.insertCityAtPosition(cityToInsert, tempToInsert, position);
                break;
            case 8:
                System.out.println("Exiting system...");
                scanner.close();
                return;
            case 7:
                system.displayAllCitiesAndTemperatures(); // Show all cities and temperatures
                break;
            default:
                System.out.println("Invalid choice, please try again.");
        }
    }
  }
}
```

OUTPUTS



```
public static void main(String[] args) {
    WeatherForecastingSystem weatherSystem = new WeatherForecastingSystem();
```

```
erode | 39.0°C | 102.2°F
namakkal | 38.0°C | 100.4°F

Weather Forecasting System
1. Add City
2. Update Temperature
3. Delete City
4. Insert City at Position
5. Display Temperature of City
6. Display All Records
0. Exit
Enter your choice: 5
Enter city name to display: salem
Temperature in salem is 30.0°C / 86.0°F

Weather Forecasting System
1. Add City
2. Update Temperature
3. Delete City
4. Insert City at Position
5. Display Temperature of City
6. Display All Records
0. Exit
Enter your choice:
```



```
public static void main(String[] args) {
    WeatherForecastingSystem weatherSystem = new WeatherForecastingSystem();
```

```
Weather Forecasting System
1. Add City
2. Update Temperature
3. Delete City
4. Insert City at Position
5. Display Temperature of City
6. Display All Records
0. Exit
Enter your choice: 6
City | Temperature (°C) | Temperature (°F)
salem | 30.0°C | 86.0°F
erode | 39.0°C | 102.2°F
omalur | 29.0°C | 84.2°F

Weather Forecasting System
1. Add City
2. Update Temperature
3. Delete City
4. Insert City at Position
5. Display Temperature of City
6. Display All Records
0. Exit
Enter your choice:
```

Faculty Incharge **Academic Coordinator**

HOD