Relazione del Progetto dell'esame di Programmazione [TWM]

Analisi del problema e descrizione della soluzione adottata

Il progetto consiste in una interfaccia utente grafica con pulsanti e aree di testo che permettono di leggere i voti degli studenti da un file, e di analizzarli creando le medie e un istogramma delle frequenze dei voti. Inoltre viene richiesta la possibilità di ordinare o meno i voti e le medie, e di salvare le medie e l'istogramma.

Per l'interfaccia si è scelto di utilizzare solamente il pacchetto *java.awt* per questioni di praticità in quanto visto parzialmente a lezione. Sono quindi state create delle *TextArea* per la visualizzazione di voti, medie e istogramma; per fare in modo che queste ultime mantenessero la stessa dimensione proporzionata e si adattassero alle dimensioni della finestra, è stato usato un layout a griglia (*GridLayout*) con una riga e tre colonne. Per far interagire l'utente con i dati sono stati aggiunti dei *Button* (per leggere i voti da file e calcolare medie e istogramma) e delle *Checkbox* (per ordinare o meno i voti e le medie), queste ultime sono state scelte per migliorare l'usabilità poiché permettono all'utente di visualizzare facilmente lo stato di ordinamento dei dati. È stata infine aggiunta una *MenuBar* in cui si trova nuovamente la possibilità di ordinare i dati e in aggiunta vi è anche quella di salvarli su file; anche qui per questioni di miglioramento dell'usabilità si è scelto di far stampare all'utente il contenuto delle aree di testo visualizzate.

Per l'ordinamento dei dati si è scelto l'algoritmo ricorsivo *MergeSort*, ritenuto adatto in quanto ha una complessità temporale nel caso peggiore di O(nlogn); migliore rispetto ad altri algoritmi come l'InsertionSort e il SelectionSort che nel peggiore dei casi hanno complessità O(n^2).

Per l'istogramma si è dovuto modificare il font scegliendone uno monospace per avere la corretta rappresentazione grafica.

Per calcolare le medie è stato creato un algoritmo (di complessità O(n^2)) che per ogni nome controlla tutti i successivi: quando due nomi sono uguali somma i voti ed elimina il nome corrispondente, in questo modo evita di calcolare più medie per lo stesso alunno. Le medie sono state arrotondate a due cifre dopo la virgola, per una più facile interpretazione dei dati, usando il metodo *rint* della classe Math; non è stato utilizzato il metodo *round* perché ritenuto più impreciso.

In fase di acquisizione dati da file è stato usato il metodo *split* della classe String per "spezzare" più volte la stringa iniziale in modo da creare una matrice di dati (chiave-valore).

Gli errori dovuti a errato utilizzo dell'interfaccia vengono gestiti e appaiono nella riga di comando. Altrimenti si avrebbe potuto usare la classe JOptionPane per farli apparire in una nuova finestra (ma non volendo usare le Swing non è stato fatto).

Listato completo del programma

Progetto.java

```
import java.io.*;

/**
*
* @author Eleonora Macuglia
*/
public class Progetto {

    public static String percorsoInputFile = "";
    public static String percorsoOutMediaFile = "";
    public static String percorsoOutIstoFile = "";

    /**
    *
    * Classe iniziale
    *
    * @param args gli argomenti della linea di comando
    */
    public static void main(String args[]) {

        if (args.length < 3){
            System.out.println("Argomenti insufficienti");
            System.out.println("Utilizzo del programma: java NomeProgramma FileConVoti
FileMedie FileIstogramma");
            System.exit(0);
        }else{
            percorsoInputFile = args[0];
            percorsoOutMediaFile = args[1];
            percorsoOutIstoFile = args[2];

            GUI GUIProgetto = new GUI();
        }
    }

    /**
     * Classe per ordinare una matrice bidimensionale
     *
     * @param input matrice da ordinare
     * @return la matrice ordinata
     */
    public static String[][] ordinaMatrice(String[][] input){

        String[][] output = new String[input.length][2];

        for(int i = 0; i < input.length; i++){
            output[i][0] = input[i][0];
            output[i][1] = input[i][1];
        }

        mergeSort(output);
        // Utilizzo mergesort, ordino la prima colonna della matrice

        return output;
```

```
    }

    /**
     * Algoritmo di ordinamento MERGESORT
     * O(nlogn)
     *
     * @param array da ordinare
     */
    private static void mergeSort(String[][] array){

        if(array.length>=2){ // Divide finche' non si ottengono array da "fondere" di
dimensione 1

            String[][] sinistra = new String[array.length/2][2];  // Creo l'array a
sinsitra con la meta' dei nomi
            for(int i = 0; i < sinistra.length; i++){  // Copio la prima parte
                sinistra[i][0] = array[i][0];
                sinistra[i][1] = array[i][1];
            }

            String[][] destra = new String[array.length - sinistra.length][2]; // Creo
l'array a destra con l'altra meta' dei nomi
            for(int i = 0; i < destra.length; i++){  // Copio la seconda parte
                destra[i][0] = array[sinistra.length + i][0];
                destra[i][1] = array[sinistra.length + i][1];
            }

            mergeSort(sinistra); // Ricorsione fino a array di lunghezza 1
            mergeSort(destra);
            merge(array, sinistra, destra);
        }
    }
    private static void merge(String[][] array, String[][] sinistra, String[][]
destra){
        int i = 0;
        int j = 0;

        for(int indice=0; indice < array.length; indice++){
            // SE (j >= destra.length) e' VERO allora significa che ho "finito" tutti
gli elementi a destra, quindi il resto e' per forza a sinistra
            // SE (i< sinistra.length) viceversa
            if((j >= destra.length) || ((i< sinistra.length) &&
sinistra[i][0].compareToIgnoreCase(destra[j][0]) <0)){ // compareToIgnoreCase per
ignorare le maiuscole
               array[indice][0] = sinistra[i][0];
               array[indice][1] = sinistra[i][1];

               i++; // indice relativo a sinistra
            }else{
               array[indice][0] = destra[j][0];
               array[indice][1] = destra[j][1];

               j++; // indice relativo a destra
            }
        }
    }

    /**
     * Classe per riempire una matrice con i dati caricati da file
     *
```

```
     * @param percorso percorso del file da caricare
     * @return la matrice biedimensionale di chiavi-valore
     */
    public static String[][] caricaDaFile(String percorso){
        String contenuto = "";
        try{
            FileInputStream f = new FileInputStream(percorsoInputFile);

            int tmp = f.read();  //leggo il primo byte per riempire il buffer caratteri
tmp
            while (tmp > 0){
                contenuto += (char)tmp;
                tmp = f.read();
            }
            f.close();
        }catch(IOException e){
            System.out.println("File di input inesistente! path: " +
percorsoInputFile);
        }

        String[] righe = contenuto.split("\\r?\\n");  // Divido la stringa in base alle
righe
        String[][] matriceFinale = new String[righe.length][2];

        for (int i=0; i<righe.length; i++){
            String[] valori = righe[i].split(" ");  //separo gli elementi della stringa
in coppie chiave-valore (separati da uno spazio nel file)
            if (valori.length < 2){
                System.out.println("Dati in input non validi");
            }else{
                matriceFinale[i][0] = valori[0];
                matriceFinale[i][1] = valori[1];
                try{
                    int tmp = java.lang.Integer.parseInt(valori[1]);
                }catch(Exception e){
                    System.out.println("Voto non valido");
                }
            }
        }

        return matriceFinale;
    }

    /**
     * Classe per scrivere dei dati in un file
     *
     * @param percorso percorso del file
     * @param stringaDaScrivere stringa da scrivere sul file
     */
    public static void scriviSuFile(String percorso, String stringaDaScrivere){
        try{
            FileOutputStream file = new FileOutputStream(percorso, false);
            PrintStream output = new PrintStream(file);

            output.print(stringaDaScrivere);

            output.flush();
            output.close();

            System.out.println("Dati salvati in " + percorso);
```

```
        }catch(FileNotFoundException e){
            System.out.println("Il file " + percorso + " non puo' essere creato");
        }
    }

}
```

GUI.java

```
import java.awt.*;
import java.awt.event.*;

/**
*
*@author Eleonora Macuglia
*/
public class GUI extends Frame{
    private TextArea tVoti, tMedie, tIsto;
    private Checkbox checkOrdinaVoti, checkOrdinaMedie;

    /**
    *
    * Costruttore
    */
    public GUI(){
        super();
        setBounds(200, 200, 800, 400);
        setTitle("Progetto");

        addWindowListener(new GUIEvents());

        Panel panelNorth = new Panel();
        Panel panelSouth = new Panel();
        Panel panelCenter = new Panel();

        Button btnLeggiVoti = new Button("Leggi voti");
        btnLeggiVoti.addActionListener(new ClickEventLeggiVoti());

        Button btnCalcola = new Button("Calcola");
        btnCalcola.addActionListener(new ClickEventCalcola());

        Button btnEsci = new Button("Esci");
        btnEsci.addActionListener(new ClickEventEsci());

        tVoti = new TextArea();
        tMedie = new TextArea();
        tIsto = new TextArea();
        tIsto.setFont(new Font("monospaced", Font.PLAIN, 12)); //Setto il font
dell'istogramma su un font monospace con dimensione 12 e font PLAIN

        MenuBar menuBar = new MenuBar();
        Menu menuFile = new Menu("File");
        Menu menuModifica = new Menu("Modifica");

        MenuItem pulsOrdinaVoti = new MenuItem("Ordina voti");
        pulsOrdinaVoti.addActionListener(new ClickEventBtnOrdinaVoti());
        MenuItem pulsOrdinaMedie = new MenuItem("Ordina medie");
        pulsOrdinaMedie.addActionListener(new ClickEventBtnOrdinaMedie());
```

```
        menuModifica.add(pulsOrdinaVoti);
        menuModifica.add(pulsOrdinaMedie);

        MenuItem pulsSalva = new MenuItem("Salva");
        pulsSalva.addActionListener(new ClickEventSalva());
        MenuItem pulsEsci = new MenuItem("Esci");
        pulsEsci.addActionListener(new ClickEventEsci());;

        menuFile.add(pulsSalva);
        menuFile.add(pulsEsci);

        menuBar.add(menuFile);
        menuBar.add(menuModifica);

        setMenuBar(menuBar);

        checkOrdinaVoti = new Checkbox("Ordina voti", false);
        checkOrdinaVoti.addItemListener(new ItemListener(){
            public void itemStateChanged(ItemEvent ie){
                isVotiOrdinati = checkOrdinaVoti.getState();
                ChangeOrdinamentoVoti(isVotiOrdinati);
            }
        });

        checkOrdinaMedie = new Checkbox("Ordina medie", false);
        checkOrdinaMedie.addItemListener(new ItemListener(){
            public void itemStateChanged(ItemEvent ie){
                isMedieOrdinate = checkOrdinaMedie.getState();
                ChangeOrdinamentoMedie(isMedieOrdinate);
            }
        });

        add(panelNorth, BorderLayout.NORTH);
        panelNorth.add(btnLeggiVoti);
        panelNorth.add(btnCalcola);
        panelNorth.add(checkOrdinaVoti);
        panelNorth.add(checkOrdinaMedie);

        add(panelCenter, BorderLayout.CENTER);
        panelCenter.setLayout(new GridLayout(1,3)); //creo layout a griglia: 1 riga e 3
colonne
        panelCenter.add(tVoti);
        panelCenter.add(tMedie);
        panelCenter.add(tIsto);

        add(panelSouth, BorderLayout.SOUTH);
        panelSouth.add(btnEsci);

        setVisible(true);
    }

    private boolean isVotiOrdinati = false;
    private boolean isMedieOrdinate = false;
    private RegistroVoti registroVoti = null;
    private Istogramma nuovoIstogramma = null;
    private String[][] ContenutoInputFile; // Rimane NON ordinato

    /**
    *
```

```
    * Ascoltatore per la GUI
    */
    class GUIEvents extends WindowAdapter {
        public void windowClosing(WindowEvent e) {
            System.out.println("Window closing");
            System.exit(0);
        }
    }

    /**
    *
    * Ascoltatore per il pulsante "Leggi voti"
    */
    public class ClickEventLeggiVoti implements ActionListener{
        public void actionPerformed(ActionEvent e){

            ContenutoInputFile = Progetto.caricaDaFile(Progetto.percorsoInputFile);

            String out = "";
            for(int i = 0; i < ContenutoInputFile.length; i++){
                out += ContenutoInputFile[i][0] + " " + ContenutoInputFile[i][1] +
"\n";
            }

            tVoti.setText(out);
        }
    }

    /**
    *
    * Ascoltatore per il pulsante "Calcola"
    */
    public class ClickEventCalcola implements ActionListener{
        public void actionPerformed(ActionEvent e){
            try{
                registroVoti = new RegistroVoti(ContenutoInputFile);
                tMedie.setText(registroVoti.toString());

                nuovoIstogramma = new Istogramma(ContenutoInputFile);
                tIsto.setText(nuovoIstogramma.toString());
            }catch(NullPointerException error){
                System.out.println("Devi prima leggere i voti!");
            }
        }
    }

    /**
    *
    * Ascoltatore per il pulsante "Esci"
    */
    public class ClickEventEsci implements ActionListener{
        public void actionPerformed(ActionEvent e){
            System.exit(0);
        }
    }

    /**
    *
    * Ascoltatore per il pulsante "Salva"
    */
```

```
    public class ClickEventSalva implements ActionListener{
        public void actionPerformed(ActionEvent e){
            Progetto.scriviSuFile(Progetto.percorsoOutMediaFile, tMedie.getText()); //
Potrei usare come argomento registroVoti.toString(), ma voglio salvare quello
effettivamente mostrato
            Progetto.scriviSuFile(Progetto.percorsoOutIstoFile, tIsto.getText());
        }
    }

    /**
    *
    * Ascoltatore per il pulsante "Ordina voti"
    */
    public class ClickEventBtnOrdinaVoti implements ActionListener{
        public void actionPerformed(ActionEvent e){

                ChangeOrdinamentoVoti(!isVotiOrdinati);
        }
    }

    /**
    *
    * Metodo per passare da un ordinamento all'altro dei voti
    *
    * @param stato indica se e' ordinata(true) o non ordinata(false)
    */
    private void ChangeOrdinamentoVoti(boolean stato){
        try{
            isVotiOrdinati = stato;
            checkOrdinaVoti.setState(isVotiOrdinati);
            String[][] daVisualizzare = null;

            if(isVotiOrdinati){
                daVisualizzare = Progetto.ordinaMatrice(ContenutoInputFile);
                //visualizzione ordinata
            }else{
                daVisualizzare = ContenutoInputFile;
                //visualizzazione NON ordinata
            }

            // VISUALIZZAZIONE del contenuto di daVisualizzare
            String out = "";
            for(int i = 0; i < daVisualizzare.length; i++){
                out += daVisualizzare[i][0] + " " + daVisualizzare[i][1] + "\n";
            }

            tVoti.setText(out);

        }catch(NullPointerException error){
            System.out.println("Devi prima leggere i voti!");
        }
    }

    /**
    *
    * Ascoltatore per il pulsante "Odina medie"
    */
    public class ClickEventBtnOrdinaMedie implements ActionListener{
        public void actionPerformed(ActionEvent ae){
                ChangeOrdinamentoMedie(!isMedieOrdinate);
```

```
        }
    }

    /**
    *
    * Metodo per passare da un ordinamento all'altro delle medie
    *
    * @param stato indica se e' ordinata(true) o non ordinata(false)
    */
    public void ChangeOrdinamentoMedie(boolean stato){
        try{
            isMedieOrdinate = stato;
            checkOrdinaMedie.setState(isMedieOrdinate);
            String[][] daVisualizzare = null;

            if(isMedieOrdinate){
                daVisualizzare =
Progetto.ordinaMatrice(registroVoti.registroVotiMedie);
                //visualizzione ordinata
            }else{
                daVisualizzare = registroVoti.registroVotiMedie;
                //visualizzazione NON ordinata
            }

            // VISUALIZZAZIONE del contenuto di "daVisualizzare"
            String out = "";
            for(int i = 0; i < daVisualizzare.length; i++){
                out += daVisualizzare[i][0] + " " + daVisualizzare[i][1] + "\n";
            }

            tMedie.setText(out);

        }catch(NullPointerException e){
            System.out.println("Devi prima calcolare le medie!");
        }
    }
}
```

RegistroVoti.java

```
import java.lang.*;
import java.io.*;

/**
 *
 * @author Eleonora Macuglia
 */
public class RegistroVoti
{
    public String[][] registroVoti; // matrice pubblica per eventuale utilizzo,
attualmente non è necessario emetterla pubblica
    public String[][] registroVotiMedie;

    /**
    *
    * Costruttore
    */
    public RegistroVoti(String[][] matriceInput){
```

```
        registroVoti = new String[matriceInput.length][2]; // Mi copio l'array perche'
andro' a modificarlo
        for(int i = 0; i < matriceInput.length; i++){
            registroVoti[i][0] = matriceInput[i][0];
            registroVoti[i][1] = matriceInput[i][1];
        }

        String[][] tmpMatrice = new String[registroVoti.length][2];

        String tmpNome = "";
        float sommaVoti = 0;
        float numeroVoti = 0;
        int totaleStudenti = 0;

        /*
        Algoritmo che fa le medie di ogni studente mantenendo l'ordine con cui sono
elencati i nomi (no ordine alfabetico)
        complessita' O(n^2)
        */
        for(int i = 0; i < registroVoti.length; i++)
        {
            if(registroVoti[i][0].length() == 0) //Il nome e' vuoto, e' già stato
"usato"! continuo il ciclo
                continue;

            tmpNome = registroVoti[i][0];

            // Ciclo che va' a trovare tutti i voti dell'attuale studente (variabile
tmpNome)
            for(int j = 0; j < registroVoti.length; j++){
                if(registroVoti[j][0].length() == 0)
                  continue;

                if(tmpNome.compareTo(registroVoti[j][0]) == 0){
                    int voto  = java.lang.Integer.parseInt(registroVoti[j][1]); // In
fase di caricamento dei dati avevo gia' controllato che il contenuto della cella fosse
effettivamente un numero! lo salto e lo do' per buono!
                    sommaVoti += voto;
                    numeroVoti++;
                    registroVoti[j][0] = "";
                }
            }

            // Salvo lo studente nell'array temporaneo "grande"
            float tmpMediaVoto = (sommaVoti/numeroVoti);
            tmpMatrice[totaleStudenti][0] = tmpNome;
            tmpMatrice[totaleStudenti][1] = Math.rint(tmpMediaVoto*100.0)/100.0 +""; //
Math.rint arrotonda al numero pari piu' vicino (se equidistante)

            sommaVoti = 0;
            numeroVoti = 0;

            totaleStudenti++; // e' un nuovo studente!
        }

        //Non e' strettamente necessario ma cosi' ho sempre a disposizione un
registroVoti "pulito"
        for(int i = 0; i < matriceInput.length; i++){
            registroVoti[i][0] = matriceInput[i][0];
            registroVoti[i][1] = matriceInput[i][1];
```

```
        }

        // Copio gli studenti dall'array temporaneo in quello "definitivo"
        registroVotiMedie = new String[totaleStudenti][2];
        for(int i= 0; i <totaleStudenti; i++){
            registroVotiMedie[i][0]= tmpMatrice[i][0];
            registroVotiMedie[i][1]= tmpMatrice[i][1];
        }
    }

    /**
    *
    * Sovrascrivo il metodo toString
    */
    @Override public String toString(){

        String output = "";

        for(int i= 0; i <registroVotiMedie.length; i++){
                    output += registroVotiMedie[i][0] + " " + registroVotiMedie[i][1]
+"\r\n";
        }

        return output;
    }
}
```

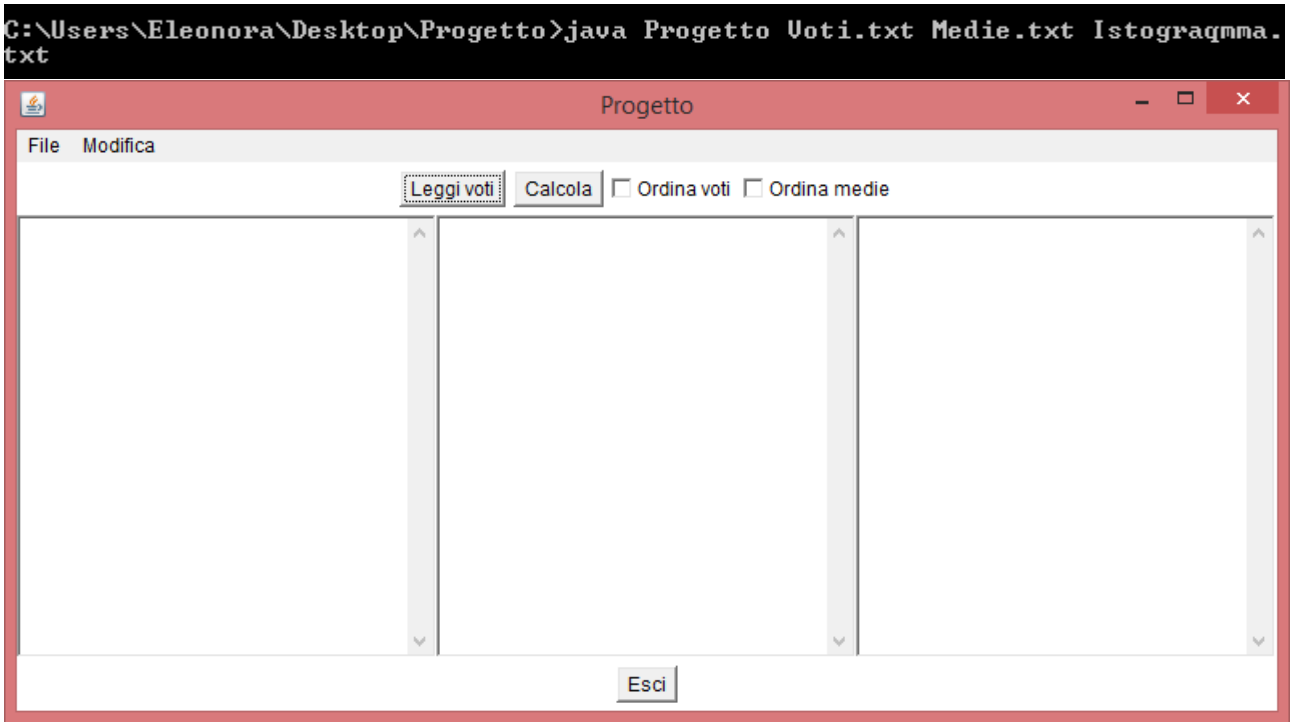
Istogramma.java

```
import java.io.*;

/**
 *
 * @author Eleonora Macuglia
 */
public class Istogramma
{
    public String[][] registroVoti; // matrice pubblica per eventuale utilizzo,
attualmente non è necessario emetterla pubblica

    /**
    *
    * Costruttore
    */
    public Istogramma(String[][] matriceInput){
        registroVoti = matriceInput;
    }

    /**
    *
    * Sovrascrivo il metodo toString
    */
    @Override public String toString(){
        byte[] conteggioVoti = new byte[10];
        String output = "";
        byte piuAlto = 0;

        //conto la frequenza dei voti e la salvo nell'array "conteggioVoti"
        for(int i = 0; i < registroVoti.length; i++){
```

```
            int voto = java.lang.Integer.parseInt(registroVoti[i][1]);
            conteggioVoti[voto-1] ++;
        }

        //setto piuAlto in base al voto con frequenza maggiore
        for(int i = 0; i < conteggioVoti.length; i++){
            if(piuAlto < conteggioVoti[i]){
                piuAlto = conteggioVoti[i];
            }
        }

        for(int i=piuAlto; i>0; i--){
            String sub = "";
            for(int j=0; j < conteggioVoti.length; j++){
                if(conteggioVoti[j] >= i){
                    sub += "* ";
                }else{
                    sub += "  ";
                }
            }
            output += sub + "\r\n";
        }

        output += "1 2 3 4 5 6 7 8 9 10";

        return output;
    }

}
```

Prova di esecuzione

Quando il programma viene eseguito da terminale si apre inizialmente una interfaccia vuota.

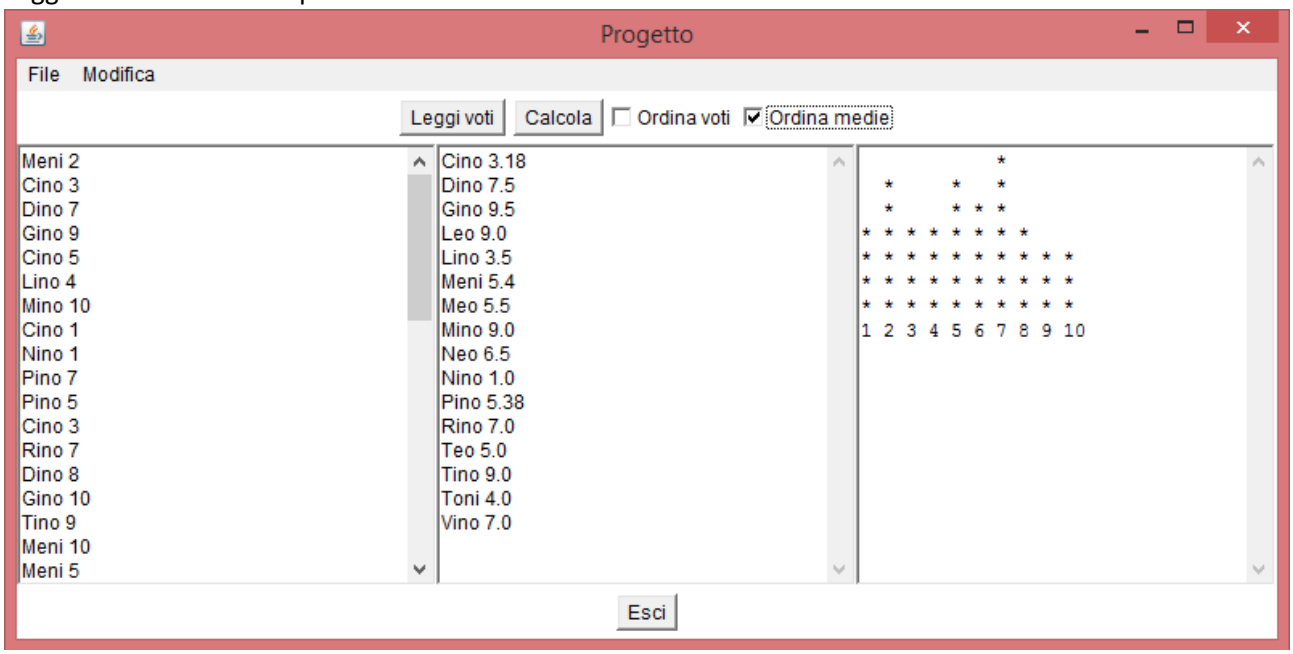


L'utente prima di compiere qualsiasi altra azione deve leggere i dati dal file cliccando sul pulsante "Leggi voti", in caso contrario appariranno dei messaggi di errore.

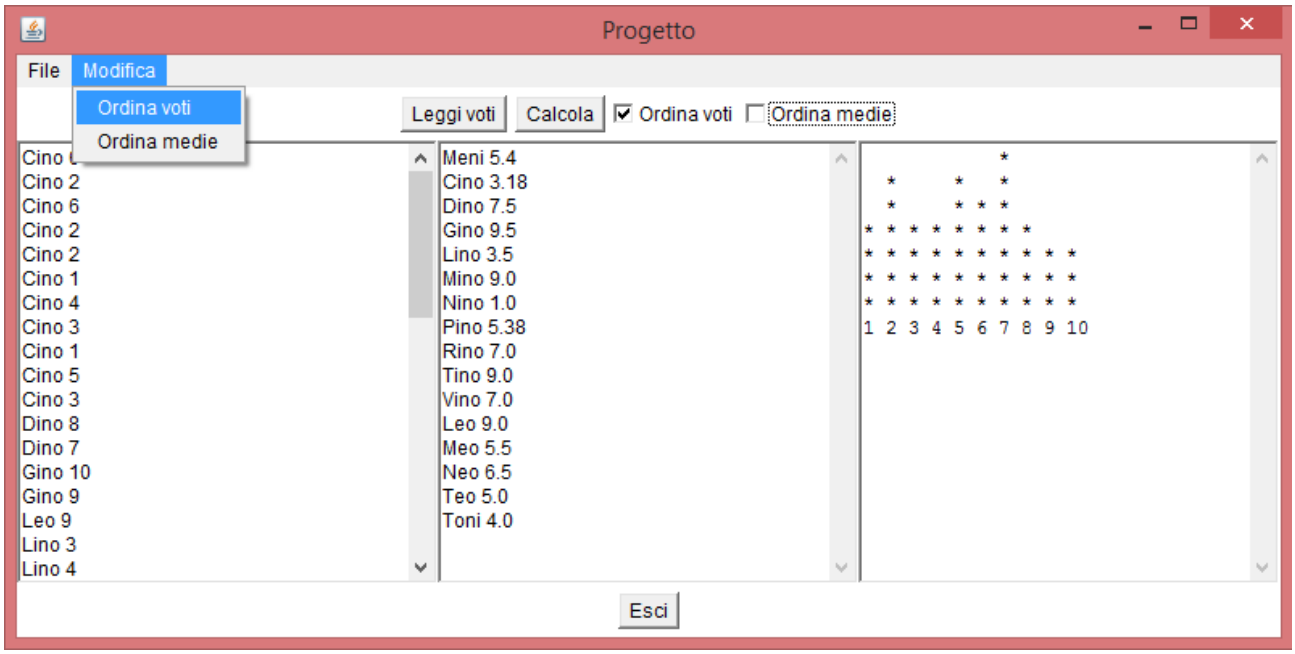
Una volta caricati i dati è possibile compiere le varie operazioni di calcolo e ordinamento.

Esempi:

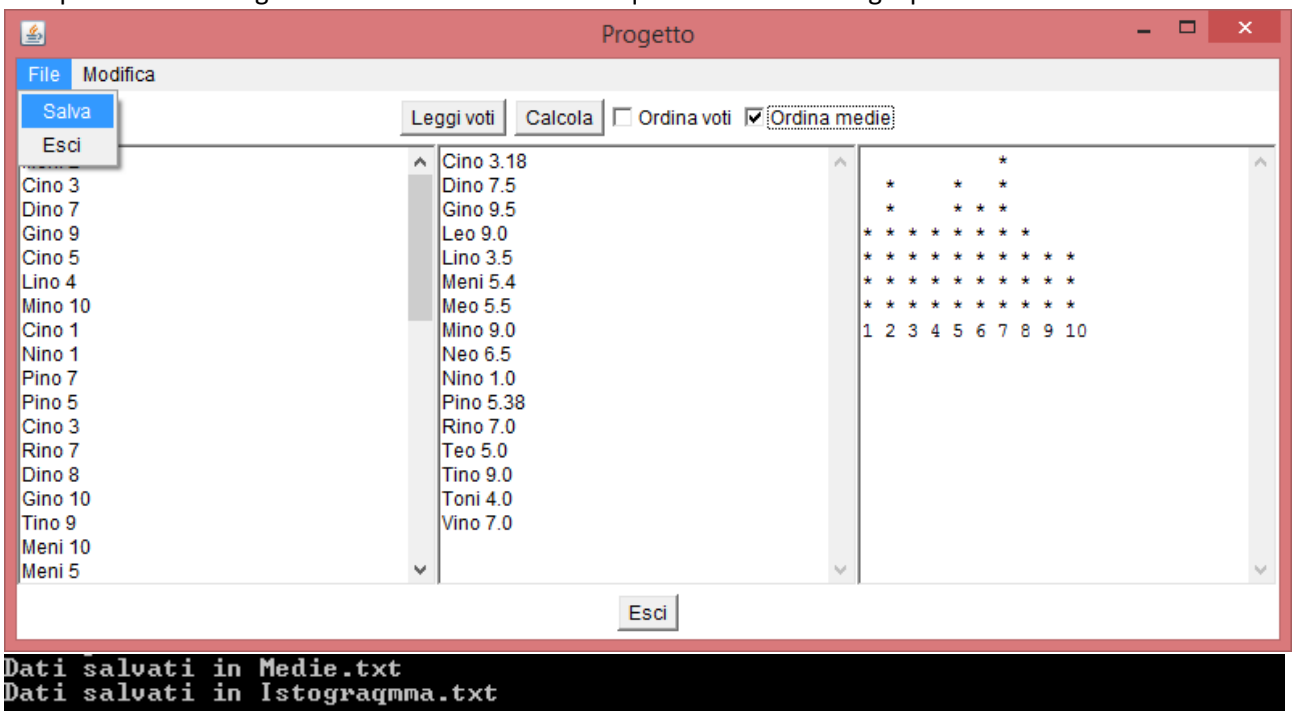
Leggi voti >> Calcola >> spunta Ordina medie

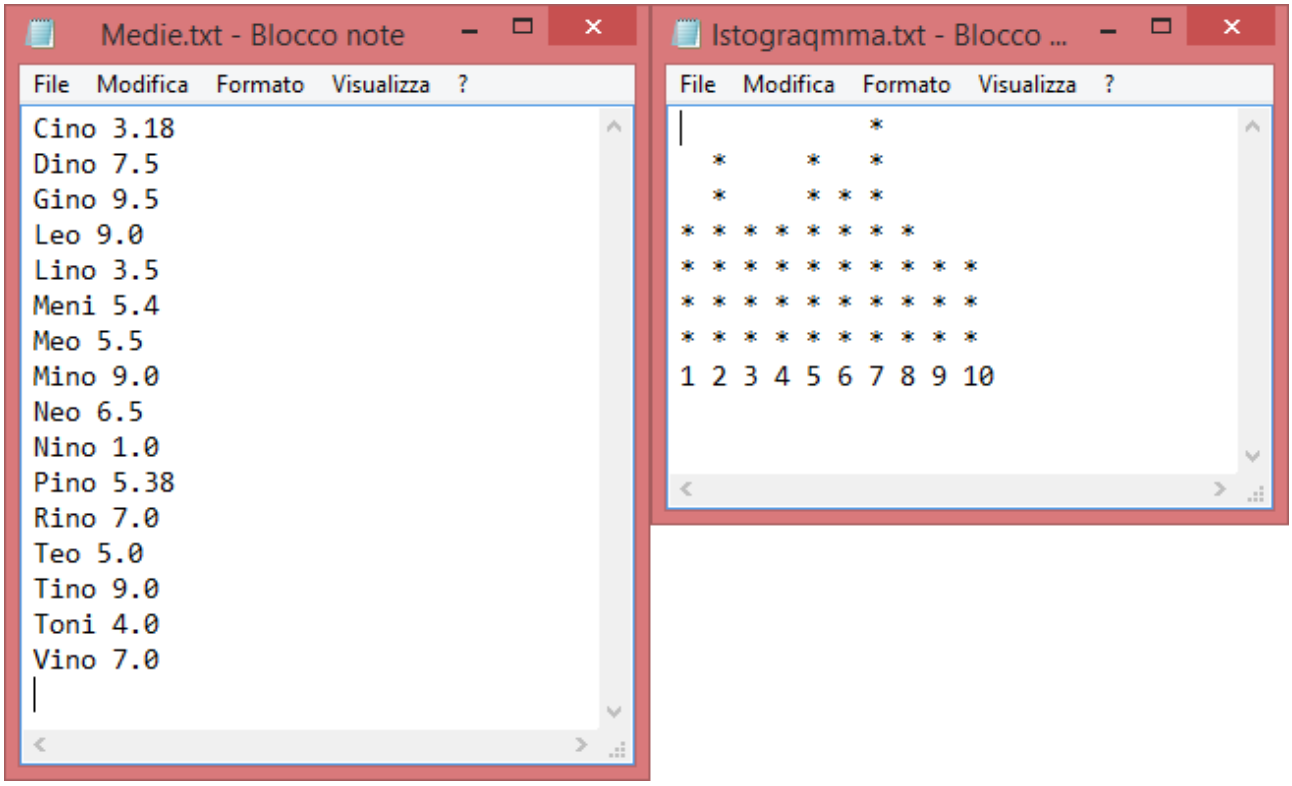


Leggi voti >> Calcola >> Modifica >> Ordina voti



Infine l'utente può salvare i due file contenenti medie e istogramma: con File >> Salva verranno creati i due file specificati nella riga di comando iniziale. In caso questi due file siano già presenti verranno sovrascritti.




Medie.txt - Blocco note
File Modifica Formato Visualizza ?
Cino 3.18
Dino 7.5
Gino 9.5
Leo 9.0
Lino 3.5
Meni 5.4
Meo 5.5
Mino 9.0
Neo 6.5
Nino 1.0
Pino 5.38
Rino 7.0
Teo 5.0
Tino 9.0
Toni 4.0
Vino 7.0
Istograqmma.txt - Blocco ...
File Modifica Formato Visualizza ?
1 2 3 4 5 6 7 8 9 10