SOLID PRINCIPLES

SREEJAYA V S

CONTENTS

- Single Responsibility Principle (SRP)
- Open/Closed Principle (OCP)
- Liskov Substitution Principle (LSP)
- Interface Segregation Principle (ISP)
- Dependency Inversion Principle (DIP)

Single Responsibility Principle (SRP):

The **Movie** class has the responsibility of representing movie information (title and release year).

The **MovieRating** class represents the combination of a movie and its rating.

The **RatingCalculator** interface defines a single responsibility: calculating movie ratings.

The **RatingOperations** interface defines operations related to movie ratings, adhering to a single responsibility.

Open/Closed Principle (OCP):

The **RatingCalculator** interface is open for extension (new implementations can be added, like **AdvancedRatingCalculator** or **BasicRatingCalculator**) but closed for modification (existing implementations don't need to be changed to accommodate new ones).

The **MovieRatingService** class is open for extension (new rating calculation strategies can be added) but closed for modification (existing code doesn't need to change when a new **RatingCalculator** is introduced).

Liskov Substitution Principle (LSP):

The **MovieRating** class can be substituted for its base class (**Object**) without affecting the correctness of the program.

The **BasicRatingCalculator** and **AdvancedRatingCalculator** can be substituted for the **RatingCalculator** interface without affecting the correctness of the program.

Interface Segregation Principle (ISP):

The **RatingCalculator** and **RatingOperations** interfaces are focused on specific responsibilities, avoiding a "fat" or general-purpose interface.

Clients (like **MovieRatingService**) are not forced to depend on interfaces they do not use.

Dependency Inversion Principle (DIP):

High-level modules (e.g., **MovieRatingService**) depend on abstractions (**RatingCalculator**), not on concrete implementations (**BasicRatingCalculator**, **AdvancedRatingCalculator**).

The choice of rating calculator implementation is injected into the **MovieRatingService** via the constructor, promoting flexibility and ease of testing.

CONCLUSION

RECAP OF SOLID PRINCIPLES APPLIED IN THE MOVIE RATING SYSTEM.

INCREASED MAINTAINABILITY: CHANGES IN ONE PART OF THE SYSTEM DON'T AFFECT OTHERS.

ENHANCED FLEXIBILITY: EASIER TO ADD NEW FEATURES WITHOUT MODIFYING EXISTING CODE.

IMPROVED TESTABILITY: COMPONENTS CAN BE TESTED IN ISOLATION, PROMOTING ROBUST TESTING STRATEGIES.