Automobili

Progetto di Programmazione II --- Appello di Febbraio 2019

Introduzione

In questo esame vi chiederemo di scrivere un insieme di classi per che riguardano automobili, patenti ecc. Il testo del progetto è da considerarsi una traccia, decidete voi quali altri classi o metodi aggiungere, o come ristrutturare il progetto se lo ritenete necessario.

Dovrete consegnare un file zip contenente:

- i file sorgente
- un PDF contenente una descrizione del progetto e che evidenzi specialmente le cose (classi o metodi) che avete deciso di aggiungere rispetto a questa traccia
- se decidete di realizzare un main, un manuale utente che ne descriva il funzionamento.

Classe Data

È una classe le cui istanze rappresentano date (gg/mm/aaaa). Per realizzare la classe, basatevi su delle classi della libreria standard di Java, descrivendo nella documentazione javadoc (e/o nel PDF che descrive il progetto) quali classi della libreria standard avete usato e come. 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).
- `static Data oggi()` : restituisce la data odierna.

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, il numero di passeggeri per cui è omologato ecc.), 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).

Fra gli altri, deve avere i seguenti metodi:

- `void faiViaggio(int km)` : fa un viaggio del numero specificato di chilometri.
- `boolean necessitaTagliando()` : dice se la macchina necessita di un tagliando (in data odierna); 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()` : fa il tagliando all'automobile in data odierna.
- `boolean faiRevisione()` : 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 veicolo (cui l'infrazione è comminata), 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). Per semplicità, potete assumere che quando l'infrazione è comminata a un certo veicolo le sue conseguenze ricadano sul possessore del veicolo.

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.). La classe deve anche tenere conto di eventuali infrazioni pendenti sul guidatore (che potrebbero impedire al guidatore di usare la patente).

Fra gli altri, deve avere i seguenti metodi:

- `void rinnovaPatente()` : rinnova la patente in data odierna.
- `boolean puoGuidare()` : restituisce `true` se il guidatore può guidare (in data odierna), 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
    - in questo momento la patente non è ritirata.

- `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.
- `List<Veicolo> veicoli()` : restituisce i veicoli associati al guidatore.
- `int daPagare()` : calcola la somma complessiva delle infrazioni non ancora pagate.
- `void aggiungiPunti(int p)` : aggiunge `p` punti (senza mai superare 30).

Classe MotorizzazioneCivile

Le istanze di questa classe rappresentano gli uffici di motorizzazione civile regionale. Questi uffici si occupano di immatricolare i veicoli, rilasciare e rinnovare le patenti, rilevare e comminare le infrazioni.

Fra gli altri, deve avere i seguenti metodi:

- `void aggiungiGuidatore(Guidatore g)` : aggiunge un nuovo guidatore, consegnandogli in data odierna una patente.
- `void aggiungiVeicolo(Veicolo v)` : aggiunge un nuovo veicolo, immatricolato in data odierna.
- `void aggiungiInfrazione(Infrazione inf)` : rileva e commina un'infrazione: in particolare toglie il numero specificato di punti al guidatore e calcola la data fino alla quale il guidatore non potrà guidare, nel caso che gli venga ritirata la patente.