Supermercato

Progetto di Programmazione dell'Appello di Giugno 2017

Introduzione

In questo esame vi chiederemo di scrivere un insieme di classi per la modellazione di un supermercato.

Classi

Descriviamo ora nei dettagli come realizzare le classi necessarie al progetto. Se lo ritenete, potete apportare cambiamenti alle segnature dei metodi proposti e anche – se lo ritenete – alla struttura delle classi, restando ovviamente all'interno dei requisiti esposti nel paragrafo introduttivo.

N.B. (1): Oltre ai metodi qui indicati, aggiungete a ogni classe un valido metodo `toString()` e, dove lo ritenete, anche un metodo `equals` e un metodo `hashCode` (con le segnature appropriate e che soddisfino i contratti previsti).

N.B. (2): Nel seguito, useremo il termine *lista di X* per indicare un qualunque tipo che consenta di conservare delle sequenze di oggetti di tipo `X`: potete implementare una lista con un array (`X[]`) oppure usando ad esempio delle `ArrayList<X>` (pacchetto `java.util`). Nelle segnature useremo sempre `X[]`, ma resta inteso che potete modificarlo in `ArrayList<X>`, purché lo facciate in modo consistente (sostituite *tutti* gli array con delle liste).

Classe Reparto

Il supermercato è diviso in reparti (verdura, surgelati, latticini, ecc.). Ogni reparto ha un nome e un responsabile (identificato dal suo cognome e nome), come evidenziato dal costruttore:

- `Reparto(String nomeReparto, String nominativoResponsabile)`

Esempio d'uso:

```
repartoBevande = new Reparto("Bevande", "Giovanni Verdini");
```

Il nome del reparto lo identifica univocamente.

Classe UnitaDiMisura

Le istanze di questa classe rappresentano nomi di unità di misura (ad es., litro, kilogrammo, ettogrammo, numero, ecc.). Quando qualcuno crea una unità di misura, deve decidere se si tratta di una *unità di riferimento* oppure no; nel secondo caso si parla di *unità derivate* e occorre fornire qual è l'unità di riferimento, e quante unità derivate contiene un'unità di riferimento. Più precisamente, la classe ha i seguenti costruttori e metodi:

- `UnitaDiMisura(String nome, String sigla)`: crea un'unità di riferimento con il nome indicato e la sigla indicata.
- `UnitaDiMisura(String nome, String sigla, UnitaDiMisura riferimento, double qta)`: crea un'unità derivata con il nome dato e la sigla data, in cui l'unità indicata è quella di riferimento, e occorrono `qta` unità derivate per fare un'unità di riferimento.
- `UnitaDiMisura riferimento()`: restituisce l'unità di riferimento (l'unità stessa su cui è invocato, se si tratta di un'unità di riferimento).
- `double qta()`: quante unità di questo tipo servono per fare un'unità di riferimento (restituisce 1.0 se questa è già un'unità di riferimento).

Ecco alcuni esempi:

```
    kg = new UnitaDiMisura("chilogrammo", "kg");
    g = new UnitaDiMisura("grammo", "g", kg, 1000);
    hg = new UnitaDiMisura("etto", "hg", kg, 10);
    l = new UnitaDiMisura("litro", "l");
    ml = new UnitaDiMisura("millilitri", "ml", l, 1000);

    num = new UnitaDiMisura("numero", "n");
    dozzina = new UnitaDiMisura("dozzina", "dozz", num, 1.0/12);
```

Classe Prodotto

Il supermercato commercializza prodotti. Ogni prodotto è caratterizzato da un reparto, da un nome commerciale, da un identificatore alfanumerico (che lo identifica univocamente), da un numero che rappresenta la quantità di prodotto contenuto in una confezione, dall' `UnitaDiMisura` usata per misurare il prodotto, dal prezzo di questo prodotto (in euro).

- `Prodotto(String nome, Reparto reparto, String id, double quantita, double prezzo, UnitaDiMisura mis)`: un prodotto di nome `nome` (identificatore `id`), del reparto `reparto`, costa `prezzo` EUR e contiene la data quantità del prodotto in ogni confezione (`quantita`), misurata in `mis`.
- `double prezzo()`: restituisce il prezzo.
- `double prezzoUnitario()`: restituisce il prezzo riferito a un'unità della misura di riferimento.

Ad esempio:

```
birraMoretti = new Prodotto("Birra Moretti classica", repartoBevande, "37712341234AF3", 660, 1.25, ml);
System.out.println(birraMoretti.prezzoUnitario())
```

stamperà 1.8939394 poiché una bottiglia contiene 660 millilitri, e costa 1.25 EUR, per cui il prezzo al litro è 1000*1.25/660=1.8939394.

Classe Offerta

Un'offerta è un evento che si applica a uno specifico prodotto, e che (sotto opportune condizioni) fa sì che il cliente che acquista quel prodotto ottenga uno sconto percentuale sul suo prezzo. Un'offerta si applica solo se una certa quantità del prodotto viene acquistata, e in certi casi si applica solo ai possessori della carta fedeltà.

- `Offerta(String nomeOfferta, Prodotto p, double percSconto, int numeroMinimo, boolean soloCartaFedelta)`: un'offerta che si applica al prodotto `p`, dando luogo a una certa percentuale di sconto `percSconto`, solo a patto che si acquistino almeno `numeroMinimo` prodotti di quel tipo, e applicabile solo a chi possegga la carta fedeltà se il boolean `soloCartaFedelta` è `true`.
- `double prezzo(int qta, boolean carta)`: restituisce il prezzo complessivo del prodotto cui si riferisce questa offerta, posto che se ne comprino `qta` unità (e che si possegga la carta fedeltà, se `carta` è `true`).

Il nome dell'offerta la identifica univocamente. Ad esempio:

```
offBirraMoretti = new Offerta("PiùBirraPerTutti", birraMoretti, 25, 3, true)
```

indica che i clienti del supermercato possessori di carta fedeltà che acquistino almeno tre bottiglie di Birra Moretti avranno il 25% di sconto su ciascuna birra, mentre

```
System.out.println(offBirraMoretti.prezzo(5, false))
System.out.println(offBirraMoretti.prezzo(2, true))
System.out.println(offBirraMoretti.prezzo(3, true))
```

stamperà, in ordine, 6.25 (poiché se si comprano 5 bottiglie ma non si possiede la carta fedeltà non si applica l'offerta, quindi il prezzo è 5*1.25=6.25), 2.50 (poiché anche possedendo la carta fedeltà, due bottiglie sono troppo poche) e 2.8125 (poiché se si possiede la carta

fedeltà ogni bottiglia costa il 25% in meno).

La classe `Offerta` contiene anche un metodo statico:

- `static double totale(Offerta[] off, Prodotto[] p, int[] qta, boolean carta)` : a questo metodo viene passata una lista di offerte, una lista di prodotti e una lista di quantità acquistate; potete assumere che `p.length==qta.length` e che `p` non contenga tutti prodotti diversi. Il metodo restituisce il prezzo totale pagato dall'utente che abbia acquistato i prodotti della lista `p` nelle quantità specificate dalla lista `qta` , sapendo che il supermercato sta applicando le offerte `off` e che il cliente possiede la carta se `carta==true` .

Classe Supermercato

Questa classe rappresenta un supermercato. Il costruttore:

- `Supermercato()` : crea un supermercato.

Si danno inoltre i seguenti metodi:

- `void aggiungiReparto(Reparto r)` : aggiunge il reparto `r` .
- `void aggiungiProdotto(Prodotto p)` : aggiunge il prodotto `p` .
- `boolean cancellaProdotto(Prodotto p)` : cancella il prodotto `p` , se esiste, oppure non fa niente; restituisce `true` se e solo se il prodotto esisteva.
- `void aggiungiOfferta(Offerta o)` : aggiunge l'offerta `o` .
- `boolean cancellaOfferta(Offerta o)` : cancella l'offerta `o` , se esiste, oppure non fa niente; restituisce `true` se e solo se l'offerta esisteva.
- `double prezzo(Prodotto[] p, boolean carta)` : restituisce il prezzo della spesa costituita dalla lista di prodotti `p` (assumendo che il cliente abbia la carta fedeltà se `carta==true`). **IMPORTANTE**: un prodotto può apparire molte volte nella lista `p` .