Università di Pisa

Corso di laurea magistrale in Informatica Umanistica

Relazione progetto
Programmazione Java

Bari Ilenia 590445
Perfetti Paolo 551050

Anno accademico 2019/2020

Abstract

Il progetto da noi elaborato ha come fine quello di gestire un magazzino per un supermercato che vende varie tipologie di prodotti: prodotti generici (non alimentari) prodotti alimentari da banco e prodotti alimentari confezionati. Tutti i prodotti hanno come attributi un *nome* che li caratterizza, una *quantità* che indica il numero di prodotti presenti in magazzino e una *quantità minima* sotto la quale necessitano di essere riordinati. I prodotti alimentari da banco, oltre agli attributi già elencati, presentano anche l'attributo *"categoria merceologica"* che può avere il valore a scelta di *"prodotti da forno"* o *"salumi e formaggi"* o *"carni"* o *"pesce"*. I prodotti alimentari confezionati, oltre agli attributi elencati per i prodotti generici, presentano anche l'attributo *"modalità di conservazione"*, che può avere il valore di *"a temperatura ambiente"* o di *"in frigorifero"*.

Il programma deve permettere di poter inserire ed eliminare un certo prodotto, di prelevarne o di depositarne una certa quantità, e di effettuare delle ricerche all'interno del magazzino in base al nome, alla categoria merceologica, alla modalità di conservazione oppure optare per la visualizzazione dell'intero magazzino senza alcun filtro. Il contenuto della lista dei prodotti verrà salvato su file binario che sarà possibile ricevere anche in lettura.

Descrizione del programma

Il programma si compone di tre packages così formati, sulla base di una divisione organizzativa:

1. **Prodotti**: contiene al suo interno le classi ProdottoBanco, ProdottoConfezionato e Magazzino;

2. **eccezioni**: sono presenti le classi EccezioneCategoria, EccezioneConservazione, EccezionePrelievo, EccezioneQuantita, EccezioneQuantitaMinima, InputMismatchException;

3. **UsaMagazzino**: composta dalla classe UsaMagazzino entro la quale è contenuto il main e MetodiUsaMagazzino.

Package Prodotti

Il package Prodotti si occupa della gestione delle funzionalità delle classi dei singoli prodotti e del magazzino.

ProdottoGenerico

La classe ProdottoGenerico è la superclasse da cui ereditano le altre e al suo interno sono presenti:

- variabili d'istanza,quali di tipo String *nome*, e int *quantita* e *quantitamin*, alle quali sono stati applicati come modificatore di visibilità "protected". La variabile *nome* indica il nome del singolo prodotto, *quantita* la quantità del singolo prodotto e *quantitamin* rappresenta la soglia al di sotto della quale un prodotto deve essere riordinato;
- un costruttore;
- il metodo **to String** che restituisce la stringa contenente le informazioni del prodotto, quali *nome***,** *quantità* e *quantità minima*;
- il metodo **visualizza** che descrive lo stato dell'oggetto, prima stampando il nome della variabile e poi il valore.
- il metodo di tipo boolean ***equals*** per accertare o meno la presenza di un prodotto.

ProdottoBanco

La classe ProdottoBanco è una delle due sottoclassi di ProdottoGenerico e quindi eredita da questa le variabili *nome***,** *quantita* e *quantitamin*. Con la seguente classe si definisce una particolare tipologia di prodotto alimentare che presenta informazioni aggiuntive relative alla categoria merceologica. Le opzioni considerate, che rientrano in tale categoria, sono salumi e formaggi, prodotti da forno, pesce e carne.

E' presente all'interno di questa sottoclasse:

- un costruttore che richiama il costruttore della superclasse, a cui viene aggiunta la variabile *categoria* di tipo String;
- il metodo **visualizza**, già presente nella classe ProdottoGenerico, ma che con un meccanismo di overriding viene modificato con l'aggiunta del messaggio stampato e del valore relativo a *categoria*.

ProdottoConfezionato

La classe ProdottoConfezionato è l'altra sottoclasse di ProdottoGenerico, al pari di ProdottoBanco rappresenta un tipo di prodotto alimentare che prevede come informazioni aggiuntive rispetto alla superclasse, quelle relative alla modalità di conservazione, se a temperatura ambiente o in frigorifero.

Si compone di:

- un costruttore che richiama il costruttore della superclasse, a cui viene inserita la variabile *modcons* di tipo String che rappresenta la modalità di conservazione;
- il metodo **visualizza**, anche questo modificato attraverso il meccanismo di overriding con l'aggiunta del messaggio stampato e del valore relativo alla modalità di conservazione del prodotto.

Magazzino

La classe Magazzino rappresenta un magazzino entro il quale vengono creati i prodotti e gestite le operazioni richieste su di essi, ed è così composto:

- due variabili con visibilità private: *nomefile* di tipo String che identifica il nome del file da attribuire al file binario in lettura o scrittura; e *modificato* di tipo boolean inizializzato a false, utilizzato come flag per tenere traccia di eventuali modifiche non salvate su file;
- Viene creato un vettore "*lista*" della classe ProdottoGenerico con visibilità private, che andrà a contenere tutti i prodotti inseriti.
- Viene creato un costruttore al quale viene passato come parametro la variabile *nomefile* e, successivamente viene creato un oggetto **ObjectInputStream** denominato *nome_file*, che legge il contenuto del vettore presente nel file binario in input. Se il file non dovesse esistere, viene lanciata la *FileNotFoundException* con in aggiunta un messaggio che informa l'utente che il file non esiste e che verrà creato alla prima modifica, nel caso in cui il file non contenga alcun oggetto scatterà *ClassNotFoundException*, per altri eventuali errori di input/output verrà lanciata l'*IO Exception*.

I metodi presenti all'interno della classe gestiscono le singole possibili operazioni. Abbiamo in tal caso:

- dei metodi di controllo di tipo booleano tra cui: **controllopresenza** ritorna *true* se il prodotto è già presente nella lista, o *false* in caso contrario; **controlloCat** ritorna *true* o *false* se la variabile *categoria* corrisponde o meno alle opzioni stabilite, e in maniera analoga il metodo **controllaCons** confronta la variabile *modcons* con le alternative accettabili relative alla modalità di conservazione.
- **inserisciGenerico**, **inserisciBanco** *e* **inserisciConfezionato** permettono di inserire i vari tipi di prodotti alla lista.
- Il metodo non consente di aggiungere una quantità o quantità minima negativa, da cui scaturirebbe il lancio delle eccezioni dedicate (EccezioneQuantita e EccezioneQuantitaMinima).
  Nel caso in cui richiamando il metodo **controllopresenza** ritorni *false*, si procede all'inserimento del prodotto a cui seguirà un messaggio per il corretto inserimento, in caso contrario viene stampato un messaggio di errore.
- **eliminaProdotto** consente di scansionare l'intero vettore e nel caso in cui il nome inserito corrisponda al nome di un prodotto esistente, questo verrà eliminato, altrimenti si riceverà un messaggio di errore.
- **prelevaProdotto** e **riforniscProdotto** gestiscono il prelievo e il deposito di una data quantità di prodotto. In entrambi i metodi se la quantità inserita è negativa viene richiamata EccezioneQuantita. Per quanto riguarda il prelievo, nel caso in cui la quantità richiesta da prelevare sia maggiore della quantità in magazzino, viene lanciata EccezionePrelievo, Nel caso in cui non venga sollevata alcuna eccezione verrà prelevata o depositata la quantità richiesta e verrà stampato un messaggio con la nuova quantità aggiornata.
- **cercaNome** consente di effettuare una ricerca sulla base del nome richiesto, quindi viene scansionato il vettore contenente l'elenco dei prodotti, se questo è presente al suo interno, viene richiamato il metodo **visualizza** della classe a cui appartiene.
- **cercaCat** e **cercaCons** sono due modalità di ricerca all'interno del magazzino, dopo essersi assicurati che l'input sia corretto, e che il prodotto appartenga a una determinata classe, viene consentita la ricerca che restituisce il prodotto se presente, oppure un messaggio che informa circa l'assenza del prodotto.
- **visualizza**, consente di visualizzare l'intera lista di tutti i prodotti, a prescindere dalla classe a cui appartengono
- **prodottiDaOrdinare** visualizza tutti i prodotti la cui quantità è minore della soglia di riordino.
- **daSalvare** è un metodo booleano che restituisce l'esito del flag per tener traccia delle modifiche
- **salva** permette di salvare la lista generata da tutte le operazioni sul file binario attraverso la serializzazione ad oggetti.

Package UsaMagazzino

Il package UsaMagazzino si occupa della gestione delle funzionalità della classe **UsaMagazzino**, che contiene il main del programma, e della classe **MetodiUsaMagazzino**, che contiene i metodi relativi al funzionamento del main.

UsaMagazzino

La classe UsaMagazzino è la classe che contiene il main del programma, una prima parte contiene la dichiarazione delle variabili di tipo String *nome*, *modcons* e *categoria*, di tipo in *quantita* e *quantitamin* e delle variabili di tipo char *s* e *scelta*. L'utente inserisce il nome del file sul quale intende salvare le proprie operazioni; dopodiché viene creato un registro utilizzando il suddetto file.

In seguito viene presentato all'utente un menu con interfaccia testuale nel quale vengono dichiarate le possibili scelte. Digitando la propria scelta, in seguito al messaggio di richiesta, a prescindere che la lettera inserita sia maiuscola o minuscola, sarà possibile effettuare le seguenti operazioni:

- **[A]**ggiungi un nuovo prodotto, se selezionato richiede di effettuare una seconda scelta per definire il tipo di prodotto da voler aggiungere: "*0"* per il prodotto generico, "*1"* per il prodotto da banco, "*2"* per il prodotto confezionato e "*3"* se si desidera tornare al menu principale. Se l'opzione selezionata non esiste, verrà richiesto di inserirla nuovamente, fino a che questa non sia corretta. Per ogni opzione verranno richiesti in input i parametri *nome*, *quantita*, *quantitamin* e *categoria* o *modcons* nel caso dei prodotti da banco o confezionati, da passare ai vari metodi per aggiungere i prodotti. Attraverso il costrutto try-catch vengono gestite le seguenti eccezioni : EccezioneQuantita, EccezioneQuantitaMinima, InputMismatchException. Nel caso in cui venga inserita una categoria o una modalità di conservazione non valida viene richiesto di reinserirla. La scelta sulla gestione di questa eventualità si basa sull'osservazione che, trattandosi di una stringa, questa può essere più facilmente oggetto di errori di digitazione, mentre l'inserimento di una quantità negativa difficilmente può essere causa di un errore involontario, di conseguenza merita un'eccezione, discorso analogo per l'eccezione InputMismatchException, che in questo caso non comporta un arresto del programma;
- **[E]**limina prodotto, consente di eliminare un prodotto dalla lista sulla base del nome scelto dall'utente;
- **[P]**releva prodotto, chiede di inserire il nome del prodotto in questione e poi la quantità che si desidera prelevare. Tutto ciò controllando, tramite il costrutto try-catch, che non siano presenti le eccezioni EccezioneQuantita, EccezionePrelievo ed InputMismatchException;
- **[D]**eposita prodotto fa inserire le informazioni relative al nome del prodotto e alla quantità di questo che si desidera depositare. Nel caso in cui venissero inseriti valori non corretti, verranno lanciate le eccezioni EccezioneQuantita ed InputMismatchException;
- **[C]**erca prodotto, chiede all'utente di inserire nuovamente una scelta per il filtro della ricerca. Digitare *"0"* per cercare un prodotto in base al nome, "*1"* per cercare un prodotto in base alla modalità di conservazione, "*2"* per cercare un prodotto in base alla categoria merceologica, "*3"* per visualizzare tutti i prodotti presenti nel magazzino, "*4"* per tornare al menu principale. In questo caso non sono previste eccezioni, siccome, anche qui in caso di errato inserimento della categoria o della modalità di conservazione viene richiesto di reinserire l'opzione per i motivi sopra descritti.
- **[V]**isualizza prodotti da ordinare, stampa la lista dei prodotti che necessitano di un riordino, quindi quelli la cui quantità è inferiore a quella minima;
- **[S]**alva dati, salva le operazioni fino a qui effettuate;
- **[O]**esci, consente di interrompere il programma. Se ci sono modifiche da salvare, stampa un messaggio che consente di salvare le modifiche rispondendo "S" o "N". In entrambi i casi stampa un messaggio per informare l'utente di essere uscito dal programma.

MetodiUsaMagazzino

Nella classe MetodiUsaMagazzino sono presenti le variabili e i metodi relativi al funzionamento del main. Vengono inizialmente dichiarate le variabili String *categoria*, *modcons*, *nome*, e int *quantita* e *quantitamin*. Sono presenti inoltre:

- Due metodi di controllo: **controlloscelta** e **controlloscelta2**; prendono come parametro la variabile s e che restituisce come valore di ritorno *true* se viene inserito *"0", "1", "2" o "3"* nella prima o *"0", "1", "2", "3" o "4",* nella seconda restituendo in entrambe, false in caso contrario. Questi metodi sono stati creati per ottimizzare il controllo sui filtri per quanto riguarda le opzioni di Aggiungi prodotto e Cerca prodotto.
- Metodi d'inserimento dei parametri, sono stati creati per quelle variabili che vengono richieste più spesso. Questi sono **InserisciNome**, **InserisciQ**, **InserisciQmin**, **InserisciCat** e **InserisciCons**. Tutte stampano il messaggio relativo alla richiesta di inserimento e la possibilità di inserirle attraverso l'oggetto scanner *input*.