LABORATORIO DI PROGRAMMAZIONE 2010-2011
PROVA D'ESAME

Andrea Trentini - DICo - UniMi

19 gennaio 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 aule semi-informatizzate per dei corsi. Le aule hanno una capienza che va rispettata e sono dotate di strumenti di supporto (PC, proiettori).
Le classi da realizzare sono le seguenti:

- *Aula*, rappresenta un'aula
- *Persona*, rappresenta una persona generica (sara' una superclasse di altre seguenti)
- *Studente*, rappresenta uno studente, ha una matricola
- *Docente*, rappresenta un docente, contiene info sulla materia insegnata
- *Proiettore*, rappresenta un proiettore, va connesso ad un PC (vedi seguente)
- *PC*, rappresenta un PC, gestisce la connessione con i proiettori
- *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 nelle parentesi, 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).

1.1 Aula

Gestisce capienza (max) e minimo di persone prenotate per poter far partire il corso. Deve contenere un elenco delle persone prenotate e un elenco dei presenti.
Deve esporre i seguenti metodi **pubblici**:

- *void setCapienza(int cap)*, imposta la capienza massima
- *void setMinimo(int min)*, imposta il minimo per far partire il corso
- *boolean raggiuntoMinimo()*, e' stato raggiunto il minimo?
- *double percentualePrenotati()*, percentuale di prenotati su capienza

- *double percentualePresenti()*, percentuale di presenti su capienza
- *void assegnaPC(PC pc)*, assegna un PC
- *void assegnaDocente(Docente doc)*, assegna un docente
- *void prenota(Persona p)*, prenota una certa persona su quest'aula (si era gia' prenotato? raggiunta capienza? etc.)
- *void annullaPrenota(Persona p)*, disdice certa persona su quest'aula (si era prenotato?)
- *void entra(Studente s)*, entra un corsista (si era prenotato?)
- *void esce(Studente s)*, esce un corsista (era entrato?)

1.2 Persona

Contiene degli attributi (di tipo opportuno) per memorizzare nome, cognome e indirizzo di mail.
Oltre ai costruttori opportuni deve esporre i metodi **pubblici** di accesso ai dati, con controlli di coerenza.

1.3 Studente

Estende *Persona* aggiungendo la parte di gestione della matricola.
Oltre ai costruttori opportuni deve esporre i metodi **pubblici** di accesso ai dati, con controlli di coerenza.

1.4 Docente

Estende *Persona* aggiungendo la parte di gestione della materia insegnata.
Oltre ai costruttori opportuni deve esporre i metodi **pubblici** di accesso ai dati, con controlli di coerenza.

1.5 PC

Contiene degli attributi (di tipo opportuno) per gestire le connessioni con i proiettori.
Oltre agli opportuni costruttori deve esporre i seguenti metodi **pubblici**:

- *void connetti(Proiettore p)*, connette un proiettore (era gia' stato connesso?)
- *void disconnetti(Proiettore p)*, disconnette un proiettore (era stato connesso prima?)
- *double potenzaTotale()*, restituisce la potenza totale consumata dai proiettori attualmente connessi (in Watt)

1.6 Proiettore

Contiene degli attributi (di tipo opportuno) per memorizzare la potenza consumata dalla lampada.
Oltre agli opportuni costruttori deve esporre i seguenti metodi **pubblici**:

- *double potenza()*, restituisce la potenza consumata (in Watt)

1.7 Main

Deve contenere il metodo *main()* in cui vanno istanziate le classi realizzate e opportunamente testate, ad es. creando almeno un'aula, degli studenti, 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.

QUESTA
SEZIONE
PRESEN
PER CO
PLETEZ