Self-bar

**Data consegna: 27 dicembre 2016**

**Autori:**

*Fagioli Giulio, 6006222*  giulio.fagioli@stud.unifi.it

*Salani Lorenzo, 5979949* lorenzo.salani@stud.unifi.it

Descrizione dell’esercizio

L’esercizio scelto è Self-Bar, che consiste nell’implementazione di un sistema per effettuare ordini al tavolo di un bar tramite un tablet.

Ogni tavolo ha il proprio tablet dove vengono selezionati ed acquistati i prodotti, l’applicazione permette la scelta fra Cocktail e Caffè e permette di aggiungere alla bevanda scelta fino ad un massimo di 2 ingredienti.

Ai caffè base Arabica, Orzo e Decaffeinato è possibile aggiungere Cacao, Latte e Panna, mentre ai Cocktail base Martini, Bloody Mary e Margarita è possibile aggiungere Soda, Lime e Appetizer.

Il cliente ordinando una bevanda dal tablet può effettuare le seguenti operazioni:

* Aggiungere una bevanda all’ordine
* Rimuovere una bevanda dall’ordine
* Visualizzare l’elenco delle bevande ordinate
* Visualizzare il prezzo singolo e totale delle bevande
* Pagare scegliendo il metodo di pagamento

La parte sinistra dell’interfaccia dell’applicazione permetterà all’utente di aggiungere al carrello la bevanda e le eventuali aggiunte ad essa e fornirà il prezzo finale del prodotto.

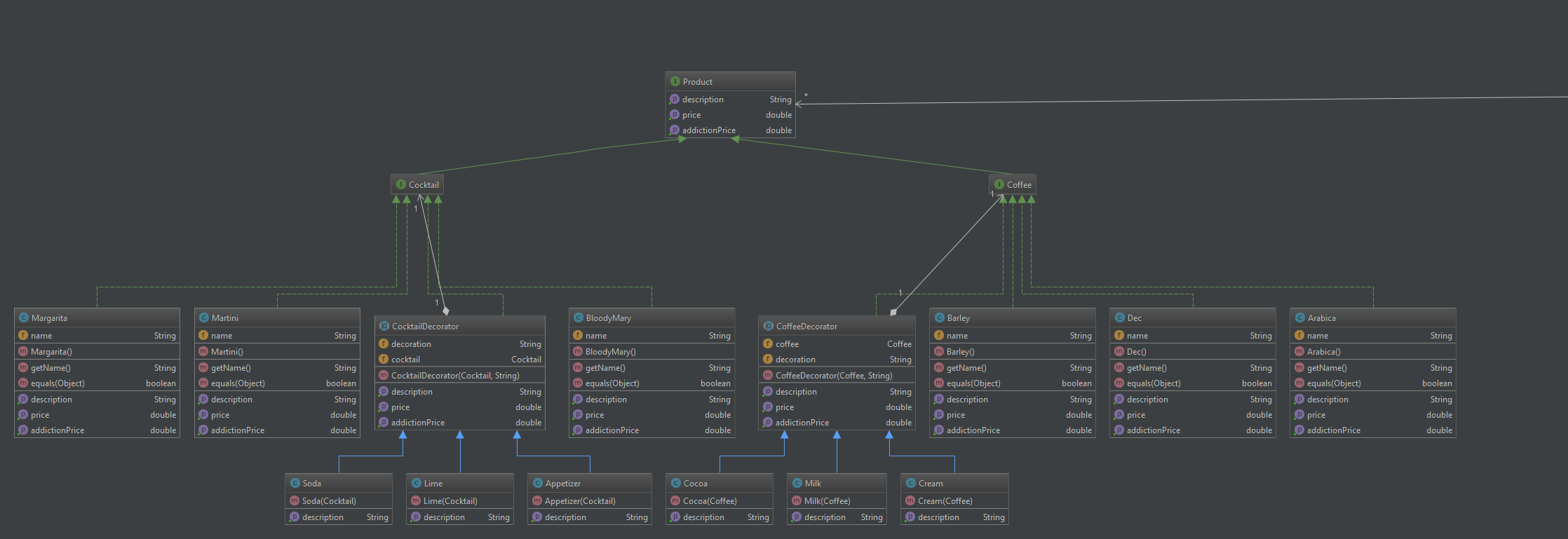
Nella parte destra dell’interfaccia avremo il carrello che mostrerà tutti gli oggetti aggiunti con relativo prezzo e prezzo totale.

Ad ogni aggiunta o rimozione l’interfaccia aggiornerà l’elenco degli ordini ed i relativi prezzi.

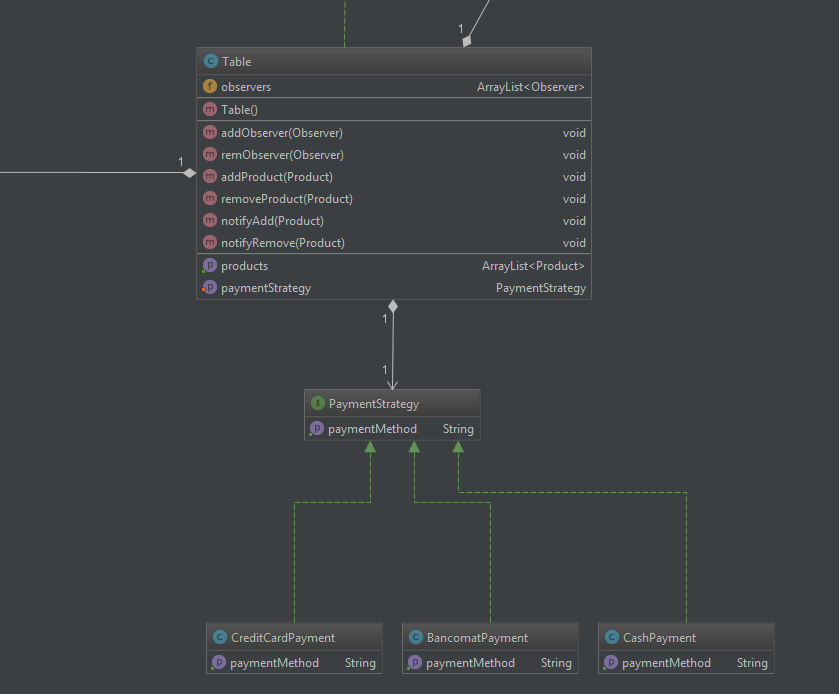
Dopo aver selezionato i prodotti da acquistare l’utente deve selezionare una modalità di pagamento fra bancomat, contanti e carta di credito, per poter effettuare il pagamento.

UML Applicazione

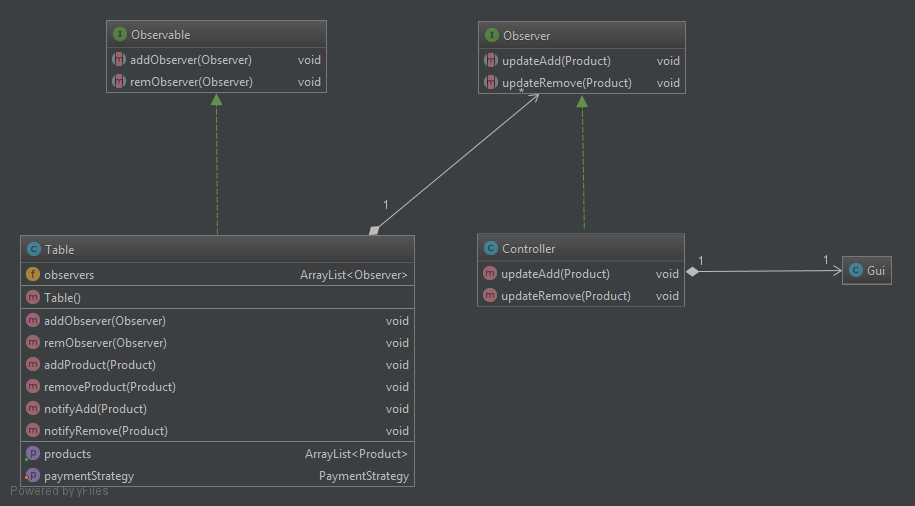
Uml decorazioni:



Uml pagamento:



Uml interfaccia grafica:



Scelte di design

La struttura principale del programma, ovvero quella che permette di effettuare le aggiunte ad una bevanda di base, è composta da due decoratori, uno per i Caffè e uno per i Cocktail.

Entrambe le strutture Decorator implementano l’interfaccia Product e sono composte dall’interfaccia principale Caffè e Cocktail, implementata da tutte le bevande di base e dalla classe astratta del rispettivo decoratore.

Una bevanda base è creata implementando l’interfaccia Coffee e Cocktail e le classi concrete realizzate sono Arabica, D’orzo e Decaffeinato come tipi di caffè, mentre Martini, Bloody Mary e Margarita come cocktail.

Un decoratore per un caffè o per un cocktail estende rispettivamente la classe astratta CoffeeDecorator e CocktailDecorator ed i decoratori concreti per i caffè sono Panna, Latte e Cacao, mentre quelli per i cocktail sono Appetizer, Soda o Lime.

Il prezzo di ogni bevanda è calcolato aggiungendo i prezzi di tutte le decorazioni al prezzo base di ogni prodotto, cosi anche la sua descrizione è ricavata concatenando il nome del prodotto alle descrizioni degli oggetti al suo interno.

La classe Table utilizzerà una lista di Product per memorizzare i prodotti inseriti nel carrello, per effettuare il pagamento è stato utilizzato il pattern Strategy permettendo la scelta a run-time del tipo di pagamento.

Tramite l’utilizzo della strategia di pagamento si potrà effettuare il pagamento scegliendo fra Contanti, Carta di credito e Bancomat.

L’interfaccia grafica è definita nella classe Gui ed è stata definita anche una classe Controller per modificare e aggiornare i componenti dell’interfaccia, in base ai prodotti inseriti e alle scelte dell’utente. La scelta di separare l’interfaccia dalla struttura del programma, tramite la classe Controller, permette di isolare la Gui, che si occupa solo della visualizzazione dei componenti.

Abbiamo realizzato la notifica di un oggetto aggiunto al Table tramite il pattern Observer, dove l’oggetto osservabile è il Table mentre l’oggetto osservatore è Controller; per ogni oggetto aggiunto o rimosso dal Table, esso notificherà a tutti gli Observer questo cambiamento; il controller ad una notifica di aggiunta o rimozione provvederà a aggiornare l’interfaccia.

Descrizione …..

Sono stati implementati diversi tipi di test