Il progetto per il gestore di un autonoleggio vuole permettere all'utente di registrare dei mezzi, effettuare prenotazioni su di essi, rimuovere e visualizzare prenotazioni, rimuovere e visualizzare mezzi. Il gestore è stato sviluppato principalmente in versione testuale, con una successiva implementazione di interfaccia grafica. Rispettivamente, i main() delle due modalità si trovano nelle seguenti classi:

- Testuale: AutonoleggioTestConsole.java
- GUI: FinestraMain.java

Per quanto riguarda la GUI si specifica che non sono state implementate tutte le funzioni previste dalla versione testuale, per la quale sono state aggiunte alcune features che vanno oltre la traccia in modo da rendere più agevole e, si spera, verosimile la gestione un programma in condizioni reali.

Versione testuale:

Il programma ha iniziato il suo sviluppo all'interno delle classi AutonoleggioTestDemo e Autonoleggio. Il primo, da ora in poi gestore, si occupa di creare a ogni avvio un oggetto della classe Autonoleggio (nel cui costruttore è previsto il passaggio per caricare eventuali file binari contenenti ArrayList<Veicolo>), di gestire un ciclo do-while con tutte le sue funzionalità, previste dalla traccia, che l'utente gestisce con input da tastiera e la parte di chiusura e salvataggio dei dati su un file binario.

La classe Autonoleggio invece gestisce la totalità dei metodi implicati dal gestore. Inoltre conserva al suo interno un ArrayList di oggetti Veicolo, identificati univocamente tramite numero di targa, che rappresentano il fulcro dei suoi metodi. Come da traccia, ogni oggetto Veicolo in concreto viene rappresentato come Autovettura o Furgone (da cui ereditano variabili e metodi e che implementano le proprie varabili e metodi) e contiene a sua volta un ArrayList di oggetti Prenotazione, che impegnano il Veicolo in una certa data (in questo caso LocalDate) per un certo richiedente. Analizziamo in concreto le funzionalità del gestore e quali classi create vi sono coinvolte, al netto delle librerie fornite da Java.

1) [I] mmatricola una vettura (AutonoleggioTestDemo, Autonoleggio, Veicolo, Autovettura OR Furgone)

Come suggerito, l'Autonoleggio gestisce gli input di stringhe targa e modello dall'utente, ne verifica la validità e crea un'istanza di Autovettura o di Furgone, aggiungendola all'ArrayList<Veicolo> di Autonoleggio. Si sottolinea che ogni istanza di Autovettura o Furgone è creata come TIPO APPARAENTE/REALE, in quanto i metodi implementati fanno ampio uso di cicli for-each che coinvologono oggetti di tipo Veicolo e operano su variabili di questa superclasse. L'intenzione è quella di scongiurare l'eventualità di errori a runtime.

2) [R] imuovi una vettura (AutonoleggioTestConsole, Autonoleggio, Veicolo)

Come suggerito, l'Autonoleggio gestisce un input stringa targa e rimuove il Veicolo corrispondente dal proprio parco mezzi, ammesso che il veicolo esista e che non sia flaggato come prenotato, ovvero non abbia oggetti Prenotazione all'interno del suo ArrayList<Prenotazione>.

3) [L] ista mezzi disponibili (AutonoleggioTestConsole, Autonoleggio, Veicolo, Autovettura AND/OR Furgone)

Visualizza un elenco degli Veicolo presenti in parco mezzi. Si gestisce l'input dell'utente che può decidere se visualizzare una sola sottoclasse di Veicolo oppure tutti indiscriminatamente.

4) [P] renota un veicolo (AutonoleggioTestConsole, Autonoleggio, Veicolo, Prenotazione)

Metodo per aggiungere un oggetto Prenotazione all'interno dell'ArrayList dedicato all'interno di ciascun Veicolo. L'Autonoleggio gestisce la parte iniziale della prenotazione, ovvero l'input targa del Veicolo da prenotare. Dopodiché il costruttore di Prenotazione prenderà in input 3 valori per giorno, mese e anno, creerà un oggetto LocalDate corrispondente ed effettuerà una serie di controlli sulla validità (a questo scopo è stata creata un'eccezione ErroreFormatoData, principalmente riservata alle prenotazioni effettuate nel passato rispetto all'invocazione del metodo). Se tutto va a buon fine, l'Autonoleggio aggiunge la Prenotazione all'interno del Veicolo richiamato per il metodo e il Veicolo verrà flaggato come prenotato e pertanto non più rimuovibile dal parco mezzi dell'Autonoleggio.

5) [C] ancella Prenotazione (AutonoleggioTestConsole, Autonoleggio, Veicolo, Prenotazione)

Rimuove un oggetto Prenotazione all'interno dell'ArrayList corrispondente di un Veicolo, selezionato da input tramite corrispondenza di targa. Dopodiché l'utente seleziona l'indice della prenotazione da rimuovere. Se la Prenotazione era l'ultima contenuta nell'ArrayList, il Veicolo viene flaggato come non prenotato e pertanto è possibile procedere alla rimozione dal parco mezzi di Autonoleggio.

6) [V] isualizza prenotazioni veicolo/data (AutonoleggioTestConsole, Autonoleggio, Veicolo, Prenotazione, SortNome OR SortData)

Il metodo chiede all'utente se vuole visualizzare quegli oggetti Veicolo del parco mezzi che non hanno oggetti Prenotazione contenenti una data specifica, da gestire in input, o se vuole visualizzare tutti gli oggetti Prenotazione appartenenti a un singolo Veicolo, la cui targa va acquisita in input.

Nel primo caso, viene anche chiesto se l'utente vuole stampare le prenotazioni in ordine di data o nome tramite due Comparator creati apposta (SortNome/SortData).

Fatti salvi alcuni lievi ritocchi, i metodi precedenti ricalcavano la traccia. Di seguito i metodi che aggiungono funzionalità aggiuntive.

7) [A] ggiungi veicolo di stato (AutonoleggioTestConsole, Autonoleggio)

Come si sarà intuito dalla lettura finora, all'utente è richiesto molte volte l'inserimento di dati. Ciò può risultare seccante per operazioni multiple su uno stesso mezzo, come l'inserimento o la visualizzazione di prenotazioni. Il veicolo di stato è una variabile pubblica di Autonoleggio che conserva in memoria un Veicolo e lo passa come argomento delle varie funzioni presenti nel ciclo do-while, saltando la fase di input per la targa. Se un veicolo viene rimosso dall'Autonoleggio mentre è salvato nella variabile veicolo di stato, questa viene impostata a null.

8) [E] limina veicolo di stato (AutonoleggioTestConsole, Autonoleggio)

Imposta il veicolo di stato a null.

9) [S] etta operatore (AutonoleggioTestConsole, Autonoleggio)

Imposta tramite input un nuovo operatore per l'Autonoleggio. Feature estetica.

10) [M] odifica dati veicolo (AutonoleggioTestConsole, Autonoleggio, Veicolo, Furgone OR Autovettura)

Una volta ottenuto in input un numero di targa, permette di modificare o targa o modello di un Veicolo. Avrei potuto implementare anche una selezione del veicolo da modificare tramite indice (come per la rimozione delle prenotazioni) ma ho preferito questo sistema perché in teoria un autonoleggio dovrebbe avere decine e decine di mezzi e risulta difficoltoso individuare in una lunga lista quello su cui intervenire.

11) [T] ermina programma (AutonoleggioTestConsole, Autonoleggio, Veicolo, Autovettura, Furgone, Prenotazione)

Esce dal ciclo do-while e salva l'intero ArrayList<Veicolo> dell'Autonoleggio utilizzato dal gestore per un futuro riutilizzo.

Versione GUI:

Il programma è gestito interamente dalla FinestraMain, che visualizza le 6 funzioni principali richieste dalla traccia e al suo avvio gestisce la creazione di un Autonoleggio, caricando eventuali file binari esistenti contenenti ArrayList<Veicolo> e sovrascrivendolo al termine. Per le 6 funzioni sono state create 8 classi differenti di finestra derivate da JFrame, ognuna creata avendo come argomento la FinestraMain, alla quale si torna alla fine di ogni operazione, e l'Autonoleggio, sul cui parco mezzi si opera. Avrei potuto impostare la GUI in maniera più corretta facendo derivare tutte le finestre non direttamente da JFrame ma da una finestra intermedia che presentasse i caratteri comuni a tutte, ma alla fine mi sono trovato più a mio agio impostando il lavoro per singoli frame. Fondamentale è il fatto che ogni finestra venga creata passando come argomenti una FinestraMain e un Autonoleggio, in modo da poter realizzare la finestra principale come un hub e poter modificare il parco mezzi dell'autonoleggio e degli oggetti ivi contenuti da qualsiasi finestra della GUI.

Il programma dialoga con l'utente tramite finestre di dialogo (FinestraDialogo) e JLabel contenuti in ogni finestra oltre alla FinestraMain. La versione GUI non prevede la scorciatoia del veicolo di stato, la possibilità di modificare i dati di un veicolo e di cambiare operatore.