

# Relazione Progetto ListaDiArticoli

## 1. Introduzione

Il progetto implementa un sistema per la gestione centralizzata di liste di articoli (es. liste della spesa o inventari). Il software permette di creare più liste indipendenti che condividono però un'anagrafica comune di articoli e categorie. È stata data particolare importanza alla persistenza dei dati e alla robustezza del codice tramite una gestione delle eccezioni.

## 2. Architettura del Sistema

Il sistema è strutturato secondo il pattern **MVC (Model-View-Controller)** per separare la logica del dominio dall'interfaccia utente.

### Modello (Package modello)

Questa sezione contiene la logica di business e le strutture dati fondamentali.

#### Classe Articolo

Rappresenta l'entità base del sistema.

- **Costruttore:** `public Articolo(String nome, String categoria, double prezzo, String nota).` Inizializza i campi validando il nome tramite l'espressione regolare:  
`[a-zA-Z0-9àèéìòù][a-zA-Z0-9\sàèéìòù]*`
- **Gestione Default:** Se non specificati, la categoria viene impostata su "Non categorizzato", il prezzo a 0 e la nota come stringa vuota.
- **Uguaglianza:** Il metodo `equals()` è ridefinito per considerare uguali due articoli con lo stesso nome e categoria, ignorando il case.

#### Classe ListaDiArticoli

Gestisce una collezione di prodotti suddivisa in articoli attivi e cancellati.

- **Soft Delete:** Il metodo `cancellaArticolo()` non elimina l'oggetto ma lo sposta nella lista `articoliCancellati`. È possibile ripristinare un elemento tramite `recuperaArticolo()`.
- **Iteratore:** Implementa `Iterable<Articolo>`. L'iteratore personalizzato attraversa prima gli articoli attivi e poi quelli nel cestino.
- **Calcolo Totale:** Il metodo `calcoloPrezzoTotale()` somma il prezzo dei soli articoli presenti nella lista attiva.

#### Classe GestioneListe

Funge da centro di controllo statico del sistema.

- **Registri Globali:** Mantiene liste statiche per `listeArticoli`, categorie e articoli totali nel sistema.
- **Persistenza:** Include i metodi `salvaSistema()` e `caricaSistema()` per la gestione dei dati su file `dati_sistema.txt` tramite `PrintWriter` e `BufferedReader`.

## Eccezioni (Package modello.exception)

Il sistema utilizza eccezioni personalizzate che estendono Exception per gestire gli errori in modo robusto:

- **ArticoloException:** Lanciata per nomi non validi o prezzi negativi.
- **ListaDiArticoliException:** Gestisce errori durante l'inserimento o la rimozione da una lista specifica.
- **GestioneListeException:** Segnala problemi nei registri globali o nel caricamento dei dati.

## 3. Interfaccia Utente

Il progetto fornisce due interfacce indipendenti, separate dalla logica del dominio.

### Interfaccia Grafica (Package gui.grafica)

Sviluppata con le librerie **Swing**, segue rigorosamente il pattern MVC:

- **Vista:** Composta da pannelli specializzati come PannelloListe (riepilogo globale), PannelloCategorie (gestione anagrafica) e ContentListaPanel (visualizzazione tabellare degli articoli).
- **Controllo:** Classi come ControlloGestore e ControlloLista implementano ActionListener per gestire i comandi dell'utente e aggiornare il modello.
- **Feedback Visivo:** Gli articoli nel cestino vengono evidenziati graficamente con un colore differente nella tabella.

### Interfaccia da Riga di Comando (Package gui.rigaComando)

La classe InterfacciaRigaDiComando permette l'interazione testuale tramite menu numerici.

- **Metodi principali:** visualizzaMenu(), selezionaLista(), cercaArticoloInLista().
- **Ricerca:** Permette di trovare articoli tramite prefisso sia tra gli attivi che tra i cancellati.

## 4. Guida all'Esecuzione

Il codice sorgente è fornito nella cartella src. L'applicazione può essere avviata in due modi:

1. **Avvio tramite Main:** Eseguendo la classe main.Main, all'utente viene presentato un menu testuale per scegliere tra l'avvio dell'interfaccia grafica (tasto 1) o dell'interfaccia da riga di comando (tasto 2).
2. **Avvio diretto GUI:** È possibile avviare direttamente l'ambiente grafico eseguendo la classe gui.GestoreGui (che contiene un proprio metodo main).

### Comandi per l'avvio da terminale:

Compilazione (dalla cartella principale del progetto)

```
mkdir -p bin
```

```
javac -d bin -sourcepath src src/main/Main.java
```

Esecuzione tramite Main (Scelta GUI o CLI)

```
java -cp bin main.Main
```

**Note operative:**

- **Caricamento:** Al lancio, il sistema tenta di ripristinare i dati dal file dati\_sistema.txt. Se il file è assente, il sistema si inizializza vuoto.
- **Salvataggio:** In fase di chiusura, se sono state apportate modifiche, il programma richiede conferma per il salvataggio dei dati.

## 5. Testing (Package modello.test)

Il sistema include classi di test JUnit 5 per ogni entità del dominio:

- **ArticoloTest:** Verifica la validazione dei dati e l'uguaglianza tra oggetti.
- **ListaDiArticoliTest:** Controlla la logica del cestino, dell'iteratore e del calcolo dei prezzi.
- **GestioneListeTest:** Testa la gestione globale delle categorie e delle liste.