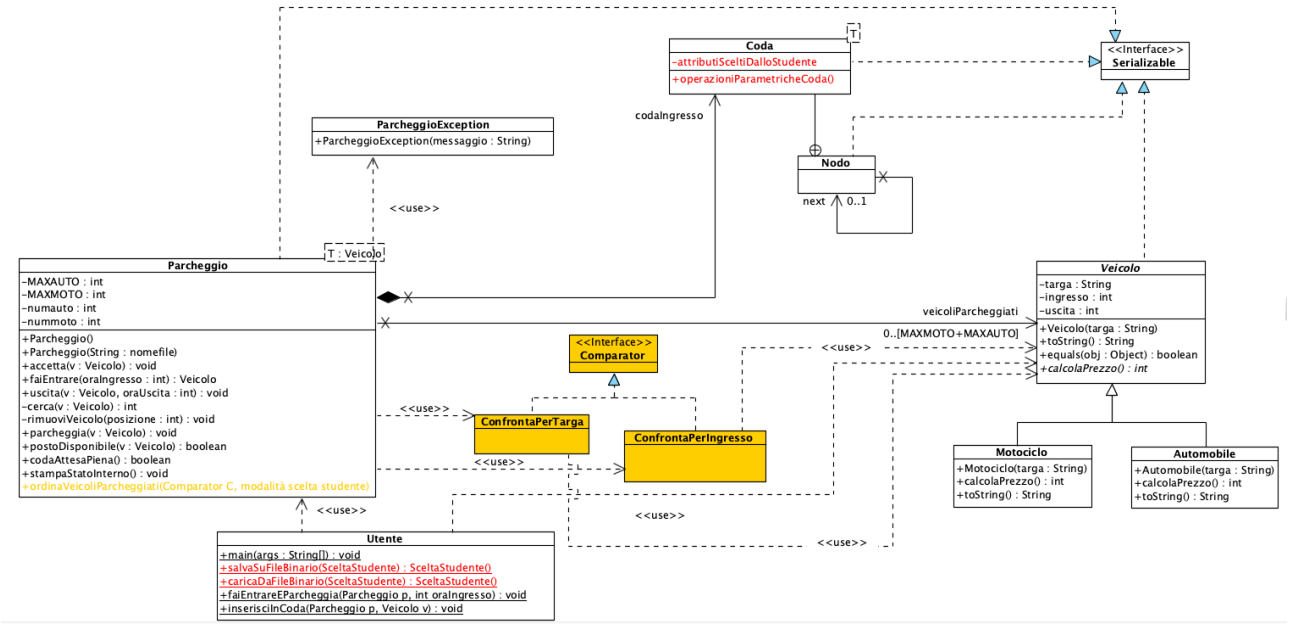
Esercitazione pre-Esame del 11 Giugno 2025 - UPDATE.

Proff. Amalfitano - Vittorini

Si vuole implementare una applicazione per la gestione automatica di un parcheggio. Il software è modellato dal diagramma UML riportato di seguito. Tutto ciò che è in rosso deve essere specificato nel dettaglio dagli studenti.



Requisiti del software.

1. Il parcheggio può contenere un numero massimo di moto e di auto.
2. Il parcheggio può gestire una coda esterna di veicoli in attesa di entrare. La coda esterna è realizzata mediante una coda linkata generica. Il parcheggio limita la coda per contenere al più 5 veicoli, controllando il numero di elementi in essa presenti.
3. Il parcheggio ha un vettore di veicoli al suo interno che modella gli spazi di parcheggio.
4. Il metodo `accetta`, serve ad **aggiungere un veicolo alla coda** di attesa. Questo metodo deve essere richiamato dalla classe utente e all'atto della chiamata il metodo solleva un'apposita eccezione se il parcheggio ha esaurito i posti interni per il tipo di veicolo che ha provato ad aggiungersi in coda oppure se la coda ha già raggiunto il numero massimo di elementi.
5. Il metodo `faiEntrare` **preleva il primo veicolo dalla coda** di attesa e **setta l'orario** di ingresso del veicolo prelevato. Solleva un'eccezione se non ci sono posti disponibili per il tipo di veicolo.
6. Il metodo `parcheggia` aggiunge un veicolo nella prima posizione disponibile dell'elenco delle vetture del parcheggio ed aggiorna il numero di posti disponibili.
7. Il metodo `postoDisponibile` verifica la disponibilità di posti nel parcheggio per un tipo di veicolo.
8. Il metodo `uscita` rimuove un veicolo dall'insieme dei veicoli del parcheggio. Prima di rimuovere il veicolo il parcheggio **setta l'orario di uscita** e **calcola il prezzo** da pagare del veicolo. Il prezzo da pagare dipende dal tipo di veicolo ed è calcolato dal metodo `calcolaPrezzo`. Il prezzo da pagare è calcolato come `(uscita - ingresso+1)*2` per i motocicli e `(uscita - ingresso+1)*4` per le automobili.
9. I metodi `cerca` e `rimuoviVeicolo` sono metodi privati di supporto dell'implementazione del metodo `uscita`. Lo studente può decidere se usufruire o meno di tali metodi.
10. Nel diagramma UML mancano i metodi di get e set degli attributi. Lo studente implementi i metodi che ritiene necessari.

11. Il parcheggio è serializzabile. Questo richiede che tutte le classi coinvolte siano serializzabili. La classe utente effettua la serializzazione e la deserializzazione. Utilizza il metodo `stampaStatoInterno` della classe parcheggio per verificare l'esito delle operazioni.
12. **[NEW]: Il parcheggio può ordinare i veicoli parcheggiati in ordine crescente o decrescente in base alla targa o all'orario di ingresso. A tale scopo il parcheggio utilizza un apposito comparator.**
13. **ParcheggioException deve essere sempre gestita.**
14. La classe utente include alcuni metodi statici di supporto per facilitare il test delle funzionalità principali del sistema:
    a. inserisciInCoda(Parcheggio p, Veicolo v): prova a inserire un veicolo nella coda di attesa, gestendo l'eventuale eccezione in caso di coda piena.
    b. faiEntrareEParcheggia(Parcheggio p, int oraIngresso): esegue l'ingresso del primo veicolo in coda e lo parcheggia, gestendo l'eventuale eccezione in caso di mancanza di posti.
    c. Un metodo per salvare lo stato del parcheggio su file binario nella cartella Files mediante serializzazione.
    d. Un metodo per caricare un oggetto Parcheggio da un file binario precedentemente salvato.

    Tali metodi possono essere richiamati dal main per verificare il corretto comportamento del sistema in condizioni normali e in presenza di errori gestiti tramite eccezioni.