ECS160-HW4 Design Patterns Documentation

Design Patterns Implemented

Singleton Pattern

Where: Configuration class

The Singleton pattern was chosen for the Configuration class because we need a single, globally accessible configuration object that stores analysis type and JSON filename. Only one instance of configuration should exist during the application's lifetime. It provides a convenient way to access configuration parameters from any part of the application without passing the object around.

Composite Pattern

Where: PostComponent interface with Post (leaf) and Thread (composite) implementations

The Composite pattern was chosen for handling posts and threads because it allows treating individual posts and thread hierarchies uniformly through a common interface. Clients can interact with both single posts and threads through the same API. It simplifies operations like recursive statistics calculation by hiding the differences between simple and composite objects. This pattern naturally represents the hierarchical relationship between posts and their replies.

Visitor Pattern

Where: PostVisitor interface with various concrete visitor implementations

The Visitor pattern was chosen for computing statistics because it separates algorithms from the object structure they operate on. New operations can be added without modifying the post/thread classes. Each statistic calculation is encapsulated in its own visitor class. It enables traversing the composite structure while applying different operations.