Lecture_13

Decorator Patter

- Allows behavior to be added to individual object dynamically without altering the structure
- The main purpose is to extend functionality of classes or objects at runtime
- Follows the **Open closed principle** : Lecture_12
- Ex
  - Applying different formatting styles to a text string (bold, italic, underline)
- Optional features

Strategy Pattern

- Define a family of algorithms, encapsulate each one as a separate class and make them interchangeable.
- Flexibility and modular
- **When To Use**
  - Multiple versions or variations of an algorithm are required
  - Asses or utilize data that should not be accessed outside the class
  - The behavior of a class should be defined at runtime
  - Conditional statements are complex and hard to use

Observer Pattern

- Behavioral pattern that defines a one to many dependency between object . It allows multiple observer objects to be notified when the state of the subject changes
- Promotes loose coupling between objects and enhances flexibility
- **Subject** : Contains a list of observers
- **Observer** : Contains the state of the object

Potential Memory Leak

- Care must be taken in the lifecycle of observers, especially when they are long lived and the subjects lifecycle is shorter, If observers are not

properly managed memory leaks may occur

- Use Weak dependency because of this