Relazione del progetto di Programmazione Java

Gestione di una concessionaria di auto

Eleonora Rossi

2020/21

1 Introduzione

Il programma riguarda la gestione di una concessionaria di auto, in particolare l'archivio delle auto in vendita sia usate che nuove. Il programma è costituito da quattro classi: Auto, AutoNuova, AutoUsata (descritte nel paragrafo 2) e ArchivioConcessionaria; quest'utlima **contiene il metodo main**. Tutte le classi sono all'interno del package concessionaria.

Il metodo main ha al suo interno l'interfaccia testuale principale costituita da un menù grazie al quale è possibile accedere a tutte le funzionalità implementate (alcuni esempi sono aggiungiAuto(), rimuoviAuto(), visualizzaAutoOrdinate ecc.). Hanno una certa rilevanza i metodi salvaElenco() e caricaElenco(), i quali permettono di salvare e caricare l'elenco delle auto rispettivamente, attraverso la serializzazione. Infine il metodo esci() consente di terminare l'esecuzione del programma, oltre che salvarne, se si desidera, lo stato.

All'interno della classe ArchivioConcessionaria sono presenti poi ulteriori metodi ausiliari volti al controllo dell'input immesso dall'utente come controlloTarga(), controlloAnno() e ottieniData(), quest'ultimo permette inoltre di formattare l'input immesso nel formato data richiesto. Ho deciso di realizzare dei metodi specifici in quanto richiamati ed utilizzati in più parti del programma, così da evitare un'eccessiva ridondanza di codice.

2 Gestione dell'ereditarietà e descrizione delle classi

Una prima scelta implementativa ha riguardato la gestione dell'ereditarietà fra le classi. Ho deciso di rendere le classi AutoNuova e AutoUsata estensioni della classe Auto. Le sottoclassi infatti definiscono oggetti con una struttura più ricca e realizzano funzionalità aggiuntive rispetto alla superclasse già definita.

Le classi presentano variabili d'istanza private, rese accessibili e modificabili, quando necessario, tramite metodi pubblici get e set, seguendo così la logica dell'incapsulamento.

3 Gestione delle strutture dati

A questo punto è stato necessario decidere che tipo di struttura dato utilizzare per rappresentare la lista di auto. *Array* è una struttura dati non dinamica, ovvero la sua dimensione è fissata al momento della creazione, in questo caso tuttavia non era possibile sapere a priori il numero di auto da inserire, di conseguenza la scelta si è ridotta a *Vector* e *ArrayList*. Le due strutture sono molto simili fra loro, ciò che le distingue è la loro velocità nelle operazioni: essendo *ArrayList* non sincronizzata opera più velocemente rispetto a Vector. L'elenco di auto è stato quindi rappresentato come *ArrayList*.

Anche la data di consegna, membro della classe AutoNuova, ha richiesto un attimo di riflessione. Inizialmente ho definito tale dato come *String*, ma *Date* è risultato più opportuno, in quanto la classe che definisce tale oggetto presenta già diversi metodi che sono stati utili per realizzare alcune funzionalità come ricercare la prossima auto nuova da consegnare presente in RicercaConsegna().

4 Gestione della visualizzazione

Altre scelte implementative hanno riguardato la visualizzazione delle informazioni relative alle auto. Mi sono chiesta se rendere reperibili le variabili d'istanza nelle classi Auto, AutoNuova, AutoUsata attraverso dei metodi pubblici e successivamente stampare le varie informazioni all'interno della classe ArchivioConcessionaria, oppure implementare dei metodi all'interno di tali classi volti a stampare direttamente le variabili dell'oggetto.

La seconda ipotesi tuttavia è parsa dal punto di vista logico più immediata rispetto alla prima, in quanto i vari metodi get, che altrimenti sarebbe dovuti essere implementati nella classe Auto al fine di recuperare le informazioni, non avrebbero avuto altro utilizzo che non per la stampa di tali informazioni. Di conseguenza all'interno della classe Auto ho ridefinito la classe toString tramite *override* e realizzato un metodo astratto visualizzaAuto() (motivo per il quale la classe Auto è astratta). Successivamente ho ridefinito tale metodo astratto all'interno di ciascuna sottoclasse in modo tale da stampare le informazioni necessarie.