Laboratorio di Programmazione
Edizione 1 - Turni A, B, C

ESAME del 2 Febbraio 2015

Avvertenze

- Nello svolgimento dell'elaborato è possibile usare qualunque classe delle librerie standard di Java
- Non è invece ammesso l'uso delle classi del package `prog` allegato al libro di testo del Prof. Pighizzini e impiegato nella prima parte del corso
- Si consiglia CALDAMENTE l'utilizzo dello script "checker.sh" per compilare ed effettuare una prima valutazione del proprio elaborato

Tema d'esame

Lo scopo è realizzare un modello di cassa da supermercato, i prodotti vengono "passati" e la cassa tiene conto del totale da pagare e dei prodotti fin lì inseriti. I prodotti si possono "stornare" ed è poi possibile pagare con contanti (carta e monete).
Le classi da realizzare sono le seguenti (dettagli nelle sezioni successive):

- `Importo`: rappresenta una cifra di denaro, in euro e centesimi, ma non è contante spendibile
- `Denaro`: classe astratta, rappresenta del contante in centesimi
- `Banconota`: sottoclasse di `Denaro`, rappresenta moneta cartacea
- `Moneta`: sottoclasse di `Denaro`, rappresenta moneta metallica
- `Prodotto`: classe che descrive un prodotto, ha un prezzo e un nome
- `Cassa`: classe che descrive la cassa, tiene traccia dei prodotti di cui sopra.

Specifica delle classi

Le classi (**pubbliche**!) dovranno esporre almeno i metodi **pubblici** specificati sotto, più eventuali altri metodi; in alcuni casi le definizioni dei metodi sono incomplete (vanno aggiunti i tipi mancanti).
Gli attributi (campi) delle classi devono essere *privati*; per leggere e modificarne i valori, creare opportunamente, e solo dove necessario, i metodi di accesso (`set` e `get`).
Si suggerisce di utilizzare, dove possibile, le classi parametriche opportunamente istanziate (es. *ArrayList<E>* invece di *ArrayList*).
Ogni classe (eccetto quelle con metodo `main`) deve avere il metodo `toString` che rappresenti lo stato delle istanze.

class Importo

Rappresenta una cifra di denaro in euro e centesimi, ma non è contante. Implementa Comparable<Importo>.
Deve disporre dei seguenti costruttori e metodi pubblici oltre ad eventuali altri metodi che riterrete opportuni:

- `Importo(int valoreInCentesimi)`
  Costruttore che accetta il valore in centesimi.
- `int inCentesimi()`
  Restituisce il valore in centesimi.
- `String toString()`
  Fornisce una descrizione testuale dell'istanza.

class Denaro

Classe astratta che rappresenta del contante in centesimi. Deve disporre dei seguenti metodi pubblici oltre ad eventuali altri metodi che riterrete opportuni.

- `int getValoreInCentesimi()`
  Restituisce il valore in centesimi.
- `Importo getImporto()`
  Restituisce l'importo corrispondente al valore in centesimi.
- `void setValoreInCentesimi(int val)`
  Modifica il valore in centesimi. Servirà alle sottoclassi per creare istanze solo per valori validi (vedi sotto).
- `String toString()`
  Fornisce una descrizione testuale dell'istanza.

class Banconota

Sottoclasse di Denaro che rappresenta le monete cartacee. Ha un solo costruttore che accetta un int che rappresenta un valore in euro; tale valore deve essere valido (5, 10, 20, 50, 100, 200, 500) e i valori non validi devono generare un'eccezione. Ricordare il toString().

class Moneta

Sottoclasse di Denaro che rappresenta le monete metalliche. Ha un solo costruttore che accetta un int che rappresenta un valore in centesimi; tale valore deve essere valido (1, 2, 5, 10, 20, 50, 100, 200), i valori non validi devono generare un'eccezione. Ricordare il toString().

class Prodotto

Classe che rappresenta un prodotto da "passare" sulla cassa, è caratterizzato da un nome e da un Importo che rappresenta il suo prezzo. Deve implementare l'interfaccia Comparable<Prodotto>, in modo che i prodotti possano essere ordinati in base al prezzo, e disporre dei seguenti costruttori e metodi pubblici oltre ad eventuali altri metodi che riterrete opportuni:

- `Prodotto(String nome, int prezzoInCentesimi)`
  Costruisce un prodotto con nome e prezzo in centesimi specificati. Solleva un'eccezione se il nome è `null` oppure se il prezzo è negativo.
- `Importo getPrezzo()`
  Restituisce il prezzo del prodotto.
- `int prezzoInCentesimi()`
  Restituisce il prezzo del prodotto, in centesimi.
- `String toString()`
  Fornisce una descrizione testuale del prodotto.

class Cassa

Rappresenta la cassa del supermercato, permette la scansione del prodotto ("passaggio") e la rimozione ("storno"). Fornisce informazioni sul contenuto. Accetta pagamento parziale e sa dire se il pagamento copre la spesa. Deve disporre dei seguenti costruttori e metodi pubblici oltre ad eventuali altri metodi che riterrete opportuni:

- `Cassa()`
  Costruttore senza argomenti, costruisce una cassa vuota (memo: impostare i campi).
- `void passa(Prodotto p)`
  Scansione prodotto, viene aggiunto alla lista e concorre alla formazione del totale da pagare.
- `void paga(Denaro d)`
  Accetta pagamenti parziali, verranno passate istanze di Banconota e Moneta. Questo metodo verrà invocato più volte fino al raggiungimento/superamento del totale da pagare.
- `int getPagamentoParzialeInCentesimi()`
  Fornisce lo stato del pagamento parziale, in centesimi.
- `boolean pagato()`
  Restituisce true se il denaro immesso finora copre la spesa totale.
- `Importo azzeramento()`
  Azzera lo stato della cassa e segnala l'importo eventualmente pagato finora.
- `Prodotto storna(int i)`
  Storna l'i-esimo prodotto dalla cassa, lo rimuove.
- `Prodotto storna(Prodotto p)`
  Storna il prodotto p dalla cassa, se c'è, lo rimuove restituendolo al chiamante.
- `Importo calcolaResto()`

Calcola il resto, se c'è, zero altrimenti.

- `int quanti()`
  Restituisce il numero di prodotto "passati".
- `Importo totale()`
  Fornisce l'importo del totale da pagare.
- `int totaleInCentesimi()`
  Fornisce il totale da pagare, in centesimi.
- `Prodotto piuCostoso()`
  Fornisce il prodotto più costoso passato finora.
- `void salvatempo(String nomeFile)`
  Apre file e legge i prodotti da passare, formato:

```
nome,prezzoincentesimi
nome,prezzoincentesimi
nome,prezzoincentesimi
nome,prezzoincentesimi
nome,prezzoincentesimi
nome,prezzoincentesimi
...,...
```

  (1 prodotto per riga)
  ATTENZIONE, deve lanciare un'eccezione se il file non esiste, non è leggibile, non è un file regolare, etc.
- `String toString()`
  Fornisce una descrizione testuale del contenuto della cassa.

Consegna

Si ricorda che le classi devono essere tutte *public* e che vanno consegnati tutti i file *.java* prodotti.
NON vanno consegnati i *.class*.
NON vanno consegnati i file relativi al meccanismo di autovalutazione (*Test_*.java, AbstractTest.java, *.sh*).
Per la consegna, eseguite l'upload dei SINGOLI file sorgente (NON un file archivio!) dalla pagina web: http://upload.di.unimi.it

---

ATTENZIONE!!! NON VERRANNO VALUTATI GLI ELABORATI CON ERRORI DI COMPILAZIONE. UN SINGOLO ERRORE DI COMPILAZIONE INVALIDA **TUTTO** L'ELABORATO.

Per ritirarsi fare l'upload di un file vuoto di nome `ritirato.txt`.
