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

ESAME del 13 giugno 2016

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.
- Ricordarsi, quando si programma: *Repetita NON iuvant* o DRY (*Don't Repeat Yourself*).

ESERCIZIO FILTRO

INIZIARE PRIMA CON QUESTO, se non si è in grado di portare a termine questo esercizio NON PROSEGUIRE.
La correttezza di questo esercizio è condizione NECESSARIA affinché i docenti correggano il resto del compito.
Realizzare una classe `Stat`, dotata del solo `main`, che letta da linea di comando una sequenza di interi con valori da 1 a 10, calcoli e visualizzi la media aritmetica, la moda, la mediana dei valori letti.
Si ricorda che la moda è il valore che compare piú frequentemente (nel caso vi siano piú valori con frequenza massima, se ne scelga uno arbitrariamente), mentre la mediana è il valore "centrale" (assumendo che i valori siano disposti in ordine crescente): piú precisamente, se il numero di valori è dispari, la mediana corrisponde al valore che occupa la posizione $(n+1)/2$, altrimenti essa corrisponde alla media aritmetica dei valori nelle posizioni $n/2$ e $n/2+1$.
Qualora uno dei valori inseriti non sia un intero nel range [1,10] l'applicazione terminerà visualizzando un messaggio d'errore.
Si ricorda che per ordinare un array è possibile utilizzare uno dei metodi `sort` definiti nella classe Arrays.
Ecco un possibile **esempio** di esecuzione:

```
$ java Stat 1 3 2 9 10 2 5
media: 4.5714285
moda: 2
mediana: 3
```

Tema d'esame

Lo scopo dell'esercizio è realizzare un modello di carta prepagata per acquisti sia in euro che in altra valuta. Le diverse classi java permettono di descrivere entità e concetti quali carta, importo, operazione, ecc.
Viene fornito un main (nella classe Main) per effettuare qualche test di funzionamento. Lo studente può modificarlo a proprio piacimento per realizzare test ulteriori. La classe Main NON va consegnata.
Le **classi** da realizzare sono le seguenti:

1. `CartaPrepagata`: rappresenta una carta prepagata.
2. `Importo`: rappresenta un generico importo, classe **astratta**.
3. `ImportoInEuro`: sottoclasse concreta di Importo, rappresenta importi in euro.
4. `ImportoInValuta`: sottoclasse concreta di Importo, rappresenta importi in valuta diversa dall'euro.
5. `Operazione`: rappresenta una generica operazione con la carta, classe **astratta**.
6. `Ricarica`: sottoclasse concreta di Operazione, rappresenta ricariche.
7. `Addebito`: sottoclasse concreta di Operazione, rappresenta addebiti.

Specifica delle classi

Le classi (**pubbliche**!) dovranno esporre almeno i metodi e costruttori **pubblici** indicati, piú eventuali altri metodi e costruttori se ritenuti opportuni. Gli attributi (campi) delle classi devono essere dichiarati **privati**. Per leggere e modificarne

i valori, creare opportunamente, e solo dove indicato, i metodi di accesso (`set` e `get`). Se si usano classi che utilizzano tipi generici, si suggerisce di utilizzarne le versioni opportunamente istanziate (es. `ArrayList<String>` invece di `ArrayList`). Ogni classe deve avere il metodo `toString` che descriva lo stato delle istanze.

1 CartaPrepagata

Classe che rappresenta una carta prepagata, caratterizzata dal nome del titolare, dal plafond (l'importo massimo caricabile sulla carta) e dal pin di 5 cifre. Deve tenere traccia dei movimenti (addebiti e ricariche) fatti. Permette al piú tre operazioni con pin errato: ogni volta visualizzerà un messaggio del tipo "Attenzione, pin errato. Hai ancora n tentativi". Dopo il terzo tenativo con pin errato la carta sarà bloccata e non sarà piú possibile fare operazioni. Deve implementare almeno i seguenti metodi:

- `public CartaPrepagata(String nome, int plafond, String pin)` costruttore che accetta nome del titolare, plafond e pin di 5 cifre. Solleva un'eccezione per pin non valido o nome nullo.
- `public boolean addebita(Importo i, String pin)` addebita l'importo sulla carta. Solleva un'eccezione se l'importo supera la ricarica disponibile.
- `public boolean ricarica(Importo i, String pin)` carica un importo sulla carta. Se l'importo non è in euro o se la ricarica è tale da far superare il plafond, solleva un'eccezione.
- `public ImportoInEuro getResiduo(String pin)` restituisce l'importo disponibile sulla carta.
- `public Importo[] getMovimentiOrdinati()` restituisce la lista delle operazioni fatte con la carta, prima le ricariche e poi gli addebiti, in ordine crescente di importo.
- `public String toString()` restituisce una rappresentazione dello stato dell'oggetto.

2 Importo

Classe **astratta**, rappresenta un generico importo costituito da due int, valuta e centesimi della valuta (es. euro e eurocent). Deve implementare l'interfaccia `Comparable` (ordine crescente di valore in euro) e almeno i seguenti metodi:

- `public Importo(int val, int cent)` costruttore che accetta come parametri la parte intera e la parte in centesimi dell'importo.
- `public int getValoreInCentesimi()` restituisce il valore in centesimi dell'importo.
- `public abstract ImportoInEuro getImportoInEuro()` restituisce l'importo in euro equivalente all'importo.
- `public ImportoInEuro somma(Importo i)` restituisce l'importo in euro equivalente alla somma dei due importi.
- `public ImportoInEuro sottrai(Importo i)` restituisce l'importo in euro equivalente alla differenza dei due importi.
- `public String toString()` restituisce una rappresentazione dello stato dell'oggetto

3 ImportoInEuro

Sottoclasse concreta di Importo, rappresenta solo importi in euro. Deve implementare almeno i seguenti metodi:

- fare opportuni *override* dei metodi della classe da cui questa deriva.

4 ImportoInValuta

Sottoclasse concreta di Importo, rappresenta solo importi in valuta diversa da euro. Ha come attributi anche il nome della valuta (String) e il tasso di cambio dalla valuta a euro (double). Deve implementare almeno i seguenti metodi:

- fare opportuni *override* dei metodi della classe da cui questa deriva.

5 Operazione

Classe **astratta**, rappresenta una generica operazione sulla carta prepagata, caratterizzata da un importo. Deve implementare l'interfaccia `Comparable` (prima gli addebiti e poi le ricariche, in ordine crescente di importo) e almeno i seguenti metodi:

- `public Operazione(Importo i))` costruttore che accetta un importo.
- `public Importo getImporto()` restituisce l'importo dell'operazione.
- `public String toString()` restituisce una rappresentazione dello stato dell'oggetto

6 Ricarica

Sottoclasse concreta di Operazione, rappresenta ricariche. Deve implementare almeno i seguenti metodi:

- fare opportuni *override* dei metodi della classe da cui questa deriva.

7 Addebito

Sottoclasse concreta di Operazione, rappresenta addebiti. Deve implementare almeno i seguenti metodi:

- fare opportuni *override* dei metodi della classe da cui questa deriva.

Raccomandazioni

Affinché l'elaborato sia valutato, è richiesto che sia le classi sviluppate che i test risultino *compilabili*. A tal fine i metodi/costruttori che non saranno sviluppati dovranno comuque avere una implementazione fittizia come la seguente:

```
public <tipo> <nomeDelMetodo> (<lista parametri>) {
   throw new UnsupportedOperationException();
}
```

Si suggerisce quindi di dotare da subito le classi di tutti i metodi richiesti, implementandoli in modo fittizio, e poi di sostituire man mano le implementazioni fittizie con implementazioni che rispettino le specifiche.

Consegna

Si ricorda che le classi devono essere tutte *public* e che vanno consegnati tutti (e soli) 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 nella sessione del prof. Trentini.
Si ricorda di fare upload spesso, in ogni caso i docenti vedranno solo l'ultima versione caricata di ogni file.

*** ATTENZIONE!!! ***

NON VERRANNO VALUTATI GLI ELABORATI CON ERRORI DI COMPILAZIONE O LE CONSEGNE CHE NON RISPETTANO LE SPECIFICHE (ad esempio consegnare un archivio zippato è sbagliato).
UN SINGOLO ERRORE DI COMPILAZIONE O DI PROCEDURA INVALIDA **TUTTO** L'ELABORATO.

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