LabProg 2011-2012 - Tema d'Esame
TURNI A, B, C
Appello del 1 febbraio 2012

Avvertenza. Non è ammesso l'uso delle classi del package *prog.io* allegato al libro di testo del corso.

1 Testo esame

Lo scopo è realizzare un modello di prenotazione per visite guidate.
Le classi da realizzare sono le seguenti (dettagli nelle sezioni successive):

- *Persona*: classe astratta, un possibile visitatore, ha nome, cognome
- *Adulto*: sottoclasse concreta di Persona
- *Studente*: sottoclasse concreta di Persona, ha un'età
- *Senior*: sottoclasse concreta di Persona, ha un'età
- *Turno*: visita per un certo gruppo, ha un massimo numero di partecipanti e la lista delle persone prenotate
- *Evento*: l'evento per cui si vogliono organizzare le visite, ha un nome, un prezzo e la lista di turni di visita organizzati.
- *Main*: contiene il metodo *main()* e deve permettere di testare TUTTI i metodi almeno una volta. Inoltre deve gestire le richieste di prenotazione che riceve in input.

Tali classi dovranno esporre almeno i metodi specificati nelle sezioni seguenti. Eventuali metodi di servizio possono essere aggiunti a piacimento. Ogni classe deve avere il *toString()* che rappresenti lo stato delle istanze e i **costruttori** adeguati per gli attributi che vengono dichiarati. Dato che gli attributi devono essere privati creare opportunamente i metodi di accesso (set&get). Si suggerisce, anche dove non segnalato, di utilizzare, se esistenti e se applicabili, le classi parametriche (es. *ArrayList<E>* invece di *ArrayList*). Alcuni controlli di coerenza vengono suggeriti nel testo, potrebbero essercene altri a discrezione. I tipi di ritorno possono essere variati (ad es. boolean invece di void se si vuole ottenere un feedback sul successo dell'operazione) previa autorizzazione del docente. Consiglio di posporre l'implementazione dei controlli di coerenza, fatelo come ultima operazione, prima realizzate un sistema funzionante.

1.1 Persona

Rappresenta un possibile visitatore, ha (attributi) nome e cognome.

1.2 Studente, Senior, Adulto

Rappresentano tipologie di visitatori, per Studente e Senior è previsto un campo età e il relativo metodo d'accesso.

1.3 Turno

Rappresenta un gruppo di visitatori, ha (attributi) il massimo numero di partecipanti ammessi e la lista delle persone prenotate, oltre al numero progressivo di turno (il primo turno creato ha numero 1, il secondo 2, ecc.).
Deve inoltre esporre i seguenti metodi **pubblici**:

- *boolean addPrenotazione(Persona p)*: accetta una nuova prenotazione (*Suggerimento.* Cosa succede se una stessa persona cerca di prenotarsi piú volte?)
- *getListaPrenotati()*: restituisce la lista dei nominativi delle persone prenotate
- *int getNrTurno*: restituisce il numero del turno

1.4 Evento

Rappresenta un evento per cui ci si può prenotare, è caratterizzato dal nome, dal prezzo del biglietto e dalla lista di turni di visita organizzati.
Deve inoltre esporre i seguenti metodi **pubblici**:

- *void addTurno(Turno t)*: aggiunge un turno
- *Turno getTurno(int i)*: restituisce l'i-esimo turno della lista
- *double getPrezzo()*: restituisce il prezzo del biglietto

1.5 Main

Deve contenere il solo metodo *main()*, implementato secondo le specifiche seguenti.
Il metodo *main* accetta come argomento passato dalla riga di comando il nome di un file. Per prima cosa, il programma istanzia un oggetto di classe *Evento*, usando come valori degli attributi il nome "Galleria degli Uffizi" e il prezzo del biglietto di 5 euro e mezzo. L'evento deve essere dotato di due turni, il primo di 4 (quattro) persone e il secondo di 3 (tre) persone. Il programma tenta quindi di aprire il file, gestendo le eccezioni nel modo opportuno. Supponendo adesso che l'argomento passato corrisponda al nome di un file esistente, il programma apre il file, e ne legge il contenuto una riga alla volta. Il programma assume che il formato di ciascuna riga del file sia il seguente:

```
nome cognome [categoria età] turno
```

Ad esempio, il file potrebbe contenere:[1]

```
Gigi Lupo studente 16 turno 2
Toto Gatti over65 72 turno 2
Pippo Leone turno 1
Beppe Topo studente 24 turno 2
Kiki Gallo turno 2
Pinco Pallo over65 22 turno 3
Tizio Caio studente 99 turno 2
J.S. Bach turno 34
```

[1] Il termine `over65` è qui usato come sinonimo di "senior".

Ciascuna riga corrisponde quindi a una prenotazione. Per ogni prenotazione il programma deve dare la conferma a video:

```
Gigi Lupo:  turno 2 - prenotazione
effettuata.
```

Se invece i posti per quel turno sono esauriti, il programma deve visualizzare:

```
Tizio Caio:  turno 2 - spiacente,
posti esauriti.
```

Nel caso che il turno richiesto sia inesistente, il programma visualizza:

```
J.S. Bach:  turno 34 - spiacente,
turno inesistente.
```

Potete aggiungere a vostra discrezione qualche controllo di coerenza e di formato sui dati forniti in ingresso; in questo caso, il programma dovrà visualizzare i messaggi opportuni, sulla falsariga di quanto appena indicato. (*Suggerimenti.* Qual è, ad esempio, il problema coi dati dichiarati da `Pinco Pallo`? E cosa fa il vostro programma se il file fornito in ingresso contiene una riga non formattata secondo le specifiche?)
Completata la lettura del file, il programma deve stampare le seguenti informazioni di riepilogo: per ogni turno di visita, le liste dei nominativi dei visitatori in ordine alfabetico (cognome, nome) e per ciascuno il prezzo del biglietto (adulto paga intero, senior gratis, studente: minore di 18 gratis, altrimenti paga 50%). Infine, il programma deve visualizzare il numero totale su tutti i turni di prenotazioni accettate a prezzo intero, a metà prezzo e gratis, oltre che il prezzo medio del biglietto. Usate lo schema seguente per la visualizzazione.

```
Prenotazioni a prezzo intero:  ...
Prenotazioni a meta' prezzo:  ...
Prenotazioni gratis:  ...
Costo medio per prenotazione:  ...
```

Per testare il *main* usate il file `prenotazioni.txt` allegato a questo tema d'esame.

2 Consegna

Ricordo che le classi devono essere tutte *public* e che vanno consegnati tutti i file *.java* prodotti. NON vanno consegnati i *.class* Per la consegna, eseguite l'upload dei singoli file sorgente dalla pagina web: http://upload.dico.unimi.it.