UNIVERSITÀ
DEGLI STUDI
DI BERGAMO

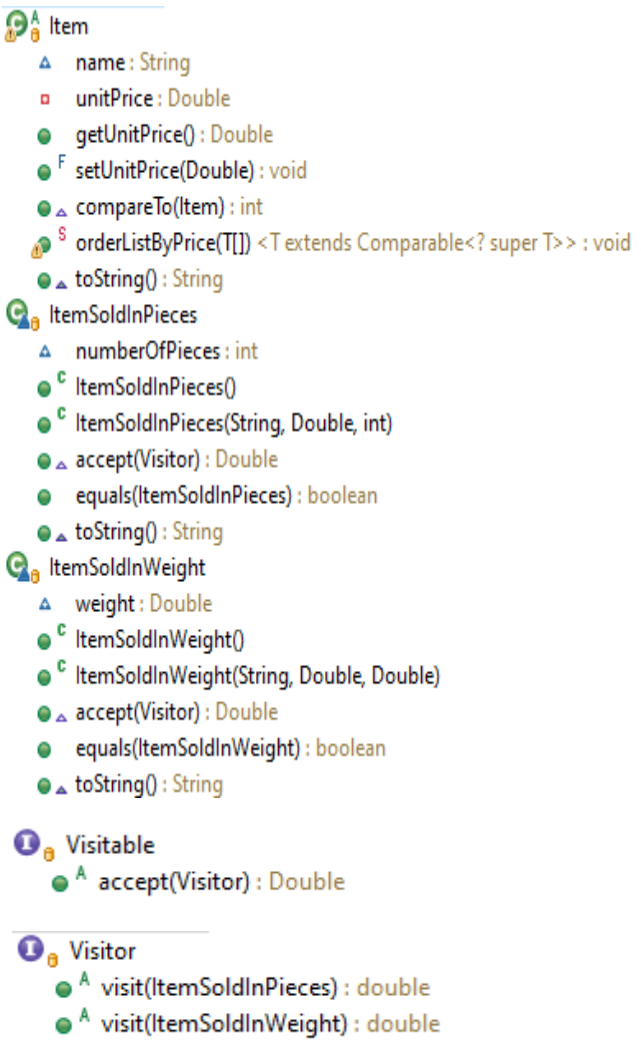
Progetto PA: Java

"Supermarket"

Boschetti Kevin, matricola n. 1057400

In questo progetto ho voluto sviluppare un programma in Java per emulare la gestione base di un supermercato. Ho cercato di utilizzare vari costrutti propri del linguaggio visti a lezione, aggiungendo anche dei commenti nel codice per rendere esplicito quando venivano sfruttati.
Ho scelto di implementare due tipologie di prodotti, quelli venduti in base alla quantità e quelli venduti in base al peso. Si possono aggiungere gli articoli a dei carrelli di cui poi può essere estratto il costo del contenuto. Questo varierà in base alla carta fedeltà, presente o meno, e al giorno della settimana.

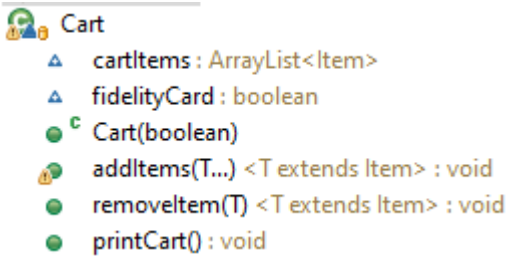
Item.java



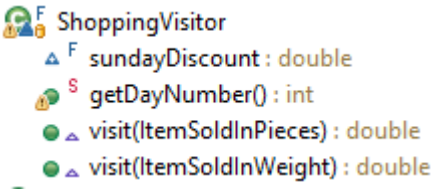
All'interno del file *Item.java* si trova la classe astratta di Item. Ho scelto di renderla tale in quanto volevo che fosse possibile creare un prodotto che facesse parte di una delle due classi che estendono Item, cioè *ItemSoldInPieces* o *ItemSoldInWeight*. Ogni prodotto è composto da un *name* e da un prezzo unitario che corrisponde al prezzo/pezzo oppure al prezzo/kg. Questo campo è private, quindi per permetterne la lettura e la modifica ho i metodi *getUnitPrice* e *setUnitPrice*, impedendo la modifica alle sottoclassi di Item dichiarandoli *final*. Ho aggiunto la possibilità di stampare un articolo facendo override di *toString*, e di comparare due prodotti tramite la ridefinizione di *compareTo*. Sfruttando questo ultimo metodo è anche possibile ordinare una lista di elementi tramite una funzione che utilizza le *wildcard* e fa *superbinding*.
Ho voluto aggiungere anche l'utilizzo del *visitor pattern*, infatti la classe *Item*, e quindi anche le sue *sottoclassi*, implementano due *interfacce*. *Visitable* permette ad un prodotto di essere visitato da un oggetto di visitor, appartenente alla seconda interfaccia *Visitor*.

La classe *ItemSoldInPieces* ha un campo che la caratterizza, ovvero il numberOfPieces. Inoltre fa override del metodo *accept* così da permetterne un'implementazione specifica come vedremo in seguito. Presenta anche un *overload* del metodo *equals* che confronta due oggetti in base a tutti i campi caratteristici. Infine ho utilizzato l'identificatore *super* per ridefinire il *toString* andando dapprima a chiamare quello di *Item*, per poi accodarci gli elementi presenti solo in *ItemSoldInPieces*.
ItemSoldInWeight presenta elementi molto simili a quelli già citati per i prodotti venduti a pezzi, uniche differenze sono il campo *weight* e il modo in cui sono ridefiniti *equals* e *toString*.

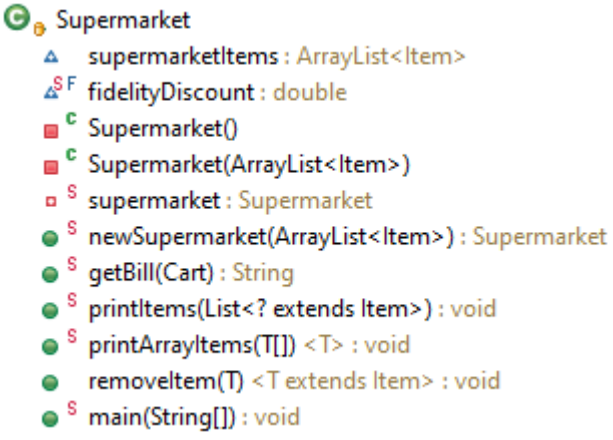
Supermarket.java



In *Supermarket.java* ho implementato tre classi, *Cart*, *ShoppingVisitor* e *Supermarket*.
In *Cart* possiamo trovare *cartItems*, una lista di *Item* generici che potranno appartenere a una delle sue sottoclassi, e un booleano *fidelityCard* che indica se applicare al carrello del cliente uno sconto fedeltà. Tramite il metodo *addItems*, grazie dei *bounded generics* e di *varaagrs* si possono aggiungere uno o molteplici prodotti al carrello, mentre con *removeItem* si possono togliere. Infine si può stampare il contenuto del carrello con *printCart*.



La classe *ShoppingVisitor* è l'implementazione del *visitor pattern*, racchiude quindi il modo in cui viene calcolato il prezzo di un prodotto in base alla sua classe, *ItemSoldInPieces* o *ItemSoldInWeight*. In particolare è *final*, quindi non può essere estesa, e calcola il prezzo di un prodotto considerando il giorno della settimana: se è domenica tramite la variabile *sundayDiscount* applica uno sconto del 10%.



La classe *Supermarket* rappresenta infine l'implementazione finale del progetto, infatti contiene anche il *main*. Un supermercato è composto da una lista *supermarketItems* con tutti i prodotti vendibili, e da uno sconto fedeltà per i clienti in possesso della carta fedeltà che a titolo di esempio ho impostato al 15%.

Ho scelto di poter creare un solo supermercato, utilizzando così il *singleton pattern*, che permette la creazione di una sola istanza dell'oggetto e solo se richiesta esplicitamente. Alla creazione richiede come argomento la lista dei prodotti oppure ha anche un costruttore base. Tramite il metodo statico *getBill* è possibile richiedere il conto per un carrello specifico, mentre con *removeItem* si può eliminare un prodotto dalla lista dei prodotti venduti dal supermercato.
Ho scelto poi di implementare la possibilità di gestire delle liste di prodotti sia con delle classi proprie di java, sia tramite dei semplici array. A questo scopo per la stampa del contenuto delle liste troviamo rispettivamente i metodi *printItems* e *printArrayItems*.

Di seguito l'output del main in cui ho effettuato alcuni test delle funzionalità del programma:

```
---------------- Prodotti creati ----------------
detersivo
dentifricio
dopobarba
cesto banane
noci
mele
---------------- Creo liste di prodotti ----------------
---------------- creo supermarket ----------------
---------------- aggiungo dopobarba----------------
Prodotti venduti a pezzi:
detersivo
dentifricio
dopobarba
Prodotti venduti a peso:
noci
mele

---------------- rimuovo dopobarba ----------------
Prodotti venduti a pezzi:
detersivo
dentifricio
Prodotti venduti a peso:
noci
mele

---------------- creo due carrelli, con e senza carta fedeltà ----------------
---------------- riempio i due carrelli ----------------
---------------- stampo prodotti nel carrello ----------------
Prodotti nel carrello:
venduti a pezzi:
detersivo
dentifricio
venduti a peso:
cesto banane
noci
```

```
---------------- rimuovo detersivo ----------------
Prodotti nel carrello:
venduti a pezzi:
dentifricio
venduti a peso:
cesto banane
noci

---------------- stampo conto carrelli ----------------
cliente fedele: 11.67€
cliente non fedele: 13.73€
```