Un’azienda di autonoleggio vuole tenere traccia dei veicoli disponibili e fornire ai propri clienti delle informazioni al riguardo.

Parte 1 - Creazione di una classe

- Crea una classe per il veicolo, con attributi come targa, modello e anno di produzione
- Istanzia nel main qualche veicolo e mostra le loro informazioni.
- Aggiungi un metodo per calcolare l’età del veicolo partendo dall’anno di produzione, e aggiungi l’informazione nella stampa delle informazioni del veicolo.
- Modifica la classe per gestire anche la marca del veicolo, separatamente dal modello, e implementa quindi 3 getter, uno per la marca (es. BMW), uno per il modello (es. X1) e uno per il nome completo del modello (es. BMW X1).

Parte 2 - Uso delle librerie

- Implementa l’overload del costruttore, in modo da avere adesso un costruttore in cui viene passato come parametro sia la marca che il modello e uno in cui viene passato invece il nome completo del modello, che dovrà essere scomposto in marca e modello separando la stringa sul primo spazio.
- Riscrivi il metodo per calcolare l’età del veicolo in modo da non dover scrivere l’anno corrente in alcuna parte del codice come costante e da non doverlo passare come parametro. Cerca su internet la giusta funzione di libreria utilizzata per recuperare la data dal sistema e, da questa, estrarre l’anno.
- Riscrivi il metodo che mostra le informazioni del veicolo utilizzando il metodo toString. Ricorda che il metodo toString non va mai richiamato esplicitamente

Parte 3 - Collezioni di oggetti

- Implementa una classe Concessionario, che permette di vendere veicoli. Il concessionario ha una lista di veicoli in vendita nella quale vanno aggiunti e rimossi veicoli tramite dei metodi specifici. Quando un veicolo viene rimosso, viene incrementato il guadagno del concessionario, aggiungendo il prezzo di vendita del veicolo.
- Implementa una funzionalità di ricerca del veicolo, che restituisca un array di veicoli di una certa marca data in input e un altro che restituisca un array dei veicoli prodotti in un intervallo compreso tra un minimo e un massimo.
- Implementa una funzionalità di ricerca del veicolo per targa. I metodo, se lo trova, restituisce sempre un solo veicolo. La classe Concessionario, quindi, non può contenere più di un veicolo con la stessa targa. L’aggiunta di un veicolo con targa duplicata deve fallire comunicando un errore un valore di ritorno.
- Verificare che gli array restituiti dai metodi di ricerca contengano solo ed esattamente i veicoli corretti. Nell’implementazione di questo punto, la collezione di veicoli dovrà essere iterata solo una volta, e, se si usa una ArrayList di appoggio, poi questa non può essere iterata nuovamente. Trovare il modo di convertire, quindi, l’ArrayList in un array.

Parte 4 - Ereditarietà

- Implementare una classe per le auto come sottoclasse del veicolo che, in aggiunta, ha una dimensione del bagagliaio e un numero di posti per le quali sono omologate. Dopo l’implementazione della classe, l’unica modifica fatta al di fuori della classe dovrà essere la creazione di alcune istanze per la loro aggiunta nel concessionario.

Nella stampa delle informazioni non dovrà essere modificato niente al di fuori della classe Auto perché basterà l'implementazione corretta del metodo toString.

- Rendere astratta la classe Veicolo e implementare una classe per le moto che permetta di creare istanze di veicoli che non hanno un bagagliaio. Portare il getter del numero di posti dalla classe auto alla classe Veicolo ed implementarlo nella classe moto in modo da restituire sempre il valore 2. Non possono esistere moto omologate per un numero di passeggeri diverso da 2. Anche in questo caso, andranno aggiunte al concessionario alcune istanze di moto per il test.
- Aggiungere al concessionario degli overload dei metodi di ricerca, che accettino in più un parametro che può avere i valori "auto", "moto" o "qualsiasi" e che quindi restituiscano solo veicoli del tipo corrispondente. Per implementare il parametro che indica il tipo del veicolo, utilizzare una enum. Per restituire gli elementi di tipo corretto, sarà necessario verificare il tipo delle istanze tramite instanceof.
- Aggiungere al concessionario un metodo che restituisca le auto con un bagagliaio di una certa capienza minima passata come argomento. Dal momento che nel concessionario dovrà a questo punto esserci una sola collezione di veicoli e non una di auto e una di moto separate, e dal momento che il getter della capienza del bagagliaio è nelle auto ma non nei veicoli, le istanze su cui si itera andranno downcastate da Veicolo a Auto, controllando tramite instanceof che non si downcastino mai oggetti di classe sbagliata.

Parte 5 - Eccezioni

- Riscrivere il metodo di aggiunta del veicolo in modo da eliminare il boolean restituito che serve ad indicare la riuscita aggiunta del veicolo. Il metodo non restituirà quindi nessun valore, ma lancerà un'eccezione nel caso in cui già esista un veicolo con la stessa marca. L'eccezione dovrà essere implementata tramite una classe specifica creata apposta, non possono essere utilizzate classi di libreria.
- Catturare l'eccezione in maniera appropriata nel main per la notifica all'utente.

Parte 6 - File

- Eliminare la creazione delle istanze di test all'inizio del main e, invece, riempire il concessionario tramite lettura di dati da un file.

Parte 7 - File (sfida)

- I metodi di aggiunta e rimozione dei veicoli dovranno adesso modificare anche il file utilizzato per riempire il concessionario all'inizio del main, in modo da rendere permanenti le modifiche. Avvii successivi dell'applicazione, quindi, dovranno mantenere le modifiche effettuate tra l'una e l'altra.