

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

import java.io.File;
import java.io.IOException;

/**
 * Classe adibita all'esecuzione del Progetto.
 *
 * @author Filippo Fontanelli , Francesca Brogi
 */
public class Main extends RegistroFrequenze {

    /**
     *
     */
    private static final long serialVersionUID = 7L;
    /**
     * @param args
     */
    /** *****MAIN***** */
    public static int controllo;

    public static void main(String[] args) throws IOException,
        ClassNotFoundException {
        menu();
    }

    /**
     * menu iniziale
     *
     * @throws IOException
     * @throws ClassNotFoundException
     */
    public static void menu() {

        System.out
            .println("\n\n\t\t\t\t\t --- GESTIONE REGISTRO FIRME DI FREQUENZA --
\n");
        System.out
            .println("\t\t\t\t\t --- + Filippo Fontanelli && Francesca Brogi + ---\n
\n");

        int y = 2;
        while (true) {

            System.out.println("\t\t\t\t\t 1 -> CREA NUOVO REGISTRO \n");
            System.out.println("\t\t\t\t\t 2 -> CARICA REGISTRO \n");
            System.out.println("\t\t\t\t\t 3 -> ESCI \n");
            controllo = SafeInput.interoUnoTre();
```

```
switch (controllo) {
case 1:
    // Creo un nuovo registro prendendo come input il nome e il
    // numero minimo di lezioni

    boolean ok = false;
    String title = "";
    do {
        System.out.println("\n");
        System.out
            .println("Inserire il nome del Registro da Creare : (es:
Lip)");

        title = SafeInput.stringaNonVuota();

        File f = new File(title + ".txt");
        y = 0;

        // Controllo se il file esiste altrimenti lancio
        // un'eccezione

        if (f.exists()) {
            System.out
                .println("Attenzione: e' gia' presente un archivio
con questo nome!");
            System.out.println("Inserire un altro nome.....\n");
        } else
            ok = true;

    } while (!ok);
    // instanzio la variabile reg che avevo precedentemente
    // dichiarato
    reg = new RegistroFrequenzeImp(title, title);

    int x = 0;

    do {
        System.out
            .println("Inserire il numero di lezioni minimo per l
'ammissione all'esame : ");
        x = SafeInput.readInt();
    } while (x < 0);

    // Modifico il numero di lezioni obbligatorio di default con
    // quello ottenuto da input
    reg.setNumObbligatorio(x);

    // se tutto e' proceduto correttamente proseguo con la creazione
    // altrimenti ritorno al menu iniziale
    if (y == 0) {

        // eseguo il menu relativo alla creazione di un nuovo
        // registro e appena esso ha finito l'esecuzione eseguo il
        // menu relativo alle varie operazioni
    }
}
```

```
        // che possiamo effettuare sul registro corrente

        menuNewRegistro();

        menuRegistro();
    }

    else
        menu();
    break;

case 2: {

    // effettuo il caricamento da file di un registro

    boolean inizializzata = false;
    String tipe = "";
    System.out.println();

    // Do la possibilita' di scegliere se caricare un file binario o
    // testuale
    do {
        try {
            System.out
                .println("Caricare file Binario o Testuale? : (T/B)")
                ;

            tipe = Input.readLine();
            if (tipe.equals((String) "T")
                || tipe.equals((String) "B")
                || tipe.equals((String) "b")
                || tipe.equals((String) "t"))
                inizializzata = true;

        } catch (java.lang.NumberFormatException e) {
            // Se il valore inaserto nn e' un numero entro nel
            // catch e non setto il booleano
            System.out.print("Attenzione parametro errato!\n");
        }
    } while (!inizializzata);

    boolean xx = false;

    // Ricevo da input il nome del file da caricare
    do {
        System.out
            .println("Inserire il nome del Registro da Caricare
(Vuoto per tornare al menu'; es : Lip)");
        title = SafeInput.readLine();
        if (title.equals("")) {
            menu();
            reg = null;
        }
    }
```

```
// instanzio la variabile reg che avevo precedentemente
// dichiarato
reg = new RegistroFrequenzeImp(title);

// Eseguo il load, binario o testuale a seconda della scelta
try {
    if (tipe.equals((String) "B")
        || tipe.equals((String) "b")) {
        reg = reg.loadBinario(title + ".bin");
    } else {
        reg.loadTesto();
    }
    xx = true;
} catch (IOException e) {
    System.out
        .println("\nAttento. Errore di caricamento; prova ad
inserire un altro nome. \n");
    xx = false;
} catch (NullPointerException e) {
    System.out
        .println("\nAttento. Errore di caricamento; prova ad
inserire un altro nome. \n");
    xx = false;
} catch (ClassNotFoundException e) {
    System.out
        .println("\nAttento. Errore di caricamento; prova ad
inserire un altro nome. \n");
    xx = false;
}
} while (!xx);

// eseguo il menu relativo alle varie operazioni
// che possiamo effettuare sul registro corrente
menuRegistro();
}



break;

case 3:
    // Salvo il Registro Corrente, sia nel file binario che in
    // quello testuale
    try {
        if (reg != null) {
            reg.saveTesto();
            reg.saveBinario();
            System.out
                .println("\n -- Archivio salvato correttamente -- \n"
);
        }
    } catch (java.io.IOException e) {
        System.out
            .println("Errore durante il salvataggio nell'archivio!\n"
);
    }
};
```

```

        } catch (ClassNotFoundException e) {
            System.out
                .println("Errore durante il salvataggio nell'archivio!\n"
);
        }
    }
    System.out
        .println("\n\t\t\t*****FINE ESECUZIONE*****");
    System.exit(0);
    break;
}

// nel caso in qui non si sia effettuata nessuna operazione, a causa di
// un'errore
// viene rieseguito il menu iniziale, in modo da non interrompere
// l'esecuzione
if (y == 0)
    menu();
}

/**
 *
 * Menu Relativo al Registro Corrente. Dove  possibile scegliere se accedere
 * alle opzioni di visualizzazione  o di modifica registro
 *
 * @throws ClassNotFoundException
 *
 */
public static void menuRegistro() {

    while (true) {
        // Creazione del menu
        System.out.println("\t\t\t 1 ->  VISUALIZZARE REGISTRO \n");
        System.out.println("\t\t\t 2 ->  MODIFICARE REGISTRO \n");
        System.out.println("\t\t\t 3 ->  ESCI \n");
        controllo = SafeInput.interoUnoTre();

        switch (controllo) {
            case 1:
                // eseguo il menu relativo alla visualizzazione di tutte le
                // caratteristiche
                // del Registro
                visualizzaReg();
                break;
            case 2:
                // eseguo il menu relativo alla modifica di tutte le componenti
                // del Registro
                modifyReg();
                break;
            case 3: {
                // Salvo il Registro Corrente, sia nel file binario che in
                // quello testuale
                try {

```

```

        if (reg != null) {
            reg.saveTesto();
            reg.saveBinario();
            System.out
                .println("\n -- Archivio salvato correttamente -- \n"
);
        }
    } catch (java.io.IOException e) {
        System.out
            .println("Errore durante il salvataggio nell'archivio!\n"
);
    } catch (ClassNotFoundException e) {
        System.out
            .println("Errore durante il salvataggio nell'archivio!\n"
);
    }
}
return;
}
}

/**
 *
 * Menu Relativo alla Creazione di un Nuovo Registro. Dove ♦ possibile
 * scegliere tra: 1 -> INSERIRE L'ELENCO DEGLI STUDENTI 2 -> INSERIRE
 * L'ELENCO DELLE LEZIONI 3 -> ESCI
 *
 */
public static void menuNewRegistro() {

    while (true) {
        // Creazione del menu
        System.out
            .println("\t\t\t 1 ->  INSERIRE L'ELENCO DEGLI STUDENTI \n");
        System.out
            .println("\t\t\t 2 ->  INSERIRE L'ELENCO DELLE LEZIONI \n");
        System.out.println("\t\t\t 3 ->  ESCI \n");
        controllo = SafeInput.interoUnoTre();

        switch (controllo) {
            case 1:
                // eseguo il menu relativo all'inserimento studenti
                inserisciStud();
                break;
            case 2:
                // eseguo il menu relatio all'inserimento lezioni
                inserisciLezDate();
                break;
            case 3: {
                Salva();
                return;
            }
        }
    }
}

```

```
    }

    }

}

/**
 *
 * Menu Relativo alla Modifica del Registro corrente. Dove ♦ possibile
 * scegliere tra : 1 -> MENU PRESENZE 2 -> MENU STUDENTI 3 -> MENU LEZIONI 4
 * -> ESCI
 */

public static void modifyReg() {
    // Creazione del menu
    System.out.println("\t\t\t 1 ->  MENU PRESENZE \n");
    System.out.println("\t\t\t 2 ->  MENU STUDENTI \n");
    System.out.println("\t\t\t 3 ->  MENU LEZIONI \n");
    System.out.println("\t\t\t 4 ->  ESCI \n");
    controllo = SafeInput.interoUnoQuattro();

    switch (controllo) {
    case 1:
        // chiama il metodo relativo al menu delle presenze
        modifyPresenze();
        break;
    case 2:
        // chiama il metodo relativo al menu degli studenti
        modifyStudenti();
        break;
    case 3:
        // chiama il metodo relativo al menu delle lezioni
        modifyLezioni();
        break;
    case 4: {
        Salva();
    }
    return;
    }
}

/**
 *
 * Menu Relativo alla Modifica dell'elenco Lezione del Registro corrente.
 * Dove ♦ possibile scegliere tra : 1 -> INSERIRE NUOVA LEZIONE 2 ->
 * MODIFICARE LEZIONE 3 -> ELIMINARE LEZIONE 4 -> ESCI
 */


public static void modifyLezioni() {

    while (true) {
        // Creazione del menu
```

```

System.out.println("\t\t\t 1 ->  INSERIRE NUOVA LEZIONE \n");
System.out.println("\t\t\t 2 ->  MODIFICARE LEZIONE \n");
System.out.println("\t\t\t 3 ->  ELIMINARE LEZIONE \n");
System.out.println("\t\t\t 4 ->  ESCI \n");
controllo = SafeInput.interoUnoQuattro();

switch (controllo) {
case 1:
    // eseguo il menu relatio all'inserimento lezioni
    inseriscilezDate();
    break;
case 2:
    // eseguo il metodo relatio alla modifica di una determinata
    // lezione
    modifyLezioniInterno();
    break;
case 3:
    // eseguo il metodo relatio all'eliminazione di una
    // determinata lezione
    eliminaLezioniInterno();
    break;
case 4: {
    Salva();
}
    return;
}
}

/**
 *
 * Menu Relativo alla Modifica dell'elenco Presenze del Registro corrente.
 * Dove  possibile scegliere tra : 1 -> INSERIRE PRESENZE 2 -> ELIMINARE
 * PRESENZE 3 -> ESCI
 *
 */

public static void modifyPresenze() {

    while (true) {
        // Creazione del menu
        System.out.println("\t\t\t 1 ->  INSERIRE PRESENZE \n");
        System.out.println("\t\t\t 2 ->  ELIMINARE PRESENZE \n");
        System.out.println("\t\t\t 3 ->  ESCI \n");
        controllo = SafeInput.interoUnoTre();

        switch (controllo) {
case 1:
            // eseguo il menu relativo all'inserimento presenze
            inserimentoPresenze();
            break;
case 2:

```



```
        // eseguo il menu relatio all'eliminazione di una determinata
        // lezione
        eliminaPresenzeLezione();
        break;
    case 3: {
        // chiama il metodo relativo al salvataggio
        Salva();
    }
    return;
}

}

}

/**
 *
 * Menu Relativo all'Inserimento delle Presenze nel Registro corrente. Dove
 * ➡ possibile scegliere tra : 1 -> INSERIMENTO PARTENDO DALLA LEZIONE 2 ->
 * INSERIMENTO PARTENDO DALLO STUDENTE 3 -> ESCI
 *
 */
public static void inserimentoPresenze() {

    while (true) {
        // Creazione del menu
        System.out
            .println("\t\t\t 1 ->  INSERIMENTO PARTENDO DALLA LEZIONE \n");
        System.out
            .println("\t\t\t 2 ->  INSERIMENTO PARTENDO DALLO STUDENTE \n");
        System.out.println("\t\t\t 3 ->  ESCI \n");
        controllo = SafeInput.interoUnoTre();

        switch (controllo) {
            case 1:
                // eseguo il metodo relatio all'inseriemnto di una determinata
                // presenza
                // avendooc come riferimento una lezione
                inserimentoPresenzeLezione();
                break;
            case 2:
                // eseguo il metodo relatio all'inseriemnto di una determinata
                // presenza
                // avendooc come riferimento uno studente
                inserimentoPresenzeStudente();
                break;
            case 3: {
                Salva();
            }
            return;
        }
    }
}
```

```

/**
 *
 * Menu Relativo all'Inserimento delle Presenze partendo dalla lezione. Dove
 *  possibile scegliere tra : 1 -> CREA UNA NUOVA LEZIONE 2 -> INSERIRE LE
 *  PRESENZE DI UNA LEZIONE ESISTENTE 3 -> ESCI
 *
 */
public static void inserimentoPresenzeLezione() {

    if (reg.getStudenti().size() > 0) {
        while (true) {
            // Creazione del menu
            System.out.println("\t\t\t 1 ->  CREARE UNA NUOVA LEZIONE\n");
            System.out
                .println("\t\t\t 2 ->  INSERIRE PRESENZE DI UNA LEZIONE
ESISTENTE\n");
            System.out.println("\t\t\t 3 ->  ESCI \n");
            controllo = SafeInput.interoUnoTre();

            switch (controllo) {
                case 1:
                    // eseguo il menu relatio all'inserimento lezioni
                    inserisciLezDate();
                    break;
                case 2:
                    // controllo se ci sono degli studenti nel Registro
                    // corrente, se ci sono procedo, altrimenti stampo
                    // un messaggio di errore e termino l'esecuzione del metodo
                    if (!(reg.getStudenti().size() > 0)) {
                        System.out
                            .println("Nel registro non ci sono Studenti...quindi
e' impossibile inserire una presenza.\n");
                        return;
                    } else {
                        // controllo se ci sono degli lezioni nel Registro
                        // corrente, se ci sono procedo eseguendo
                        // il metodo relativo all'inserimento di una presenza
                        // facendo riferimento ad una lezione
                        // gia esistente , altrimenti stampoun messaggio di
                        // errore e termino l'esecuzione del metodo
                        if (reg.getLezioni().size() > 0)

                            InserimentoPresenzeInternoEsistente();

                        else {
                            System.out
                                .println("Nel registro non ci sono Lezioni...
quindi e' impossibile inserire una presenza.\n");
                            return;
                        }
                    }
                    break;
                case 3:

```

```
        Salva();
        return;
    }
} else {
    System.out
        .println("Nel registro non ci sono Studenti...quindi e'
impossibile inserire una presenza.\n");
}
return;
}

/**
 *
 * Menu Relativo alla Modifica degli Studenti del Registro corrente. Dove
 * possibile scegliere tra : 1 -> INSERIRE NUOVO STUDENTE 2 -> MODIFICARE
 * STUDENTE 3 -> ELIMINARE STUDENTE 4 -> ESCI
 *
 */
public static void modifyStudenti() {

    while (true) {
        // Creazione del menu
        System.out.println("\t\t\t 1 ->  INSERIRE NUOVO STUDENTE \n");
        System.out.println("\t\t\t 2 ->  MODIFICARE STUDENTE \n");
        System.out.println("\t\t\t 3 ->  ELIMINARE STUDENTE \n");
        System.out.println("\t\t\t 4 ->  ESCI \n");
        controllo = SafeInput.interoUnoQuattro();

        switch (controllo) {
            case 1:
                // chiama il metodo relativo all'inserimento studenti
                inserisciStud();
                break;
            case 2:
                // chiama il metodo relativo alla modifica studenti
                modifyStudentiInterno();
                break;
            case 3:
                // chiama il metodo relativo all'eliminazione studenti
                eliminaStudenteInterno();
                break;
            case 4: {
                // chiama il metodo relativo al salvataggio
                Salva();
            }
            return;
        }
    }
}

/**
```

```

*
* Menu Relativo alla Visualizzazione del Registro corrente. Dove
* possibile scegliere tra : 1 -> VISUALIZZA REGISTRO SU VIDEO 2 ->
* VISUALIZZA STATISTICHE CONSUNTIVE 3 -> ESCI
*
*/
public static void visualizzaReg() {

    while (true) {
        // Creazione del menu
        System.out.println("\t\t\t 1 ->  VISUALIZZA REGISTRO SU VIDEO \n");
        System.out
            .println("\t\t\t 2 ->  VISUALIZZA STATISTICHE CONSUNTIVE \n");
        System.out.println("\t\t\t 3 ->  ESCI \n");
        controllo = SafeInput.interoUnoTre();

        switch (controllo) {
            case 1:
                // chiama il metodo relativo al menu della visualizzazione registro/
                studenti/lezioni
                visualizzaRegTwo();
                break;
            case 2:
                // chiama il metodo relativo al menu della visualizzazione delle
                statistiche consuntive
                visualizzaStatistiche();
                break;
            case 3:
                // chiama il metodo relativo al salvataggio
                Salva();
                return;
        }
    }

}

/**
*
* Menu Relativo alla Visualizzazione del Registro corrente. Dove
* possibile scegliere tra : 1 -> VISUALIZZA REGISTRO 2 -> VISUALIZZA ELENCO
* STUDENTI 3 -> VISUALIZZA ELENCO LEZIONI 4 -> ESCI
*
*/
public static void visualizzaRegTwo() {

    while (true) {
        // Creazione del menu
        System.out.println("\t\t\t 1 ->  VISUALIZZA REGISTRO \n");
        System.out.println("\t\t\t 2 ->  VISUALIZZA ELENCO STUDENTI \n");
        System.out.println("\t\t\t 3 ->  VISUALIZZA ELENCO LEZIONI \n");
        System.out.println("\t\t\t 4 ->  ESCI \n");
        controllo = SafeInput.interoUnoQuattro();
    }
}

```

```

        switch (controllo) {
        case 1:
            // chiama il metodo che provvedera' alla visualizzazione del registro
            visualizzaRegInterno();
            break;
        case 2:
            // chiama il metodo che provvedera' alla visualizzazione degli
            studenti
                visualizzaRegStudenti();
            break;
        case 3:
            // chiama il metodo che provvedera' alla visualizzazione delle
            lezioni
                visualizzaRegLezioni();
            break;
        case 4:
            return;
        }
    }

}


/**
 *
 * Metodo che provvede alla Visualizzazione dell'elenco Studenti.
 *
 */
public static void visualizzaRegStudenti() {
    System.out.println(" Studenti :\n ");
    //ciclo che stampa a video uno studente per riga
    for (int i = 0; i < reg.getStudenti().size(); i++) {
        System.out.print("\t" + (i + 1) + " - "
            + reg.getStudenti().getElemReg().get(i).toString() + "\n");
    }
    System.out.println();
    System.out.println("\t\t\t Premere un invio per tornare al menu'\n");
    Input.readLine();
    return;
}

/**
 *
 * Metodo che provvede alla Visualizzazione dell'elenco Lezioni.
 *
 */
public static void visualizzaRegLezioni() {
    System.out.println("Elenco delle lezioni :");
    //ciclo che stampa a video una lezione per riga
    for (int i = 0; i < reg.getLezioni().size(); i++) {
        System.out.print("\t" + (i + 1) + " - "
            + reg.getLezioni().getElemReg().get(i).toString() + "\n");
    }
    System.out.println();
}

```

```
        System.out.println("\t\t\t Premere un invio per tornare al menu'\n");
        Input.readLine();
        return;
    }

    /**
     *
     * Metodo che provvede alla Visualizzazione del Registro corrente su video.
     *
     */
    public static void visualizzaRegInterno() {
        //ciclo che stampa a video una presenza per riga, andando cosi a comporre il
registro
        reg.printReg();
        System.out.println("\t\t\t Premere un invio per tornare al menu'\n");
        Input.readLine();
        return;
    }

    /**
     *
     * Menu Relativo alla Visualizzazione delle Statistiche Consuntive sulla
     * base del Registro corrente. Dove  possibile scegliere tra : 1 ->
     * VISUALIZZA STATISTICHE STUDENTI 2 -> VISUALIZZA STATISTICHE LEZIONI 3 ->
     * ESCI
     *
     */
    public static void visualizzaStatistiche() {

        while (true) {
            // Creazione del menu
            System.out
                .println("\t\t\t 1 ->  VISUALIZZA STATISTICHE STUDENTI \n");
            System.out
                .println("\t\t\t 2 ->  VISUALIZZA STATISTICHE LEZIONI \n");
            System.out.println("\t\t\t 3 ->  ESCI \n");
            controllo = SafeInput.interoUnoQuattro();

            switch (controllo) {
                case 1:
                    // chiama il metodo relativo al menu della visualizzazione delle
statistiche studenti
                    statisticheStudenti();
                    break;
                case 2:
                    // chiama il metodo relativo al menu della visualizzazione delle
statistiche lezioni
                    statisticheLezioni();
                    break;
                case 3:
                    // chiama il metodo relativo al salvataggio
                    Salva();
            }
        }
    }
}
```

```

        return;
    }
    return;
}

/**
 *
 * Menu Relativo all'Inserimento degli Studenti nel Registro corrente. Dove
 *  possibile scegliere tra : 1 -> CARICARE STUDENTI DA FILE 2 ->
 *  INSERIMENTO MANUALE 3 -> ESCI
 *
 */
public static void inserisciStud() {

    // Creazione del menu
    System.out.println("\t\t\t 1 ->  CARICARE STUDENTI DA FILE \n");
    System.out.println("\t\t\t 2 ->  INSERIMENTO MANUALE \n");
    System.out.println("\t\t\t 3 ->  ESCI \n");
    controllo = SafeInput.interoUnoTre();

    switch (controllo) {
    case 1: {
        String file = "";
        //do while che termina nel momento in cui il file esiste o quando
passiamo il carattere
        //per tornare al menu
        do {
            System.out
                .println("Inserire il nome del file (* per tornare al menu)")
);
            file = Input.readLine();
            if (!file.equals("")) {

                if (file.equals(""))
                    return;
                else {

                    File f = new File(file);

                    if (f.exists()) {
                        //se il file esiste carico il file contenente gli
studenti
                        reg.loadStudenti(file);
                    } else {
                        file = "";
                    }
                }
            }
        } while (file.equals(""));
        break;
    }
}

```

```

    case 2:
        // chiama il metodo relativo all'inserimento studenti da tastiera
        inserimentoStudMan();
        break;

    case 3:
        // chiama il metodo relativo al salvataggio
        Salva();
        return;
    }
    // chiama il metodo relativo al salvataggio
    Salva();
    return;
}

/**
 *
 * Menu Relativo all'Inserimento delle Lezioni nel Registro Corrente. Dove
 * possibile scegliere tra : 1 -> CARICARE LEZIONE DA FILE 2 -> INSERIRE
 * MANUALE 3 -> ESCI
 *
 */
public static void inserisciLezDate() {

    // Creazione del menu
    System.out.println("\t\t\t 1 ->  CARICARE LEZIONI DA FILE \n");
    System.out.println("\t\t\t 2 ->  INSERIMENTO MANUALE \n");
    System.out.println("\t\t\t 3 ->  ESCI \n");
    controllo = SafeInput.interoUnoTre();

    switch (controllo) {
    case 1: {
        String file = "";
        //do while che termina nel momento in cui il file esiste o quando
passiamo il carattere
        //per tornare al menu
        do {
            System.out.println("Inserire il nome del File"
                + " (* per tornare al menu')");
            file = Input.readLine();
            if (!file.equals("")) {
                if (file.equals("*"))
                    return;
                else {
                    File f = new File(file);

                    if (f.exists()) {
                        reg.loadLezioni(file);
                    } else {
                        file = "";
                    }
                }
            }
        }
    }
    }
}

```



```

        }
        } while (file.equals(""));
        break;
    }
    case 2:
        // chiama il metodo relativo all'inserimento lezioni da tastiera
        inserimentoLezMan();
        break;

    case 3:
        // chiama il metodo relativo al salvataggio
        Salva();
        return;
    }
    return;
}

/**
 *
 * Metodo relativo all'eliminazione di una lezione del Registro Corrente.
 *
 */
public static void eliminaLezioniInterno() {
    //prima di procedere, controllo che il registro lezioni contenga qualche
    elemento, se e' vuoto
    //manda un messaggio di avvertimento e ritorna al menu precedente
    if (reg.getLezioni().size() > 0) {
        //chiama il metodo lezione, il quale ritorna l'indice della lezione
        scelta
        int index = Lezione();
        if (index >= 0) {
            return;
        } else {
            try {
                //inserisco il nome dello studente all'interno dell'elenco
                studenti
                //di una determinata lezione
                reg.getLezioni().removeLezioni(
                    reg.getLezioni().getElemReg().get(index));
                Salva();
                reg.loadTesto();
                //salvo e ricarico il file
                System.out.println("Lezione eliminata...\n");
            } catch (IOException e) {
                System.out.println("Eliminazione non riuscita!!");
            }
        }
    } else {
        System.out
            .println("Nel registro non ci sono Lezioni...quindi e'
            impossibile modificarne una.\n");
    }
}

```

```
// chiama il metodo relativo al salvataggio
Salva();
return;
}

/**
 *
 * Metodo relativo alla Modifica di una lezione del Registro Corrente.
 *
 */
public static void modifyLezioniInterno() {
    //prima di procedere, controllo che il registro lezioni contenga qualche
    elemento, se e' vuoto
    //manda un messaggio di avvertimento e ritorna al menu precedente
    if (reg.getLezioni().size() > 0) {
        int index = Lezione();
        if (index != -1)
            //chiamo il metodo che si occupa della modifica
            reg.getLezioni().modify(index);
    } else {
        System.out
            .println("Nel registro non ci sono Lezioni...quindi e'
impossibile modificarne una.\n");
    }
    return;
}

/**
 * Metodo ausiliario per gestire la selezione di uno <code>Studente</code>.
 *
 */
public static int Studente() {

    if (reg.getStudenti().getElemReg().size() <= 0) {
        System.out
            .println(" Nel Registro non ci sono Studenti...quindi e'
impossibile visualizzarli...\n ");
        return -1;
    }

    System.out.println(" Studenti :\n ");

    for (int i = 0; i < reg.getStudenti().size(); i++) {
        System.out.print("\t" + (i + 1) + " - "
            + reg.getStudenti().getElemReg().get(i).toString() + "\n");
    }

    int mat = 0;

    while (mat == 0) {
        System.out
            .println("Inserire il numero dello Studente ( -1 per uscire ) :");
    }
}
```

```
        mat = SafeInput.readInt();
    }
    if (mat == -1) {
        // System.out.println("Lezione inesistente...\n");
        return -1;
    }

    return mat - 1;
}

/**
 * Metodo ausiliario per gestire la selezione di uno <code>Lezione</code>.
 *
 */
private static int Lezione() {

    if (reg.getLezioni().getElemReg().size() <= 0) {
        System.out
            .println(" Nel Registro non ci sono lezioni...quindi e'
impossibile visualizzarle...\n ");
        return -1;
    }

    System.out.println("Elenco delle lezioni :");

    for (int i = 0; i < reg.getLezioni().size(); i++) {
        System.out.print("\t" + (i + 1) + " - "
            + reg.getLezioni().getElemReg().get(i).toString() + "\n");
    }
    int nome = 0;
    boolean inizializzata = false;
    while (!inizializzata) {
        System.out
            .println("Inserire il numero della lezione ( -1 per uscire ) :");
        nome = SafeInput.readInt();

        if (nome == -1)
            inizializzata = true;

        if (nome <= reg.getLezioni().size() + 1 && nome > 0)
            inizializzata = true;
        else
            System.out.println("Intero Immeso Errato..Reinserire...\n");
    }
    if (nome == -1) {
        // System.out.println("Lezione inesistente...\n");
        return -1;
    }

    return nome - 1;
}
```

```
/**
 * Metodo ausiliario per gestire l'inserimento da tastiere di una lezione.
 *
 */
public static void inserimentoLezMan() {
    Data date = new Data();

    while (date.equals(new Data())) {
        System.out
            .println("Inserire la data della lezione(vuoto per terminare l
'inserimento)");
        date = SafeInput.readDate();
    }

    if (!date.equals(new Data())) {
        System.out.println("\nInserire il Nome della lezione");
        String Nome = SafeInput.readLineSafe();
        System.out.println("\nInserire l'Argomento della lezione");
        String Arg = SafeInput.readLineSafe();

        // creazione della foto ed inserimento in archivio.
        Lezione l = new Lezione(date, Nome, Arg, new RegStudenti(), 0);
        try {
            reg.getLezioni().addLezione(l);
            System.out.println("Inserimento effettuato.\n");
        } catch (DuplicatedStudentiException e) {
            System.out
                .println("\nATTENZIONE!! foto gia' presente in archivio!! \n"
);
            System.out.println("Errore occorso durante l'inserimento.\n");
            System.out
                .println("Controlla se il nome della foto inserita e il
percorso dove cercare il file sono corretti.\n");
        }
    }
    // chiama il metodo relativo al salvataggio
    Salva();
    return;
}

/**
 * Metodo ausiliario per gestire l'inserimento da tastiera di uno studente.
 *
 */
public static void inserimentoStudMan() {
    String Cognome = "..";
    // while (!Cognome.equals("")) {
    System.out.println("Inserire il Cognome dello studente");
    Cognome = SafeInput.readLineSafe();
    // }

    System.out.println("\nInserire il Nome dello studente");
```

```
String Nome = SafeInput.readLineSafe();
System.out.println("\nInserire il numero di matricola");
int Mat = SafeInput.readInt();

// creazione della foto ed inserimento in archivio.
try {
    Studente s = new Studente(Cognome, Nome, Mat, new RegLezioni(), 0);
    reg.getStudenti().addStudenti(s);
} catch (NullPointerException e) {
    System.out
        .println("Dati immessi errati...Inserimento non riuscito\n");
    return;
} catch (DuplicatedStudentiException e) {
    System.out
        .println("Dati immessi errati...Inserimento non riuscito\n");
    return;
}
System.out.println("\n\t\tInserimento avvenuto con successo!!\n");
// chiama il metodo relativo al salvataggio
Salva();
}

/**
 * Metodo ausiliario per gestire l'eliminazione di uno <code>Lezione</code>.
 *
 */
public static void eliminaStudenteInterno() {

    if (reg.getStudenti().size() > 0) {
        int index = Studente();
        if (index == -1)
            return;
        else {
            reg.getStudenti().removeStudenti(
                reg.getStudenti().getElemReg().get(index));
            System.out.println("Studente eliminato...\n");
        }
    } else {
        System.out
            .println("Nel registro non ci sono Studenti...quindi e'
impossibile modificarne una.\n");
    }
    // chiama il metodo relativo al salvataggio
    Salva();
    return;
}

/**
 * Metodo ausiliario per gestire la selezione dello <code>Lezione</code> da
 * Modificare.
 *
 */
public static void modifyStudentiInterno() {
```

```
        if (reg.getStudenti().size() > 0) {
            int index = Studente();
            if (index == -1)
                return;
            else
                reg.getStudenti().modify(index);
        } else {
            System.out
                .println("Nel registro non ci sono Studenti...quindi e'
impossibile modificarne una.\n");
        }
        // chiama il metodo relativo al salvataggio
        Salva();
        return;
    }

    /**
     * Metodo che provvede alla selezione della <code>Lezione</code> di cui
     * desideriamo vedere le statistiche, e alla sua stampa su video.
     *
     */
    public static void statisticheLezioni() {

        char x = ' ';
        int index = Lezione();
        if (index < 0) {
            System.out.println("\t\t\t Uscire o riprovare? ( q / r)'\n");
            x = SafeInput.readChar();
            if (x == 'R' || x == 'r') {
                statisticheStudenti();
            } else
                return;
        } else {
            System.out.println(" Alla lezione : "
                + reg.getLezioni().getElemReg().get(index).toString()
                + " sono stati presenti i seguenti studenti...");
            System.out.println();
            System.out.println(reg.getLezioni().getElemReg().get(index)
                .getList().toString());
            System.out.println("Per un totale di "
                + reg.getLezioni().getElemReg().get(index).getNumPresenze()
                + " Studenti.");
            System.out.println();
        }
    }

    /**
     * Metodo che provvede alla selezione dello <code>Studente</code> di cui
     * desideriamo vedere le statistiche, e alla sua stampa su video.
     *
     */
    public static void statisticheStudenti() {
```

```

char x = ' ';
int index = Studente();
if (index < 0) {
    System.out
        .println("\t\t\t\t Uscire o riprovare? ( Qualsiasi tasto / r)'\n");
    x = SafeInput.readChar();
    if (x == 'R' || x == 'r') {
        statisticheStudenti();
    } else
        return;
} else {
    System.out.println("\n\tLo studente "
        + reg.getStudenti().getElemReg().get(index).toString()
        + " e' stato presente alle seguenti lezioni: ");
    System.out.println("-----");
    System.out.println(reg.getStudenti().getElemReg().get(index)
        .getList().toString());
    System.out.println("-----");
    System.out.println("Per un totale di "
        + reg.getStudenti().getElemReg().get(index)
        .getNumPresenze() + " presenze.\n");
    if (reg.getStudenti().getElemReg().get(index).getNumPresenze() >= reg
        .getNumObbligatorio())
        System.out
            .println("Tale numero di presenze permette allo studente "
                + reg.getStudenti().getElemReg().get(index)
                .toString()
                + " di esser ammesso all'esame finale.");
    else
        System.out
            .println("Tale numero di presenze non permette allo studente
"
                + reg.getStudenti().getElemReg().get(index)
                .toString()
                + " di esser ammesso all'esame finale.");
    System.out.println();
}
}

/**
 *
 * Metodo che provvede all'inserimento di una presenza, relativa ad uno
 * Studente esistente.
 */
public static void inserimentoPresenzeStudente() {
    if (reg.getLezioni().size() > 0 || reg.getStudenti().size() > 0) {
        // TODO Auto-generated method stub
        int index = Studente();
        if (index == -1)
            return;
    }
}

```

```

        else {
            int indexS = Lezione();
            if (reg.getLezioni().getElemReg().get(indexS).getList().size() < 0) {
            }
            try {
                reg.getStudenti().getElemReg().get(index).addElemList(
                    reg.getLezioni().getElemReg().get(indexS));
                reg.getLezioni().getElemReg().get(indexS).addElemList(
                    reg.getStudenti().getElemReg().get(index));
                System.out.println("Presenza inserita...\n");
                Salva();
                return;
            } catch (DuplicatedLezioniException e) {
                System.out.println("Studente gia presente...\n");
            } catch (DuplicatedStudentiException e) {
                System.out.println("Lezione gia presente...\n");
            }
        }
    } else {
        System.out
            .println("Nel registro non ci sono Lezioni o Studenti...quindi e'
impossibile inserire una presenza.\n");
    }
    return;
}

/**
 *
 * Metodo che provvede all'eliminazione di una presenza.
 *
 */
public static void eliminaPresenzeLezione() {

    int index = Lezione();
    if (reg.getLezioni().getElemReg().get(index).getList().size() <= 0) {
        System.out
            .println(" In tale lezione non è stato presente nessuno Studente,
quindi e' impossibile eliminarne uno...\n ");
        return;
    }
    if (index >= 0) {
        System.out.println(" Studenti :\n ");

        for (int i = 0; i < reg.getLezioni().getElemReg().get(index)
            .getList().size(); i++) {
            System.out.print("\t"
                + (i + 1)
                + " - "
                + reg.getLezioni().getElemReg().get(index).getList()
                    .getElemReg().get(i).toString() + "\n");
        }
    }
}

```



```

        int mat = 0;

        while (mat == 0) {
            System.out
                .println("Inserire il numero dello Studente ( -1 per uscire )
:");

            mat = SafeInput.readInt();
        }

        if (mat == -1)
            return;

        int indexS = mat - 1;

        try {
            System.out.println(reg.getStudenti().getElemReg().get(indexS)
                .toString());
            reg.getLezioni().getElemReg().get(index).removeElemList(
                reg.getStudenti().getElemReg().get(indexS));
            reg.getStudenti().getElemReg().get(indexS).removeElemList(
                reg.getLezioni().getElemReg().get(index));

            System.out.println("\t****Presenza eliminata****\n");
            Salva();
            return;
        } catch (NullPointerException e) {
            System.out
                .println("\t****Presenza non elimanta a causa di un'errore
interno****\n");
        }

    }

    return;
}

/**
 *
 * Metodo che provvede all'inserimento di una presenza, relativa ad una
 * lezione esistente.
 *
 */
public static void InserimentoPresenzeInternoEsistente() {

    int index = Lezione();
    if (index != -1) {
        char input = ' ';
        int i = 0;
        while (input != 'E' && i < reg.getStudenti().size()) {
            do {
                System.out.println(" Studente : "
                    + reg.getStudenti().getElemReg().get(i).toString());
                System.out

```

```

        .println("Aggiungere Presenza ? (Y / N / E per uscire
dall'inserimento)");
        input = SafeInput.readChar();
        if (input != 'Y' && input != 'N' && input != 'E'
            && input != 'y' && input != 'n')
            System.out
                .println("Input errato..inserisci Y oppure N oppure E
per uscire!!");

        } while (input != 'Y' && input != 'N' && input != 'E'
            && input != 'y' && input != 'n');

        if (input != 'E' || input != 'e') {
            if (input == 'Y' || input == 'y') {
                try {
                    reg.getLezioni().getElemReg().get(index)
                        .addElemList(
                            reg.getStudenti().getElemReg().get(
                                i));
                    reg.getStudenti().getElemReg().get(i).addElemList(
                        reg.getLezioni().getElemReg().get(index));
                } catch (DuplicatedLezioniException e) {
                    System.out.println("Studente gia presente...\n");
                    i++;
                } catch (DuplicatedStudentiException e) {
                    System.out.println("Lezione gia presente...\n");
                } catch (IllegalArgumentException e) {
                    System.out.println("Formato Dati Errati...\n");
                }
            }
        } else
            return;

        i++;
    }
    System.out.println("Presenze inserite...\n");
    // chiama il metodo relativo al salvataggio
    Salva();
    return;
} else
    return;

}

/**
 *
 * Metodo che provvede al Salvataggio del Registro attuale sia in formato
 * Testuale che Binario.
 *
 */
public static void Salva() {
    // Salvo il Registro Corrente, sia nel file binario che in
    // quello testuale

```

```
try {
    if (reg != null) {
        reg.saveTesto();
        reg.saveBinario();
    }
} catch (java.io.IOException e) {
    System.out
        .println("Errore durante il salvataggio nell'archivio!\n");
} catch (ClassNotFoundException e) {
    System.out
        .println("Errore durante il salvataggio nell'archivio!\n");
}
}
```