Food Donation System

A Java Console Application using OOP & Design Patterns

K.Vineesha | Roll no: 24501A05A0

Prasad V Potluri Siddhartha Institute of Technology

Using technology to reduce food waste and increase social impact.

PROBLEM & OBJECTIVE

Addressing Food Waste: A Software Solution

The Problem Statement

- Significant food waste daily (homes, hostels, functions).
- NGOs and volunteers struggle with quick donor connections.
- Lack of simple platforms for tracking donations.

Our Project Objective

- Develop a menu-driven Java system to:
- Connect Donors, Volunteers, and Admin seamlessly.
- Track food donations efficiently.
- Improve awareness on food waste impact.
- Ensure better coordination among stakeholders.

> This project addresses a real-world social problem using robust software design.

OOP PRINCIPLES

Object-Oriented Programming in Action

Class & Object

Illustrated by `Donation`, `Main`, and `FoodFactory` classes as blueprints for entities and processes.

Encapsulation

The `Donation` class bundles data and methods, protecting internal state and promoting data integrity.

Abstraction

Users interact through simplified menus, hiding complex backend details for ease of use.

Modularity

Separate methods for Donor, Volunteer, and Admin roles ensure clear code organisation.

Reusability

Common input methods (`safeInt`, `safeString`) are reused across the application, reducing redundancy.

> The project demonstrates a strong foundation in Object-Oriented Design principles.

ARCHITECTURE

System Architecture & Design Approach

This project adopts a modular layered architecture for clarity and scalability.

1

Presentation Layer

User interface with dedicated menus for Donor, Volunteer, Admin, and Awareness.

2

Business Logic Layer

Manages core functionalities: donation handling, report generation, data validation, and application flow control.

3

Model Layer

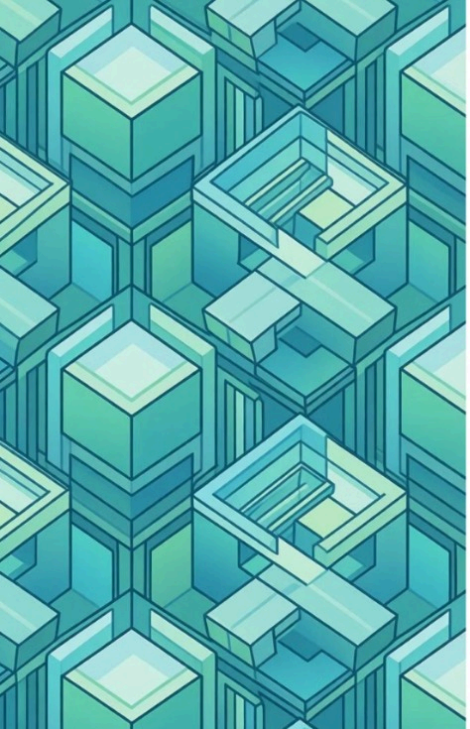
Defines data structures, like the `Donation` class, representing real-world entities.

4

Utility Layer

Contains helper classes such as `FoodFactory` and methods for input validation.

> This clear separation improves readability, scalability, and maintainability.



DESIGN PATTERNS

Applied Design Patterns for Robustness

1

Factory Pattern

Implemented in `FoodFactory` to centralise food type creation, allowing scalable addition of new types.

`Example: FoodFactory.getFoodType(choice)`

2

Singleton-style Shared Resource

`Scanner` and `ArrayList` are globally shared, ensuring a single source of truth for critical data.

3

Controller Pattern

The `Main` class acts as a controller, orchestrating application flow between various modules.

4

Model Pattern

The `Donation` class serves as a model, accurately representing real-world donation entities.

FEATURES & CONCLUSION

Key Features & Project Conclusion

Key System Features

Donor

Add, view, update donations; track contribution history.

Volunteer

Accept donations, filter by area, submit feedback.

Admin

Monitor system performance, view comprehensive statistics.

Awareness Module

Promotes social impact and reduces food waste.

Conclusion

This project demonstrates the effective application of:

- **Core Java** for fundamental programming.
- **OOP Principles** for structured and maintainable code.
- **Design Patterns** for robust and scalable solutions.

Ultimately, solving real-world challenges through thoughtful software engineering.