Esame di Programmazione II, 9 settembre 2025

(si consegnino i file `.java` delle classi richieste negli esercizi)

Si crei un progetto Eclipse e il package `it.univr.dadi`*. Si copino al suo interno le classi del compito. Non si modifichino le dichiarazioni dei metodi e delle classi. Si possono definire altri campi, metodi o costruttori non richiesti dal compito, ma devono essere* `private`*. Si possono definire altre classi non richieste dal compio, che in tal caso vanno consegnate. La soluzione che verrà consegnata dovrà compilare, altrimenti non verrà corretta.*

L'interfaccia `Dado`, già completa e da non modificare, rappresenta un dado, cioè un oggetto che ha un numero fissato di facce e che può essere *lanciato*, ottenendo un numero intero fra 1 e il numero di facce del dado (inclusi). La classe `Lanci` implementa il lancio ripetuto di alcuni dadi, forniti al costruttore. I suoi oggetti possono essere stampati, poiché il `toString()` restituisce il risultato dei lanci, e possono essere trasformati in una stringa che disegna un istogramma con il risultato dei lanci, tramite il metodo `frequenze()`.

Per esempio, si consideri il seguente codice della classe `Main`, già fatta e completa:

```
System.out.println("Lanciamo 20 volte due dadi a sei facce");
Lanci l = new Lanci(20, new D6(), new D6());
System.out.println("Lanci ottenuti: " + l);
System.out.println(l.frequenze());

System.out.println("Lanciamo 10000 volte un dado a sei facce");
l = new Lanci(10000, new D6());
System.out.println(l.frequenze());

System.out.println("Lanciamo 10000 volte un dado a sei facce truccato");
l = new Lanci(10000, new D6Truccato());
System.out.println(l.frequenze());

System.out.println("Lanciamo 10000 volte un dado a sei facce truccatissimo");
l = new Lanci(10000, new D6Truccatissimo());
System.out.println(l.frequenze());

System.out.println("Lanciamo 10000 volte un dado a otto facce, uno a sei facce truccato
  e uno a dieci facce");
l = new Lanci(10000, new D8(), new D6Truccato(), new D10());
System.out.println(l.frequenze());

System.out.println("Lanciamo 10000 volte tre dadi a sei facce, usando frecce");
l = new LanciFrecce(10000, new D6(), new D6(), new D6());
System.out.println(l.frequenze());

System.out.println("Lanciamo 10000 volte tre dadi a sei facce, usando frecce alternate");
l = new LanciFrecceAlternate(10000, new D6(), new D6(), new D6());
System.out.println(l.frequenze());

System.out.println("Lanciamo -10000 volte tre dadi a sei facce, usando frecce alternate");
new LanciFrecceAlternate(-10000, new D6(), new D6(), new D6());
```

L'esecuzione di tale codice dovrebbe stampare qualcosa del tipo:

```
Lanciamo 20 volte due dadi a sei facce
Lanci ottenuti: [8, 8, 4, 8, 10, 8, 4, 7, 8, 5, 5, 6, 5, 6, 8, 6, 6, 8, 6, 8]
  2:  (0.0%)
  3:  (0.0%)
  4: ********** (10.0%)
  5: *************** (15.0%)
  6: ************************* (25.0%)
  7: ***** (5.0%)
  8: ****************************************** (40.0%)
  9:  (0.0%)
 10: ***** (5.0%)
 11:  (0.0%)
 12:  (0.0%)

Lanciamo 10000 volte un dado a sei facce
  1: ***************** (16.7%)
  2: ***************** (16.3%)
  3: ***************** (16.5%)
  4: ****************** (17.3%)
  5: ***************** (16.1%)
  6: ****************** (17.0%)

Lanciamo 10000 volte un dado a sei facce truccato
  1: ********** (10.5%)
  2: ********** (10.3%)
  3: ********* (10.0%)
  4: ********** (10.2%)
  5: ***************************** (29.5%)
  6: ***************************** (29.4%)

Lanciamo 10000 volte un dado a sei facce truccatissimo
  1:  (0.0%)
  2:  (0.0%)
  3:  (0.0%)
  4:  (0.0%)
  5:  (0.0%)
  6: ******************************.....******************** (100.0%)

Lanciamo 10000 volte un dado a otto facce, uno a sei facce truccato
    e uno a dieci facce
  3:  (0.2%)
  4:  (0.5%)
  5:  (0.8%)
  6: * (1.3%)
  7: ** (2.3%)
  8: *** (3.5%)
  9: **** (4.9%)
 10: ***** (5.6%)
 11: ******* (7.1%)
 12: ******** (8.2%)
```

```
 13: ******** (8.3%)
 14: ********* (9.7%)
 15: ********* (9.8%)
 16: ******** (8.2%)
 17: ******* (7.8%)
 18: ****** (6.5%)
 19: ***** (5.3%)
 20: **** (4.1%)
 21: ** (2.9%)
 22: * (1.7%)
 23:  (0.9%)
 24:  (0.4%)

Lanciamo 10000 volte tre dadi a sei facce, usando frecce
  3:  (0.4%)
  4: > (1.5%)
  5: --> (3.1%)
  6: ---> (4.5%)
  7: ------> (7.2%)
  8: --------> (9.5%)
  9: ----------> (11.0%)
 10: ----------> (12.9%)
 11: ----------> (12.6%)
 12: ----------> (11.5%)
 13: --------> (9.9%)
 14: -----> (6.4%)
 15: ---> (4.5%)
 16: -> (2.8%)
 17: > (1.6%)
 18:  (0.4%)

Lanciamo 10000 volte tre dadi a sei facce, usando frecce alternate
  3:  (0.4%)
  4: > (1.3%)
  5: => (2.5%)
  6: ---> (4.8%)
  7: =====> (6.7%)
  8: --------> (9.9%)
  9: ==========> (11.2%)
 10: ----------> (13.0%)
 11: ==========> (12.6%)
 12: ----------> (11.7%)
 13: ========> (9.3%)
 14: -----> (6.8%)
 15: ===> (4.9%)
 16: -> (3.0%)
 17: > (1.3%)
 18:  (0.6%)
Lanciamo -10000 volte tre dadi a sei facce, usando frecce alternate:
java.lang.IllegalArgumentException: Il numero di lanci richiesto deve essere positivo
```

Esercizio 1 (3 **punti).** Si completi la classe astratta `AbstractDado`, che implementa il codice comune a tutti i dadi, cioè quello per la gestione del numero di facce del dado.

Esercizio 2 (3 **punti).** Si scriva una sottoclasse astratta `DadoUniforme` di `AbstractDado`, che implementa un dado uniforme, in cui cioè la probabilità di ottenere una faccia, lanciandolo, è la stessa per ciascuna faccia. Questa classe dovrà avere un costruttore che specifica il numero di facce del dado e dovrà implementare il metodo `lancio()`.

Esercizio 3 (3 **punti).** Si scrivano le sottoclassi concrete `D6`, `D8` e `D10` di `DadoUniforme`, che implementato dadi uniformi a sei, otto e dieci facce, rispettivamente.

Esercizio 4 (3 **punti).** Si scriva una sottoclasse `D6Truccato` di `AbstractDado` che implementa un dado a sei facce truccato, poiché, lanciandolo, le facce 5 e 6 escono ciascuna mediamente nel 30% dei casi, mentre le facce 1, 2, 3 e 4 escono ciascuna mediamente nel 10% dei casi.

Esercizio 5 (2 **punti).** Si scriva una sottoclasse `D6Truccatissimo` di `AbstractDado` che implementa un dado a sei facce truccatissimo, poiché, lanciandolo, esce sempre e soltanto la faccia 6.

Esercizio 6 (8 **punti).** Si completi la classe concreta `Lanci`, il cui costruttore deve lanciare i dadi forniti, insieme e ripetutamente, per il numero di volte `quanti` indicato al costruttore. Il metodo `toString()` deve restituire la stampa dei risultati di tali lanci. Il metodo `frequenze()` deve restituire una stringa istogramma, come quelli che si vedono nell'esecuzione dell'esempio `Main` della pagina precedente. La classe `Lanci` usa asterischi per rappresentare le barre degli istogrammi.

Esercizio 7 (4 **punti).** Si completi la sottoclasse concreta `LanciFrecce` di `Lanci`, che si differenzia da `Lanci` solo perché rappresenta le barre degli istogrammi con delle frecce, come nel penultimo esempio della pagina precedente.

Esercizio 8 (5 **punti).** Si completi la sottoclasse concreta `LanciFrecceAlternate` di `Lanci`, che si differenzia da `Lanci` solo perché rappresenta le barre degli istogrammi con delle frecce, alternativamente fatte con il carattere `'-'` e con il carattere `'='`, come nell'ultimo esempio della pagina precedente.