Distributore Automatico di Bevande

Un distributore automatico è un oggetto che contiene bevande, ha un incasso totale e può essere in stato di manutenzione oppure attivo.

Un distributore può essere in `manutenzione` (si presume che quando è in manutenzione non può erogare bevande né accettare denaro) oppure `attivo` (in questo caso può compiere tutte le operazioni normali). Quando passa da manutenzione ad attivo, il distributore esegue un check di tutte le bevande presenti.

Se il distributore è `attivo`, sarà possibile:

- `inserire` denaro (che va ad accumularsi nel credito temporaneo dell'acquisto)
- `acquistare` una bevanda se il credito è sufficiente (Es: se una bevanda costa 2.50€ e sono stati inseriti 5€, deve restituire 2.50€ di resto)
- `rifornire` il distributore con nuove bevande (ogni bevanda ha un nome, un prezzo e una quantità)
- `ottenere` l'incasso totale del distributore
- verificare la `disponibilità` di una bevanda

Il distributore accetta solo monete/banconote valide: 0.50€, 1€, 2€, 5€. Se viene inserito un importo non valido, questo viene rifiutato. Il distributore eroga la bevanda solo se questa è disponibile e se il credito inserito è sufficiente.

Dopo aver letto con attenzione il testo soprariportato, scrivere una classe `Distributore` che ne descriva le caratteristiche e le azioni che esso può fare.

Le caratteristiche saranno codificate attraverso delle variabili di esemplare:

- stato del distributore (manutenzione/attivo)
- array/ArrayList di bevande disponibili
- incasso totale
- credito temporaneo dell'acquisto in corso

Le azioni saranno dei metodi invocabili sul Distributore.

Implementare quindi:

1. Un Costruttore che inizializza un Distributore vuoto (senza bevande) e attivo
2. Tutti i metodi getter/setter necessari
3. Il metodo toString
4. I seguenti metodi caratteristici:

- `inserisciDenaro(double importo)` : restituisce true se l'importo è valido e viene accettato
- `acquistaBevanda(String nome)` : tenta l'acquisto e restituisce il resto se necessario
- `aggiungiBevanda(String nome, double prezzo, int quantità)` : aggiunge una bevanda al distributore
- `getIncassoTotale()` : restituisce l'incasso totale
- `getBevandaDisponibile(String nome)` : verifica se una bevanda è disponibile
- `setManutenzione(boolean stato)` : mette/toglie il distributore dalla modalità manutenzione

Scrivere una classe tester `DistributoreTester.java` che verifichi:

- Creazione distributore
- Aggiunta bevande
- Inserimento importi validi e non validi
- Acquisto con resto
- Acquisto bevanda non disponibile
- Cambio stato manutenzione/attivo

Carta di Credito

Una carta di credito è un oggetto che ha un proprietario, un saldo, un PIN per l'autenticazione e può essere bloccata o attiva.

Una carta può essere `bloccata` (si presume che quando è bloccata non può effettuare alcuna operazione) o `attiva`. Il blocco avviene automaticamente dopo 3 tentativi errati di inserimento del PIN, mentre lo sblocco può avvenire solo inserendo il PIN corretto.

Se la carta è `attiva`, sarà possibile:

- `prelevare` denaro (solo se il PIN inserito è corretto e l'importo non supera il limite giornaliero di 500€)
- `versare` denaro sul conto
- `verificare` il saldo
- `cambiare` il PIN (richiede il PIN vecchio)

La carta memorizza le ultime 5 operazioni effettuate con data e importo. Ogni operazione di prelievo richiede la verifica del PIN e del limite giornaliero. Se un'operazione viene rifiutata, viene comunque registrata nel report delle operazioni.

Dopo aver letto con attenzione il testo soprariportato, scrivere una classe `CartaDiCredito` che ne descriva le caratteristiche e le azioni.

Le caratteristiche saranno codificate attraverso delle variabili di esemplare:

- proprietario della carta
- saldo attuale
- PIN
- stato (bloccata/attiva)
- contatore tentativi PIN errati
- lista ultime operazioni
- totale prelevato oggi

Implementare quindi:

1. Un Costruttore che inizializza una carta con proprietario, PIN e saldo iniziale
2. Tutti i metodi getter/setter necessari
3. Il metodo toString
4. I seguenti metodi caratteristici:
   - `verificaPIN(int pin)` : verifica il PIN e gestisce i tentativi
   - `preleva(double importo, int pin)` : effettua un prelievo se possibile
   - `versa(double importo)` : effettua un versamento
   - `getSaldo()` : restituisce il saldo attuale
   - `cambiaPin(int vecchioPin, int nuovoPin)` : cambia il PIN
   - `getUltimeOperazioni()` : restituisce report ultime operazioni

Scrivere un'opportuna classe tester che dimostri il funzionamento di tutti i metodi presentati.

Biblioteca Digitale

Una biblioteca digitale è un sistema che gestisce ebook, utenti registrati e prestiti digitali.

Un ebook può essere `disponibile` (cioè può essere prestato) o `in prestito` (quando è già assegnato a un utente). Il prestito ha una durata standard di 14 giorni, ma può essere esteso una sola volta per altri 7 giorni.

Se un ebook è `disponibile`, sarà possibile:

- `prestarlo` a un utente registrato
- `visualizzare` le sue informazioni
- `ottenere` suggerimenti di ebook simili per genere
- `verificare` la sua disponibilità

La biblioteca tiene traccia di tutti i prestiti effettuati e può generare statistiche sui generi più letti e suggerimenti personalizzati basati sulla cronologia di lettura degli utenti.

Implementare una classe `BibliotecaDigitale` con le seguenti caratteristiche:

Le caratteristiche saranno codificate attraverso delle variabili di esemplare:

- catalogo degli ebook disponibili
- lista utenti registrati
- registro dei prestiti attivi
- storico dei prestiti completati

Implementare quindi:

1. Un Costruttore che inizializza una biblioteca vuota
2. Tutti i metodi getter/setter necessari
3. Il metodo toString
4. I seguenti metodi caratteristici:
   - `aggiungiEbook(String titolo, String autore, String genere)` : aggiunge un ebook al catalogo
   - `registraUtente(String nome, String email)` : registra un nuovo utente
   - `prestaEbook(String titolo, String utente)` : presta un ebook se disponibile
   - `restituisciEbook(String titolo, String utente)` : registra la restituzione
   - `estendiPrestito(String titolo, String utente)` : prolunga il prestito se possibile
   - `getSuggerimenti(String utente)` : genera suggerimenti personalizzati
   - `getStatistiche()` : restituisce statistiche sui prestiti

Scrivere un'opportuna classe tester che dimostri il funzionamento di tutti i metodi presentati.