Dipartimento di Informatica Sistemistica e Comunicazione
Corso di Laurea in Informatica

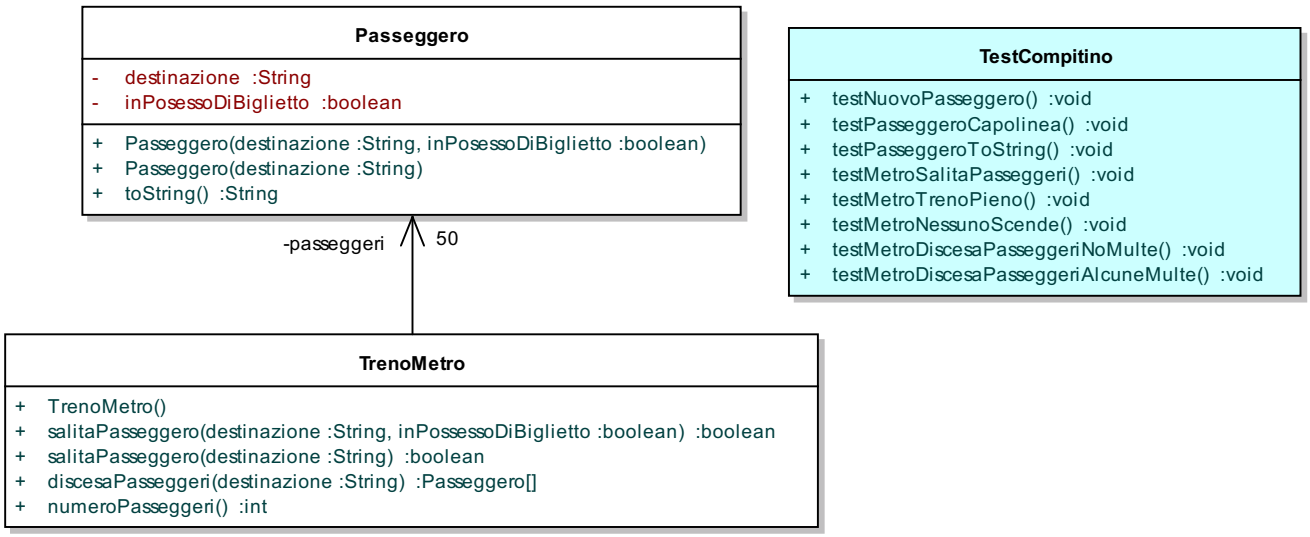
Programmazione 2

22 Aprile 2015 – Primo Compitino

Testo parte di pratica

Un treno della metropolitana è in grado di ospitare fino a 50 passeggeri. Ogni passeggero che sale sul treno ha una stazione di destinazione e può essere o meno in possesso di un biglietto valido. Ogni volta che il treno si ferma a una stazione tutti i passeggeri destinati a quella stazione scendono. I passeggeri senza biglietto che scendono in una stazione rischiano una multa.

Implementare le classi come rappresentate dal seguente diagramma UML. Il diagramma non include gli eventuali metodi di incapsulamento che **DEVONO** essere individuati correttamente e codificati. Infine, la classe `TestCompitino` contiene un insieme di test che devono essere fatti girare di volta in volta in modo da verificare la corretta realizzazione del software. **Come requisito minimo per ottenere una valutazione positiva, lo studente deve garantire che la sua implementazione non presenti errori di compilazione e superi almeno 3 casi di test fra quelli dati.**



Classe `Passeggero`:

- ✓ rappresenta un passeggero della metropolitana. È caratterizzato da una fermata di destinazione (`destinazione`) e da un attributo che indica il possesso di un biglietto valido (`inPossessoDiBiglietto`)
- ✓ `destinazione` è mutabile, `inPossessoDiBiglietto` è immutabile, ma accessibile dall'esterno
- ✓ definisce un costruttore che inizializza gli attributi. Se il valore del parametro `destinazione` è `null` il corrispondente attributo viene inizializzato con la stringa `"CAPOLINEA"`
- ✓ effettua l'overloading del costruttore definendone uno ulteriore che inizializza `destinazione` e pone `inPossessoDiBiglietto` a TRUE.
- ✓ il metodo `toString` restituisce una stringa con l'informazione relativa al passeggero (destinazione e possesso del biglietto)

Classe `TrenoMetro`:

- ✓ rappresenta un vagone della metro della capienza di 50 passeggeri
- ✓ definisce un costruttore che inizializza opportunamente il metro in accordo con la specifica UML
- ✓ l'attributo `passeggeri` non deve essere accessibile dall'esterno
- ✓ il metodo `salitaPasseggero(destinazione, inPossessoDiBiglietto)` aggiunge un `passeggero` con destinazione `destinazione` e possesso del biglietto `inPossessoDiBiglietto` al treno.

L'inserimento è possibile solamente se il treno non è pieno. Il metodo ritorna `true` nel caso l'inserimento avvenga con successo, altrimenti ritorna `false`.

- ✓ il metodo `salitaPasseggero(destinazione)` aggiunge un nuovo `passeggero` dotato di biglietto valido con destinazione `destinazione`. L'inserimento è possibile solamente se il treno non è pieno. Il metodo ritorna `true` nel caso l'inserimento avvenga con successo, altrimenti ritorna `false`.
- ✓ il metodo `discesaPasseggeri(destinazione)` rimuove dall'elenco dei passeggeri tutti i passeggeri con destinazione `destinazione`. Inoltre il metodo ritorna un array che contiene l'insieme di passeggeri con biglietto non valido che sono stati rimossi. L'array restituito deve avere esattamente la stessa dimensione dell'elenco di passeggeri restituito (se l'elenco passeggeri è vuoto il metodo deve ritornare un array di lunghezza pari a 0).
- ✓ Il metodo `numeroPasseggeri()` restituisce il numero di passeggeri che sono presenti sul treno.