ACS 560 SOFTWARE ENGINEERING

HW 12 - DESIGN PATTERN SUMMARY

In this project, I implemented the **Observer Design Pattern** to decouple forms and buttons from their dependent components, enhancing flexibility and scalability. The pattern allowed forms and buttons to act as subjects, notifying observers whenever a state change occurred, such as form submission or button interaction. For instance, in SalesForm.java and ItemDetailsForm.java, user actions trigger updates in other parts of the system, like real-time UI updates or backend data synchronization, without tightly coupling these components. This design ensured that new observers could be added or removed dynamically without modifying the core logic.

Additionally, the use of the Observer pattern streamlined communication across the application. By centralizing notifications, it reduced redundancy in code and made it easier to manage dependencies. The forms became reusable, as they no longer relied directly on the components they impacted. This separation of concerns led to a cleaner codebase and facilitated easier testing, debugging, and future enhancements. For example, adding new features, such as sending email notifications or logging interactions, could be achieved by introducing a new observer without altering existing logic.

The impact of this implementation is significant in terms of maintainability and modularity. By adhering to this pattern, the project gained a robust architecture capable of handling changes efficiently, meeting both current functional requirements and accommodating future scalability needs. It also enhanced the user experience by enabling real-time feedback mechanisms and improving overall responsiveness.