Punto Cartesiano

Si scriva una classe PuntoCartesiano che rappresenti un punto geometrico in coordinate cartesiane (x, y). Oltre a un costruttore, serviranno i metodi seguenti:

- `p.dist(q)` restituisce la distanza euclidea fra i punti p e q
- `p.translate(q)` sposta il punto p, traslandolo di q
- `p.zero()` - sposta il punto p alle coordinate (0,0)

Cineteca

Testo

Si progetti una classe `Cineteca` rappresentante un insieme di film. Ogni film è un oggetto avente i seguenti attributi: `titolo`, `regista`, `annoUscita`, `volteVisto`.
Si può assumere che il titolo rappresenti univocamente un film, ossia che non esistono due film diversi aventi lo stesso titolo.

- `c.add(titolo, regista, anno)` Il metodo aggiunge il film alla cineteca e restituisce il numero totale dopo l'aggiunta.
- `c.remove(titolo)` Il metodo rimuove il film, se presente, e restituisce il numero di film dopo l'operazione. Il numero va comunque restituito anche in caso non sia possibile rimuovere alcun film)
- `c.count()` Numero di film presenti nella cineteca.
- `c.vedi(titolo)` Se il film è presente, incrementa di uno il suo numero di visualizzazioni e restituisce il numero di visualizzazioni totale; altrimenti, restituisce -1.

Cineteca

Esempio

```
c = new Cineteca();
c.add("Il Signore degli Anelli", "Peter Jackson", 2001) -> 1
c.add("Guida galattica per autostoppisti", "Garth Jennings", 2005) -> 2
c.vedi("Il Signore degli Anelli") -> 1
c.vedi("Il Signore degli Anelli") -> 2
c.add("Spaceballs", "Mel Brooks", 1987) -> 3
c.remove("Il nome della rosa") -> 3
c.count() -> 3
```

Collezione

Testo

Si scriva una classe `Collezione` che rappresenta una struttura dati astratta. Oltre al costruttore, la classe deve definire i metodi:

- `occurrences(o)` che restituisce il numero di occorrenze di `o`.
- `len()` che restituisce il numero di elementi presenti nella struttura dati.
- `isEmpty()` che restituisce true se la struttura dati è vuota, false altrimenti.

Infine si specializzino le classi `Coda` e `Pila` che implementano i metodi:

- `add(o)` che aggiunge l'elemento alla collezione.
- `remove()` che rimuove l'elemento dalla collezione e lo restituisce.

Nel caso di `Coda` l'elemento rimosso è il primo inserito (FIFO), mentre nel caso di `Pila` l'elemento rimosso è l'ultimo inserito (LIFO). Si definisca infine una funzione `mediaCollezioni` che data una lista di collezioni restituisce la loro lunghezza media, indipendentemente dall'implementazione.

Collezione

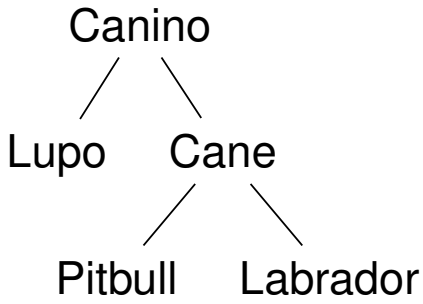
Esempi

```
c=new Coda()
c.add(1)
c.add(2)
c.add(3)
c.remove() -> 1

p = new Pila()
p.add(1)
p.add(2)
p.add(3)
p.remove() -> 3
p.remove() -> 2

mediaCollezioni([c,p]) -> 2
```

Animali



Si riproduca la tassonomia in figura come una gerarchia di classi. Ogni classe dovrà implementare almeno un costruttore, che dato il nome e l'età dell'animale costruisce l'oggetto. Inoltre, definire nelle classi appropriate i metodi: `abbaia`, `bevi`, `mangia`, `ulula`. Ognuno di questi aggiorna lo stato interno dell'animale, incrementando un contatore `num_actions` che indica il numero di azioni eseguite.

Si scriva infine una funzione `contaAzioni`, che data una lista di animali e un riferimento a una classe, restituisce la somma del numero di azioni compiute dagli animali istanza della classe.

Animali

Esempio:

```
p = new Pitbull("Fido", 2);
l = new Labrador("Britta", 3);
w = new Lupo("Pippo", 5);

p.abbaia()
p.mangia()
l.bevi()
w.mangia()
w.ulula()

contaAzioni([p, l, w], Cane) -> 3
```