



Problema 16

8 Maggio 2015

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, 10, 20, 50, 100 (1 euro) e 200 (2 euro) centesimi, e serva un solo tipo di bevanda, un bicchierino di caffè 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 caffè 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 caffè (comprensive dei bicchierini) e la riserva di monete. Il costo di un bicchierino di caffè è stabilito una volta per tutte, all'inizio.

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

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

Normalmente il metodo `richiediCaffe` consuma una dose (e un bicchierino) di caffè 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 caffè 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 caffè sono esaurite
- `"in funzione"` se c'è disponibilità di caffè e il credito è zero
- `"resto non disponibile per x cent"` se c'è disponibilità di caffè, il credito di *x* cent è superiore al costo, ma nel caso di erogazione non ci sarebbe modo di restituire il resto
- `"credito: x cent"` se c'è disponibilità di caffè 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.

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.caricaCaffe( 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
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
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
```