Relazione progetto Metodologie di Programmazione

Autori

Nome e Cognome	Matricola	Esito Scritto	Email
Andrea Temin	5948774	Non effettuato	Andrea.temin@stud.unifi.it
Lorenzo Morandi	5931789	Positivo	Lorenzo.morandi@stud.unifi.it

Note:

- L'interfaccia grafica è stata realizzata tramite l'utilizzo di un Wizard. Per questo le classi all'interno del pacchetto GUI hanno del codice generato dall'IDE che abbiamo utilizzato.
- Gli schemi UML completi sono allegati alla consegna;
- Per lanciare l'applicazione è sufficiente eseguire la classe "MainClass".

Sommario

Schema delle Figure

Esercizio Scelto Funzionalità del Sistema

Esercizio Scelto

Per il nostro progetto abbiamo fatto riferimento al primo esercizio del foglio 'Proposte di esercizi' presenti sulla pagina moodle del corso: "Shop online".
Nello specifico lo Shop/Negozio offre prodotti di natura informatica come Tastiera, mouse, monitor etc…

Specifiche e funzionalità del sistema:

- Nel programma è possibile inserire manualmente un prodotto nel magazzino riempendo i campi : Nome, Descrizione, Prezzo, Giacenza che si trovano sul lato sinistro della finestra "Gestione Magazzino". Premendo sul bottone "Inserisci >>", l'articolo verrà inserito nel magazzino e comparirà sulla jList nel lato destro della medesima finestra.

- Se si desidera invece rimuovere un elemento dal magazzino si può selezionare gli oggetti presenti nella jList e poi premere, "Rimuovi Elemento".

- Per creare un pacchetto formato da più articoli:
  1. Selezionare la quantità di pacchetti da creare,
  2. selezionare gli oggetti dalla jList e poi premere "Crea Pacchetto" .

- Nella finestra "Gestione Magazzino", in basso, si trovano i bottoni "Ricerca Prezzo Limite" e "Ricerca Giacenza Limite". Questi, come si intuisce dal nome, hanno la funzione di trovare rispettivamente il prezzo più alto e il più basso oppure la giacenza maggiore e minore degli articoli introdotti fino a quel momento.

- L' utente può aggiungere al suo carrello i pacchetti o gli articoli singoli disponibili nel magazzino. Quindi selezionando uno o più oggetti dalla jList, presente nel riquadro di sinistra della finestra "Interfaccia utente", e poi cliccando sul bottone "Inserisci" si aggiunge di una unità tutti gli item selezionati.

- Per rimuovere un Item dal carrello il procedimento è speculare all'inserimento, premendo il tasto rimuovi.

- Per accedere alla sezione di pagamento occorre premere sul bottone "Paga Carrello" nella finestra "Interfaccia Utente". Questo bottone aprirà una nuova finestra.

- La nuova finestra: “Seleziona Metodo di Pagamento” permette all’utente scegliere se pagare con Bonifico o Carta di Credito, in più include la possibilità di inserire un codice che permette (a chi lo possiede) di avere uno sconto su ogni articolo.
  Con l’ultimo bottone “Paga” si effettua il pagamento definitivamente.

- Concluso il pagamento si apre un’ultima finestra: “Scontrino” con stampato:
  - Il metodo di Pagamento scelto dall’utente.
  - La lista degli Item acquistati.
  - Gli sconti effettuati.
  - Il totale importo.
  - L’Ora e la Data del pagamento.

Motivazioni sulle scelte di design più significative

Rappresentazione Degli Articoli

Gli articoli sono stati rappresentati utilizzando il pattern Composite. Il quale consente alle classi client che utilizzano Item di ignorare la differenza tra oggetti composti (Pacchetto) e oggetti singoli (Articolo singolo).

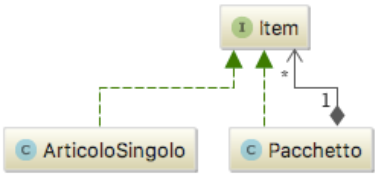

Figura 1. UML Composite

Applicazione degli sconti ai singoli item

Abbiamo utilizzato il pattern decorator in modo che sia possibile applicare particolari sconti ai singoli articoli (pacchetti o articoli singoli).
Ogni decoratore dovrà estendere la classe “DecoraItem” e la nuova classe creata dovrà riscrivere i metodi che intende decorare.
Ad esempio abbiamo implementato la classe “ScontoUniversitariDecorator” che applica uno sconto del 20%.

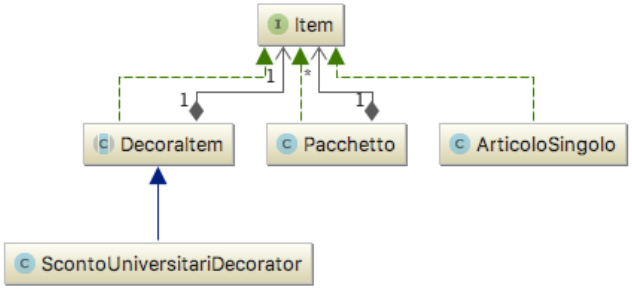

Figura 2. UML Decorator su Composite

Descrizione classe Carrello.

La classe "Carrello" implementa l'interfaccia "CarrelloInterface" e rappresenta l'insieme degli Item che l'utente vorrebbe acquistare. L'insieme degli articoli scelti dall'utente è rappresentato da una lista di "OggettiMagazzinabili".

L'Applicazione di Sconti all'intero Carrello

Abbiamo utilizzato il pattern decorator in modo che sia possibile applicare particolari sconti all'intero carrello.
Ogni decoratore dovrà estendere la classe "DecoraCarrello" e la nuova classe creata dovrà riscrivere i metodi che intende decorare.
Ad esempio abbiamo implementato la classe "ScontoOltreCentoDecorator" che applica uno sconto del 10%.

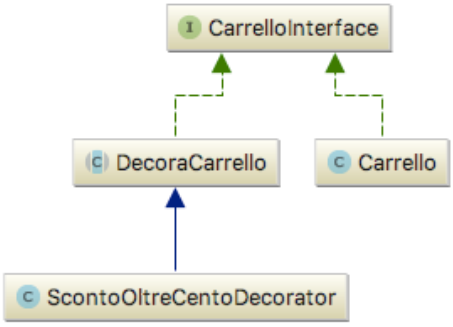

Figura 3. UML Decorator Carrello

Rappresentazione degli articoli all'interno del magazzino

Ogni oggetto che deve essere contenuto nel magazzino deve implementare l'interfaccia "OggettoImmagazzinabile". Abbiamo implementato la classe "RegistroMagazzino" che rappresenta un singolo oggetto presente nel magazzino.

Esso è formato da:

- Codice: è univoco all'interno del magazzino e viene incrementato ad ogni inserimento di un nuovo Item;

- Oggetto: identifica una foglia o un composto della gerarchia degli articoli;
- Giacenza: indica la disponibilità dell'oggetto nel magazzino.

L'interfaccia OggettoImmagazzinabile quindi la classe RegistroMagazzino estendono clonable che permette di crearne una copia.

Rappresentazione del Magazzino.

Dovendo esistere una sola istanza della classe "Magazzino" è stato applicato il pattern Singleton. La classe Magazzino implementa l'interfaccia "Warehouse" e ne sovrascrive i metodi. La classe Magazzino è composta da una lista di OggettoImmagazzinabile.

Esecuzione dei Report all'interno del Magazzino.

Grazie a due Bottoni nella finestra "Gestione Magazzino" è possibile avviare un Visitor che effettui una ricerca all'interno del magazzino stampando i risultati su una nuova finestra.
La gerarchia di Visitor è costituita da una classe astratta "CercaMaxMinMagazzinoABSVisitor" che tramite il pattern Template definisce i metodi comuni ad ogni Visitor:

- "CalcolaMaxMin(Iterator<OggettoImmagazzinabile> merce)": scorre l'iterator applicando ad ogni elemento una BiFunction. Il metodo ritorna l'OggettoImmagazzinabile più grande o più piccolo in base alla funzione utilizzata.

Ogni Visitor dovrà estendere la classe CercaMaxMinMagazzinoABSVisitor definendo le due BiFunction necessarie al metodo CalcolaMaxMin.

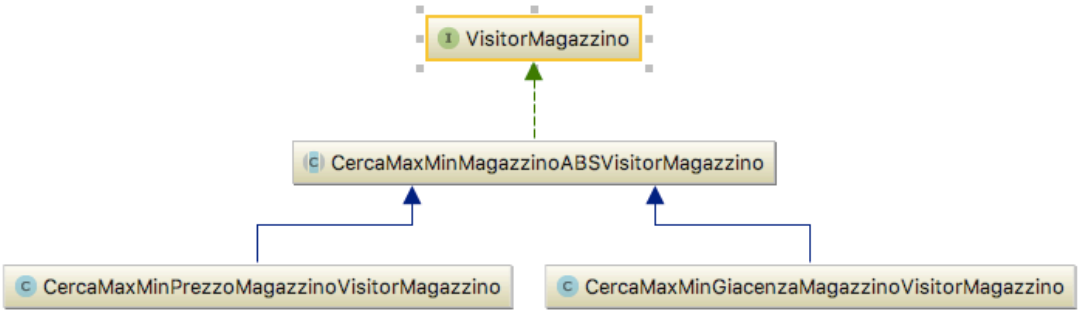

Figura 4. UML Visitor Magazzino

Utilizzo del pattern Observer

Per tenere aggiornata l'interfaccia grafica ad ogni cambiamento di stato del magazzino e del carrello abbiamo implementato il pattern Observer.
Sono state rese osservabili le interfacce "CarrelloInterface" e "Warehouse"
Le classi dell'interfaccia grafica: "InserimentoElementiMagazzino" e "InterfacciaUtente" sono osservatori del Magazzino, in oltre la classe InterfacciaUtente osserva il Carrello.

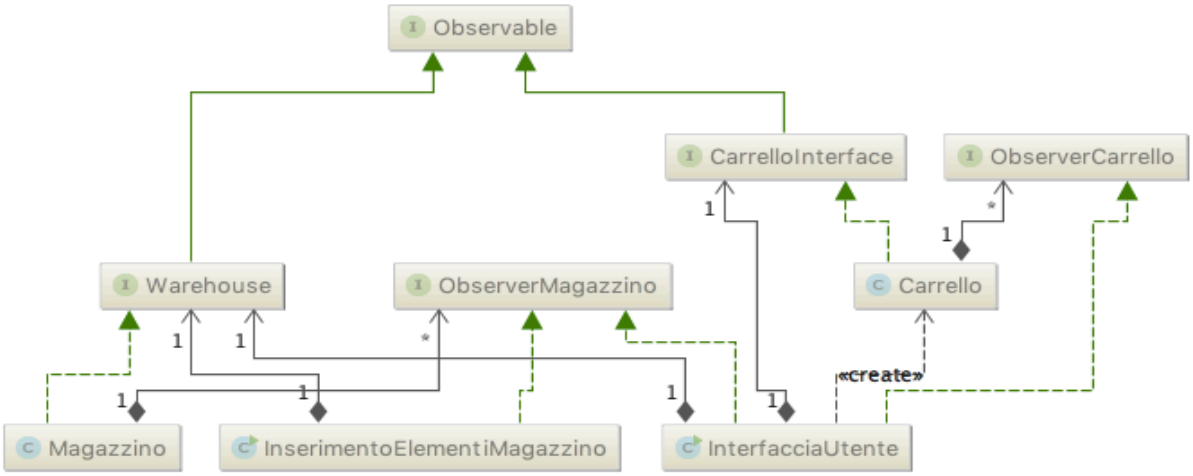

Figura 5. UML Observer

Strategie di pagamento

Per implementare diverse strategie di pagamento abbiamo utilizzato il pattern Strategy.
Ogni nuovo metodo di pagamento dovrà estendere la classe "PagamentoAbstractStrategy".
Abbiamo simulato due tipi di pagamento: Bonifico e Carta di Credito.

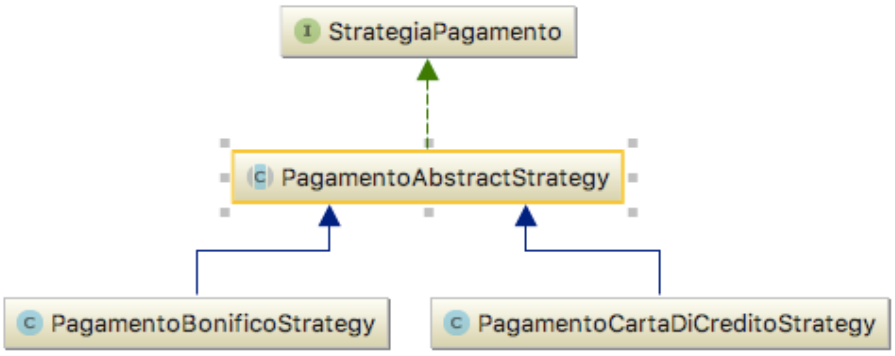

Figura 6. UML Strategy

Fase di Testing

I test sono stati realizzati mediante l'impiego di Junit 4 e sono stati scritti seguendo le specifiche che abbiamo visto a lezione:

- I test sono stati scritti in modo incrementale rispetto alla stesura del codice;
- Ogni classe concreta ha una sua classe di test nella quale vengono eseguiti i test di tutti i metodi della classe;