**Name of your final project: Poster Collector App**

**Group:** SE-2219

**Team members: Adilkhasymkyzy Aruzhan, Imanbazarova Akzhan**

# Project Overview

## Project Information

**Project Name:** Poster Collector App  
**Programming Language:** Java

## Project Idea

The Poster Collector App is designed to manage a collection of movie posters, allowing users to interact with and enhance their experience through various features. The project aims to showcase the implementation of several design patterns to achieve flexibility, maintainability, and extensibility in the codebase.

## Purpose of the Work

The primary purpose of this project is to create a practical example that demonstrates the application of key design patterns in software development. By building a Poster Collector App, we aim to provide a clear illustration of how design patterns can be effectively employed to address common challenges in designing and maintaining software systems.

## Objectives of the Work

* Implement features for adding and deleting movie posters.
* Enhance posters with additional information such as authors and comments.
* Implement different strategies for collecting posters.
* Use a factory pattern to create instances of posters.
* Ensure a single, centralized instance of the poster collection using the singleton pattern.
* Notify users about updates to the poster collection using the observer pattern.

# Main Body

## Explanations of Features and Design Patterns

### 1. Add a New Poster

**Description:** Users can add a new movie poster to the collection by providing a name, choosing an author, leaving a comment, and selecting a collection strategy.

**Implementation:** The user is prompted to choose between basic and advanced collection strategies. Basic strategy directly adds the poster to the collection, while the advanced strategy enhances the poster with author and comment information before adding it.

### 2. Delete a Poster

**Description:** Users can delete an existing movie poster from the collection by entering the name of the poster they wish to remove.

**Implementation:** The application searches for the poster by name and, if found, removes it from both the poster collection and the list of collected poster names.

### 3. Show Collected Poster Names

**Description:** Users can view the names of all the movie posters currently collected in the system.

**Implementation:** The application displays a list of collected poster names, providing a quick overview of the posters present in the collection.*.*

### 4. Show Poster Collection History

**Description:** The application maintains a history of actions performed on the poster collection, including when posters were added or deleted.

**Implementation:** The history is displayed, allowing users to understand the sequence of operations performed on the posters.

### Design Patterns

#### Observer Pattern

**Description:** Used to notify collectors about new posters added to the collection.

**Implementation:** The **PosterCollector** class acts as an observer, receiving updates when new posters are added.

#### Decorator Pattern

**Description:** Enhances posters with additional features such as authors and comments.

**Implementation:** The **PosterDecorator** class wraps the basic **ConcretePoster** class, allowing for dynamic enhancement of the poster display.

#### Strategy Pattern

**Description:** Defines different strategies for collecting posters.

**Implementation:** The **PosterCollectionStrategy** interface is implemented by the **BasicCollectionStrategy** and **AdvancedCollectionStrategy** classes, providing flexibility in how posters are collected.

#### Factory Pattern

**Description:** Employed to create instances of posters.

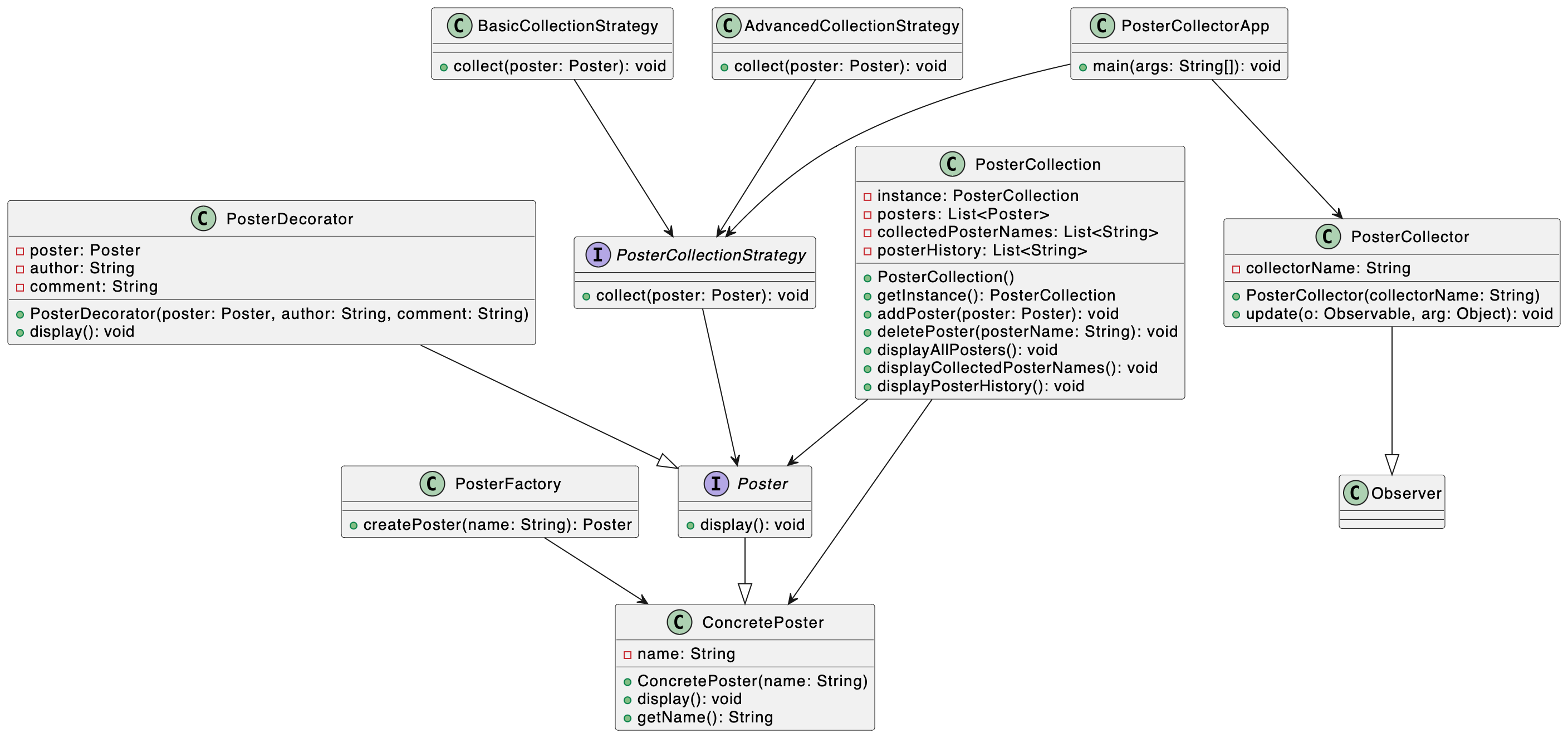
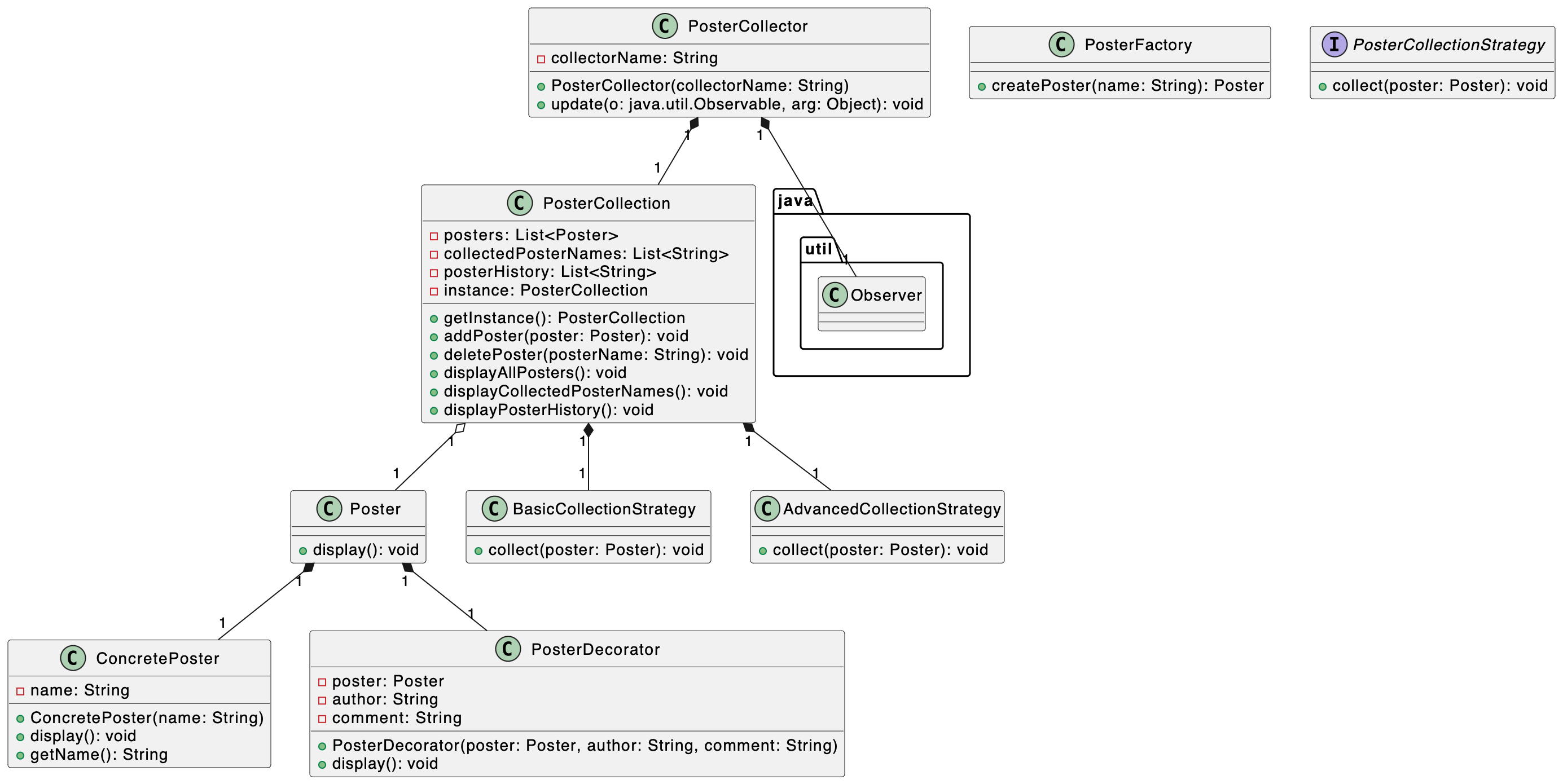
**Implementation:** The **PosterFactory** class creates **ConcretePoster** instances based on the provided name.

#### Singleton Pattern

**Description:** Ensures a single instance of the **PosterCollection** class.

**Implementation:** This ensures a single point of access to the poster collection and avoids multiple instances that could lead to inconsistencies.

## UML Diagram



# Conclusion

## Key Points

The Poster Collector App successfully demonstrates the implementation of key design patterns, including Observer, Decorator, Strategy, Factory, and Singleton. Each pattern addresses specific challenges in the project, contributing to a modular and maintainable codebase.

## Project Outcomes and Challenges

**Outcomes:**

* Successful implementation of features and design patterns.
* Clear illustration of the benefits of design patterns in enhancing code structure and flexibility.

**Challenges:**

* Balancing the complexity of design patterns with the simplicity required for an educational example.
* Ensuring effective communication between objects in the Observer and Strategy patterns.

## Future Improvements

**Potential Improvements:**

* Implement a graphical user interface (GUI) for a more user-friendly experience.
* Expand the range of enhancements in the Decorator pattern.
* Introduce additional collection strategies to further demonstrate the flexibility of the Strategy pattern.

In conclusion, the Poster Collector App provides a practical and educational example of design pattern implementation in Java. It serves as a reference for developers looking to understand and apply these patterns in their projects.