Lorenzo Mirto Bertoldo

PROGRAMMAZIONE AD OGGETTI

Scelte Progettuali:
Il progetto è svolto interamente in inglese per preferenza personale. Javadoc, annotazioni, nomi di classi e variabili varieranno da quelli proposti da quelle già proposte nella consegna, ma saranno comunque una traduzione letterale.
Ho scritto le mie classi di Eccezioni (3: `ValidationException` `DuplicateNameException`, `MissingNameException`) che estendono la classe Exception.

Questa scelta è stata fatta per permettermi di eventualmente aggiungere metodi e funzionalità alle classi di eccezioni, come pratica in vista di eventuale espansione progettuale.
L'eccezione Validation viene lanciata quando l'input fornito non è coerente o compatibile con l'operazione che si sta cercando di effettuare.
La DuplicateName solitamente per quando si sta cercando di creare un oggetto dallo stesso nome di uno esistente.
La MissingName quando si sta cercando di accedere ad un oggetto tramite un identificatore che non è presente in memoria.
Il progetto è diviso in 4 parti principali: MODEL, VIEW, CONTROLLER e TEST.

MODEL: Contiene 3 classi principali
-Article

la classe di più basso livello nel modello. Essa possiede 4 campi (String name, float cost, String category, int quantity). L'articolo è identificato dal suo nome e, dato che un oggetto di tipo Article potrà solo essere creato all'interno di una istanza ShoppingList, anche dalla ShoppingList in cui è inserito. (Posso creare oggetti Article con lo stesso nome, a patto che siano in ShoppingList differenti)
Lo user di conseguenza NON potrà quindi assegnare lo stesso nome a due articoli da prezzo o categoria differente, ma dovrà optare per l'assegnamento di nomi differenti (es: YogurtFaje, YogurtMuller).
All'oggetto Article, come da requisito, se dovesse essere assegnata una category non presente nella lista di categorie (spiegata in ListManager) verrà assegnata una String costante "Uncathegorized". Allo stesso modo, se una categoria dovesse essere cancellata dalla lista in questione, verrà effettuato un controllo su ogni Article di ogni ShoppingList affinché non venga lasciato in memoria alcun oggetto con una category non più esistente.
E' stata anche soddisfatta la richiesta (nello specifico nel metodo .addArticle() di ShoppingList e con i costruttori in overloading di Article) di settare i campi di category e quantity al valore predefinito ("Uncathegorized", 1) nel caso di input vuoto da parte dell'utente, non sarà tuttavia possibile inserire la stringa vuota per una entry per la natura del metodo .next() della classe importata Scanner per gestire gli input da terminale, dal momento che essa necessita di Token+Separatore(Invio), di conseguenza un invio senza alcun altro tipo di Token non permette l'accettazione dell'input.

-ShoppingList

la classe ShoppingList si tratta della rappresentazione di una lista della spesa, contraddistinta da un campo name, che sarà anche l'identificatore (non potrò quindi creare un nuovo oggetto lista dallo stesso nome di un altro già esistente). L'altro campo della classe è un ArrayList<Article>.
Essa implementa l'interfaccia Iterable e la scelta di utilizzo tra Iterator o costrutto For-Each nei metodi di ShoppingList, ma in particolare per ListManager (la classe successiva che andrò a descrivere) sarà determinata dalla complessità delle operazioni del metodo stesso.

A proposito di queste operazioni di iterazione su ArrayList, in diversi metodi ho inserito una mia variabile di controllo (solitamente chiamata articleMatch) che mi permette di fermare un ciclo presto in caso trovi il match di un campo di un oggetto che sto confrontando: questo perché per scelta progettuale i nomi delle liste e i nomi degli articoli all'interno di una singola non potranno mai essere ripetuti. → il primo match sarà anche l'unico in una determinata collezione di dati.

-ListManager

Questa classe rappresenta una collezione di ShoppingLists, ovvero un ArrayList<ShoppingLists>, e un secondo ArrayList<String> di strighe questa volta, perché è qua che conterrò le mie categorie.
Essa non ha identificatore, dal momento che runtime sarà possibile avere istanziato un solo oggetto di questa classe e non diversi. In essa sono racchiusi i metodi per l'inserimento e modifica di liste, e i metodi che vanno ad operare sulle ShoppingList, ma che interessano l'intera collezione di liste.
Uno di questi è il .setDefaultCategory(), che permette di andare ad modificare il dato category su ogni istanza di Article presente in ogni istanza di ShoppingList.

VIEW:

Contiene 2 classi principali, che verranno create direttamente dal main, nel quale, attraverso uno scanner, verrà creato un blocco decisionale per cui l'utente potrà scegliere per quale interfaccia optare.

-CommandLineInterface :
essa si fa passare come parametro l'oggetto Scanner istanziato nel main e avrà un campo manager, ovvero una singola istanza di oggetto ListManager, con accesso tutte le funzionalità implementate nelle 3 classi di model.
La CommandLineInterface è composta da un metodo .executeInterface(), un grosso ciclo while su uno switch il cui unico punto di ritorno al chiamante risiede nell'opzione Exit.
Le altre opzioni richiamano altri metodi della classe in cui sono descritte sequenza di istruzioni che permettono di effettuare (in ordine:)
-opzioni di visualizzazione liste e categorie
-opzioni di aggiunta e rimozione liste
-opzioni di aggiunta e rimozione categorie
-opzioni di aggiunta e rimozione articoli
-opzioni di ricerca articoli e calcolo prezzi
In queste operazioni vengono messe assieme tutte le operazioni definite nelle 3 classi di modello ed è a questo livello che funzionalità implicite (come la gestione dei campi category degli oggetti in seguito alla rimozione di categoria) vengono soddisfatte.

-GraphicUserInterface
è l'altra classe del modello che sostanzialmente funziona come la precedente, tuttavia essa avrà degli eventi da gestire (click di bottoni per invocare le stesse funzionalità che anche la CommandLineInterface soddisfa), di conseguenza il main istanzierà sia un oggetto di tipo GraphicUserInterface, sia uno di tipo Controller a cui passerà quello di GUI appena creato come parametro.
La classe definisce una GUI suddivisa in 3 colonne con funzionalità per Liste, Categorie e Articoli.

CONTROLLER:

ho avuto problemi a capire come gestire l'interfaccia. La gui non può essere utilizzata a causa degli eventListener non correttamente utilizzati. Le due classi interessate sono GraphicUserInterface e Controller.