Laboratorio di Programmazione
ESAME del 11 luglio 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.
- Si consiglia CALDAMENTE l'utilizzo dello script "checker.sh" (se non è eseguibile renderlo tale col comando `chmod`) per compilare ed effettuare una prima valutazione del proprio elaborato. Si consiglia anche di leggere il sorgente dei `Test_*.java` per capire cosa devono offrire le classi da sviluppare.
- Ricordarsi, quando si programma: *Repetita NON iuvant* o DRY (*Don't Repeat Yourself*).
- Un corpo di metodo (escluso un `main`) più lungo di una decina di righe è un buon indizio di "strada sbagliata"
- Se avete dubbi sulla interpretazione del testo chiedete!

ESERCIZIO FILTRO

===>>> INIZIARE PRIMA CON QUESTO, se non si è in grado di portare a termine questo esercizio... NON PROSEGUIRE!!! (la correzione del resto dell'elaborato è subordinata alla correttezza di questo primo esercizio)

Realizzare una classe `Stat`, dotata del solo `main`, che letta da linea di comando una serie di coppie di numeri v_i p_i che rappresentano ciascuna un voto e la sua probabilità, calcoli e visualizzi il valore atteso, secondo la formula fornita sotto.

- i valori v_i sono di tipo `int` compresi tra 18 e 30.
- i valori p_i sono di tipo `float` compresi tra 0 e 1 e tali che $\sum_i p_i = 1.0$

Ecco due esempi:

```
18 0.15 21 0.15 22 0.05 24 0.05 25 0.25 26 0.05 28 0.18 30 0.12
24 0.5 25 0.2 30 0.3
```

Si ricorda che il valore atteso (E) è definito da

$$E = \sum_i v_i \cdot p_i$$

Qualora uno dei valori v_i inseriti non sia nel range [18,30] o uno dei p_i non sia nel range [0,1] l'applicazione terminerà visualizzando un messaggio d'errore. A parte ciò, SI ASSUMA che

l'input (gli argomenti dell'applicazione) sia CORRETTO sintatticamente e semanticamente, e non vuoto. Ecco un possibile **esempio** di esecuzione:

```
$ java Stat  18 0.15 21 0.15 22 0.05 24 0.05 25 0.25 26 0.05 28 0.18 30 0.12
E : 24.340002
```

Tema d'esame

Lo scopo dell'esercizio è realizzare un insieme di operandi e operatori. Gli *operandi* sono gli argomenti, i valori da trattare, mentre le *operazioni* calcolano delle funzioni semplici su insiemi di *operandi*.
Le **classi** da realizzare sono le seguenti:

1. `Operando`: rappresenta ("wrapper") un valore double
2. `Operazione`: classe ASTRATTA che rappresenta una generica operazione da applicare ad un insieme di operandi
3. `Somma`: sottoclasse di `Operazione`, calcola la somma degli operandi che contiene
4. `Massimo`: sottoclasse di `Operazione`, calcola il massimo tra gli operandi che contiene
5. `Media`: sottoclasse di `Operazione`, calcola la media degli operandi che contiene
6. `Moltiplicazione`: sottoclasse di `Operazione`, calcola il prodotto degli operandi che contiene
7. `SommaUnici`: sottoclasse di `Operazione`, calcola la somma degli operandi che contiene, ESCLUDENDO le ripetizioni

Specifica delle classi

Le classi (**pubbliche**!) dovranno esporre almeno i metodi e costruttori **pubblici** indicati, più 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 Operando (Comparable)

Rappresenta un generico valore `double`. Implementa l'interfaccia `Comparable<Operando>` Deve implementare almeno i seguenti metodi:

- `public Operando(double valore)` costruttore che accetta il valore interno iniziale
- `public Operando()` costruttore che imposta a 0 il valore interno
- `public double getValore()` restituisce il valore corrente
- `public String toString()` ritorna una rappresentazione dello stato dell'oggetto

2 Operazione

Classe **astratta**, "contiene" un insieme (potenzialmente infinito) di operandi, le sue sottoclassi completeranno l'implementazione realizzando il calcolo effettivo. Deve implementare almeno i seguenti metodi:

- `public void addOperando(Operando o)`: aggiunge un operando all'insieme corrente, NON effettua il calcolo dell'operazione
- `public int getNumeroOperandi()`: restituisce il numero degli operandi attualmente contenuti
- `public List<Operando> getOperandi()`: restituisce una struttura dati compatibile con `List<Operando>` contenente tutti gli operandi attualmente presenti in operazione
- `public void sort()`: ordina gli operandi in base al loro valore
- `public String toString()`: restituisce una forma testuale dello stato dell'oggetto
- `public double ultimo()`: restituisce il valore dell'ultimo operando del contenitore
- `public abstract double calcola()`: efffettua il calcolo (e restituisce il valore relativo) effettivo, verrà implementato nelle sottoclassi, deve lanciare un'eccezione se l'operazione non ha operandi su cui lavorare

3 Somma

Sottoclasse di Operazione, calcola la somma degli operandi, ad es. se la lista dei valori è: 3, 5, 8, 3, 3 il risultato diventa 3+5+8+3+3=22.

4 Massimo

Sottoclasse di Operazione, calcola il massimo tra gli operandi, ad es. se la lista dei valori è: 3, 5, 8 il risultato diventa 8.

5 Media

Sottoclasse di Operazione, calcola la media degli operandi, ad es. se la lista dei valori è: 4, 9, 8, 7, 3 il risultato diventa (4+9+8+7+3)/5=6,2.

6 Moltiplicazione

Sottoclasse di Operazione, calcola il prodotto degli operandi, ad es. se la lista dei valori è: 3, 5, 8 il risultato diventa 3*5*8=120.

7 SommaUnici

Sottoclasse di Operazione, calcola la somma degli operandi considerando **una sola volta** gli operandi uguali, ad es. se la lista dei valori è: 3, 5, 8, 3, 3 il risultato diventa 3+5+8=16.

Consegna

Si ricorda che:

- le classi devono essere tutte *public*
- vanno consegnati tutti (e soli) i file *.java* prodotti
- si consiglia di fare upload successivi e frequenti, i docenti vedono solo l'ultima versione di ogni file
- NON va consegnato un file "archivio"! (NO tar, zip, etc.)
- NON vanno consegnati i *.class*
- NON vanno consegnati i file relativi al meccanismo di autovalutazione (*Test_*.java, AbstractTest.java, *.sh*)
- eseguite l'upload dei SINGOLI file sorgente (http://upload.di.unimi.it) nella sessione "Trentini"

*** 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`.
