UNIVERSITÀ DEL PIEMONTE ORIENTALE

ShoppingManager - Relazione

Nome: Alessandro Fatone
Matricola: 20019780

1 Introduzione

Questa relazione è stata redatta per il corso di Programmazione ad Oggetti, parte del modulo di Paradigmi di Programmazione, per il corso di Informatica del Dipartimento di Scienze e Informazione Tecnologica, presso la sede di Vercelli. L'applicazione sviluppata è rivolta alla gestione e manutenzione di varie liste della spesa, offrendo funzionalità per organizzare e modificare gli articoli e le categorie in modo efficace. Il codice sorgente è stato scritto in inglese, inclusi i nomi delle classi e dei package, per garantire una maggiore coerenza con le convenzioni di sviluppo internazionali. L'interfaccia utente, invece, è realizzata in italiano. Il documento descrive la progettazione e implementazione dell'applicazione in Java, con una panoramica delle scelte architetturali, delle classi principali, dei metodi pubblici e delle eccezioni.

2 Strumenti utilizzati

Per la realizzazione del progetto sono stati utilizzati i seguenti strumenti:

- **Gradle**: per la gestione delle dipendenze dei test (JUnit5), per organizzare la struttura del progetto e per la generazione della documentazione Javadoc.
- **Java 21**: la versione di Java utilizzata per lo sviluppo.
- **IDE**: il progetto è stato sviluppato con Visual Studio Code, ma il suo funzionamento è stato testato anche con IntelliJ IDEA.
- **LaTeX**: utilizzato per la generazione di questa documentazione.

3 Struttura del progetto

L'applicazione è strutturata in due sezioni principali, seguendo la convenzione nota come **Standard Directory Layout**:

- **src/main/java**: contiene le classi principali e il codice applicativo dell'applicazione.
- **src/test/java**: contiene i test unitari per verificare il comportamento delle classi di sistema (contenute nel package `model`).

Le classi principali dell'applicazione seguono il pattern **MVC** (Model-View-Controller) per l'interfaccia grafica. Questo approccio è stato scelto perché l'applicazione offre sia un'interfaccia a riga di comando (CLI) sia un'interfaccia grafica (GUI). La struttura del progetto è organizzata come segue: nella root del progetto si trovano il package `model`, il package `ui` e la classe `StartApp`, che rappresenta il punto di ingresso dell'applicazione. Di seguito vengono descritti i contenuti dei package principali:

- **model**: il package che contiene le classi di sistema e le eccezioni personalizzate.
- **ui**: package principale per l'interfaccia utente, suddiviso in due sottopackage:
  - **cli**: gestisce l'interfaccia a riga di comando. È suddiviso nei seguenti sottopackage:
    - **menu**: contiene le classi che rappresentano i vari menu dell'interfaccia CLI.
    - **base**: include la classe base dei menu, a cui i menu CLI fanno riferimento.
  - **gui**: gestisce l'interfaccia grafica. Oltre a includere la classe `MainGUI`, responsabile dell'inizializzazione dell'interfaccia grafica, il package `gui` è suddiviso in due sottopackage principali:
    - **view**: package che contiene i componenti grafici, inclusi i vari `frame`, `panel` e la `menu bar`.
    - **controller**: package che contiene le classi di controller che gestiscono l'interazione tra il `model` e la `view`.

4 Classi e Relazioni

In questa sezione vengono descritte le classi principali dell'applicazione e le loro relazioni, con un'illustrazione delle scelte progettuali adottate. Come già definito precedentemente, la classe `StartApp` è il punto d'avvio dell'applicazione e si trova nella radice del progetto. La sua unica funzionalità è quella di avviare la classe `UIHandlerMenu`, che è un menu CLI che consente all'utente di selezionare l'interfaccia che preferisce (tra CLI e GUI).

4.1 Classi del package MODEL

Le classi presenti nel package `model` rappresentano il nucleo logico dell'applicazione, gestendo i dati e le operazioni principali.

4.1.1 Classe Article

La classe `Article` rappresenta un articolo all'interno dell'applicazione, consentendo di gestire informazioni come nome, costo, quantità e categoria. Fornisce funzionalità per creare e manipolare un articolo, applicando delle convalide sui campi per garantire l'integrità dei dati. La classe include metodi per accedere e modificare i dettagli dell'articolo, assicurando che valori non validi siano gestiti correttamente e che vengano impostati valori di default quando necessario. Viene incluso anche un campo pubblico denominato `DEFAULT_CATEGORY`, che definisce la categoria predefinita `"Non Categorizzati"` da assegnare all'articolo se non viene specificata una categoria. Questo campo è dichiarato pubblico poiché viene utilizzato anche nelle altre classi che gestiscono le categorie, le liste o una singola lista della spesa, permettendo di centralizzare il valore della categoria predefinita.

4.1.2 Classe CategoryManager

La classe `CategoryManager` gestisce le categorie degli articoli, consentendo di aggiungere, rimuovere e aggiornare le categorie. Permette di mantenere un elenco centralizzato delle categorie disponibili, applicando convalide per garantire che ogni categoria sia valida e non duplicata. La classe utilizza un `Set` per memorizzare le categorie, implementato come `HashSet`. Questo approccio è stato scelto per garantire che ogni categoria sia unica, poiché un `Set` non permette duplicati. Inoltre, il `HashSet` fornisce operazioni di inserimento, ricerca e rimozione efficienti in termini di complessità temporale. Vengono anche utilizzate le collezioni per garantire una maggiore flessibilità nel gestire gruppi di oggetti. In particolare, il metodo `updateCategoryInAllLists` accetta una `Collection` di liste della spesa per aggiornare le categorie in tutte le liste fornite, rendendo il codice riutilizzabile con diverse implementazioni di collezioni (come `ArrayList` o `LinkedList`).

4.1.3 Classe InputOutputList

La classe `InputOutputList` gestisce le operazioni di input e output per il salvataggio e il caricamento delle liste della spesa da file. Fornisce metodi per scrivere le informazioni di una lista su un file e per caricare una lista da un file, gestendo eccezioni per garantire l'integrità dei dati.

- `void saveToFile(ShoppingList shoppingList, File filePath)`: salva una lista della spesa nel file specificato, sovrascrivendo il file se esiste già. Solleva una `FileOperationException` se si verifica un errore durante il salvataggio.
- `void loadFromFile(ShoppingList shoppingList, CategoryManager categoryManager, File filePath)`: carica una lista della spesa da un file, sostituendo gli articoli presenti nella lista. Utilizza il `CategoryManager` per aggiornare le categorie degli articoli. Solleva una `FileOperationException` se il file non esiste o se si verifica un errore durante il caricamento, e una `InvalidInputException` se i dati nel file non sono validi.
- `String checkListsDir()`: verifica l'esistenza della directory predefinita per il salvataggio delle liste della spesa (`lists`) e la crea se non esiste. Solleva una `FileOperationException` se non è possibile creare la directory o se un file con lo stesso nome esiste già ma non è una cartella.

4.1.4 Classe ListManager

La classe `ListManager` gestisce le liste della spesa, consentendo di aggiungere, rimuovere e ottenere liste specifiche. Organizza le liste utilizzando una struttura a mappa, dove ogni lista è associata a un nome unico, rendendo il recupero rapido ed efficiente. Per gestire le liste, la classe utilizza una `Map` implementata come `HashMap`. Questa scelta è motivata dalla necessità di associare a ciascun nome di lista un oggetto `ShoppingList` corrispondente, evitando duplicati nei nomi delle liste e garantendo tempi di accesso rapidi. La `HashMap` consente infatti operazioni di inserimento, ricerca e rimozione efficienti. Inoltre, il metodo che restituisce tutte le liste della spesa (ovvero `getShoppingLists`) utilizza una `Collection`, offrendo flessibilità nel trattamento del gruppo di liste senza specificare una struttura dati rigida. Questo rende il codice più versatile e permette di interagire con l'insieme delle liste in modo generico.

4.1.5 Classe ShoppingList

La classe `ShoppingList` rappresenta una lista della spesa e include metodi per aggiungere, rimuovere, cercare articoli, oltre a calcolare il totale della quantità e del costo di tutti gli articoli presenti. La classe implementa l'interfaccia `Iterable` per permettere l'iterazione sugli articoli contenuti nella lista. `ShoppingList` utilizza una mappa `Map` (implementata come `HashMap`) per memorizzare gli articoli, con il nome dell'articolo come chiave. Questo approccio garantisce che ogni articolo nella lista abbia un nome univoco, migliorando l'efficienza delle operazioni di aggiunta, rimozione e ricerca. L'uso di `HashMap` permette di ottenere prestazioni ottimali grazie alla gestione in tempo costante delle operazioni di accesso, fondamentale per gestire quantità numerose di articoli in modo efficace. La classe utilizza anche altre collezioni come `List`, `ArrayList` e `Collection` per lavorare in modo flessibile con gruppi di articoli, supportando sia l'accesso immutabile (per evitare modifiche accidentali) che l'ordinamento dei risultati di ricerca.

4.2 Eccezioni personalizzate

Le eccezioni personalizzate definite nell'applicazione vengono utilizzate per gestire situazioni specifiche che possono verificarsi durante l'esecuzione, come operazioni su file non riuscite, input non validi o elementi non trovati. Queste eccezioni migliorano la leggibilità del codice e facilitano la gestione degli errori, fornendo messaggi specifici che aiutano nella diagnosi dei problemi.

4.2.1 Eccezioni per elementi non trovati

Questa categoria di eccezioni si trova nel package `domain` e include le classi `ArticleNotFoundException`, `CategoryNotFoundException` e `ListNotFoundException`, tutte estensioni di `Exception`. Queste eccezioni vengono sollevate quando un elemento specifico, come un articolo, una categoria o una lista della spesa, non è presente o non può essere trovato. Ogni eccezione è dotata di un costruttore che accetta una stringa di messaggio, utilizzata per fornire dettagli sull'errore specifico. Questo approccio consente di creare istanze di eccezioni con messaggi personalizzati, facilitando la comprensione del contesto dell'errore per l'utente o il programmatore.

4.2.2 Classe FileOperationException

La classe `FileOperationException` estende `Exception` ed è utilizzata per gestire errori durante operazioni di input/output sui file, come errori di lettura o scrittura. Fa parte del package `io` e possiede due costruttori:

- `FileOperationException(String message)`: crea un'istanza dell'eccezione con un messaggio che descrive l'errore di file.
- `FileOperationException(String message, Throwable cause)`: crea un'istanza dell'eccezione con un messaggio e una causa, permettendo di tracciare la causa dell'errore.

4.2.3 Classe InvalidInputException

La classe `InvalidInputException` estende `Exception` ed è utilizzata per segnalare che un input fornito è invalido, come un valore nullo o non conforme. Anche questa classe fa parte del package `io`, però essa ha un solo costruttore:

- `InvalidInputException(String message)`: crea un'istanza dell'eccezione con un messaggio che descrive il problema dell'input non valido.

4.3 Classi del package CLI

Il package `cli` contiene le classi che gestiscono l'interfaccia a riga di comando dell'applicazione, fornendo un'interazione testuale per l'utente. Come già citato in precedenza, `cli` è formata da due sottopackage: il package `menu` che contiene le varie classi per i menu, e il package `base` che contiene la classe di base su cui fanno riferimento i menu. Ogni classe presente nel package `menu` si occupa della visualizzazione di un singolo menu CLI specifico, al fine di rispettare il principio della separazione delle responsabilità.

4.3.1 Classe CategoryMenu

La classe `CategoryMenu` fornisce un'interfaccia a riga di comando per la gestione delle categorie all'interno dell'applicazione. Estendendo la classe `BaseMenu`, questa classe consente all'utente di visualizzare le categorie esistenti, aggiungerne di nuove o rimuovere quelle non più necessarie. Ogni volta che una categoria viene rimossa, la classe si occupa anche di aggiornare le liste della spesa, sostituendo la categoria rimossa con la categoria predefinita definita nella classe `Article`. Il menu, visualizzato ciclicamente fino a quando l'utente sceglie di uscire, presenta opzioni per aggiungere o rimuovere categorie, con una gestione delle eccezioni per assicurare che l'input dell'utente sia valido. Le eccezioni `InvalidInputException` e `CategoryNotFoundException` vengono sollevate per gestire rispettivamente i casi di input non valido e di categorie non trovate, garantendo una gestione degli errori robusta e informativa.

4.3.2 Classe ListMenu

La classe `ListMenu` fornisce un'interfaccia a riga di comando per gestire le liste della spesa all'interno dell'applicazione. Estendendo la classe `BaseMenu`, consente agli utenti di creare nuove liste, rimuovere liste esistenti e gestire una lista selezionata. Il `ListMenu` interagisce con `ListManager` per eseguire operazioni sulle liste della spesa e con `CategoryManager` per garantire la coerenza delle categorie associate alle liste. Durante l'esecuzione del menu, l'utente può scegliere tra diverse opzioni fino a quando non decide di tornare al menu principale. Inoltre, la classe gestisce le eccezioni che possono sorgere a causa di input non validi o della mancanza di liste specificate, fornendo messaggi di errore appropriati per migliorare l'interazione dell'utente.

4.3.3 Classe MainMenu

La classe `MainMenu` rappresenta il menu principale dell'interfaccia a riga di comando dell'applicazione, estendendo `BaseMenu`. Serve come punto di accesso centrale, permettendo all'utente di navigare tra le principali funzionalità dell'applicazione: gestione delle liste della spesa e delle categorie, ritorno alla selezione dell'interfaccia utente, e terminazione del programma. La classe `MainMenu` utilizza un ciclo per mantenere attivo il menu fino a quando l'utente decide di uscire o di tornare alla selezione dell'interfaccia. Include opzioni per accedere al `ListMenu` per la gestione delle liste e al `CategoryMenu` per la gestione delle categorie. Ogni opzione viene presentata all'utente tramite il metodo `displayMenu`, che mostra le scelte disponibili. Le scelte dell'utente vengono gestite in `start`, e se l'utente sceglie di terminare o tornare indietro, viene richiesta una conferma per evitare la perdita di modifiche non salvate. La struttura della classe garantisce una navigazione intuitiva e un controllo sicuro sull'accesso alle diverse funzionalità dell'applicazione.

4.3.4 Classe ShoppingMenu

La classe `ShoppingMenu` è un menu CLI per la gestione di una singola lista della spesa, ed estende `BaseMenu`. Fornisce funzionalità avanzate per aggiungere, rimuovere, cercare articoli e svuotare la lista. Inoltre, include opzioni per salvare e caricare la lista da un file, utilizzando la classe `InputOutputList` per le operazioni di input/output. All'avvio, il `ShoppingMenu` mostra all'utente le varie opzioni di gestione e rimane attivo fino a quando l'utente non decide di tornare al menu precedente. Tra le sue funzioni principali, `ShoppingMenu` gestisce le seguenti operazioni:

- **Aggiunta e rimozione di articoli**: consente all'utente di inserire nuovi articoli con nome, costo, quantità e categoria, controllando anche la presenza della categoria nel `CategoryManager`. In caso di categoria non presente, l'utente può scegliere se aggiungerla o impostare la categoria predefinita.
- **Ricerca articoli**: permette di cercare articoli nella lista per nome o per categoria, fornendo risultati in base ai parametri specificati dall'utente.
- **Gestione della lista**: include la possibilità di svuotare completamente la lista, previa conferma da parte dell'utente.
- **Salvataggio e caricamento da file**: salva la lista in un file o la carica da un file, aggiornando sia gli articoli della lista che le categorie, utilizzando i metodi di `InputOutputList`.

Questa classe consente all'utente di mantenere un controllo completo sulla lista della spesa, con un'interfaccia CLI intuitiva per la gestione e il mantenimento della lista su file.

4.3.5 Classe UIHandlerMenu

La classe `UIHandlerMenu` è il menu di selezione dell'interfaccia utente, permettendo di scegliere tra l'interfaccia a riga di comando e l'interfaccia grafica. Questa classe, che estende `BaseMenu`, offre all'utente un punto di ingresso per scegliere il tipo di interazione preferito con l'applicazione, mantenendo una separazione chiara tra i due ambienti. Il metodo principale `start()` avvia un ciclo in cui l'utente può selezionare l'interfaccia desiderata o terminare il programma. Le opzioni disponibili includono:

- Avvio dell'interfaccia CLI, che istanzia e avvia il `MainMenu`.
- Avvio dell'interfaccia GUI, che sospende temporaneamente l'esecuzione della CLI finché la GUI non viene chiusa, garantendo la sincronizzazione tramite un oggetto `lock`.
- Uscita dall'applicazione, che chiude l'intero programma con conferma dell'utente.

`UIHandlerMenu` utilizza `SwingUtilities` per avviare la GUI in un thread separato, consentendo alla CLI di attendere il termine della GUI. Questa struttura offre flessibilità, consentendo all'utente di passare facilmente tra le interfacce o uscire dal programma in modo sicuro e intuitivo.

4.3.6 Classe BaseMenu

La classe `BaseMenu` è una classe astratta che fornisce le funzionalità di base per la gestione dei menu a riga di comando nell'applicazione. Serve come superclass per le altre classi di menu, implementando metodi di utilità per la gestione dell'input, l'interazione con l'utente e la gestione di errori e conferme. Questa classe utilizza un oggetto `Scanner` per leggere l'input da console e mette a disposizione metodi per leggere stringhe, numeri interi e decimali dall'utente, con gestione degli errori di formattazione. Inoltre, `BaseMenu` include metodi per mostrare messaggi e errori, come `showMessage()` e `showError()`, e per visualizzare liste di elementi con `displayItems()`. Un'altra funzione importante è la gestione delle conferme, con il metodo `confirmQuestion()` che richiede risposte sì/no all'utente. Essendo una classe astratta, `BaseMenu` definisce i metodi `start()` e `displayMenu()` come metodi astratti, imponendo alle classi derivate di implementare le proprie logiche per avviare e visualizzare i menu specifici. La presenza di metodi per controllare se una collezione è vuota o per terminare l'applicazione, come `isCollectionEmpty()` ed `exitProgram()`, consente di standardizzare la gestione dei menu nell'applicazione, migliorando la coerenza dell'interfaccia e facilitando l'estensione della CLI.

4.4 Classi del package GUI

Il package `gui` contiene le classi che gestiscono l'interfaccia grafica dell'applicazione. Come è già stato definito in precedenza, oltre a contenere la classe MainGUI, esso è suddiviso in due sottopackage principali che rispettano il pattern **MVC**: la `view` che contiene le classi per i componenti grafici, e il `controller` che fa da intermediario tra le classi di **model** e **view**.

4.5 Classi del package VIEW

Il package `view` fornisce l'interfaccia grafica dell'applicazione, organizzando le classi che gestiscono la visualizzazione e l'interazione con i dati principali in una singola schermata principale. Include pannelli per la gestione di articoli, categorie, liste della spesa e dettagli di un articolo selezionato, oltre a finestre di dialogo per l'inserimento di nuovi articoli e menu per l'accesso alle opzioni principali.

4.5.1 FRAME

Il package `frame` contiene la classe `MainFrame`, uno dei vari componenti della `view`, che è responsabile della visualizzazione della finestra principale e dell'organizzazione dei pannelli di interfaccia grafica. Estende `JFrame` e contiene vari componenti grafici, tra cui una barra dei menu (`MenuBar`) e diversi pannelli (`ListPanel`, `ArticlePanel`, `DetailPanel`, `CategoryPanel`) che visualizzano e gestiscono diverse funzionalità dell'applicazione. La classe configura questi pannelli utilizzando un layout di tipo `GridBagLayout` per distribuire i componenti nella finestra principale, assegnando loro spazi e posizioni specifiche. `MainFrame` imposta inoltre alcune proprietà della finestra, come la dimensione e il comportamento alla chiusura. In questo modo, la classe funge da contenitore centrale dell'interfaccia grafica, coordinando la disposizione e l'interazione dei pannelli e della barra dei menu per fornire un'esperienza utente coerente.

4.5.2 PANEL

Il package `panel` contiene le classi per la gestione dei pannelli dell'interfaccia grafica dell'applicazione. Ogni classe rappresenta un pannello specifico che visualizza e gestisce diverse informazioni dell'applicazione, come articoli, categorie, dettagli di un articolo e liste della spesa. Di seguito viene fornita una descrizione delle principali classi incluse in questo package:

- La classe `ArticlePanel` è un pannello dedicato alla visualizzazione e gestione degli articoli. Estendendo `BasePanel`, fornisce una lista grafica degli articoli e visualizza informazioni sul numero totale e sul costo complessivo degli articoli presenti. Questo pannello è organizzato con un layout `BorderLayout` e include una `JList` gestita da un `DefaultListModel` per la visualizzazione dinamica degli articoli. Comprende inoltre etichette informative (`totalArticlesLabel` e `totalCostLabel`) e pulsanti (`addArticleButton` e `removeArticleButton`) per aggiungere e rimuovere articoli, facilitando l'interazione con altre parti dell'applicazione tramite metodi getter.

- La classe `CategoryPanel` fornisce un'interfaccia grafica per la visualizzazione e gestione delle categorie. Anch'essa estende `BasePanel` e utilizza un layout `BorderLayout` per separare i componenti, con una lista (`JList`) gestita da un `DefaultListModel` e pulsanti (`addCategoryButton` e `removeCategoryButton`) per aggiungere o rimuovere categorie. I metodi getter permettono di accedere ai componenti principali, supportando l'integrazione con altre funzionalità dell'applicazione.

- La classe `DetailPanel` rappresenta un pannello dedicato alla visualizzazione dei dettagli di un articolo selezionato. Utilizzando un layout `BorderLayout`, questa classe offre un'area di testo non modificabile (`JTextArea`) avvolta in uno `JScrollPane` per permettere lo scorrimento, garantendo una visualizzazione chiara e completa dei dettagli anche in caso di descrizioni lunghe. Questo pannello consente l'aggiornamento dei dettagli visualizzati da altre parti dell'applicazione grazie al metodo getter che restituisce l'area di testo.

- La classe `ListPanel` è un pannello dedicato alla visualizzazione e gestione delle liste della spesa. Fornisce una lista grafica (`JList`) gestita da un `DefaultListModel`, avvolta in uno `JScrollPane` per facilitare la navigazione tra liste numerose. Include anche un pannello dei pulsanti con `addListButton` e `removeListButton` per aggiungere e rimuovere liste, offrendo un'interfaccia intuitiva per la gestione delle liste della spesa. La classe fornisce metodi getter per accedere ai componenti principali e aggiornare dinamicamente i dati visualizzati.

In sintesi, ogni classe all'interno del package `panel` è progettata per gestire in modo specifico un singolo tipo di contenuto, in moda da garantire la modularità e la manutenibilità dell'interfaccia grafica dell'applicazione.

4.5.3 INPUT

All'interno del package `input` si trova la classe `ArticleInputDialog`, che viene utilizzata per raccogliere i dettagli di un nuovo articolo quando l'utente preme il pulsante "Aggiungi articolo". Questa finestra di dialogo consente di inserire nome, costo, quantità e categoria dell'articolo, con vari controlli di validazione per garantire che i dati siano corretti. La classe presenta campi di testo per l'inserimento dei dettagli, un pulsante di conferma per validare e registrare i dati, e un pulsante di annullamento per chiudere la finestra senza salvare. In caso di input valido, l'utente può creare un oggetto `Article` tramite il metodo `getArticle`, che restituisce i dettagli inseriti sotto forma di istanza della classe `Article`.

4.5.4 BAR

Le classi `MenuBar` e `FileChooser` appartengono al package `bar` e forniscono funzionalità legate alla gestione delle opzioni di menu dell'interfaccia grafica, oltre a supportare le operazioni di salvataggio e caricamento delle liste.

- La classe `MenuBar` rappresenta una barra dei menu personalizzata per l'interfaccia utente. Organizza e visualizza le opzioni di menu suddivise in diverse categorie: il menu `File`, che include le opzioni per salvare e caricare le liste; il menu `Ricerca`, per cercare articoli per nome o per categoria; e il menu `Altro`, che consente di rimuovere tutti gli articoli o tornare alla selezione dell'interfaccia.
- La classe `FileChooser` è un selettore di file personalizzato che traduce le etichette e i suggerimenti dei pulsanti in italiano. Questa classe è inclusa nel package `bar` poiché è utilizzata in relazione alle opzioni di salvataggio e caricamento delle liste presenti nella barra dei menu. Attraverso una serie di configurazioni di `UIManager`, `FileChooser` modifica testi e suggerimenti degli elementi dell'interfaccia per garantire una presentazione coerente e localizzata, migliorando l'esperienza utente nelle operazioni di gestione dei file.

4.6 Classi del CONTROLLER

Il package `controller` contiene tutte le classi responsabili della gestione delle interazioni tra l'interfaccia grafica e la logica applicativa del progetto. Ogni controller nel package è specializzato in un'area specifica dell'applicazione e coordina le operazioni tra i vari componenti dell'interfaccia e i manager dei dati. `MainController` funge da controller principale, inizializzando i manager e i controller specifici, e garantendo la coerenza tra i diversi moduli. `ListController`, `ArticleController`, `CategoryController` e `MenuBarController` gestiscono rispettivamente le operazioni su liste della spesa, articoli, categorie e barra dei menu, permettendo all'utente di interagire con i dati tramite l'interfaccia.

4.6.1 Classe ArticleController

La classe `ArticleController`, parte del package `controller`, gestisce le operazioni relative agli articoli all'interno di una lista della spesa, integrando l'interfaccia grafica con la logica applicativa. Questo controller collega i pannelli `ArticlePanel` e `DetailPanel` con il `ListManager` e il `CategoryController` per permettere una gestione completa degli articoli. Quando l'utente preme il pulsante "Aggiungi Articolo", viene visualizzata una finestra di dialogo, `ArticleInputDialog`, che consente l'inserimento dei dettagli dell'articolo, successivamente aggiunto alla lista selezionata. `ArticleController` gestisce anche la rimozione degli articoli selezionati e l'aggiornamento dei dettagli visualizzati, rispondendo agli eventi generati dall'interfaccia utente e aggiornando le informazioni, come il numero totale di articoli e il costo complessivo, in tempo reale. La classe controlla inoltre l'esistenza della categoria inserita per l'articolo, interfacciandosi con il `CategoryController` per aggiungere nuove categorie o applicare la categoria predefinita in caso di mancata conferma da parte dell'utente. Infine, `ArticleController` si occupa della sincronizzazione della vista con il modello, mostrando i dettagli completi degli articoli selezionati nel pannello dedicato e assicurando la coerenza dei dati tra la lista visualizzata e il modello sottostante.

4.6.2 Classe CategoryController

La classe `CategoryController` è parte del package `controller` e si occupa della gestione delle categorie nell'interfaccia grafica dell'applicazione, sincronizzando il modello dei dati con la vista. Questa classe controlla l'aggiunta, rimozione e visualizzazione delle categorie, utilizzando il `CategoryPanel` per visualizzare le categorie disponibili e il `CategoryManager` per la gestione delle operazioni a livello di modello. Alla creazione, il `CategoryController` inizializza le categorie disponibili e aggiunge listener ai pulsanti di aggiunta e rimozione presenti nel `CategoryPanel`. Quando l'utente decide di aggiungere una nuova categoria, viene richiesto l'inserimento del nome della categoria tramite una finestra di dialogo. Se il nome è valido, la categoria viene aggiunta sia nel `CategoryManager` che nel `CategoryPanel`, aggiornando così la vista per riflettere immediatamente le modifiche. Analogamente, quando si richiede la rimozione di una categoria selezionata, il controller verifica la selezione, rimuove la categoria dal `CategoryManager` e aggiorna tutti gli articoli e liste che utilizzano tale categoria. Viene anche gestito il caso in cui una categoria non esista, fornendo un messaggio di errore all'utente. In questo modo, il `CategoryController` centralizza la logica delle categorie, assicurando una gestione uniforme e coerente delle operazioni legate alle categorie, sincronizzando efficacemente il modello e l'interfaccia utente.

4.6.3 Classe ListController

La classe `ListController` appartiene al package `controller` e gestisce le operazioni relative alle liste della spesa all'interno dell'interfaccia grafica dell'applicazione. Questa classe coordina l'interazione tra la vista, rappresentata dal `ListPanel`, e la logica applicativa delle liste, gestita dal `ListManager`. Inizialmente, il `ListController` si occupa di popolare la vista con le liste della spesa esistenti e aggiunge dei listener ai pulsanti per consentire all'utente di aggiungere o rimuovere liste. Quando viene richiesta la creazione di una nuova lista, il controller mostra una finestra di dialogo per ottenere il nome della lista dal'utente e, se valido, aggiunge la lista al `ListManager` e aggiorna la vista per riflettere l'operazione. La rimozione di una lista selezionata avviene tramite una conferma da parte dell'utente e, se confermata, il controller procede a eliminare la lista dal `ListManager`, aggiornando la vista e cancellando gli articoli associati. Inoltre, `ListController` comunica con l'`ArticleController` per aggiornare dinamicamente la visualizzazione degli articoli in base alla lista selezionata. Se nessuna lista è selezionata, mostra un messaggio informativo all'utente. La classe assicura una gestione centralizzata e reattiva delle liste della spesa, sincronizzando i dati tra il modello e la vista in modo intuitivo e coerente.

4.6.4 Classe MainController

La classe `MainController` è il controller principale dell'applicazione e coordina l'interazione tra i vari componenti dell'interfaccia grafica e la logica applicativa. Ha il compito di inizializzare e gestire i diversi manager, come `ListManager` e `CategoryManager`, che si occupano rispettivamente della gestione delle liste della spesa e delle categorie. La classe controlla la finestra principale dell'interfaccia, rappresentata da `MainFrame`, e gestisce la creazione dei controller specifici per le diverse parti dell'applicazione, tra cui `ListController`, `ArticleController`, `CategoryController` e `MenuBarController`. Quest'ultimo si occupa della gestione della barra dei menu, facilitando l'accesso alle principali opzioni dell'applicazione, come il salvataggio e il caricamento delle liste, e offre un punto di controllo centralizzato per le interazioni con l'utente. `MainController` si occupa di configurare le relazioni tra i controller per garantire che le operazioni eseguite su una parte dell'interfaccia abbiano riflesso su altre. Ad esempio, collega il `ListController` con l'`ArticleController` per assicurare che la selezione di una lista aggiorni automaticamente la visualizzazione degli articoli associati. Inoltre, il controller gestisce le azioni che devono essere eseguite al ritorno alla schermata principale, consentendo un'integrazione fluida delle funzionalità. L'architettura di `MainController` centralizza la logica di gestione dell'interfaccia e della sincronizzazione dei dati, assicurando che i vari componenti dell'applicazione operino in modo coordinato e coerente.

4.6.5 Classe MenuBarController

La classe `MenuBarController` è responsabile della gestione delle azioni principali nella barra dei menu dell'interfaccia grafica dell'applicazione. Questo controller permette di eseguire operazioni fondamentali, come il salvataggio e il caricamento delle liste della spesa, la ricerca di articoli per prefisso o categoria, e la rimozione di tutti gli articoli dalla lista selezionata. `MenuBarController` gestisce anche l'uscita dall'interfaccia grafica, consentendo all'utente di tornare alla schermata principale dell'applicazione. Per realizzare queste operazioni, il controller interagisce con vari componenti dell'applicazione: `ListController` per ottenere e manipolare la lista attualmente selezionata, `CategoryManager` e `CategoryController` per gestire le categorie associate agli articoli e aggiornare la visualizzazione della lista. Inoltre, utilizza la classe `InputOutputList` per la gestione delle operazioni di salvataggio e caricamento delle liste su file. Quando l'utente sceglie di salvare una lista, il controller apre una finestra di dialogo per selezionare il percorso di destinazione e verifica se il file esiste già, chiedendo conferma in caso di sovrascrittura. Analogamente, il caricamento di una lista permette all'utente di selezionare un file e richiede conferma prima di sovrascrivere i dati esistenti. Inoltre, sia per il salvataggio che per il caricamento di una lista, la finestra di dialogo viene impostata di default sulla cartella **lists**. La cartella **lists** viene creata automaticamente se non esiste, e se non è possibile crearla allora il `FileChooser` aprirà un'altra cartella predefinita del sistema operativo. La ricerca di articoli si basa su un criterio specificato, permettendo all'utente di filtrare gli articoli in base al prefisso del nome o alla categoria, con i risultati visualizzati in una finestra di dialogo. La rimozione di tutti gli articoli dalla lista selezionata richiede una conferma per evitare cancellazioni accidentali. In sintesi, `MenuBarController` centralizza la gestione delle funzionalità offerte dalla barra dei menu, fornendo un'interfaccia coerente e completa per la manipolazione delle liste e degli articoli.

5 Test

Il package `test/model` contiene i test unitari per verificare il corretto funzionamento delle classi principali.

5.1 ArticleTest

La classe `ArticleTest` è una classe di test per verificare il corretto funzionamento della classe `Article`. I vari test includono la verifica della creazione di oggetti `Article` con parametri validi e non validi, assicurando che i valori predefiniti vengano applicati correttamente. Per esempio, vengono testati il comportamento del costruttore e dei metodi `setName`, `setCost`, `setQuantity` e `setCategory`, sia con valori validi sia con valori non validi. In particolare, i test controllano che:

- Il costruttore e i metodi di `Article` impostino correttamente i valori predefiniti per il costo (impostato a zero se negativo), la quantità (impostata a uno se non positiva), e la categoria (impostata a `DEFAULT_CATEGORY` se nulla o vuota).
- Le eccezioni `InvalidInputException` vengano sollevate in caso di nome o categoria non validi, come un nome nullo o vuoto.
- I metodi `setName`, `setCost`, `setQuantity` e `setCategory` aggiornino correttamente i campi di `Article` quando vengono forniti valori validi.

Ogni test è isolato, con l'oggetto `Article` creato in un metodo `setUp` eseguito prima di ogni test e rimosso in `tearDown` al termine. Questo approccio garantisce che ogni test sia indipendente e che l'inizializzazione dell'oggetto di prova sia consistente tra i vari test.

5.2 CategoryManagerTest

La classe `CategoryManagerTest` è una classe di test per verificare il corretto funzionamento della classe `CategoryManager`. Garantisce la corretta gestione delle categorie, inclusa la creazione, l'aggiunta, la rimozione e l'aggiornamento delle categorie associate agli articoli nelle liste della spesa. I test eseguiti includono:

- **Test di inizializzazione**: verifica che il costruttore di `CategoryManager` crei correttamente un insieme di categorie contenente solo la categoria di default.
- **Test di aggiunta di categorie**: verifica che sia possibile aggiungere nuove categorie valide, e che vengano sollevate eccezioni per nomi di categorie nulli, vuoti, duplicati o che corrispondono alla categoria di default.
- **Test di rimozione di categorie**: verifica che sia possibile rimuovere categorie valide, e che vengano sollevate eccezioni se si tenta di rimuovere una categoria nulla, vuota, non esistente, o la categoria di default.
- **Test di aggiornamento delle categorie**: verifica che il metodo `updateCategoryInAllLists` aggiorni correttamente la categoria degli articoli nelle liste della spesa, sostituendo una categoria specifica con la categoria di default. I test controllano anche che nessuna modifica avvenga se la categoria non è presente negli articoli, e che non vengano sollevate eccezioni con liste di spesa vuote. Inoltre, verificano che vengano sollevate eccezioni quando le liste o la categoria sono nulli.

Anche in questo caso ogni test è isolato, con un oggetto `CategoryManager` creato in un metodo `setUp` eseguito prima di ogni test e rilasciato in `tearDown` al termine.

5.3 InputOutputListTest

La classe `InputOutputListTest` verifica la corretta funzionalità della classe `InputOutputList`, che gestisce le operazioni di input e output per il salvataggio e il caricamento delle liste della spesa su file. I test della classe includono:

- **Test di salvataggio su file**: verifica che il metodo `saveToFile` salvi correttamente una lista della spesa sia non vuota (controllando il contenuto del file), sia vuota (lasciando il file vuoto). Inoltre, verifica che venga sollevata un'eccezione `FileOperationException` per percorsi di file non validi.
- **Test di caricamento da file**: verifica che il metodo `loadFromFile` carichi correttamente i dati di una lista della spesa da un file contenente dati validi. Esegue anche controlli sui dati non validi, verificando che vengano sollevate eccezioni appropriate (`FileOperationException` per formato non valido e `InvalidInputException` per valori non validi di costo o quantità).
- **Test della gestione delle directory**: verifica che il metodo `checkListsDir` crei una cartella `lists` se non esiste, restituisca il percorso corretto se la cartella esiste già, e sollevi un'eccezione `FileOperationException` se esiste un file con lo stesso nome anziché una cartella.

La configurazione dell'ambiente di prova viene gestita dai metodi `setUp` e `tearDown` per creare e rimuovere file temporanei, assicurando un ambiente pulito e consistente per ogni esecuzione.

5.4 ListManagerTest

La classe `ListManagerTest` verifica la corretta funzionalità della classe `ListManager`, che gestisce le operazioni relative alle liste della spesa, come creazione, rimozione, recupero e gestione di più liste. I tipi di test eseguiti includono:

- **Test di aggiunta delle liste**: verifica che sia possibile aggiungere liste con nomi validi, controllando che le liste siano effettivamente aggiunte e accessibili. Inoltre, controlla che vengano sollevate eccezioni di tipo `InvalidInputException` quando si tenta di aggiungere una lista con nome nullo, vuoto o duplicato.
- **Test di recupero delle liste**: verifica che il metodo `getShoppingList` permetta di recuperare una lista esistente e lanci un'eccezione `ListNotFoundException` se la lista non è presente. Il test include anche la verifica del metodo `getShoppingLists` per assicurarsi che restituisca tutte le liste aggiunte o un insieme vuoto se non esistono liste.
- **Test di rimozione delle liste**: verifica che le liste possano essere rimosse correttamente e che dopo la rimozione non siano più accessibili. Controlla inoltre che venga sollevata un'eccezione `ListNotFoundException` se si tenta di rimuovere una lista non esistente.
- **Test di isolamento delle operazioni**: verifica che le operazioni su liste diverse siano indipendenti, assicurando che la rimozione o la modifica di una lista non influenzi altre liste.

I test sono progettati per garantire che `ListManager` gestisca correttamente le liste della spesa, mantenendo la coerenza e l'integrità delle operazioni di gestione delle liste.

5.5 ShoppingListTest

La classe `ShoppingListTest` è progettata per verificare la correttezza delle funzionalità della classe `ShoppingList`, che permette di gestire articoli in una lista della spesa, includendo operazioni di aggiunta, rimozione, ricerca e calcolo dei totali. Come tutte le altre classi di test, utilizza il framework `JUnit` per eseguire i diversi tipi di test sui metodi della classe. I principali tipi di test eseguiti sono:

- **Test del costruttore**: verifica che la lista della spesa sia creata correttamente con un nome valido e che venga lanciata un'`InvalidInputException` per nomi nulli o vuoti.
- **Test di aggiunta e rimozione degli articoli**: assicura che sia possibile aggiungere articoli validi alla lista, che l'aggiunta di articoli nulli generi un'`InvalidInputException`, e che sia possibile rimuovere articoli esistenti, lanciando un'`ArticleNotFoundException` per articoli non presenti.
- **Test di ricerca**: verifica che gli articoli possano essere trovati tramite prefisso o categoria, e che la ricerca sia case-insensitive. Testa inoltre le eccezioni appropriate per ricerche con stringhe nulle o vuote (`InvalidInputException`) e per categorie o prefissi non corrispondenti (`ArticleNotFoundException`).
- **Test di calcolo delle somme**: verifica che il metodo `getTotalFromList` calcoli correttamente la quantità totale e il costo degli articoli presenti nella lista e restituisca valori pari a zero quando la lista è vuota.
- **Test del metodo `iterator`**: assicura che l'iteratore della lista della spesa permetta di scorrere tutti gli articoli presenti.
- **Test del metodo `toString`**: verifica che il metodo `toString` restituisca correttamente il nome della lista.

Questi test garantiscono che la classe `ShoppingList` gestisca correttamente le operazioni comuni su una lista della spesa e che sollevi le eccezioni appropriate in presenza di errori.