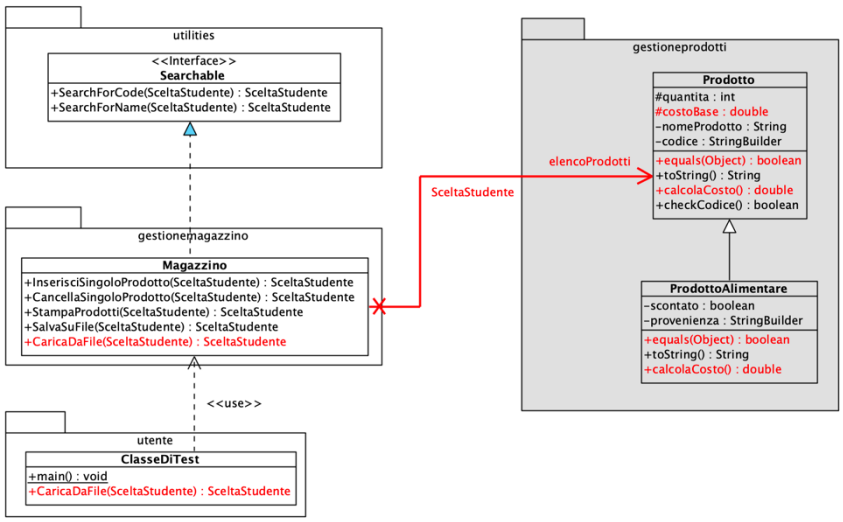
Corso di Programmazione

ESERCITAZIONE 3 del 23 maggio 2024

Proff. Amalfitano - Vittorini

Si vuole realizzare un'applicazione per la gestione di un magazzino di prodotti alimentari. Il programma riusa il codice dell'Esercitazione 1 (Esercitazione Ereditarietà Prodotto ← ProdottoAlimentare). L'applicazione deve essere strutturata in modo da rispecchiare il diagramma UML riportato. Per leggibilità, le classi del package `gestioneprodotti` non riportano i metodi di get e set, i costruttori di default, i costruttori con parametri e i costruttori di copia.



Per implementare il codice è richiesto che:

1. Lo studente esegua un refactoring del codice già disponibile dell'Esercitazione 1 in modo tale da:
    a. modificare il tipo della variabile `costoBase` da float a double e di conseguenza i parametri e i valori di ritorno delle funzioni di `set` e `get` e di tutte le funzioni impattate da tale variazione, come ad esempio i metodi `calcolaCosto`;
    b. generalizzare il metodo `equals` da overload a override.
2. Sia garantita la struttura a package riportata nel diagramma UML.
3. Lo studente scelga la relazione opportuna tra Magazzino e Prodotto. Si tenga presente che il Magazzino è in grado di gestire una collezione, denominata `elencoProdotti` di al più 50 prodotti. Lo studente garantisca la semantica del tipo di relazione scelta.
4. Si implementino i metodi esposti dalle classi del package `gestionemagazzino`. In particolare:
    a. `SearchForCode`: cerca un prodotto nell'elenco dei prodotti in magazzino in base al codice. Lo studente scelga il valore di ritorno del metodo e l'elenco dei parametri.

b. `SearchForName`: cerca un prodotto nell'elenco dei prodotti in magazzino in base al nome. Lo studente scelga il valore di ritorno del metodo e l'elenco dei parametri.

c. `InserisciSingoloProdotto`: inserisce un singolo prodotto nella prima posizione disponibile. Lo studente scelga il valore di ritorno del metodo e l'elenco dei parametri. Il metodo deve gestire opportunamente due casi di errore:

   i. il codice del prodotto che si vuole inserire è già presente il magazzino;

   ii. nell'inserimento si potrebbe andare oltre la dimensione massima della collezione.

d. `CancellaSingoloProdotto`: elimina un prodotto, con codice noto, dal magazzino. Lo studente scelga il valore di ritorno del metodo e l'elenco dei parametri. Il metodo deve gestire opportunamente il seguente caso di errore:

   i. il codice del prodotto non esiste in magazzino;

e. `StampaProdotti`: stampa a terminale i dati di tutti i prodotti presenti in magazzino.

f. `SalvaSuFile`: salva i dati di tutti prodotti presenti in magazzino in un file di testo. Lo studente scelga il valore di ritorno del metodo e l'elenco dei parametri e gestisca i casi eccezionali che potrebbero presentarsi.

5. Si implementi una classe di test che verifichi il corretto funzionamento dei metodi realizzati.

6. Affinché sia garantita la semantica del tipo di relazione scelta tra Magazzino e Prodotto la responsabilità del metodo `CaricaDaFile` deve essere asseganta o alla classe Magazzino o alla classe utente di test. Nello specifico il metodo `CaricaDaFile` carica i dati dei prodotti del magazzino da un file di testo. Lo studente scelga il valore di ritorno del metodo e l'elenco dei parametri e gestisca i casi eccezionali che potrebbero presentarsi. **[Eventualmente da svolgere a casa]**