DOCUMENT MANAGEMENT APPLICATION

Project Documentation

Table of Contents

Overview

The Document Management Application is a Java desktop program designed for uploading, managing, and assigning roles to digital documents (PDF, Word, Excel). It utilizes SQL Server for persistent storage and integrates five major design patterns for scalability and maintainability.

Design Patterns and Justification

1

Singleton Pattern

- **Purpose**: Ensures only one instance of key classes like DocumentRepository and UserAccessControl exists.
- **Justification**: Centralizes document management and role assignments to avoid inconsistencies.

2

Factory Pattern

- **Purpose**: Simplifies the creation of objects like Document and UserRole based on type.
- **Justification**: Encapsulates object creation logic, allowing easy addition of new document or role types.

3

Observer Pattern

- **Purpose**: Allows logging of document uploads without coupling the repository to the logging mechanism.
- **Justification**: Ensures decoupled, real-time notifications when documents are added.

4

Command Pattern

- **Purpose:** Encapsulates actions like file uploads and role assignments as reusable objects.
- **Justification:** Decouples the GUI from the logic, improving code modularity and testability.

5

Decorator Pattern

- **Purpose:** Dynamically adds behavior (e.g., logging) to roles without modifying their base classes.
- **Justification:** Supports extension of role functionality without altering the existing hierarchy.

Class Descriptions

1

Main Package

- **DocumentManagementApp:** Entry point for the application, managing the GUI and user interactions**.**
- **DatabaseConnection:** Handles the SQL Server connection.

2

Repository Package

DocumentRepository: Singleton class for managing documents and notifying observers.

3

Access Control Package

UserAccessControl: Singleton class for managing role assignments.

4

Factories Package

- **DocumentFactory:** Creates Document objects based on their type.
- **UserRoleFactory:** Creates UserRole objects based on role type.

5

Commands Package

- **Command (Interface):** Defines the structure for encapsulated actions.
- **UploadDocumentCommand:** Handles file uploads.
- **AssignRoleCommand:** Handles role assignments.

6

Roles and Decorators Package

- **UserRole (Interface):** Defines the behavior of user roles.
- **AdminRole, EditorRole, ViewerRole:** Concrete role implementations.
- **RoleDecorator:** Abstract class for extending role behavior.
- **LoggingRoleDecorator:** Adds logging functionality to roles.

Project Workflow

- **Document Upload:**
  - User selects a file.
  - UploadDocumentCommand creates a document and adds it to DocumentRepository.
  - The document details are saved to the database, and observers log the upload.

- **Role Assignment:**
  - User assigns a role via the GUI.
  - AssignRoleCommand creates a role and assigns it using UserAccessControl.

- **Logging Updates:**
  - Observers receive updates whenever a new document is added.

CHALLENGES AND SOLUTIONS

Challenge	Solution
Supporting multiple document types	Implemented the Factory Pattern to create documents dynamically.
Preventing inconsistent role handling	Used the Singleton Pattern for centralized role management.
Logging updates without tight coupling	Applied the Observer Pattern for decoupled logging.
Decoupling GUI from core logic	Utilized the Command Pattern for modular action handling.
Extending role functionality	Added the Decorator Pattern to dynamically extend role behaviors.

Conclusion

The Document Management Application demonstrates a robust architecture using five design patterns:

- Singleton: Centralized control over document and role management.
- Factory: Simplified object creation for documents and roles.
- Observer: Real-time updates for document uploads.
- Command: Decoupled and reusable action logic.
- Decorator: Dynamic extension of role functionality.