CLIENT -> Observer design pattern

The Observer design pattern is a behavioral design pattern used to manage the communication between objects in a one-to-many relationship. This pattern ensures that when one object changes state, all its dependents are notified and updated automatically.
Movie Class
- Methods: attach_observer(User), detach_observer(User), notifyObservers()
- This class acts as the subject in the Observer pattern. It maintains a list of observers (users) who are interested in updates. The attach_observer method adds a user to the list of observers, and detach_observer removes a user. The notifyObservers method iterates through the list of observers and calls their update method to notify them of changes.

UpcomingMovie Class (inherits from Movie):
- Methods: updateMovieStatus()
- This subclass represents a specific type of movie that is yet to be released. The updateMovieStatus method changes the movie's status from upcoming to currently in theaters. When this status is updated, the notifyObservers method is triggered to inform all registered users about the change.

User Interface
- Methods: update(notification)
- The User interface defines the update method that all concrete user classes must implement. This method is used to receive notifications from the Movie class.

UserFollower Class (implements User):
- Methods: update(notification)
- This class represents users who follow upcoming movies. The update method in this class will handle notifications about movie status changes, specifically alerting users when a movie they've followed is released.

LoyaltyProgramSub Class (implements User):
- Methods: update(notification)
- This class represents users who are part of a loyalty program. The update method in this class will handle notifications related to loyalty program details, ensuring that users receive information about new benefits or offers.

Staff Class
- Methods: answerInquery()
- This class handles user inquiries. When a staff member responds to an inquiry, it sends out a notification to the relevant user. The actual notification process would utilize the update method of the User interface.

Notification Class
- This class contains the information to be sent out as updates. It holds details like movie information, loyalty program updates, and responses to inquiries.

When a movie's status is updated from upcoming to currently in theaters, the updateMovieStatus method in the UpcomingMovie class is called. This triggers the notifyObservers method in the Movie class, which then calls the update method for each registered user. Users who are instances of UserFollower receive notifications about the movie release, while those in LoyaltyProgramSub might receive updates about related offers or programs.
For user inquiries, the Staff class's answerInquery method would create a notification and use the update method to inform the user who made the inquiry.
This design ensures a decoupled and efficient way to manage and distribute updates, adhering to the principles of the Observer pattern.