LabProg 2010-2011 - TESTO ESAME
Andrea Trentini - DICo - UniMi
7 febbraio 2011

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

1 Testo esame

Lo scopo e' modellare e realizzare un contesto di gestione di un cinema monosala con assegnazione dei posti a sedere. Le classi da realizzare sono le seguenti:

- *Cinema*: la sala di proiezione
- *Spettatore*: uno spettatore (connesso con il Biglietto)
- *Posto*: un posto a sedere, numerato (fila, nr.)
- *Biglietto*: tariffa pagata per un film (titolo), ha anche info sul Posto assegnato
- *Main*, contiene il solo metodo *main()* con alcune invocazioni di test (importante e' riuscire a testare TUTTI i metodi almeno una volta)

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. Creare opportunamente i metodi di accesso agli attributi (set&get). Si suggerisce, anche dove non segnalato, di utilizzare, se esistenti e se applicabili, le classi parametriche (es. *Vector<E>* invece di *Vector*). 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 Cinema

Contiene degli attributi (di tipo opportuno) per memorizzare titolo del film, orario e la "mappa dei posti disponibili" (suggerimento: *insieme* di istanze di Posto). Oltre ai costruttori opportuni deve esporre i metodi **pubblici** di accesso ai dati, con controlli di coerenza. Deve inoltre esporre i seguenti metodi **pubblici**:

- *Posto getPosto(... fila, ... nr)*: accede ad un Posto (per far accomodare uno spettatore, per vedere se il posto e' libero, etc.)
- *int postiLiberi()*: quanti posti sono ancora liberi?
- *int postiOccupati()*: quanti sono gia' occupati?
- *int incasso()*: totale pagato dagli spettatori al momento presenti
- *Posto getPostoLibero()*: fornisce un posto libero (non importa l'algoritmo di scelta, basta che il posto sia effettivamente libero)

Nota bene: dove mancano i tipi (indicati con "...") e' lo studente che deve decidere cosa usare. Suggerimento per chi vuole dimostrare particolare abilita' programmativa: implementare un metodo *iteratore()* che fornisca un iteratore dei posti nel cinema. Non impelagatevi in questo punto se non vi sentite sicuri di voi!

1.2 Spettatore

Contiene degli attributi (di tipo opportuno) per memorizzare un nickname e un indirizzo di mail. Oltre ai costruttori opportuni deve esporre i metodi **pubblici** di accesso ai dati, con controlli di coerenza. Deve inoltre esporre i seguenti metodi **pubblici**:

- *void compraBiglietto(Biglietto b)*: assegna un biglietto
- *void siede(Posto p)*: lo spettatore si siede in un posto, se e' libero e se il biglietto corrisponde al film (attenzione: non e' banale, dipende da come viene implementato Posto)

1.3 Posto

Contiene degli attributi (di tipo opportuno) per memorizzare fila e numero. Suggerimento: e' probabile che serva anche un reference al Cinema per implementare altri metodi sparsi qua e la' nel presente testo. Oltre ai costruttori opportuni deve esporre i metodi **pubblici** di accesso ai dati, con controlli di coerenza. Deve esporre i seguenti metodi **pubblici**:

- *void siede(Spettatore s)*: fa sedere uno spettatore (non accetta se il posto e' gia' occupato, lo spettatore deve avere il biglietto per quel posto)

1.4 Biglietto

Contiene degli attributi (di tipo opportuno) per memorizzare titolo del film, orario, posto assegnato. Oltre ai costruttori opportuni deve esporre i metodi **pubblici** di accesso ai dati, con controlli di coerenza.

1.5 Main

Deve contenere il metodo *main()* in cui vanno istanziate le classi realizzate e opportunamente testate, ad es. creando almeno un cinema (opportunamente configurato), degli spettatori, etc. e popolando le istanze per avere una situazione reale a regime.

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.