Progetto Ingegneria del Software

Studente: Fabrizio Iuvara

Matricola: 1000015982

Introduzione
Si vuole realizzare un sistema software per simulare il comportamento di un ascensore che può ospitare varie persone e può andare su diversi piani; si dovrà verificare che il peso delle persone sia tale da permetterne l'ingresso al suo interno, così come che il piano selezionato rispetti i vincoli di funzionamento.

Classi da implementare
Dopo aver effettuato l'analisi dei requisiti, si è ritenuto opportuno implementare le seguenti classi:

- Ascensore: classe che contiene le informazioni dell'ascensore
- Persona: classe che contiene le informazioni di una persona
- Piano: classe che contiene le informazioni di un piano
- Progetto: è la classe che implementa il main del progetto
- Mediatore: classe che implementa l'omonimo design pattern per gestire il corretto funzionamento del codice, gestendo le dipendenze tra le altre classi; inoltre funge da Model per il design pattern MVC
- View: classe che si occupa di mostrare i dati all'utente (dp MVC)
- Controller: classe che si occupa di ricevere input dall'utente (dp MVC)

Scelte progettuali:

- La classe Ascensore implementa il design pattern *singleton* in quanto rappresentiamo una singola istanza della classe, della quale ne gestiamo il comportamento
- La classe Piano implementa il design pattern *prototype;* la scelta deriva dal fatto che le istanze di Piano da creare possono essere molteplici e differiscono per pochi attributi, per cui anziché creare nuove istanze della classe procediamo con la clonazione
- La classe Mediatore implementa l'omonimo design pattern; questa scelta deriva dal fatto che avrei avuto molte connessioni tra le classi, rendendole non riusabili in altri contesti, per cui ci rifacciamo agli standard di progetto; in questo modo le classi non interagiscono direttamente tra di loro, ma comunicano con Mediatore, che implementa il comportamento cooperativo tra tutte le classi, e

se il comportamento dovesse cambiare, non dobbiamo modificare le classi definite in precedenza che lo utilizzano, ma soltanto il Mediatore

- Le classi View e Controller sono state create per implementare il design pattern *Model View Controller,* fondamentale per le applicazioni interattive; la classe Mediatore funge da Model in quanto contiene i dati dell'applicazione e le funzionalità principali, la classe View si occupa di mostrare i dati all'utente e le operazioni disponibili a runtime, tramite le quali è possibile impostare lo stato iniziale del sistema e lanciare l'applicativo; infine la classe Controller riceve gli input dall'utente e le trasforma in richieste di servizio per Mediatore

Funzionamento del progetto

Il funzionamento standard del progetto è il seguente: si preme 1 per generare il "palazzo" (numero di piani in cui può viaggiare l'ascensore), si può inserire un valore compreso tra 1 e 100; successivamente si creano le persone premendo il tasto 2, inserendo il nome, il peso (valori compresi tra 1 e 300 Kg) e il piano corrente in cui la persona si trova, per esempio "Giacomo 60 3"; successivamente, per controllare i dati inseriti, si potrebbero utilizzare i tasti 4 e 5 per visualizzare rispettivamente lo stato dell'ascensore (di default parte sempre dal primo piano e ha un peso massimo supportato di 300 Kg) e delle persone presenti all'interno del palazzo; infine si utilizza il tasto 6 per far chiamare l'ascensore a una persona che verrà passata in input per nome; a questo punto si può premere 7 per far partire la simulazione e dopo ciò si possono ancora utilizzare i tasti 4 o 5 per verificare il nuovo stato dell'ascensore e delle persone; si può continuare a inserire persone e chiamare l'ascensore da piani diversi e creare simulazioni più complesse; quando si vuole terminare il programma si deve premere il tasto 8.

Unit test

Ho effettuato dei test del software (presenti in AppTest.java) per verificare la corretta esecuzione del programma e per mantenerla in caso di manutenzione o refactoring del codice, testando in particolare situazioni ricorrenti all'interno della classe Mediatore perché è la principale classe da cui dipende il corretto funzionamento del software. Ad esempio, ho implementato il metodo di test: testaPesoAscensore() per verificare che l'ascensore abbia lo stesso peso interno con persone diverse la cui somma dei pesi è uguale, oppure metodi più complessi come testaViaggia() che fa chiamare l'ascensore da persone in diversi piani e ne controlla il funzionamento corretto.