Dipartimento di Informatica Sistemistica e Comunicazione
Corso di Laurea in Informatica

Programmazione 2

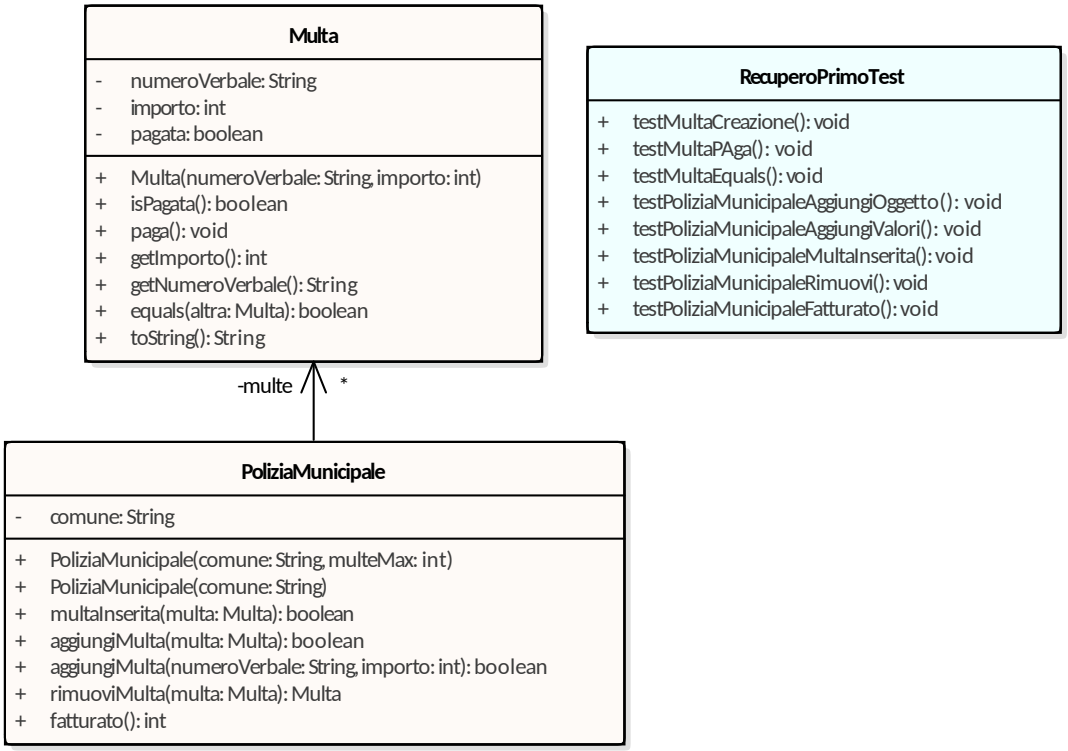
5 Luglio 2018 – Recupero Primo Compitino

Testo parte di pratica

La polizia municipale di un comune sanziona le irregolarità dei cittadini alla guida emettendo multe. Una multa ha un numero di verbale, un importo e lo stato di pagata o meno. Una multa può essere rimossa grazie ad un ricorso andato a buon fine.

Implementare le classi come rappresentate dal seguente diagramma UML. **I diagrammi delle classi contengono tutti e i soli metodi richiesti, compresi quelli di incapsulamento.**

Infine, la classe `RecuperoPrimoTest` (già fornita) 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 `Multa`:

- ✓ rappresenta una multa. È caratterizzata da un numero di verbale (`numeroVerbale`), un importo (`importo`) e dal fatto che sia stata pagata o meno (`pagata`)
- ✓ definisce un costruttore che inizializza gli attributi. Assumete che i valori passati in ingresso siano validi (stringa diversa da `null` e diversa dalla stringa vuota e importo maggiore di 0). Lo stato di pagata viene impostato nel costruttore sempre a `false`
- ✓ gli attributi hanno definiti i propri metodi di incapsulamento (`isPagata()`, `getImporto()` e `getNumeroVerbale()`). Per quanto riguarda il metodo `paga()`, questo imposta a `true` l'attributo `pagata`
- ✓ due multe sono uguali secondo il metodo `equals` se hanno lo stesso numero di verbale a prescindere dalle lettere maiuscole o minuscole
- ✓ il metodo `toString` restituisce una stringa con l'informazione relativa alla multa (numero di verbale, importo e lo stato di pagata o meno)

Classe `PoliziaMunicipale`:

- ✓ rappresenta un comando di polizia municipale di un comune. La polizia municipale fa capo ad un comune (attributo `comune`) e mantiene l'insieme delle multe emesse (associazione `multe`).
- ✓ definisce un costruttore che inizializza gli attributi. Assumete che i valori passati in ingresso siano validi (stringa diversa da `null` e diversa dalla stringa vuota e `multeMax` maggiore di 0)
- ✓ definisce un ulteriore costruttore che inizializza il comune con il valore passato in ingresso ed imposta il numero di multe massimo a 1000. Anche in questo caso si assume che il valore della stringa sia corretto
- ✓ il metodo `multaInserita(multa)` restituisce `true` se `multa` è già presente (cioè se la polizia municipale ha già una multa uguale a quella passata in ingresso). `False` in tutti gli altri casi (incluso parametro null)
- ✓ il metodo `aggiungiMulta(multa)` aggiunge `multa` nella prima posizione libera delle multe se diversa da `null` e se non già presente nelle multe. Restituisce `true` in caso di inserimento riuscito, `false` in caso contrario
- ✓ il metodo `aggiungiMulta(numeroVerbale,importo)` effettua l'overloading dell'omonimo metodo accettando però in ingresso il numero di verbale e l'importo della nuova multa che si vuole aggiungere. La multa non viene aggiunta se `numeroVerbale` è `null` o la stringa vuota, oppure se importo è minore o uguale a 0. Il metodo restituisce `true` in caso di inserimento riuscito, `false` in caso contrario
- ✓ il metodo `rimuoviMulta(multa)` rimuove dall'insieme di multe la multa uguale a `multa`. Tale operazione va a buon fine se `multa` è diverso da `null` e se la multa cercata effettivamente era presente. Se presente, il metodo restituisce il riferimento all'oggetto rimosso dall'insieme di multe. In caso contrario, il metodo restituisce `null`
- ✓ il metodo `fatturato()` calcola per tutte le multe in stato di pagato l'importo complessivo incassato dal comune