Laboratorio di Programmazione 2004/2005

Docente: Luca Tesei

Testo del progetto di esame

1 Descrizione del dominio di applicazione e delle funzionalità richieste

Si realizzi un'applicazione Java per la gestione delle prenotazioni di un insieme di aule da parte di una comunità.

Ogni aula è dotata di diversi attributi fra cui: numero dei posti, presenza di un sistema di proiezione, presenza di un sistema di videoconferenza.

Un utente (docente, tutor, tecnico, esterno) può chiedere al sistema di prenotare:

- una certa aula
- in un certo giorno
- per una certa ora singola
- per una certa attività (deve essere fornita una descrizione)

Il sistema deve fare in modo che non ci siano sovrapposizioni fra le prenotazioni.

Le aule possono essere prenotate solo nei giorni feriali (sabato escluso) e solo a partire da una certa ora "intera" (non sono ammesse cioè prenotazioni che ad esempio partono alle 9:30 - ma solo alle 9:00 o 10:00 etc.)

In ogni giornata le aule sono fruibili dalle ore 8:00 del mattino fino alle ore 19:00 della sera. Non possono essere accettate prenotazioni fuori da questo orario.

Un utente, inoltre, può richiedere sempre al sistema di cancellare una certa ora - in una certa aula, in un certo giorno - delle sue prenotazioni.

Il sistema, per aiutare il richiedente a fare le sue prenotazioni, deve anche fornire la possibilità di fare ricerche con diversi criteri. In particolare il sistema deve permettere di:

- Cercare, se c'è, un'aula libera:
  - in un dato giorno
  - per un certo numero di ore consecutive o no (se nella richiesta viene indicato che si desiderano ore consecutive allora tale vincolo deve essere rispettato, altrimenti non c'è nessun vincolo: le ore possono essere consecutive o no)
  - il cui numero di posti sia maggiore uguale a un certo numero dato
  - i cui attributi soddisfino alcune richieste date sulla presenza di strumenti didattici (videoconferenza etc.)
- Visualizzare tutte le prenotazioni di una data aula in un dato giorno

- Visualizzare tutte le prenotazioni fatte da un dato utente nell'arco di tempo individuato da due date fornite.

I risultati delle ricerche devono fornire la visualizzazione di tutte le prenotazioni dell'aula trovata, nel giorno in questione, oltre all'indicazione delle ore libere oggetto della ricerca.

Coloro che sostengono un esame da 6 CFU devono inoltre implementare le seguenti funzionalità:

- Uno stesso utente può prenotare, dopo sua conferma esplicita alla segnalazione di sovrapposizione che deve essere data dal sistema, una stessa aula in uno stesso orario per attività diverse
- Il sistema deve permettere di cercare, se c'è, la **prima** aula libera in una certa settimana (quindi se ce n'è una di lunedì il sistema deve indicare quella) per un certo numero di ore anche non consecutive ma nello stesso giorno:
  - il cui numero di posti sia maggiore uguale a un certo numero dato
  - i cui attributi soddisfino alcune date richieste sulla presenza di strumenti didattici (videoconferenza etc.)

Per quanto riguarda quest'ultima ricerca se due o più aule soddisfano la richiesta per lo stesso giorno (e questo giorno è il primo, nella settimana data, che ha almeno un'aula che soddisfa le richieste) non è importante scegliere quella le cui ore libere iniziano prima delle altre: può esserne scelta una qualsiasi.

Sia gli studenti che devono ottenere 5 CFU, sia quelli che ne devono ottenere 6, devono fornire una classe Test che illustri tutte le funzionalità implementate.

2 Suggerimenti di progettazione e implementazione

La prima fase di approccio alla realizzazione dell'applicazione richiesta è, come sappiamo, un'adeguata progettazione secondo le caratteristiche tipiche della programmazione ad oggetti.

In particolare è della massima importanza definire le classi che si vogliono scrivere e il loro esatto significato. In questa fase è bene riflettere bene sui concetti che si vogliono rappresentare e badare che ogni classe rappresenti un singolo concetto.

Per quanto riguarda l'applicazione da realizzare nel progetto il suggerimento è quello di individuare, tra eventuali altre, le seguenti classi:

`Utente` Gli oggetti di questa classe sono le persone che hanno accesso al sistema per le prenotazioni, le cancellazioni e le ricerche

`Aula` Gli oggetti di questa classe sono le aule che si possono prenotare, con i loro attributi

`GestoreAule` Un oggetto di questa classe deve mantenere una collezione di un certo numero di aule, permettendo l'aggiunta di nuove aule. Inoltre potrebbe essere in grado di ricercare, su richiesta, aule che hanno certe caratteristiche (numero di posti, presenza di macchinari, etc.)

`Prenotazione` Un oggetto di questa classe rappresenta una prenotazione di una certa aula, da parte di un certo utente, in un certo giorno per una certa ora **singola** (è bene che la granularità delle prenotazioni sia questa: in questo modo risulta più semplice la gestione di prenotazioni con ore consecutive e non)

`DatabasePrenotazioni` Un oggetto di questa classe mantiene una collezione di prenotazioni effettuate. Deve permettere l'inserimento di nuove prenotazioni (controllando che non ci siano sovrapposizioni) e la cancellazione di prenotazioni (controllando che l'utente che cancella sia colui che ha effettuato la prenotazione)

`GestoreRicerche` Un oggetto di questa classe, attraverso un certo oggetto di `DatabasePrenotazioni`, mette a disposizione dei metodi per effettuare le ricerche e visualizzazioni previste

`CaratteristicheAula` Un oggetto di questa classe può essere utile per rappresentare un tipo di attributi richiesti ad un aula durante una ricerca. Potrebbe implementare facilmente un metodo `boolean satisfies(Aula a)` che indica se la data aula soddisfa i criteri in esso definiti.

`Data` Un oggetto di questa classe rappresenta una certa data: giorno mese e anno

`DataEOra` Un oggetto di questa classe rappresenta una certa ora di un certo giorno.

Una volta individuate le classi è molto importante definire la loro interfaccia pubblica. Vanno specificati tutti i metodi con i loro parametri e il loro tipo di ritorno (in questo modo si ha la possibilità di vedere se i concetti sono stati ben modellati e se ogni classe ha la possibilità, così com'è, di effettuare tutte le operazioni a lei richieste). Si valutino anche le dipendenze e il grado di accoppiamento delle classi della propria applicazione e si cerchi di portare questi parametri a valori adeguati.

Per quanto riguarda la scrittura della classe Test sarebbe auspicabile che essa preveda di prendere una certa porzione di input (un certo numero di Aule, Utenti, Prenotazioni predefinite) da uno o più file di testo. In questo modo può essere creata velocemente una situazione tipica e abbastanza dettagliata in cui l'applicazione si trova ad operare. Poi possono essere effettuati esempi di ricerche, prenotazioni e cancellazioni.

Un consiglio: scrivere la classe Test per ultima e con in mente il tipo di presentazione che si vuole fare il giorno dell'orale. Durante lo sviluppo dell'applicazione testare le varie classi utilizzando Bluej.

Per quanto riguarda la gestione delle date e del calendario consultare le API della classe `java.util.GregorianCalendar` che permette, fra le altre cose, di calcolare se un certo giorno è feriale o no.

È bene cercare di dotare le classi `DatabasePrenotazioni` e `GestoreRicerche` di adeguati metodi che riflettano esattamente la struttura delle ricerche, prenotazioni, cancellazioni richieste. Tuttavia è auspicabile una sintesi: utilizzare gli oggetti della classe `CaratteristicheAula` e/o definirne di simili per limitare il numero di metodi delle classi di cui sopra.

3 Criteri di valutazione

Il progetto sarà valutato in base ai seguenti criteri, in ordine decrescente di importanza:

- L'applicazione fa ciò che è stato richiesto

- La struttura dell'applicazione riflette un'adeguata progettazione secondo i principi della programmazione ad oggetti
- Il codice è ampiamente commentato e documentato. Tutte le classi pubbliche sono dotate di documentazione per la generazione di pagine html con `javadoc`
- L'applicazione gestisce anche casi particolari ed eventuali situazioni di "errore" o incongruenza
- L'applicazione fa un uso intelligente delle risorse (efficienza e risparmio di spazio di memoria)
- Interfaccia dell'applicazione.

Si ricordi che:

- non è richiesta l'implementazione di una interfaccia grafica. Tuttavia, se questa viene fornita verrà valutata.
- Coloro che utilizzano particolari ambienti di programmazione (es JavaBeans, J++, etc.) devono essere comunque in grado di estrapolare i file sorgente .java delle classi che rappresentano il nucleo dell'applicazione (non importano i sorgenti automaticamente generati ad esempio da generatori visuali di interfacce grafiche)
- Il progetto va spedito come file .zip contenente tutti i sorgenti: l'applicazione deve poter essere compilata attraverso il solo uso del compilatore standard della Sun e delle librerie standard di Java (Java 2 SDK 1.4 o superiore)
- Va allegato ai sorgenti un file `README.txt` che descriva a grandi linee il tipo di approccio che si è seguito, la struttura delle classi individuata attraverso la progettazione ed eventuali dettagli di implementazione che si vogliono mettere in evidenza