Laboratorio di Programmazione a.a. 2015-16

Problema 14

18 Aprile 2016

Descrizione

Si vuole modellare alcune funzionalità di un *distributore automatico* di bevande, simile a quelli disponibili nei corridoi della nostra sede universitaria. In particolare, il modello dovrà gestire la riserva di monete, i crediti e i resti in relazione all'erogazione di una bevanda. Per semplicità, supponiamo che il distributore riceva solo monete, rispettivamente da 5 centesimi, 10, 20, 50, 100 (= 1 euro) e 200 (= 2 euro), e serva un solo tipo di bevanda, un bicchierino di caffé espresso, senza possibilità di regolare la dose di zucchero. Il fruitore ha dunque a disposizione una fessura per inserire una moneta alla volta e due pulsanti, uno per richiedere l'erogazione del caffé e il secondo per richiedere il resto (o l'intera somma introdotta se cambia idea o se l'operazione non può essere portata a termine). Il gestore ha inoltre la possibilità di (ri)caricare le dosi di caffé (comprensive dei bicchierini) e la riserva di monete. Il costo di un bicchierino di caffé è stabilito una volta per tutte, all'inizio.

Il modello è rappresentato dalla classe Java `DistributoreAutomatico`, per cui è definito il seguente protocollo:

`d = new DistributoreAutomatico( c )`	costruttore: l'intero *c* indica il costo in centesimi di un caffé
`d.caricaDosi( k )`	carica (aggiunge) nel distributore *k* dosi di caffé e *k* bicchierini
`d.caricaMonete( n, v )`	carica nel distributore *n* monete da *v* centesimi l'una per aumentare la riserva di monete da utilizzare come resto
`d.inserisciMoneta( v )`	inserisce una moneta da *v* centesimi per creare o aumentare il credito da parte del fruitore
`d.richiediCaffe()`	determina l'erogazione di un bicchierino di caffé, purché ciò sia possibile e il credito sia sufficiente
`d.richiediResto()`	determina la restituzione del credito residuo
`String s = d.leggiStato()`	restituisce una descrizione testuale dello stato del distributore
`... d.caffeEsaurito()`	verifica se il caffé è esaurito restituendo un booleano

Normalmente il metodo `richiediCaffe` consuma una dose (e un bicchierino) di caffé e ne applica il costo riducendo il credito da parte del fruitore. Tuttavia, l'operazione ha un effetto nullo nei seguenti casi: (i) il credito non è sufficiente; (ii) le dosi di caffé sono esaurite; (ii) nel caso di erogazione la riserva di monete non consentirebbe di restituire il credito residuo (resto). Inoltre, la stringa restituita dal metodo `leggiStato` ha una delle seguenti forme:

- `"caffe' esaurito"` se le dosi di caffé sono esaurite
- `"in funzione"` se c'è disponibilità di caffé e il credito è zero
- `"resto non disponibile per` *x* `cent"` se c'è disponibilità di caffé, il credito di *x* cent è superiore al costo, ma nel caso di erogazione non si potrebbe restituire il resto
- `"credito:` *x* `cent"` se c'è disponibilità di caffé e il credito è di *x* cent, inferiore o uguale al costo, oppure tale che ci sia la possibilità di restituire il resto

Realizza la classe `DistributoreAutomatico` rispettando le specifiche informali illustrate sopra.

Si intende, naturalmente, che le monete introdotte dal fruitore vadano ad integrare la riserva interna. Per semplificare la composizione del resto, e la verifica che sia possibile restituirlo utilizzando la riserva, puoi limitarti a considerare le monete disponibili per ordine decrescente di valore, di volta in volta immaginando di utilizzarne tante quante ne sono necessarie o disponibili, senza esplorare tutte le combinazioni possibili—cosa che complicherebbe alquanto la ricerca di una soluzione.

Per maggiore chiarezza, nella pagina seguente è riportato un esempio di codice che utilizza un'istanza della classe `DistributoreAutomatico`, nonché il feedback che ne risulta dall'esecuzione. Tale traccia ti servirà anche come elemento di verifica del corretto funzionamento del programma da te realizzato.

Esempio

Codice:

```
DistributoreAutomatico distributore
  = new DistributoreAutomatico( 45 );

distributore.caricaDosi( 10 );

distributore.caricaMonete( 10,   5 );
distributore.caricaMonete(  8,  10 );
distributore.caricaMonete(  5,  20 );
distributore.caricaMonete(  5,  50 );
distributore.caricaMonete(  2, 100 );
distributore.caricaMonete(  1, 200 );

System.out.println( distributore.leggiStato() );

for ( int k=0; k<5; k=k+1 ) {

  distributore.inserisciMoneta( 200 );
  System.out.println( distributore.leggiStato() );

  distributore.richiediCaffe();
  System.out.println( distributore.leggiStato() );

  distributore.richiediResto();
  System.out.println( distributore.leggiStato() );
}
for ( int k=0; k<5; k=k+1 ) {

  distributore.inserisciMoneta( 20 );
  distributore.inserisciMoneta( 20 );
  distributore.inserisciMoneta( 20 );
  System.out.println( distributore.leggiStato() );

  distributore.richiediCaffe();
  System.out.println( distributore.leggiStato() );

  distributore.richiediResto();
  System.out.println( distributore.leggiStato() );
}
for ( int k=0; k<2; k=k+1 ) {

  distributore.inserisciMoneta( 20 );
  distributore.inserisciMoneta( 20 );
  distributore.inserisciMoneta(  5 );
  System.out.println( distributore.leggiStato() );

  distributore.richiediCaffe();
  System.out.println( distributore.leggiStato() );


  distributore.richiediResto();

  System.out.println( distributore.leggiStato() );

}
```

Output:

```
in funzione
credito: 200 cent
credito: 155 cent
in funzione
credito: 200 cent
credito: 155 cent
in funzione
credito: 200 cent
credito: 155 cent
in funzione
credito: 200 cent
credito: 155 cent
in funzione
resto non disponibile per 200 cent
resto non disponibile per 200 cent
in funzione
credito: 60 cent
credito: 15 cent
in funzione
credito: 60 cent
credito: 15 cent
in funzione
credito: 60 cent
credito: 15 cent
in funzione
credito: 60 cent
credito: 15 cent
in funzione
resto non disponibile per 60 cent
resto non disponibile per 60 cent
in funzione
credito: 45 cent
in funzione
in funzione
credito: 45 cent
caffe' esaurito
caffe' esaurito
```