Dipartimento di Informatica Sistemistica e Comunicazione
Corso di Laurea in Informatica

Programmazione 2

19 Febbraio 2020 – Esame completo

Testo parte di pratica

Si consideri un `Varco` stradale che convalida i veicoli che entrano in una zona cittadina a traffico controllato applicando delle regole per l'accesso. Il `Varco` tiene traccia dei veicoli che non rispettano una data `RegolaDiAccesso` (che può essere specializzata) e che saranno quindi multati conseguentemente. Una possibile `RegolaDiAccesso` ammette solo i veicoli che hanno effettuato un pagamento. Un'altra possibile `RegolaDiAccesso` vieta l'accesso ai veicoli con motore Diesel.

Si realizzino le classi che modellano tale sistema seguendo la specifica indicata di seguito. Si provino le classi realizzate con JUnit utilizzando i test forniti nella classe EsameTest.

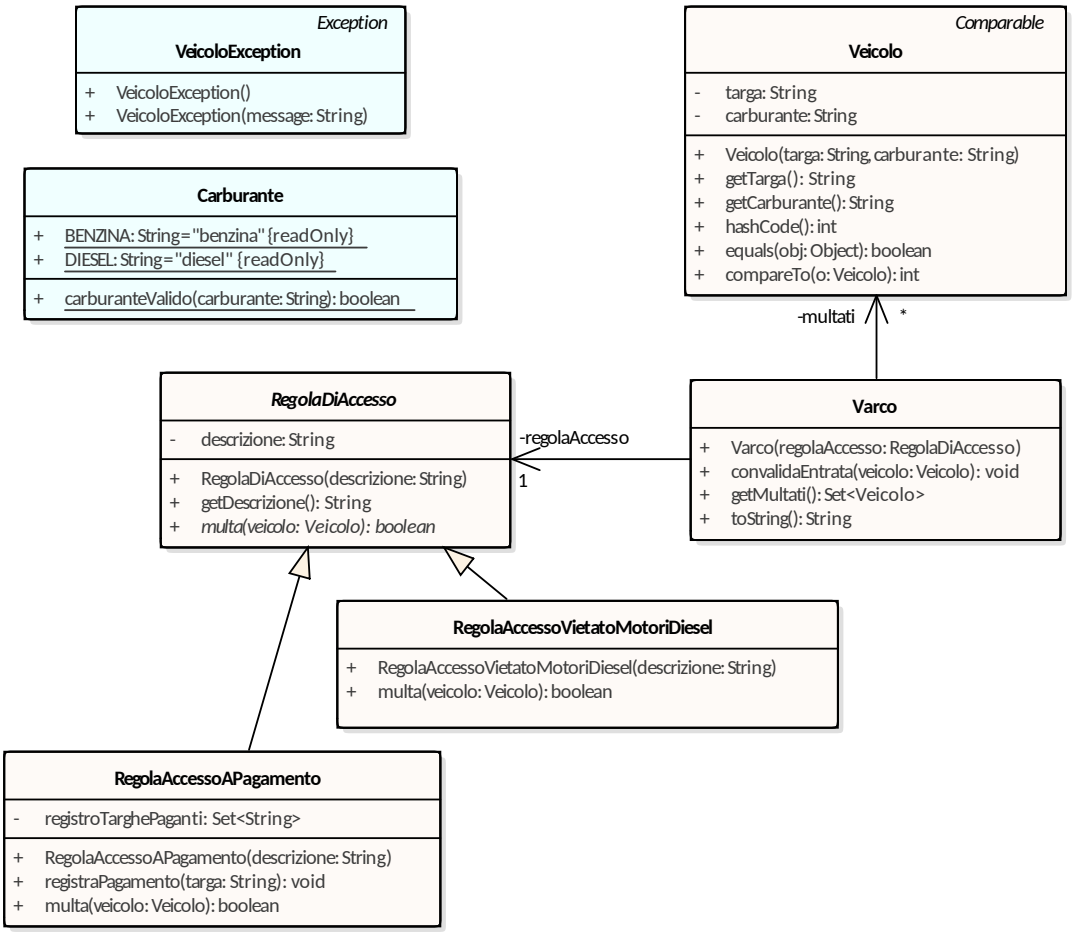
Nota: Gli elaborati che non superino almeno 4 test fra quelli dati saranno considerati insufficienti.

(**Suggerimento: si eviti di eseguire i test solo alla fine del lavoro, quando ormai sarebbe tardi per apportare correzioni; Piuttosto si eseguano i test man mano che si procede con l'implementazione, per verificare incrementalmente il lavoro via via fatto.)**

La classe `Carburante` e la classe `VeicoloException` sono fornite e vanno usate senza modifiche.

Le altre classi devono essere implementate in modo coerente al diagramma e alla specifica che seguono. **I metodi sono tutti specificati dal diagramma**: non occorre aggiungere altro.

La classe `TestEsame` non è indicata nel diagramma, ma viene anch'essa fornita.



Segue una breve descrizione delle classi.

`VeicoloException`:

- E' un tipo di eccezione che può essere sollevata dal programma

`Carburante`:

- E' una classe di utilità che implementa il metodo `carburanteValido` che ritorna `true` se la stringa passata come parametro corrisponde al nome di un carburante valido

`Veicolo`:

- E' una classe che rappresenta un generico veicolo. E' caratterizzato dagli attributi `targa` e `carburante` che sono inizializzati dal costruttore e possono essere acceduti tramite metodi getter.
- Il costruttore della classe ritorna l'eccezione `VeicoloException` se la targa è `null` o la stringa vuota, oppure se carburante è diverso dai tipi di carburanti accettati (un carburante è accettato se il metodo `carburanteValido` della classe `Carburante` ritorna `true` quando eseguito con il carburante come parametro).
- Implementa il metodo `equals` che ritorna `true` nel caso il veicolo sia confrontato con un altro veicolo con la stessa targa. Il metodo `hashCode()` utilizza il solo attributo `targa` per il calcolo dell'hash.
- Il veicolo implementa l'interfaccia `Comparable<Veicolo>` che permette di confrontare i veicoli implementando il metodo `compareTo(Veicolo)`. I veicoli sono ordinati rispetto al valore della `targa`.

`RegolaDiAccesso`:

- E' una classe astratta che rappresenta una generica regola di accesso. E' caratterizzata dall'attributo `descrizione` che viene inizializzato dal costruttore e può essere letto tramite un metodo getter.
- Definisce il metodo astratto `multa(Veicolo)` che ritorna `true` nel caso la regola di accesso produca una multa per il veicolo passato come parametro

`RegolaAccessoVietatoMotoriDiesel`:

- E' una classe che estende la classe `RegolaDiAccesso` implementando una regola che multa tutti i veicoli che hanno il carburante di tipo diesel.
- L'implementazione del metodo `multa(Veicolo)` ritorna `true` se il veicolo passato come parametro è un veicolo diesel, altrimenti ritorna `false`.

`RegolaAccessoAPagamento`:

- E' una classe che estende la classe `RegolaDiAccesso` implementando una regola che permette di registrare le targhe delle auto che hanno pagato l'accesso e multare di conseguenza i veicoli morosi.
- La classe è caratterizzata dall'attributo `registroTarghePaganti` che è un `HashSet<String>` che memorizza al suo interno l'insieme di tutte le targhe dei veicoli che hanno pagato.
- Il metodo `registraPagamento(String targa)` aggiunge la targa al `registroTarghePaganti`. Nel caso in cui la targa aggiunta sia `null` o la string vuota, il metodo non fa nulla. Nel caso la targa aggiunta esista già nella collezione, il metodo ritorna una eccezione di tipo `VeicoloException`.
- Il metodo `multa(Veicolo)` ritorna `true` se la targa del veicolo è stata precedentemente registrata tra quelle paganti, altrimenti ritorna `false`.

`Varco`:

- E' una classe che definisce un varco controllato da una regola di accesso. La classe è caratterizzata da due attributi. L'attributo `regolaAccesso` è una regola di accesso. L'attributo `multati` è un `TreeSet<Veicolo>` di veicoli che sono stati multati dalla regola di accesso. Il costruttore della classe inizializza la regola e crea la collezione vuota. Il varco può essere creato privo di una regola (parametro `null`). In questo caso nessun veicolo viene multato.
- Il metodo `convalidaEntrata(Veicolo veicolo)` verifica se il veicolo passato come parametro deve essere multato dalla regola memorizzata in `regolaAccesso`. Se la regola è `null`, il veicolo non viene mai multato. I veicoli multati sono aggiunti nella collezione `multati`.
- Il metodo `getMultati()` ritorna un riferimento alla collezione `multati`.
- Il metodo `toString` ritorna una stringa contenente la descrizione della regola di accesso e le targhe di tutti i veicoli multati.