Relazione per il corso di Linguaggi di Programmazione: Metodologie di Programmazione

Dario Di Minno Claudio Tortorelli

21 marzo 2003

Indice

1 Introduzione **3**
1.1 Descrizione del problema 3

2 Analisi **4**
2.1 Casi d'uso 4
2.2 Specifiche dell'applicazione 5

3 Progettazione **6**
3.1 Uso di pattern 6
3.2 Descrizione del sistema e interazioni fra le classi 6
3.3 Diagramma UML 7

4 Implementazione **9**
4.1 Scelte implementative 9
4.2 Documentazione delle classi 9
4.2.1 AccessoGruppi 9
4.2.2 AssociaGruppi 10
4.2.3 CancellaGruppo 10
4.2.4 CancellaUtente 11
4.2.5 Command 11
4.2.6 CreaGruppo 12
4.2.7 DisassociaGruppo 12
4.2.8 IscrizioneUtenti 13
4.2.9 LeggiListaTutor 13
4.2.10 LeggiListaUtenti 14
4.2.11 MostraDomanda 14
4.2.12 MostraGruppiAssociati 15
4.2.13 MostraRisposte 16
4.2.14 PoniDomanda 16
4.2.15 PoniRisposta 17
4.2.16 VisualizzaUtente 17
4.2.17 AccessoGruppiMenuItem 18
4.2.18 AssociaGruppiMenuItem 18
4.2.19 CancellaGruppoMenuItem 19
4.2.20 CancellaUtenteMenuItem 19
4.2.21 CreaGruppoMenuItem 20
4.2.22 DisassociaGruppoMenuItem 20
4.2.23 InfoUtenteMenuItem 21
4.2.24 IscrizioneUtentiMenuItem 22
4.2.25 MenuItem 22
4.2.26 MostraGruppiAssociatiMenuItem 23
4.2.27 Menu 23

4.2.28 MenuAmministratore 24
4.2.29 MenuTutor 25
4.2.30 MenuUtente 25
4.2.31 Disegnatore 26
4.2.32 Forum 27
4.2.33 GruppoDiscussione 28
4.2.34 Messaggio 29
4.2.35 Utente 30
4.2.36 Tutor 31
4.2.37 Amministratore 32
4.2.38 ListaUtentiForum 33
4.2.39 Sistema 35
4.3 Codice 36

1 Introduzione

Lo sviluppo di un'applicazione tipicamente prevede i seguenti passi:

- Analisi del problema
- Progettazione
- Implementazione

L'analisi del problema fornisce come risultato delle specifiche precise e non ambigue sulle quali basare la progettazione. Seguendo il paradigma della programmazione orientata agli oggetti, durante la progettazione vengono definite le *classi* e le relazioni che intercorrono tra di loro. Tale sistema viene rappresentato mediante un diagramma *UML* e implementato in un linguaggio orientato agli oggetti.

Partendo dal problema di realizzare un sistema di tutoraggio per studenti universitari, abbiamo affrontato i passi sopra descritti fino arrivare a un'effettiva implementazione.

1.1 Descrizione del problema

Un *servizio di tutoraggio per studenti universitari* è un sistema tramite il quale gli *studenti* possono usufruire dell'assistenza di un gruppo di *tutor*.

Ogni studente interessato può iscriversi facendone richiesta a un *amministratore*.

I servizi che sono forniti agli studenti possono essere diversi, come:

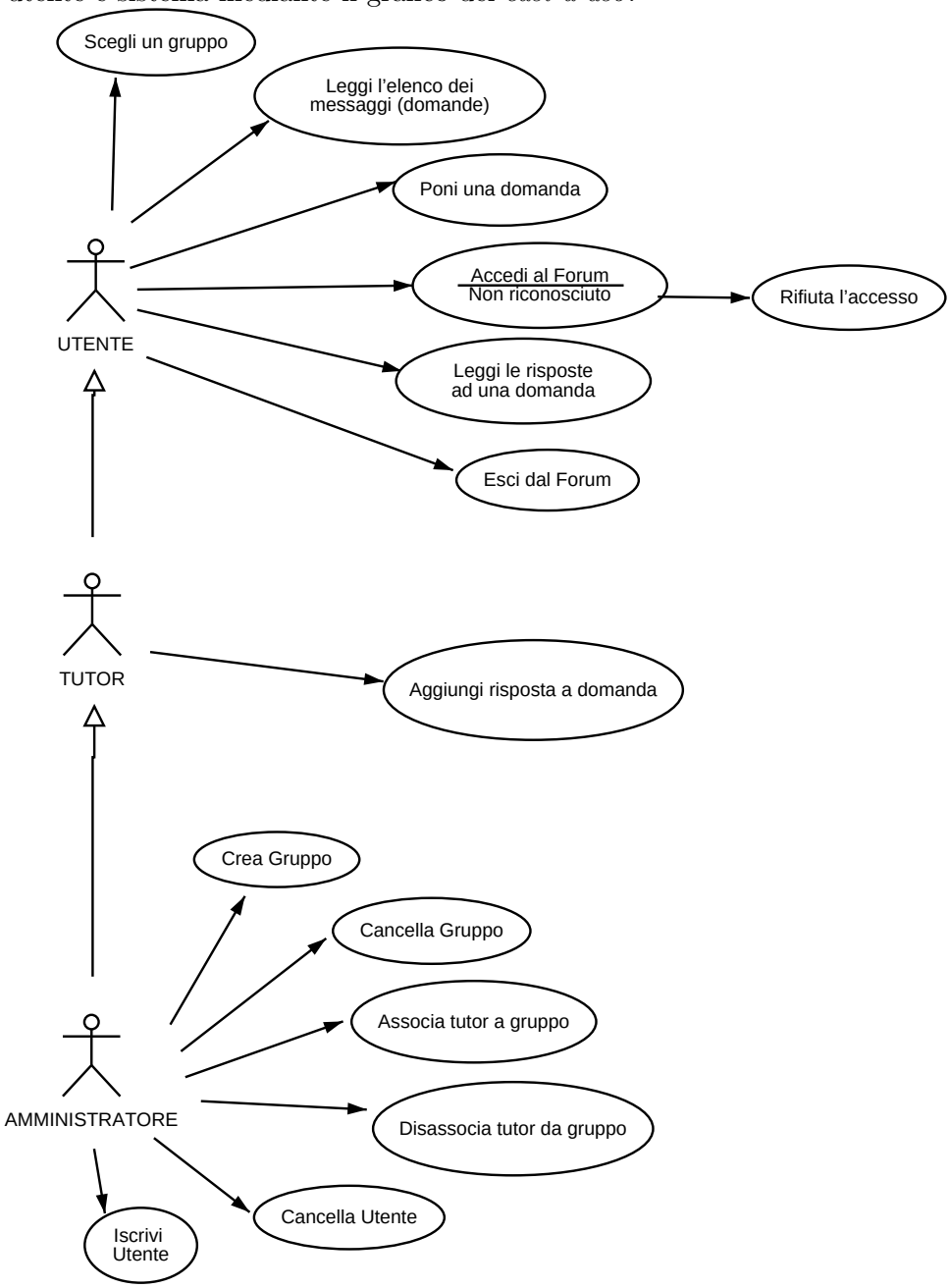
- un *forum* formato da più *gruppi di discussione*, uno per ogni materia.
- una bacheca
- posta elettronica
- una collezione di domande frequentemente poste
- e altro...

Nel nostro progetto abbiamo sviluppato il forum, in quanto parte fondamentale del sistema.

2 Analisi

2.1 Casi d'uso

Per scoprire i requisiti del sistema, abbiamo evidenziato le interazioni fra utente e sistema mediante il grafico dei *casi d'uso*:



2.2 Specifiche dell'applicazione

Dalla nostra analisi, sono derivate le seguenti specifiche:

Esistono tre tipi di utenti: Utente, Tutor, Amministratore. Ogni utente è descritto dalle sue informazioni personali, più una login e una password per essere riconosciuto dal sistema. Una volta entrato nel sistema, un utente semplice può accedere al forum, leggere le domande poste nei gruppi di discussione e porre nuove domande. I tutor sono degli utenti con la possibilità di rispondere alle domande poste nei gruppi ai quali sono associati. Gli amministratori sono tutor con la facoltà di:

- aggiungere o cancellare gruppi di discussione del forum
- gestire le associazioni fra tutor e gruppi
- gestire le iscrizioni degli utenti

Se il sistema non possiede utenti, al suo avvio crea un amministratore con login 'root' e password 'root'. Questo primo utente creerà poi gli altri.

3 Progettazione

3.1 Uso di pattern

L'utilizzo di un pattern influenza significativamente la progettazione di un sistema, o perlomeno di una sua parte. Abbiamo fatto uso del pattern Command, che ci ha suggerito le linee guida per la realizzazione del menù, che è composto da voci di menù che, se selezionate, fanno partire l'esecuzione di uno o più comandi.

L'utilizzo di questo pattern quindi ci ha portato a definire le classi Menù, MenuItem (le voci del menù) e Command. I comandi sono realizzati come classi che hanno la responsabilità dell'esecuzione del comando che implementano. Il menù viene istanziato in forme diverse a seconda del tipo di utente che accede al sistema, abilitando o meno alcune operazioni.

3.2 Descrizione del sistema e interazioni fra le classi

La classe **Sistema** ha un ruolo fondamentale nell'applicazione, in quanto crea gli oggetti che sono alla base del suo funzionamento. In particolare, questi oggetti sono:

- Forum
- ListaUtentiForum
- Disegnatore
- Menu

Inoltre il sistema dopo aver autorizzato un utente registrato a utilizzare il forum, delega agli oggetti che ha creato il recupero delle informazioni memorizzate in precedenza sui file.

Il **Forum** contiene i riferimenti ai gruppi di discussione e fornisce agli oggetti di classe Command l'accesso ai messaggi memorizzati. Tutta la gestione dei gruppi di discussione passa attraverso questa classe.

Un **GruppoDiscussione** è formato da un insieme di messaggi, strutturato come una sequenza di domande seguite da eventuali risposte. Le domande e le risposte sono tutte istanze della classe **Messaggio**. E' compito di GruppoDiscussione mantenere su file i messaggi in modo ordinato.

ListaUtentiForum ha i riferimenti agli utenti registrati nel sistema e dunque ha accesso a tutti i loro dati, permettendo il loro utilizzo alle classi che ne fanno richiesta. Questa classe si preoccupa anche di garantire la persistenza dei dati di cui è responsabile.

L'utente semplice è rappresentato dalla classe **Utente**. Da questa ereditano le classi **Tutor** e **Amministratore**, che hanno in più dei riferimenti a un insieme di gruppi per i quali hanno diritto a rispondere. Le operazioni

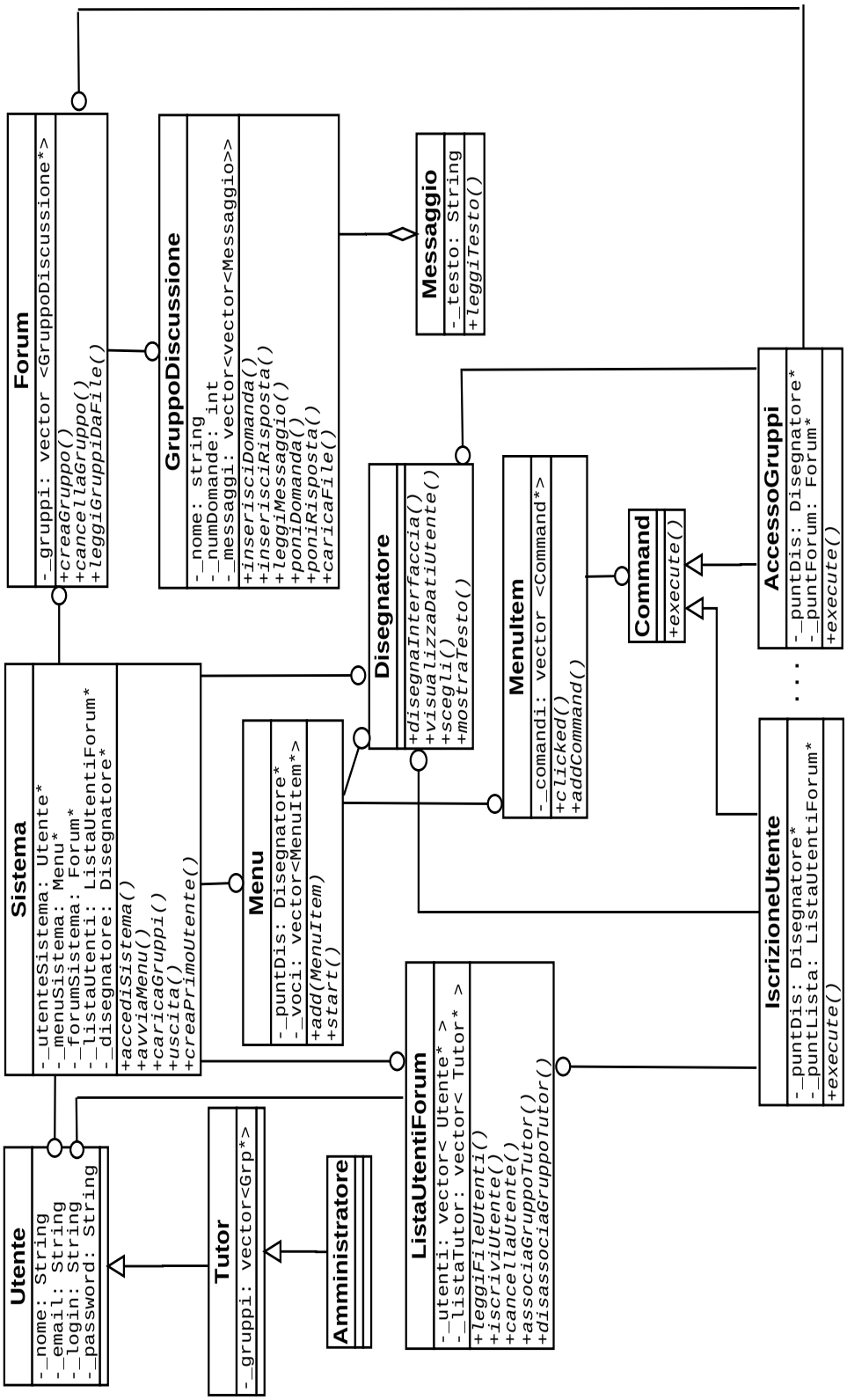
di responsabilità dell'amministratore vengono rese disponibili dal Menu che viene a lui istanziato.

Alla classe **Disegnatore** vengono indirizzate le richieste di visualizzazione dell'output in modo da dare un aspetto omogeneo all'interfaccia utente.

La classe **Menu** possiede i riferimenti ai MenuItem, ovvero le voci di menu che possono presentarsi agli utenti. I MenuItem vengono istanziati dal costruttore di questa classe a seconda dell'utente che sta utilizzando il sistema. I **MenuItem** a loro volta hanno i riferimenti a un insieme di **Command** che implementano i comandi associati alle voci di menu disponibili. Selezionare un MenuItem equivale quindi a eseguire in maniera ordinata le Command a cui fa riferimento. Tutti i comandi sono implementati nel metodo *execute()* di classi derivate da Command. Queste ultime, per eseguire i propri compiti, dispongono dei riferimenti alle classi principali del sistema (Forum, ListaUtentiForum, Disegnatore).

3.3 Diagramma UML

Il diagramma UML risultante dalla progettazione e dalle sue successive revisioni è il seguente:


Forum
-_gruppi: vector <GruppoDiscussione*>
+creaGruppo()
+cancellaGruppo()
+leggiGruppiDaFile()
GruppoDiscussione
-_nome: string
-_numDomande: int
-_messaggi: vector<vector<Messaggio>>
+inserisciDomanda()
+inserisciRisposta()
+leggiMessaggio()
+poniDomanda()
+poniRisposta()
+caricaFile()
Messaggio
-_testo: String
+leggiTesto()
Sistema
-_utenteSistema: Utente*
-_menuSistema: Menu*
-_forumSistema: Forum*
-_listaUtenti: ListaUtentiForum*
-_disegnatore: Disegnatore*
+accediSistema()
+avviaMenu()
+caricaGruppi()
+uscita()
+creaPrimoUtente()
Disegnatore
+disegnaInterfaccia()
+visualizzaDatiUtente()
+scegli()
+mostraTesto()
MenuItem
-_comandi: vector <Command*>
+clicked()
+addCommand()
Command
+execute()
AccessoGruppi
-_puntDis: Disegnatore*
-_puntForum: Forum*
+execute()
Menu
-_puntDis: Disegnatore*
-_voci: vector<MenuItem*>
+add(MenuItem)
+start()
Utente
-_nome: String
-_email: String
-_login: String
-_password: String
Tutor
-_gruppi: vector<Grp*>
Amministratore
ListaUtentiForum
-_utenti: vector< Utente* >
-_listaTutor: vector< Tutor* >
+leggiFileUtenti()
+iscriviUtente()
+cancellaUtente()
+associaGruppoTutor()
+disassociaGruppoTutor()
IscrizioneUtente
-_puntDis: Disegnatore*
-_puntLista: ListaUtentiForum*
+execute()

4 Implementazione

4.1 Scelte implementative

Compilatore utilizzato Il linguaggio di programmazione da noi usato per scrivere il codice del progetto è il C++, in particolare il compilatore *mingw*, ovvero il porting su sistema operativo *Windows* e piattaforma Intel x86 del famoso *gcc* della GNU. In particolare ci siamo avvalsi dell'ambiente di sviluppo gratuito DEV-C++ (www.bloodshed.net/dev), versione 4.9.6.0.

Librerie utilizzate Abbiamo fatto uso di librerie ANSI-C++ per la gestione di vector, stringhe, stream, date, e della libreria conio per la gestione dell'output su schermo.

4.2 Documentazione delle classi

Abbiamo suddiviso le classi nelle seguenti categorie: Command, Menu, MenuItem, Utenti, Forum, Sistema, Disegnatore, ListaUtentiForum. Segue la loro descrizione.

Command

4.2.1 AccessoGruppi

File:
AccessoGruppi.h, AccessoGruppi.cpp

Descrizione
Questa classe eredita da Command e serve a visualizzare e scegliere il gruppo al quale accedere tra l'elenco dei gruppi di discussione presenti nel sistema.

Costruttore

- `AccessoGruppi(Forum* f, Disegnatore* d, int* sceltaGruppo)`

Il costruttore di questa command richiede in input il riferimento al Forum e al Disegnatore correnti, oltre al riferimento ad un intero che è il valore di scambio tra le varie command di questo MenuItem. Nel caso della AccessoGruppi questo intero assume significato di 'sceltaGruppo'.

Metodi

- `virtual void execute()`

Esegue il comando rappresentato.

4.2.2 AssociaGruppi

File
AssociaGruppi.h, AssociaGruppi.cpp

Descrizione
Questa classe eredita da Command e serve ad associare un particolare gruppo ad un Tutor. Questa operazione permette poi al Tutor di poter rispondere a domande di quel gruppo.

Costruttore

- `AssociaGruppi(Forum* f, Disegnatore* d, ListaUtentiForum* l, int* numeroTutor)`

Il costruttore di questa command richiede in input il riferimento al Forum, al Disegnatore e alla ListaUtentiForum correnti. Inoltre vuole il riferimento ad un intero che è il valore di scambio tra le varie command di questo MenuItem. Nel caso della AssociaGruppi questo intero assume significato di 'numeroTutor', cioè l'indice del particolare tutor nella lista dei tutor.

Metodi

- `virtual void execute()`

Esegue il comando rappresentato.

4.2.3 CancellaGruppo

File
CancellaGruppo.h, CancellaGruppo.cpp

Descrizione
Questa classe eredita da Command e serve ad eliminare un particolare gruppo dal sistema.

Costruttore

- `CancellaGruppo(Forum* f, Disegnatore* d, int* numGruppo)`

Il costruttore di questa command richiede in input il riferimento al Forum e al Disegnatore correnti. Inoltre vuole il riferimento ad un intero che è il valore di scambio tra le varie command del MenuItem di cui CancellaGruppo fa parte. Nel caso della CancellaGruppi questo intero assume significato di 'numGruppo', cioè l'indice del particolare gruppo di discussione da eliminare.

Metodi

- `virtual void execute()`

Esegue il comando rappresentato.

4.2.4 CancellaUtente

File
CancellaUtente.h, CancellaUtente.cpp

Descrizione
Questa classe eredita da Command e serve ad eliminare un particolare utente dal sistema, eccetto lo stesso utente che ne fa richiesta.

Costruttore

- `CancellaUtente(Disegnatore* d, ListaUtentiForum* l, int* numUtente)`

Il costruttore di questa command richiede in input il riferimento al Disegnatore e alla ListaUtentiForum correnti. Inoltre vuole il riferimento ad un intero che è il valore di scambio tra le varie command del MenuItem di cui CancellaUtente fa parte. Nel caso di CancellaUtente questo intero assume significato di 'numUtente', cioè l'indice del particolare utente da eliminare dalla lista di utenti.

Metodi

- `virtual void execute()`

Esegue il comando rappresentato.

4.2.5 Command

File
Command.h, Command.cpp

Descrizione
Command è la classe base con funzione di interfaccia per tutte le command derivate, che hanno compiti specifici.

Costruttore

- `Command()`

Costruttore senza argomenti.

Distruttore

- `virtual ~Command()`

Distruttore virtuale: permette di chiamare distruttori specifici delle classi derivate.

Metodi

- `virtual void execute() = 0`

E' un metodo virtuale puro che implica l'implementazione nelle classi derivate da command.

4.2.6 CreaGruppo

File
CreaGruppo.h, CreaGruppo.cpp

Descrizione
Questa classe eredita da Command e serve a creare un nuovo gruppo di discussione.

Costruttore

- `CreaGruppo(Forum* f, Disegnatore* d)`

Il costruttore di questa command richiede in input il riferimento al Disegnatore e al Forum correnti.

Metodi

- `virtual void execute()`

Esegue il comando rappresentato.

4.2.7 DisassociaGruppo

File
DisassociaGruppo.h, DisassociaGruppo.cpp

Descrizione
Questa classe eredita da Command e serve ad eliminare l'associazione di un particolare tutor con un certo gruppo di discussione. Questa operazione comporta che il tutor non potrà più rispondere a domande di quel gruppo.

Costruttore

- `DisassociaGruppo(Forum* f, Disegnatore* d, ListaUtentiForum* l, int* numeroTutor)`

Il costruttore di questa command richiede in input il riferimento al Forum, al Disegnatore e alla ListaUtentiForum correnti. Inoltre vuole il riferimento ad un intero che è il valore di scambio tra le varie command del MenuItem di cui DisassociaGruppo fa parte. Nel caso di DisassociaGruppo questo intero assume significato di 'numeroTutor', cioè l'indice del particolare tutor al quale rimuovere un gruppo associato.

Metodi

- `virtual void execute()`

Esegue il comando rappresentato.

4.2.8 IscrizioneUtenti

File
IscrizioneUtenti.h, IscrizioneUtenti.cpp

Descrizione
Questa classe eredita da Command e serve ad iscrivere un nuovo utente, specificando i suoi campi e il tipo di appartenenza ('utente', 'tutor', 'amministratore').

Costruttore

- `IscrizioneUtenti(ListaUtentiForum* l, Disegnatore* d)`

Il costruttore di questa command richiede in input il riferimento a ListaUtentiForum e al Disegnatore correnti.

Metodi

- `virtual void execute()`

Esegue il comando rappresentato.

4.2.9 LeggiListaTutor

File
LeggiListaTutor.h, LeggiListaTutor.cpp

Descrizione
Questa classe eredita da Command e serve a scorrere l'elenco dei tutor registrati nel sistema, selezionandone uno in particolare.

Costruttore

- `LeggiListaTutor(ListaUtentiForum* l, Disegnatore* d, int* scelta)`

Il costruttore di questa command richiede in input il riferimento a ListaUtentiForum e al Disegnatore correnti. E' inoltre necessario il riferimento ad un intero che all'interno della classe sarà inizializzato con l'indice del tutor selezionato.

Metodi

- `virtual void execute()`

Esegue il comando rappresentato.

4.2.10 LeggiListaUtenti

File
LeggiListaUtenti.h, LeggiListaUtenti.cpp

Descrizione
Questa classe eredita da Command e serve a scorrere l'elenco degli utenti registrati nel sistema, selezionandone uno in particolare.

Costruttore

- `LeggiListaUtenti(ListaUtentiForum* l, Disegnatore* d, int* scelta)`

Il costruttore di questa command richiede in input il riferimento a ListaUtentiForum e al Disegnatore correnti. E' inoltre necessario il riferimento ad un intero che all'interno della classe sarà inizializzato con l'indice dell'utente selezionato.

Metodi

- `virtual void execute()`

Esegue il comando rappresentato.

4.2.11 MostraDomanda

File
MostraDomanda.h, MostraDomanda.cpp

Descrizione
Questa classe eredita da Command e serve a scorrere l'elenco delle domande di un gruppo, selezionandone una in particolare.

Costruttore

- `MostraDomanda(Forum* f, Disegnatore* d, int* numGruppo, int* numDomanda, bool* gruppoVuoto)`

Il costruttore di questa command richiede in input il riferimento a Forum e al Disegnatore correnti. Sono inoltre necessari i riferimenti a due interi e a un booleano. Il primo intero informa sui risultati di command precedenti e comunica in quale gruppo ricercare le domande, il secondo memorizza l'indice della domanda selezionata. Il booleano informa all'esterno command seguenti se il gruppo selezionato non ha domande.

Metodi

- `virtual void execute()`

Esegue il comando rappresentato.

4.2.12 MostraGruppiAssociati

File
MostraGruppiAssociati.h, MostraGruppiAssociati.cpp

Descrizione
Questa classe eredita da Command e serve a scorrere l'elenco dei gruppi associati ad un certo Tutor.

Costruttore

- `MostraGruppiAssociati(Disegnatore* d, ListaUtentiForum* l, int* numeroTutor)`

Il costruttore di questa command richiede in input il riferimento al Disegnatore e alla ListaUtentiForum correnti. E' inoltre necessario il riferimento a un intero contenente il numero del tutor del quale si vogliono verificare i gruppi associati.

Metodi

- `virtual void execute()`

Esegue il comando rappresentato.

4.2.13 MostraRisposte

File
MostraRisposte.h, MostraRisposte.cpp

Descrizione
Questa classe eredita da Command e serve a visualizzare l'elenco delle risposte associate ad una data domanda di un dato gruppo di discussione.

Costruttore

- `MostraRisposte(Forum* f, Disegnatore* d, int* numGruppo, int* numDomanda)`

Il costruttore di questa command richiede in input il riferimento al Forum e al Disegnatore correnti. E' inoltre necessario il riferimento a due interi contenenti il numero del gruppo e l'indice della domanda della quale si vogliono verificare le risposte.

Metodi

- `virtual void execute()`

Esegue il comando rappresentato.

4.2.14 PoniDomanda

File
PoniDomanda.h, PoniDomanda.cpp

Descrizione
Questa classe eredita da Command e serve a inserire una nuova domanda in un dato gruppo discussione.

Costruttore

- `PoniDomanda(Forum* f, Disegnatore* d, ListaUtentiForum* l, int* numGruppo, bool* gruppoVuoto)`

Il costruttore di questa command richiede in input il riferimento al Forum, al Disegnatore e alla ListaUtentiForum correnti. Inoltre sono necessari i riferimenti ad altri due valori: uno intero contenente il numero del gruppo e uno booleano che informa se il gruppo e' vuoto.

Metodi

- `virtual void execute()`

Esegue il comando rappresentato.

4.2.15 PoniRisposta

File
PoniRisposta.h, PoniRisposta.cpp

Descrizione
Questa classe eredita da Command e serve a inserire una nuova risposta ad una data domanda di un gruppo di discussione. Solo i tutor che hanno tra i gruppi associati quello in cui vogliono rispondere sono abilitati a farlo.

Costruttore

- `PoniRisposta(Forum* f, Disegnatore* d, ListaUtentiForum* l, int* numGruppo, int* numDomanda)`

Il costruttore di questa command richiede in input il riferimento al Forum, al Disegnatore e alla ListaUtentiForum correnti. Inoltre sono necessari i riferimenti ad altri due valori interi: uno contenente il numero del gruppo e un altro che informa sull'indice della domanda alla quale si vuol rispondere.

Metodi

- `virtual void execute()`

Esegue il comando rappresentato.

4.2.16 VisualizzaUtente

File
VisualizzaUtente.h, VisualizzaUtente.cpp

Descrizione
Questa classe eredita da Command e serve a visualizzare a schermo i dati di un certo utente del sistema.

Costruttore

- `VisualizzaUtente(ListaUtentiForum* l, Disegnatore* d, int* numeroUtente)`

Il costruttore di questa command richiede in input il riferimento alla ListaUtentiForum e al Disegnatore correnti. Inoltre e' necessario il riferimento al valore intero contenente l'indice dell'utente del quale si vogliono leggere i dati.

Metodi

- `virtual void execute()`

Esegue il comando rappresentato.
MenuItem

4.2.17 AccessoGruppiMenuItem

File
AccessoGruppiMenuItem.h, AccessoGruppiMenuItem.cpp

Descrizione
L'AccessoGruppiMenuItem e' la classe che raccoglie le command necessarie alla visita e alla manipolazione delle informazioni contenute nei gruppi.

Costruttore

- `AccessoGruppiMenuItem()`
- `AccessoGruppiMenuItem(string testo, Forum* f, Disegnatore* d, ListaUtentiForum* l)`

Oltre al costruttore vuoto questa classe ammette un costruttore che prende in input una stringa di testo contenente il messaggio della voce di menu', il riferimento al Forum, al Disegnatore e alla ListaUtentiForum correnti.

Metodi

- Questa classe non ha metodi propri

4.2.18 AssociaGruppiMenuItem

File
AssociaGruppiMenuItem.h, AssociaGruppiMenuItem.cpp

Descrizione
L'AssociaGruppiMenuItem e' la classe che raccoglie le command necessarie all'associazione dei tutor ai gruppi discussione di cui sono responsabili.

Costruttore

- `AssociaGruppiMenuItem()`
- `AssociaGruppiMenuItem(string testo, Forum* f, Disegnatore* d, ListaUtentiForum* l)`

Oltre al costruttore vuoto questa classe ammette un costruttore che prende in input una stringa di testo contenente il messaggio della voce di menu', il riferimento al Forum, al Disegnatore e alla ListaUtentiForum correnti che saranno a loro volta passati alle command che ne hanno bisogno.

Metodi

- Questa classe non ha metodi propri

4.2.19 CancellaGruppoMenuItem

File
CancellaGruppoMenuItem.h, CancellaGruppoMenuItem.cpp

Descrizione
La CancellaGruppoMenuItem e' la classe che raccoglie le command necessarie all'eliminazione di un dato gruppo.

Costruttore

- `CancellaGruppoMenuItem()`
- `CancellaGruppoMenuItem(string testo, Disegnatore* d, Forum* f)`

Oltre al costruttore vuoto questa classe ammette un costruttore che prende in input una stringa di testo contenente il messaggio della voce di menu', il riferimento al Forum, al Disegnatore e alla ListaUtentiForum correnti che saranno a loro volta passati alle command che ne hanno bisogno.

Metodi

- Questa classe non ha metodi propri

4.2.20 CancellaUtenteMenuItem

File
CancellaUtenteMenuItem.h, CancellaUtenteMenuItem.cpp

Descrizione
La CancellaUtenteMenuItem e' la classe che raccoglie le command necessarie all'eliminazione di un dato gruppo.

Costruttore

- `CancellaUtenteMenuItem()`
- `CancellaUtenteMenuItem(string testo, Disegnatore* d, ListaUtentiForum* l)`

Oltre al costruttore vuoto questa classe ammette un costruttore che prende in input una stringa di testo contenente il messaggio della voce di menu', il riferimento al Disegnatore e alla ListaUtentiForum correnti che saranno a loro volta passati alle command che ne hanno bisogno.

Metodi

- Questa classe non ha metodi propri

4.2.21 CreaGruppoMenuItem

File
CreaGruppoMenuItem.h, CreaGruppoMenuItem.cpp

Descrizione
La CreaGruppoMenuItem e' la classe che raccoglie le command necessarie alla creazione di un nuovo gruppo di discussione.

Costruttore

- `CreaGruppoMenuItem()`
- `CreaGruppoMenuItem(string testo, Disegnatore* d, Forum* f)`

Oltre al costruttore vuoto questa classe ammette un costruttore che prende in input una stringa di testo contenente il messaggio della voce di menu', il riferimento al Disegnatore e al Forum correnti che saranno a loro volta passati alle command che ne hanno bisogno.

Metodi

- Questa classe non ha metodi propri

4.2.22 DisassociaGruppoMenuItem

File
DisassociaGruppoMenuItem.h, DisassociaGruppoMenuItem.cpp

Descrizione
La DisassociaGruppoMenuItem e' la classe che raccoglie le command necessarie alla esclusione di un gruppo dall'elenco di quelli associati ad un tutor.

Costruttore

- `DisassociaGruppoMenuItem()`
- `DisassociaGruppoMenuItem(string testo, Forum* f, Disegnatore* d, ListaUtentiForum* l)`

Oltre al costruttore vuoto questa classe ammette un costruttore che prende in input una stringa di testo contenente il messaggio della voce di menu', il riferimento al Forum, al Disegnatore e alla ListaUtentiForum correnti che saranno a loro volta passati alle command che ne hanno bisogno.

Metodi

- Questa classe non ha metodi propri

4.2.23 InfoUtenteMenuItem

File
InfoUtenteMenuItem.h, InfoUtenteMenuItem.cpp

Descrizione
La InfoUtenteMenuItem e' la classe che raccoglie le command necessarie alla lettura dei dati relativi ad un certo utente.

Costruttore

- `InfoUtenteMenuItem()`
- `InfoUtenteMenuItem(string testo, Disegnatore* d, ListaUtentiForum* l)`

Oltre al costruttore vuoto questa classe ammette un costruttore che prende in input una stringa di testo contenente il messaggio della voce di menu', il riferimento al Disegnatore e alla ListaUtentiForum correnti che saranno a loro volta passati alle command che ne hanno bisogno.

Metodi

- Questa classe non ha metodi propri

4.2.24 IscrizioneUtentiMenuItem

File
IscrizioneUtentiMenuItem.h, IscrizioneUtentiMenuItem.cpp

Descrizione
La IscrizioneUtentiMenuItem e' la classe che raccoglie le command necessarie all'iscrizione di un nuovo utente del sistema.

Costruttore

- `IscrizioneUtentiMenuItem()`
- `IscrizioneUtentiMenuItem(string testo, Disegnatore* d, ListaUtentiForum* l)`

Oltre al costruttore vuoto questa classe ammette un costruttore che prende in input una stringa di testo contenente il messaggio della voce di menu', il riferimento al Disegnatore e alla ListaUtentiForum correnti che saranno a loro volta passati alle command che ne hanno bisogno.

Metodi

- Questa classe non ha metodi propri

4.2.25 MenuItem

File
MenuItem.h, MenuItem.cpp

Descrizione
MenuItem e' la classe base che contiene i metodi e i campi necessari a gestire e memorizzare le command. Da questa ereditano tutte le altre classi MenuItem, con la funzione aggiuntiva di caricare di specifici puntatori a command il vector _comandi.

Costruttore

- `MenuItem()`
- `MenuItem(string testo, Disegnatore* d)`

Oltre al costruttore vuoto questa classe ammette un costruttore che prende in input una stringa di testo contenente il messaggio della voce di menu' e il riferimento al Disegnatore di cui poi ogni classe figlia avrà bisogno.

Distruttore

- `~MenuItem()`

Metodi

- `void addCommand(Command* c)`
  Aggiunge un nuovo puntatore ad una command al vector _comandi.
- `string leggiTesto() const`
  Legge la stringa del MenuItem che deve apparire come voce di Menu.
- `void clicked()`
  Esegue in sequenza tutte le command puntate in _comandi, una volta che è stato selezionato il MenuItem.

4.2.26 MostraGruppiAssociatiMenuItem

File
MostraGruppiAssociatiMenuItem.h, MostraGruppiAssociatiMenuItem.cpp

Descrizione
La MostraGruppiAssociatiMenuItem e' la classe che raccoglie le command necessarie alla visualizzazione dell'elenco dei gruppi associati ad un tutor.

Costruttore

- `MostraGruppiAssociatiMenuItem()`
- `MostraGruppiAssociatiMenuItem(string testo, Disegnatore* d, ListaUtentiForum* l)`

Oltre al costruttore vuoto questa classe ammette un costruttore che prende in input una stringa di testo contenente il messaggio della voce di menu', il riferimento al Disegnatore e alla ListaUtentiForum correnti che saranno a loro volta passati alle command che ne hanno bisogno.

Metodi

- Questa classe non ha metodi propri

Menu

4.2.27 Menu

File
Menu.h, Menu.cpp

Descrizione
La Menu e' la classe base che funziona da interfaccia per le classi derivate, che sono i menu' specifici per i vari tipi di utente del sistema. Contiene i metodi e le strutture necessarie a gestire le voci del menu' e la loro presentazione. Le classi derivate si specializzano selezionando le particolari voci di menu'.

Costruttore

- `Menu()`
- `Menu(Forum* f, Disegnatore* d, ListaUtentiForum* l)`

Oltre al costruttore vuoto questa classe ammette un costruttore che prende in input il riferimento al Forum, al Disegnatore e alla ListaUtentiForum correnti che saranno a loro volta passati alle MenuItem che ne hanno bisogno.

Distruttore

- `~Menu()`

Metodi

- `void add(MenuItem* s)`
  Aggiunge un nuovo puntatore ad un MenuItem al vector _voci.
- `void start(string firma)`
  Una volta definite le voci del menu' questo metodo lo visualizza permettendo l'interazione tra utente ed applicazione.

4.2.28 MenuAmministratore

File
MenuAmministratore.h, MenuAmministratore.cpp

Descrizione
La MenuAmministratore e' una classe derivata da Menu che carica con specifici puntatori a MenuItem il vettore _voci, al fine di presentare il menu' in funzione del tipo di utente del sistema (amministratore).

Costruttore

- `MenuAmministratore()`
- `MenuAmministratore(Forum* f, Disegnatore* d, ListaUtentiForum* l)`

Questa classe ammette (oltre al costruttore vuoto) un costruttore che prende in input il riferimento al Forum, al Disegnatore e alla ListaUtentiForum correnti che saranno a loro volta passati alle MenuItem che ne hanno bisogno.

Metodi

- Questa classe non ha ulteriori metodi.

4.2.29 MenuTutor

File
MenuTutor.h, MenuTutor.cpp

Descrizione
La MenuTutor e' una classe derivata da Menu che carica con specifici puntatori a MenuItem il vettore _voci, al fine di presentare il menu' in funzione del tipo di utente del sistema (tutor).

Costruttore

- `MenuTutor()`
- `MenuTutor(Forum* f, Disegnatore* d, ListaUtentiForum* l)`

Questa classe ammette (oltre al costruttore vuoto) un costruttore che prende in input il riferimento al Forum, al Disegnatore e alla ListaUtentiForum correnti che saranno a loro volta passati alle MenuItem che ne hanno bisogno.

Metodi

- Questa classe non ha ulteriori metodi.

4.2.30 MenuUtente

File
MenuUtente.h, MenuUtente.cpp

Descrizione
La MenuUtente e' una classe derivata da Menu che carica con specifici puntatori a MenuItem il vettore _voci, al fine di presentare il menu' in funzione del tipo di utente del sistema (utente semplice).

Costruttore

- `MenuUtente()`
- `MenuUtente(Forum* f, Disegnatore* d, ListaUtentiForum* l)`

Questa classe ammette (oltre al costruttore vuoto) un costruttore che prende in input il riferimento al Forum, al Disegnatore e alla ListaUtentiForum correnti che saranno a loro volta passati alle MenuItem che ne hanno bisogno.

Metodi

- Questa classe non ha ulteriori metodi.

Disegnatore

4.2.31 Disegnatore

File:
Disegnatore.cpp, Disegnatore.h

Descrizione
Questa classe fornisce i metodi per la gestione dell'input e dell'output sulla console

Costruttori

- `Disegnatore()`
  Costruttore di base.

Metodi

- `int scegli(vector<string> opzioni, string messaggio)`
  Metodo che permette all'utente di scegliere con i tasti cursore fra un insieme di stringhe; viene restituito l'indice dell'opzione scelta
- `void pulisci()`
  Metodo per cancellare la parte dello schermo dove avviene l'interazione fra utente e applicazione
- `void disegnaInterfaccia(string firma)`
  Metodo che disegna l'interfaccia principale dell'applicazione
- `void visualizzaDatiUtente(vector<string> info)`
  Metodo per visualizzare a schermo i dati di un utente, passati in un vettore di stringhe

- `void mostraTesto(string testo)`
  Metodo che pulisce lo schermo e mostra una stringa
- `void statusbar(string s)`
  Disegna la statusbar in fondo allo schermo e avvisa l'utente con la stringa s
- `void mostraStringhe(vector<string>)`
  Metodo che pulisce lo schermo e mostra un insieme di stringhe
- `vector<string> riempiForm(vector<string>)`
  Metodo per il riempimento di un form. Vengono poste delle richieste (passate come parametro di ingresso) e viene restituito un vettore di stringhe contenente le risposte

Forum

4.2.32 Forum

File:
Forum.cpp, Forum.h

Descrizione
Questa classe contiene il vettore dei puntatori ai gruppi di discussione. Possiede i metodi per accedere ai gruppi di discussione, per distruggerli e per crearne di nuovi.

Costruttori

- `Forum()`
  Costruttore di base.

Distruttore

- `Forum()`
  Libera la memoria occupata dai gruppi di discussione.

Metodi

- `void leggiGruppiDaFile()`
  Metodo che legge i file dei gruppi (*.tut) e carica i messaggi nel vector dei gruppi di discussione
- `vector<string> leggiElencoGruppi()`
  Restituisce un vettore contenente i nomi dei gruppi di discussione

- `vector<string> leggiElencoDomande(int* numGruppo)`
  Legge le domande poste in un gruppo e le restituisce in un vettore di stringhe

- `vector<string> leggiRisposte(int numGruppo, int numDomanda)`
  Legge le risposte relative alla domanda numDomanda del gruppo di indice numGruppo, e le restituisce in un vettore di stringhe

- `void poniDomanda(int numGruppo, string domanda)`
  Metodo per aggiungere una domanda nel gruppo di discussione di indice numGruppo

- `void poniRisposta(int numGruppo, int numDomanda, string risposta)`
  Metodo per aggiungere una risposta alla domanda di indice numDomanda nel gruppo di discussione di indice numGruppo

- `void creaGruppo(string nome)`
  Crea un nuovo gruppo vuoto con il nome passato come argomento

- `bool cancellaGruppo(int numGruppo)`
  Metodo che cancella un gruppo di discussione dal forum; restituisce true in caso di successo

- `string leggiNomeGruppo(int numGruppo)`
  Restituisce il nome del gruppo di discussione di indice numGruppo

4.2.33 GruppoDiscussione

File:
GruppoDiscussione.cpp, GruppoDiscussione.h

Descrizione
Questa classe contiene i messaggi di un gruppo di discussione e i metodi per gestirli.

Costruttori

- `GruppoDiscussione()`
  Costruttore di base.

Distruttore

- `~GruppoDiscussione()`
  Distruttore. Provvede al salvataggio dei gruppi di discussione

Metodi

- `int getNumeroDomande() const`
  Restituisce il valore del campo _numDomande
- `int getNumeroRisposte(int i) const`
  Restituisce il numero di risposte per la domanda di indice i
- `void setNumeroDomande(int i)`
  Cambia il valore del campo _numDomande
- `void inserisciDomanda(Messaggio _nuovoMessaggio)`
  Inserisce una nuova domanda nel gruppo di discussione
- `void inserisciRisposta(Messaggio _nuovoMessaggio, int _numDom)`
  Inserisce una nuova risposta per la domanda di indice _numDom
- `string leggiMessaggio(int i, int j) const`
  Legge il messaggio di indice (i,j)
- `string leggiNome() const`
  Restituisce il valore del campo _nome
- `vector<string> leggiRisposte(int numDom)`
  Metodo che restituisce un vettore di stringhe contenenti le risposte date alla domanda di indice numDom
- `vector<string> leggiDomande()`
  Metodo che restituisce un vettore di stringhe contenenti le domande poste nel gruppo di discussione
- `void poniDomanda(string domanda)`
  Metodo per aggiungere una domanda al gruppo di discussione
- `void poniRisposta(int _numDom, string risposta)`
  Metodo per aggiungere una risposta alla domanda di indice numDom
- `void caricaFile(char* nomeFile)`
  Legge i messaggi di un gruppo di discussione dal file corrispondente

4.2.34 Messaggio

File
Messaggio.h, Messaggio.cpp

Descrizione
Messaggio e' una classe contenente gli oggetti che memorizzano le domande e le risposte dei gruppi di discussione.

Costruttore

- `Messaggio(string testo);`

Questa classe ammette un costruttore che prende in input il testo stesso del messaggio.

Metodi

- `string leggiTesto()`
  Metodo accessore che restituisce semplicemente il contenuto del messaggio.

Utenti

4.2.35 Utente

File:
Utente.cpp, Utente.h

Descrizione
Questa classe definisce un utente del sistema, dotato di dati personali, oltre a un login e una password. I metodi sono accessori e mutatori.

Costruttori

- `Utente()`
  Costruttore di base.
- `Utente(string nome, string cognome, string email, string login, string password)`
  Costruttore che crea un utente in base ai dati passati.

Metodi

- `string leggiNome() const`
  Restituisce il valore del campo _nome.
- `string leggiCognome() const`
  Restituisce il valore del campo _cognome.
- `string leggiEmail() const`
  Restituisce il valore del campo _email.
- `string leggiLogin() const`
  Restituisce il valore del campo _login.

- `string leggiPassword() const`
  Restituisce il valore del campo _password.

- `virtual vector <string> tuttiIDati()`
  Restituisce in un vettore di stringhe tutti i dati di un utente, così come vengono memorizzati nel file 'utenti.ute'.

- `virtual vector <string> leggiGruppiAssociati()`
  Restituisce i nomi dei gruppi associati a un utente. Per un utente semplice, l'insieme dei gruppi associati è vuoto.

- `void cambiaNome(string nuovoNome)`
  Aggiorna il valore del campo _nome al valore nuovoNome.

- `void cambiaCognome(string nuovoCognome)`
  Aggiorna il valore del campo _cognome al valore nuovoCognome.

- `void cambiaEmail(string nuovaEmail)`
  Aggiorna il valore del campo _email al valore nuovaEmail.

- `void cambiaLog(string nuovoLog)`
  Aggiorna il valore del campo _login al valore nuovoLog.

- `void cambiaPw(string nuovaPw)`
  Aggiorna il valore del campo _password al valore nuovaPw.

4.2.36 Tutor

File:
Tutor.cpp, Tutor.h

Descrizione
Tutor è una classe derivata da utente. Ha in più i gruppi associati e i metodi per la loro gestione.

Costruttori

- `Tutor(string nome, string cognome, string email, string login, string password, vector <string> gruppiAssociati)`
  Costruttore che crea un tutor in base ai dati passati.

- `Tutor(string nome, string cognome, string email, string login, string password)`
  Costruttore che crea un tutor in base ai dati passati, lasciando l'insieme dei gruppi associati vuoto.

Metodi

- `vector <string> leggiGruppiAssociati()`
  Restituisce i nomi dei gruppi associati a un tutor.
- `vector <string> tuttiIDati()`
  Restituisce in un vettore di stringhe tutti i dati di un tutor, così come vengono memorizzati nel file 'utenti.ute'.
- `void aggiungiGruppo(string gruppo)`
  Metodo che permette di aggiungere un'associazione fra il tutor e un gruppo.
- `void eliminaGruppo(string gruppo)`
  Metodo che permette di eliminare un'associazione fra il tutor e un gruppo.

4.2.37 Amministratore

File:
Amministratore.cpp, Amministratore.h

Descrizione
Questa classe è derivata da tutor. Un amministratore è un tutor che può gestire i gruppi del forum e le iscrizioni al sistema.

Costruttori

- `Amministratore(string nome, string cognome, string email, string login, string password, vector <string> gruppiAssociati)`
  Costruttore che crea un amministratore in base ai dati passati.
- `Amministratore(string nome, string cognome, string email, string login, string password)`
  Costruttore che crea un amministratore in base ai dati passati, lasciando l'insieme dei gruppi associati vuoto.

Metodi

- `virtual vector <string> tuttiIDati()`
  Restituisce in un vettore di stringhe tutti i dati di un amministratore, così come vengono memorizzati nel file 'utenti.ute'.

ListaUtentiForum

4.2.38 ListaUtentiForum

File:
ListaUtentiForum.cpp, ListaUtentiForum.h

Descrizione
Questa classe contiene la lista dei puntatori agli utenti del sistema e possiede le operazioni per gestire tale lista. Contiene inoltre una lista dei puntatori ai tutor (sottoinsieme della lista utenti).

Costruttori

- `ListaUtentiForum()`
  Costruttore di base.

Distruttore

- `~ListaUtentiForum()`
  Il distruttore di ListaUtentiForum si occupa di:
  - liberare la memoria occupata dalle liste di puntatori a utenti e tutor, compresi i gruppi associati.
  - salvare sul file 'utenti.ute' le informazioni relative agli utenti
  - salvare sul file 'gruppitutor.ute' le associazioni fra tutor e gruppi
  - salvare sul file 'gruppiamm.ute' le associazioni fra amministratori e gruppi

Metodi

- `vector <string> leggiFileUtenti(string login, string password)`
  Metodo che apre i file contenenti i dati degli utenti e inizializza le apposite strutture destinate a contenerli
- `vector <string> leggiDatiUtente(int num)`
  Restituisce i dati dell'utente di indice num in un vettore di stringhe, cosi' come sono stati letti dal file utenti.ute
- `string leggiFirmaUtente()`
  Restituisce una stringa formata da nome e cognome dell'utente corrente che viene usata per 'firmare' i messaggi del forum.
- `string leggiFirmaUtente(int num)`
  Restituisce una stringa formata da nome e cognome dell'utente di indice num. Serve per elencare nome e cognome degli utenti.

- `int numeroUtenti()`
  Restituisce il numero degli utenti del sistema
- `Utente* getUtenteCorrente()`
  Restituisce un puntatore all'utente corrente
- `int leggiNumUtenteSistema()`
  Restituisce il numero dell'utente che sta utilizzando il sistema
- `void cambiaPwUtente(int numUtente, string newPw)`
  Cambia la password dell' utente con indice num
- `void cambiaLogUtente(int numUtente, string newLog)`
  Cambia il login dell' utente con indice num
- `void cambiaNomeUtente(int numUtente, string newNome)`
  Cambia il nome dell' utente con indice num
- `void cambiaCognomeUtente(int numUtente, string newCognome)`
  Cambia il cognome dell' utente con indice num
- `void cambiaEmailUtente(int numUtente, string newEmail)`
  Cambia l'email dell' utente con indice num
- `vector <string> elencoGruppiAssociati(int numUtente)`
  Restituisce l'elenco dei gruppi associati all'utente di indice num
- `void add(Utente* u)`
  Aggiunge un puntatore a un nuovo utente alla lista degli utenti.
- `void addTutor(Tutor* t)`
  Aggiunge un puntatore a un nuovo tutor alla lista dei tutor.
- `bool iscriviUtente(vector<string> nuovoUtente)`
  Metodo che permette l'iscrizione di un utente; le informazioni per creare il nuovo utente sono passate in un vettore di stringhe. Se l'utente è un tutor, viene inoltre aggiornata la lista dei tutor.
- `bool cancellaUtente(int numUtente)`
  Metodo che permette la cancellazione dell'utente di indice numUtente. Viene inoltre disallocata la memoria occupata dall'utente che si sta cancellando. Se l'utente è un tutor, viene inoltre aggiornata la lista dei tutor.
- `vector<string> leggiFirmeTutor()`
  Metodo che restituisce un vettore di 'firme' (nome seguito da cognome) dei tutor del sistema. Viene usato per elencarli.

- `void associaGruppoTutor(string gruppo, int* numeroTutor)`
  Metodo che aggiunge a un tutor l'associazione a un gruppo. Il gruppo è passato come stringa, il numero del tutor è puntato da numeroTutor.

- `void disassociaGruppoTutor(string gruppo, int* numeroTutor)`
  Metodo che toglie a un tutor l'associazione a un gruppo. Il gruppo è passato come stringa, il numero del tutor è puntato da numeroTutor.

- `vector<string> leggiGruppiTutor(int numeroTutor)`
  Metodo che restituisce come vettore di stringhe i nomi dei gruppi associati al tutor di indice numeroTutor.

- `void rinfrescaListaTutor()`
  Aggiorna la lista dei tutor in base alla lista attuale degli utenti.

- `bool fileUtentiVuoto()`
  Controlla se il file degli utenti 'utenti.ute' è vuoto.

Sistema

4.2.39 Sistema

File:
Sistema.cpp, Sistema.h

Descrizione

Questa classe contiene i puntatori agli oggetti fondamentali e ha la responsabilità del riconoscimento degli utenti. Contiene il metodo main() che fa partire, controlla il flusso di esecuzione e fa terminare il programma.

Costruttori

- `Sistema()`
  Costruttore. Inizializza i puntatori agli oggetti di classe Disegnatore, Forum, ListaUtentiForum e Menu.

Distruttore

- `~Sistema()`
  Distruttore. Libera la memoria dagli oggetti di classe Disegnatore, Forum e ListaUtentiForum, e Menu.

Metodi

- `int main()`
  Metodo dal quale parte e termina la computazione. Crea un oggetto di tipo Sistema, chiamato tutoraggio, che si occuperà di riconoscere un utente. Se viene riconosciuto un utente valido, il sistema si preoccupa di caricare i gruppi da file ed avviare il menù

- `void accediSistema()`
  Metodo che si preoccupa del riconoscimento di un utente. Se il sistema non possiede alcun utente, ne viene creato uno con dati di default. Se viene riconosciuto un utente valido, crea anche un menù idoneo.

- `void avviaMenu()`
  Metodo che fa avviare il menu, personalizzato con la firma dell'utente corrente.

- `void caricaGruppi()`
  Metodo che fa caricare i gruppi di discussione al forum

- `void uscita()`
  Metodo che viene eseguito al termine del programma. Pulisce lo schermo e riporta i colori della console ai valori standard.

- `void creaPrimoUtente()`
  Metodo che viene chiamato se non esistono attualmente utenti nel sistema. Crea un utente di default di tipo amministratore con login 'root' e password 'root'. Questo primo utente potrà a sua volta creare altri utenti.

4.3 Codice

Nelle pagine seguenti sono i riportati i listati C++ del programma.