**Fault Models**

**Observer Strategy – Thermostat**

Il modello rappresenta un sistema semplificato di gestione di apparecchiature di riscaldamento mediante l’uso di un termostato. Ciascun apparecchio si comporta in modo specifico in base allo stato in cui si trova, dipendente dalla relazione tra la temperatura da lui rilevata tramite un sensore e quella impostata dal termostato. Se la temperatura rilevata è inferiore a quella impostata, il sistema passa nello stato ON; se le due temperature sono equivalenti, passa nello stato READY; se la temperatura rilevata è maggiore di quella impostata passa nello stato OFF. In base allo stato viene poi eseguita una procedura specifica (preparazione, accensione, spegnimento).

Il sistema è implementato mediante l’utilizzo della combinazione di due Design Patterns, Observer e Strategy. Ciascun apparecchio rappresenta un Soggetto che verrà monitorato da un Controller (con la funzione di Observer). Al variare della temperatura rilevata dal Soggetto, questa verrà notificata al suo Controllore che andrà ad eseguire una Strategia dipendente dallo stato in cui si trova il sistema.

Le criticità del sistema risiedono principalmente nell’aggiornamento dello stato da parte del controller: può fallire l’aggiornamento corretto dello stato in base alla temperatura del soggetto e l’inizializzazione della giusta procedura.

Inoltre il soggetto potrebbe non notificare correttamente il cambiamento di temperatura rilevata agli observers che lo stanno monitorando, impedendo così l’avvio delle procedure.

La rilevazione della temperatura da parte del soggetto, in un caso reale, si assume debba avvenire poi con aggiornamenti periodici nel tempo; nel modello simulato l’aggiornamento viene esplicitamente richiesto da un client che dunque può fallire nella sua richiesta non aggiornando correttamente la temperatura interna dell’apparecchio.

**Criticità del sistema**

* Mancata registrazione corretta di un Observer all’elemento controllato (Device).
* Eliminando un Observer deve essere rimosso dalla lista di Observers contenuta nel Soggetto e dunque non più notificato al cambio di stato.
* All’aggiornamento della temperatura nei singoli Device (Soggetti), non vengono notificati tutti gli Observer a lui associati.
* Al cambio di stato del Soggetto, non viene correttamente istanziata la giusta strategia (dipendente dalla temperatura rilevata).

**Defensive Copy**

Il modello rappresenta un sistema di gestione degli esami da parte di uno studente. Ciascuno studente ha un proprio libretto, in cui gli esami vengono inseriti. Il corso di laurea, utilizzando il libretto di uno studente, è in grado di calcolare la media dei voti degli esami. Sono previste funzionalità per il recupero e la modifica di un singolo esame, il recupero di tutti gli esami, l'aggiunta di un nuovo esame e la cancellazione di un singolo esame o di tutti gli esami presenti nel libretto di uno studente.

Il sistema è implementato utilizzando la tecnica della copia difensiva, ovvero invece di condividere l'oggetto originale, nel caso di studio il libretto o l'esame, viene condivisa una copia di esso, in modo da limitare gli effetti negativi di modifiche inattese.

**Criticità del sistema**

* Durante la fase di copia, l’oggetto non viene copiato correttamente in tutti i suoi attributi.
* Quando il Professore assegna un voto, questo deve necessariamente trovarsi nel *range* di valori tra 18 e 30 (compresi).
* Un esame non è inserito correttamente nel libretto dello studente.
* Il calcolo della media ponderata dei voti degli esami presenti nel libretto di uno studente è errata.
* Un esame non è correttamente rimosso dal libretto di uno studente, e dunque considerato erroneamente nel calcolo della media.

**Builder Composite**

Il modello rappresenta un sistema di creazione di espressioni booleane. Il sistema è in grado di combinare fra di loro variabili booleane, operatori logici quali AND, OR, NOT e l'uso di parentesi. Il sistema è in grado di valutare correttamente tali espressioni booleane e di stamparle a schermo, mostrando il valore assegnato alle singole variabili più la valutazione finale dell'intera espressione.

Il sistema è implementato combinato due design patterns. Il design pattern Composite si occupa di creare le componenti semplici, le variabili, e le componenti composte, gli operatori e le parentesi. Il design pattern Builder si occupa di istanziare correttamente tali componenti e di combinarle fra di loro per creare delle espressioni booleane più complesse.

**Criticità del sistema**

* Quando viene chiamato il metodo per aggiungere un componente ad un altro componente, il sistema deve essere in grado di capire il tipo dell’oggetto, referenziato tramite classe astratta, su cui viene invocato.
* Il Builder non costruisce l’oggetto desiderato (compresa l’assegnazione delle variabili).
* Un’espressione booleana non viene valutata correttamente.