Southeast University

Department of Computer Science and Engineering (CSE)

School of Sciences and Engineering

Semester: (Summer, Year: 2025)

LAB REPORT

Course Title: Introduction to Programming Language II (Java) Lab

Course Code: CSE282.2

Batch: 65

Lab Experiment Name: Implement Flower Management System using OOP Concepts in Java.

Student Details

	Name	ID
1.	Md. Mahmud Hossain	2023200000799

Submission Date : **05-09-25**

Course Teacher's Name : **Dr. Mohammed Ashikur Rahman**

Lab Report Status

Marks: ………………………………… **Signature:**....................

Comments:... **Date:**............................

Table of contents

Lab Task: Implement Flower Management System using OOP Concepts in Java.

Objective:

The objective of this lab task is to design and implement a **Flower Management System** in Java that demonstrates:

- Object-Oriented Programming (OOP) concepts such as **inheritance, polymorphism, encapsulation, and method overloading/overriding**.
- Usage of **ArrayList** and **Java Streams** for dynamic storage and data filtering.
- **Menu-driven console interaction** with input validation for real-time flower shop management.
- Handling **expiry dates** using LocalDate and DateTimeFormatter.

Understanding the requirements of the task:

The Flower Management System was designed to handle the addition, removal, and display of flowers, along with expiry tracking and support for different flower types (Rose, Tulip). Beyond the immediate requirements, future steps were delineated, including scalability with ArrayList, efficient filtering with Streams, and robust expiry management with the Java Date/Time API.

Problem:

Implement a Java program that store the flower's information's like name, expiry date and can show the flower details based on expiry date. Additionally, user can add and remove the flower details in this program.

Solution:

Analysing requirements and analysis:

Key challenges such as expiry tracking, handling multiple flower types through inheritance, scalability using ArrayList, and providing a user-friendly menu interface. The solution analysis also anticipated real-world needs, like automatic identification of expired flowers, ensuring system reliability.

Problem Analysis:

A flower shop requires a computerized system that can efficiently manage its inventory by allowing the addition of flowers, whether generic or specific types such as Rose and Tulip. Each flower must store attributes like name, expiry date, price, and type-specific details to ensure accurate record-keeping. The system should also provide the ability to remove flowers by name and display all flowers or filter them based on their expiry dates. Additionally, it must automatically identify expired flowers to maintain the quality of stock. This system addresses key challenges such as tracking expiry dates, handling different flower types through inheritance, and offering flexibility in managing an unlimited number of flowers using an ArrayList. To ensure ease of use, the program is designed with a clear and user-friendly menu-driven interface.

Background Theory:

1. **Encapsulation** – Private attributes in Flower, accessed via getters/setters.
2. **Inheritance** – Subclasses Rose and Tulip extend Flower to reuse and extend functionality.
3. **Polymorphism** – Overriding displayDetails() in subclasses provides customized behavior for different flower types.
4. **Method Overloading** – Methods like displayDetails() and addFlower() are overloaded to handle different inputs.
5. **Collections Framework** – ArrayList is used to dynamically store flowers instead of fixed-size arrays.
6. **Streams & Lambda Expressions** – Used for filtering flowers by expiry date and identifying expired flowers.
7. **Date and Time API** – LocalDate and DateTimeFormatter ensure reliable date handling and formatting.

Develop the algorithm:

The algorithm developed in a structured solution.

- A **base class (Flower)** captured core attributes and methods.
- **Subclasses (Rose, Tulip)** extended the design with specialized attributes.
- A **manager class (FlowerShop)** handled operations using ArrayList and Streams.
- A **main class (FlowerManagementSystem)** provided menu-driven interaction with validation.

Algorithm Design:

1. Define Base Class – **Flower**

- Attributes: name, expiryDate, price.
- Methods: constructors, getters/setters, displayDetails(), isExpired().

2. Define Subclasses – **Rose** and **Tulip**

- Rose: additional attributes color, hasThorns.
- Tulip: additional attributes petalCount, variety.
- Override displayDetails() for specific output.

3. Create Manager Class – **FlowerShop**

- Use an ArrayList<Flower> to store flowers.
- Methods: addFlower(), removeFlower(), displayAllFlowers(), displayFlowersByExpiryDate(), displayExpiredFlowers().
- Use **stream filtering** for expiry-based searches.

4. Implement Main Class – **FlowerManagementSystem**

- Show a menu with options (add, remove, display, search, exit).
- Take user input with validation using Scanner.
- Call corresponding methods in FlowerShop.

UML Class Diagram:

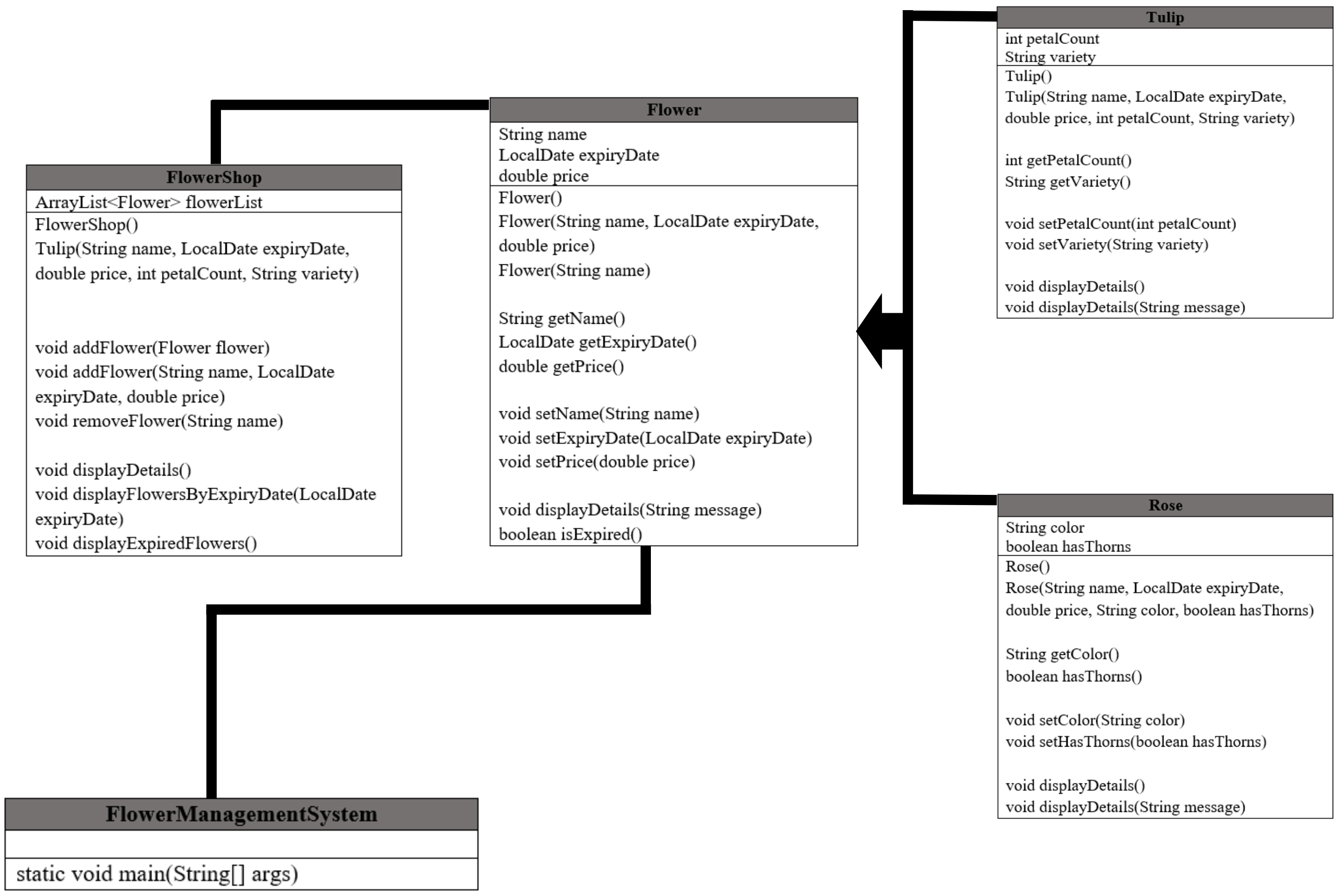

Tulip
int petalCount
String variety
Tulip()
Tulip(String name, LocalDate expiryDate, double price, int petalCount, String variety)
int getPetalCount()
String getVariety()
void setPetalCount(int petalCount)
void setVariety(String variety)
void displayDetails()
void displayDetails(String message)
Flower
String name
LocalDate expiryDate
double price
Flower()
Flower(String name, LocalDate expiryDate, double price)
Flower(String name)
String getName()
LocalDate getExpiryDate()
double getPrice()
void setName(String name)
void setExpiryDate(LocalDate expiryDate)
void setPrice(double price)
void displayDetails(String message)
boolean isExpired()
FlowerShop
ArrayList<Flower> flowerList
FlowerShop()
Tulip(String name, LocalDate expiryDate, double price, int petalCount, String variety)
void addFlower(Flower flower)
void addFlower(String name, LocalDate expiryDate, double price)
void removeFlower(String name)
void displayDetails()
void displayFlowersByExpiryDate(LocalDate expiryDate)
void displayExpiredFlowers()
Rose
String color
boolean hasThorns
Rose()
Rose(String name, LocalDate expiryDate, double price, String color, boolean hasThorns)
String getColor()
boolean hasThorns()
void setColor(String color)
void setHasThorns(boolean hasThorns)
void displayDetails()
void displayDetails(String message)
FlowerManagementSystem
static void main(String[] args)

Figure: Class Diagram

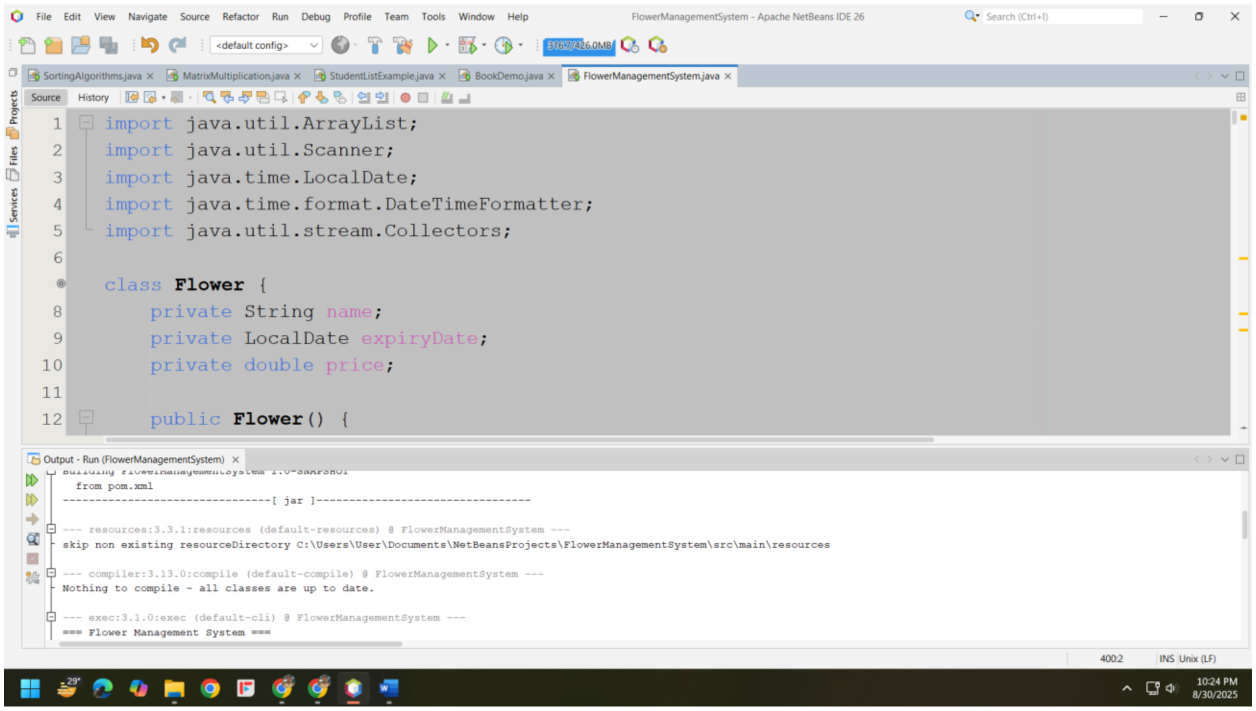
Code:

```
import java.util.ArrayList;
import java.util.Scanner;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.stream.Collectors;

class Flower {
    private String name;
    private LocalDate expiryDate;
    private double price;

    public Flower() {
        this.name = "Unknown";
        this.expiryDate = LocalDate.now().plusDays(7);
        this.price = 0.0;
    }

    public Flower(String name, LocalDate expiryDate, double price) {
        this.name = name;
        this.expiryDate = expiryDate;
        this.price = price;
    }

    public Flower(String name) {
        this.name = name;
        this.expiryDate = LocalDate.now().plusDays(7);
        this.price = 0.0;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public LocalDate getExpiryDate() {
        return expiryDate;
    }

    public void setExpiryDate(LocalDate expiryDate) {
        this.expiryDate = expiryDate;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    public void displayDetails() {
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
        System.out.println("Flower Name: " + name + ", Expiry Date: " +
                        expiryDate.format(formatter) + ", Price: $" + price);
    }

    public void displayDetails(String message) {
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
        System.out.println(message);
        System.out.println("Flower Name: " + name + ", Expiry Date: " +
                        expiryDate.format(formatter) + ", Price: $" + price);
    }

    public boolean isExpired() {
        return LocalDate.now().isAfter(expiryDate);
    }
}

class Rose extends Flower {
    private String color;
    private boolean hasThorns;

    public Rose() {
        super();
        this.color = "Red";
        this.hasThorns = true;
    }

    public Rose(String name, LocalDate expiryDate, double price, String color, boolean hasThorns) {
        super(name, expiryDate, price);
        this.color = color;
        this.hasThorns = hasThorns;
    }

    public String getColor() {
        return color;
    }

    public void setColor(String color) {
        this.color = color;
    }

    public boolean hasThorns() {
        return hasThorns;
    }

    public void setHasThorns(boolean hasThorns) {
        this.hasThorns = hasThorns;
    }

    @Override
    public void displayDetails() {
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
        System.out.println("Rose - Name: " + getName() + ", Expiry: " +
                getExpiryDate().format(formatter) + ", Price: $" + getPrice() +
                ", Color: " + color + ", Has Thorns: " + hasThorns);
    }

    @Override
    public void displayDetails(String message) {
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
        System.out.println(message);
        System.out.println("Rose - Name: " + getName() + ", Expiry: " +
                getExpiryDate().format(formatter) + ", Price: $" + getPrice() +
                ", Color: " + color + ", Has Thorns: " + hasThorns);
    }
}

class Tulip extends Flower {
    private int petalCount;
    private String variety;

    public Tulip() {
        super();
        this.petalCount = 6;
        this.variety = "Standard";
    }

    public Tulip(String name, LocalDate expiryDate, double price, int petalCount, String variety) {
        super(name, expiryDate, price);
        this.petalCount = petalCount;
        this.variety = variety;
    }

    public int getPetalCount() {
        return petalCount;
    }

    public void setPetalCount(int petalCount) {
        this.petalCount = petalCount;
    }

    public String getVariety() {
        return variety;
    }

    public void setVariety(String variety) {
        this.variety = variety;
    }

    @Override
    public void displayDetails() {
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
        System.out.println("Tulip - Name: " + getName() + ", Expiry: " +
                getExpiryDate().format(formatter) + ", Price: $" + getPrice() +
                ", Petal Count: " + petalCount + ", Variety: " + variety);
    }

    @Override
    public void displayDetails(String message) {
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
        System.out.println(message);
        System.out.println("Tulip - Name: " + getName() + ", Expiry: " +
                getExpiryDate().format(formatter) + ", Price: $" + getPrice() +
                ", Petal Count: " + petalCount + ", Variety: " + variety);
    }
}

class FlowerShop {
    private ArrayList<Flower> flowerList;

    public FlowerShop() {
        flowerList = new ArrayList<>();
    }

    public void addFlower(Flower flower) {
        flowerList.add(flower);
        System.out.println("Flower added successfully.");
    }

    public void addFlower(String name, LocalDate expiryDate, double price) {
        Flower flower = new Flower(name, expiryDate, price);
        flowerList.add(flower);
        System.out.println("Flower added successfully.");
    }

    public void removeFlower(String name) {
        boolean removed = flowerList.removeIf(flower -> flower.getName().equalsIgnoreCase(name));
        if (removed) {
            System.out.println("Flower removed successfully.");
        } else {
            System.out.println("Flower not found.");
        }
    }

    public void displayAllFlowers() {
        if (flowerList.isEmpty()) {
            System.out.println("No flowers in the shop.");
        } else {
            System.out.println("\n=== All Flowers ===");
            for (Flower f : flowerList) {
                f.displayDetails();
            }
        }
    }

    public void displayFlowersByExpiryDate(LocalDate expiryDate) {
        ArrayList<Flower> filteredFlowers = flowerList.stream()
                .filter(flower -> flower.getExpiryDate().equals(expiryDate))
                .collect(Collectors.toCollection(ArrayList::new));

        if (filteredFlowers.isEmpty()) {
            System.out.println("No flowers found with expiry date: " + expiryDate);
        } else {
            System.out.println("\n=== Flowers with Expiry Date: " + expiryDate + " ===");
            for (Flower f : filteredFlowers) {
                f.displayDetails();
            }
        }
    }

    public void displayExpiredFlowers() {
        ArrayList<Flower> expiredFlowers = flowerList.stream()
```

```
                .filter(Flower::isExpired)
                .collect(Collectors.toCollection(ArrayList::new));

        if (expiredFlowers.isEmpty()) {
            System.out.println("No expired flowers found.");
        } else {
            System.out.println("\n=== Expired Flowers ===");
            for (Flower f : expiredFlowers) {
                f.displayDetails("EXPIRED: ");
            }
        }
    }
}

public class FlowerManagementSystem {
    public static void main(String[] args) {
        System.out.println("=== Flower Management System ===");
        System.out.println("Developed for CSE282 Lab Tasks");

        FlowerShop shop = new FlowerShop();
        Scanner scanner = new Scanner(System.in);
        int choice;
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");

        do {
            System.out.println("\n--- Flower Management System Menu ---");
            System.out.println("1. Add Flower");
            System.out.println("2. Add Rose");
            System.out.println("3. Add Tulip");
            System.out.println("4. Remove Flower");
            System.out.println("5. Display All Flowers");
            System.out.println("6. Display Flowers by Expiry Date");
            System.out.println("7. Display Expired Flowers");
            System.out.println("0. Exit");
            System.out.print("Enter your choice: ");

            while (!scanner.hasNextInt()) {
                System.out.println("Please enter a valid number.");
                scanner.next();
            }
            choice = scanner.nextInt();
            scanner.nextLine();

            switch (choice) {
                case 1:
                    System.out.print("Enter flower name: ");
    String name = scanner.nextLine();

    LocalDate date = null;
    boolean validDate = false;
    while (!validDate) {
        System.out.print("Enter expiry date (YYYY-MM-DD): ");
        String dateStr = scanner.nextLine();
        try {
            date = LocalDate.parse(dateStr, formatter);
            validDate = true;
        } catch (Exception e) {
            System.out.println("Invalid date format. Please use YYYY-MM-DD format.");
        }
    }

    System.out.print("Enter price: ");
    double price = scanner.nextDouble();
    scanner.nextLine();

    shop.addFlower(name, date, price);
    break;

case 2:
    System.out.print("Enter rose name: ");
    String roseName = scanner.nextLine();

    LocalDate roseDate = null;
    boolean validRoseDate = false;
    while (!validRoseDate) {
        System.out.print("Enter expiry date (YYYY-MM-DD): ");
        String dateStr = scanner.nextLine();
        try {
            roseDate = LocalDate.parse(dateStr, formatter);
            validRoseDate = true;
        } catch (Exception e) {
            System.out.println("Invalid date format. Please use YYYY-MM-DD format.");
        }
    }

    System.out.print("Enter price: ");
    double rosePrice = scanner.nextDouble();
    scanner.nextLine();

    System.out.print("Enter color: ");
    String color = scanner.nextLine();
    System.out.print("Does it have thorns? (true/false): ");
    boolean hasThorns = scanner.nextBoolean();
    scanner.nextLine();

    Rose rose = new Rose(roseName, roseDate, rosePrice, color, hasThorns);
    shop.addFlower(rose);
    break;

case 3:
    System.out.print("Enter tulip name: ");
    String tulipName = scanner.nextLine();

    LocalDate tulipDate = null;
    boolean validTulipDate = false;
    while (!validTulipDate) {
        System.out.print("Enter expiry date (YYYY-MM-DD): ");
        String dateStr = scanner.nextLine();
        try {
            tulipDate = LocalDate.parse(dateStr, formatter);
            validTulipDate = true;
        } catch (Exception e) {
            System.out.println("Invalid date format. Please use YYYY-MM-DD format.");
        }
    }

    System.out.print("Enter price: ");
    double tulipPrice = scanner.nextDouble();
    scanner.nextLine();

    System.out.print("Enter petal count: ");
    int petalCount = scanner.nextInt();
    scanner.nextLine();

    System.out.print("Enter variety: ");
    String variety = scanner.nextLine();

    Tulip tulip = new Tulip(tulipName, tulipDate, tulipPrice, petalCount, variety);
    shop.addFlower(tulip);
    break;

case 4:
    System.out.print("Enter flower name to remove: ");
    String removeName = scanner.nextLine();
    shop.removeFlower(removeName);
    break;
```

```
                case 5:
                    shop.displayAllFlowers();
                    break;

                case 6:
                    LocalDate searchDate = null;
                    boolean validSearchDate = false;
                    while (!validSearchDate) {
                        System.out.print("Enter expiry date to search (YYYY-MM-DD): ");
                        String dateStr = scanner.nextLine();
                        try {
                            searchDate = LocalDate.parse(dateStr, formatter);
                            validSearchDate = true;
                        } catch (Exception e) {
                            System.out.println("Invalid date format. Please use YYYY-MM-DD format.");
                        }
                    }
                    shop.displayFlowersByExpiryDate(searchDate);
                    break;

                case 7:
                    shop.displayExpiredFlowers();
                    break;

                case 0:
                    System.out.println("Exiting program. Thank you!");
                    break;

                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        } while (choice != 0);

        scanner.close();
    }
}
```

Output:



```java
import java.util.ArrayList;
import java.util.Scanner;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.stream.Collectors;

class Flower {
    private String name;
    private LocalDate expiryDate;
    private double price;

    public Flower() {
```

```
Building FlowerManagementSystem 1.0-SNAPSHOT
  from pom.xml
--------------------------------[ jar ]---------------------------------

--- resources:3.3.1:resources (default-resources) @ FlowerManagementSystem ---
skip non existing resourceDirectory C:\Users\User\Documents\NetBeansProjects\FlowerManagementSystem\src\main\resources

--- compiler:3.13.0:compile (default-compile) @ FlowerManagementSystem ---
Nothing to compile - all classes are up to date.

--- exec:3.1.0:exec (default-cli) @ FlowerManagementSystem ---
=== Flower Management System ===
```

Figure: Output

Report Formatting:

It follows a logical and professional structure: Objective → Problem → Problem Analysis → Background Theory → Algorithm Design → UML Diagram → Code → Output → Conclusion. Each section is well-connected, with UML diagrams, structured explanations, and clear flow.

Conclusion:

The **Flower Management System** system allows efficient management of flowers in a shop by handling multiple flower types, expiry tracking, and flexible data storage with ArrayList. This lab enhanced understanding of:

- OOP (inheritance, polymorphism, encapsulation).
- Collections (ArrayList, LinkedList).
- Java Date/Time API.
- Menu-driven interactive programs.

This lab not only improved my Java coding skills but also helped me understand how OOP concepts apply in real-world inventory systems. I learned the importance of using Collections, Java Streams, and Date/Time APIs for efficient data management.

--End--