Packages, Pre-condizioni

25 Ottobre 2016

Esercizio

- Si supponga di voler progettare e implementare parte del sistema informativo di una azienda di trasporti su rotaia. Per ogni treno occorrerà tenere traccia delle stazioni di fermata, della stazione di partenza e di quella di arrivo, oltre che dei relativi orari. Occorre poi che ad ogni treno sia associato il numero dei posti a sedere disponibili e il numero totale di chilometri percorsi. Nei treni espressi, infine, è previsto anche un servizio ristorante, e anche per questo servizio è necessario tenere traccia del numero di posti disponibili.
- Un utente di questo sistema informativo potrebbe essere interessato a determinare il numero di fermate effettuate da ciascun treno. Inoltre, chi utilizza tale sistema informativo potrebbe essere interessato a determinare il massimo ricavo realizzabile nell'erogazione di questo servizio. Tale ricavo dipende chiaramente da un parametro, ovvero dal prezzo che ogni passeggero dovrà pagare per percorrere un chilometro. Nei treni espressi occorrerà tenere conto anche del ricavo che si presume di ottenere in ogni chilometro da ognuno dei posti disponibili nel vagone ristorante (anch'esso fornito come parametro).

Esercizio

- Implementare le classi:
- Purse:
  - addCoin(Coin)
  - getTotal()
- Coin:
  - getValue()
  - getName()
- Mettere le classi in un pacchetto “it.unisa.prog2.money” e testare le classi con una classe di collaudo MoneyTest del pacchetto di default
- Dare delle precondizioni ragionevoli per i metodi e testarle con delle asserzioni

Esercizio

- Implementare e testare la classe ContoCorrente (pacchetto “money”)
- deposita(double importo)
  - pre-condizione importo >= 0
- preleva(double importo)
  - pre-condizione importo <= saldo
- restituisciSaldo()
- restituisciNumeroConto()

- Ogni conto corrente ha un numero progressivo che lo identifica, restituito da restituisciNumeroConto()