Programmazione OO
Secondo Appello 12/02/2020

Non è consentito usare libri o appunti.

Implementare un sistema per la gestione dei tesserini di una mensa.

1. [**12 punti**] Implementare in Java
   - la classe `Tesserino` che modella i vari tipi di tesserino utilizzati per la mensa universitaria. Ogni tesserino è caratterizzato da `codice`, `nome`, `cognome`, e dalla proprietà `attivo` (quest'ultima è una variabile booleana indicante se il tesserino è utilizzabile). Corredare la classe con i metodi
     - `attiva()` che setta a `true` lo stato della variabile `attivo` (se `acceso` è già `true` lancia l'eccezione `RunTimeException`).
     - `disattiva()` che setta a `false` lo stato della variabile `attivo` (se `acceso` è già `false` lancia l'eccezione `RunTimeException`).

   - due sottoclassi
     1) `TesserinoStudente` caratterizzata da `matricola`, `scadenza`, `saldo`, `fascia`, `bonus`. Corredare la classe con i metodi
        - `double calcolaPrezzo()` che calcola il prezzo del pasto di uno studente in base alla fascia: coloro che appartengono alla fascia A il prezzo sarà di 2.50€, mentre coloro che appartengono alla fascia B il prezzo sarà 1.50€. Inoltre gli studenti vincitori di borse di studio (i cui tesserini hanno la variabile bonus settata a `true`) hanno lo sconto di 1€.
        - `boolean isBonus()` se lo studente ha vinto la borsa di studio il metodo restituisce `true`.
        - `double paga()` che simula il pagamento di un pasto. Il metodo controlla prima se il tesserino è scaduto. Se è scaduto lancia l'eccezione controllata `TesserinoScadutoException`, altrimenti sottrae al saldo il costo del pasto. Nel caso in cui il saldo è insufficiente per pagare il pasto viene lanciata l'eccezione non controllata `SaldoInsufficienteException`. Il valore restituito corrisponde all'importo pagato dallo studente.
        - `void versa(double x)` che aggiorna il saldo con la somma `x` passata come argomento. Se la somma da versare è negativa lancia l'eccezione `RunTimeException`.

     - **Nota Bene**: per ottenere la data corrente utilizzate la classe `GregorianCalendar` il cui funzionamento è descritto dalle seguenti istruzioni:

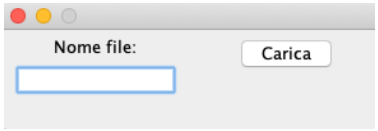
```
Calendar calendar = new GregorianCalendar();
int giorno = calendar.get(Calendar.DAY);
int mese = calendar.get(Calendar.MONTH);
int anno = calendar.get(Calendar.YEAR);
```

     2) `TesserinoPersonale` caratterizzata dalle variabili `facoltà`, `sommaSpesa`, `categoria` (che può essere docente o amministrativo). Corredare la classe con i metodi
        - `double paga()` aggiunge a `sommaSpesa` il costo del pasto che dipende dal valore di `categoria`. Nel caso di categoria docente l'importo del pasto è 1.60€, nel caso di categoria amministrativo l'importo è 4.00€. Il metodo restituisce l'importo pagato dal personale.
        - `void cambiaCategoria()` il metodo cambia la categoria del tesserino.

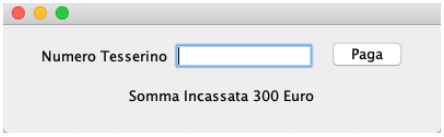
2. [**8 punti**] Scrivere la classe `PagamentoPasti` che modella una collezione di tesserini e fornisce i seguenti metodi:

   - `void aggiungiTesserino(Tesserino t)` che inserisce un tesserino nell'archivio.
   - `boolean usaTesserino(int code)` che simula il pagamento di un pasto per la persona in possesso del tesserino con codice `code`. Il metodo non gestisce le eventuali eccezioni lanciate. Restituisce `true` se il codice è presente nell'archivio, `false` altrimenti.
   - `double calcolaTotale()` che restituisce la somma incassata fino a quello istante. **N.B.** *Utilizzare una variabile che tiene traccia di tale somma durante i pagamenti.*
   - `PagamentoPasti dammiTesserinoPerTipo(int x)`, che restituisce l'elenco dei tesserini di una certa tipologia. In particolare, `x`=0 indica tipologia `TesserinoStudente`, `x`=1 `TesserinoPersonale`. Per valori di `x` diversi da 0, 1, viene la lanciata l'eccezione controllata `ParametroInvalidoException`.
   - `double dammiSommaSpesa()`, che restituisce la somma spesa da tutti i possessori di `TesserinoPersonale`.

3. [**10 punti**] Considerando le classi ai punti precedenti, scrivere un programma Java che realizzi un'interfaccia grafica per
   a) caricare da un file una lista di tesserini,



   b) effettuare il pagamento di un pasto (chiedere di inserire solo il codice del tesserino),
   c) visualizzare la somma incassata.



Ogni violazione delle regole enunciate ai punti sotto elencati comporta l'annullamento della prova (l'elaborato viene valutato 0).

1. Prima di eseguire eclipse assicurarsi che non ci siano file Java (sorgenti, bytecode, workspace,progetti, pacchetti) sul desktop.

2. Eseguire eclipse specificando un workspace sul desktop.

3. Durante la prova d'esame è vietato usare:
   a. libri e appunti sia in forma cartacea che in forma digitale
   b. supporti di memoria esterni
   c. un font di dimensione maggiore di 10 punti.

4. Non è consentito modificare i file allegati alla traccia.

5. Il nome del progetto consegnato deve cominciare con COGNOME seguito dal carattere underscore e quindi dal NOME (tutto in maiuscole). Ad esempio, il nome del progetto di Marco Rossi può essere ROSSI_MARCO, ROSSI_MARCO_P2, ROSSI_MARCO_ESERCIZIO, ROSSI_MARCO_549449384, etc.

6. Il file da consegnare deve essere creato da eclipse seguendo i passi:
   a. Seleziona "export…" nel menu file
   b. Seleziona "Archive File" in "General"
   c. Pressa "Next"
   d. Seleziona progetto da esportare
   e. Controllare il percorso del file (nell'area di testo con etichetta "To archive file:")
   f. Assicurarsi che i pulsanti radio nel pannello Options siano selezionati su "Save in zip format" e "Create directory structure for files"
   g. Pressa "Finish"

Assicurarsi che i progetti consegnati possono essere importati in eclipse come:
General → Existing Projects into Workspace