Automobili

Scritto dell'Appello di Programmazione Gennaio 2019 (Java)

Introduzione

In questo esame vi chiederemo di scrivere un insieme di classi per che riguardano automobili, patenti ecc.

Classi

Descriviamo ora nei dettagli come realizzare le classi necessarie al progetto. Se lo ritenete, potete apportare cambiamenti alle segnature dei metodi proposti e anche – se lo ritenete – alla struttura delle classi, restando ovviamente all'interno dei requisiti esposti nel paragrafo introduttivo.

N.B. (1): Oltre ai metodi qui indicati, potete aggiungere ad ogni classe un valido metodo `toString()` e, dove lo ritenete, anche un metodo `equals` e un metodo `hashCode` (con le segnature appropriate e che soddisfino i contratti previsti).

N.B. (2): Nel seguito, useremo il termine *lista di X* per indicare un qualunque tipo che consenta di conservare delle sequenze di oggetti di tipo `X`: potete implementare una lista con un array (`X[]`) oppure usando ad esempio delle `ArrayList<X>` (pacchetto `java.util`).

Classe Data

Potete supporre (***SENZA DOVERLA SCRIVERE***) di disporre di una classe `Data` le cui istanze rappresentano date (gg/mm/aaaa). La classe dispone dei seguenti costruttori e metodi:

- `Data(int g, int m, int a)`: crea la data g/m/a.
- `Data(Data d)`: crea una copia della data `d`.
- `boolean precede(Data d)`: restituisce `true` se e solo se questa data precede temporalmente (viene prima di) `d`.
- `int giorniFinoA(Data d)`: restituisce il numero di giorni fra questa data e `d`; il numero è negativo se `d` precede questa data, zero se le due date coincidono.
- `Data fra(int giorni)`: restituisce la data ottenuta aggiungendo il numero di giorni specificati a questa data (o togliendoli, se `giorni` è negativo). Ad esempio, `d.fra(1)` è il giorno dopo `d`, mentre `d.fra(-1)` è il giorno che precede `d`.
- `Data fraAnni(int anni)`: restituisce la data ottenuta aggiungendo il numero di anni specificati a questa data (o togliendoli, se `anni` è negativo).

Classe Veicolo

Le istanze di questa classe rappresentano dei veicoli. Ogni veicolo ha associato un certo numero di dati fissi (una targa, un modello, un numero di cavalli, la data di immatricolazione), e una serie di altri dati che cambiano nel tempo (il numero di chilometri percorsi in tutto, il numero di chilometri dall'ultimo tagliando, la data dell'ultimo tagliando e la data dell'ultima revisione).

Scrivete i seguenti costruttori e metodi:

- `Veicolo(String targa, String modello, int cavalli, Data immatricolazione)` : costruisce una macchina con i dati indicati; la macchina è nuova, ha fatto 0 chilometri, e la data di immatricolazione è considerata idealmente la data di ultimo tagliando e di ultima revisione.
- `void faiViaggio(int km)` : fa un viaggio del numero specificato di chilometri.
- `boolean necessitaTagliando(Data oggi)` : dice se la macchina necessita un tagliando; il tagliando è necessario se sono passati due anni dall'ultimo tagliando oppure se sono stati fatti almeno 15000 km dall'ultimo tagliando.
- `Data prossimaRevisione()` : restituisce la data della prossima revisione (la revisione si fa ogni quattro anni).
- `void faiTagliando(Data oggi)` : fa il tagliando all'automobile nella data indicata.
- `boolean faiRevisione(Data oggi)` : fa la revisione in data odierna, posto che non necessiti di tagliando e che manchino al massimo 30 giorni alla data della prossima revisione; se una di queste condizioni non è soddisfatta, restituisce `false` e non fa nessuna revisione; altrimenti, fa la revisione e restituisce `true` .

Classe Infrazione

Le sue istanze rappresentano infrazioni al codice della strada.

Un'infrazione è caratterizzata da una descrizione, da una data, da un certo valore monetario che dovrà essere pagato, da un numero di punti che verranno tolti dalla patente (eventualmente zero), e da un certo numero di giorni per cui la patente sarà ritirata (eventualmente zero).

Scrivete i seguenti costruttori e metodi:

- `Infrazione(String descrizione, Data data, int euro, int punti, int giorniDiRitiro)` : costruisce un'infrazione con i dati specificati.
- `Data data()` : restituisce la data dell'infrazione.
- `int euro()` : restituisce l'importo dell'infrazione.
- `int puntiTolti()` : restituisce il numero di punti tolti dalla patente.
- `int giorniDiRitiro()` : restituisce il numero di giorni per cui la patente sarà ritirata.

Classe Guidatore

Le sue istanze rappresentano dei guidatori. Un guidatore è identificato da un insieme di dati fissi (nome, cognome, codice fiscale, data di nascita, data di conseguimento della patente) e da alcuni dati variabili (veicolo posseduto, elenco di infrazioni non ancora pagate, elenco infrazioni pagate, data di ultimo rinnovo della patente, ecc.). E' anche indicata una data di *prossima utilizzabilità*, che è il primo giorno in cui si può tornare a guidare dopo che la patente è stata ritirata.

Scrivete i seguenti costruttori e metodi:

- `Guidatore(String nome, String cognome, String cf, Data nascita, Data conseguimentoPatente)` : costruisce il guidatore specificato. All'inizio, il guidatore ha 30 punti sulla patente. La data di prossima utilizzabilità e la data di ultimo rinnovo sono entrambe poste uguali a `conseguimentoPatente` .
- `void comminaInfrazione(Infrazione inf)` : aggiunge l'infrazione specificata all'elenco delle infrazioni non pagate; inoltre
  - toglie il numero specificato di punti (quando si arriva a 0 non si tolgono ulteriori punti)
  - calcola la data (diciamo `d`) ottenuta a partire dalla data dell'infrazione aggiungendo i giorni di ritiro patente previsti; questa è la data di prossima utilizzabilità (a meno che l'attuale data di prossima utilizzabilità sia

successiva a `d`, nel qual caso non la data di prossima utilizzabilità non cambia).

- `void rinnovaPatente(Data oggi)`: rinnova la patente in data odierna.
- `boolean puoGuidare(Data oggi)`: restituisce `true` se il guidatore può guidare, ovvero se sono verificate tutte le seguenti condizioni:
  - ci sono punti sulla patente
  - l'ultimo rinnovo della patente è stato ≤ di 10 anni (3650 giorni) fa
  - la data di prossima utilizzabilità è ≤ a oggi.
- `boolean paga(Infrazione inf)`: controlla che l'infrazione `inf` compaia nell'elenco delle infrazioni non pagate; se sì, lo sposta nell'elenco di quelle pagate e restituisce `true`; altrimenti restituisce `false`.
- `void associa(Veicolo v)`: associa al guidatore il veicolo indicato. In ogni momento al guidatore può essere associato un solo veicolo.
- `Veicolo veicolo()`: restituisce il veicolo associato al guidatore.
- `int daPagare()`: calcola la somma complessiva delle infrazioni non ancora pagate.
- `void aggiungiPunti(int p)`: aggiunge `p` punti (senza mai superare 30).