```java
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import java.io.*;
import java.util.*;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableColumn;



public class Finestra extends JFrame
{

    private static Vector elenco = new Vector (1,1); //vettore che conterrà le voci del file
    private static String nomeFile = ""; //conterrà il nome del file da aprire
    private static final String msg = "Messaggio : "; //stringa iniziale per i messaggi delle azioni
effettuate
    private static boolean modificato = false; //serve per gestire se salvare o no il file
    private String estensioneFile = ".rbc"; //estensione del file
    private static String percorso = ""; //percorso dell'ultima cartella aperta
    private static String nomeFileTemp = ""; //nome del file senza percorso

    private JPanel pannello = new JPanel(); //pannello principale

    private JMenuBar mb = new JMenuBar (); //barra dei menu
    //voci della barra dei menu
    private JMenu file = new JMenu ("File");
    private static JMenu modifica = new JMenu ("Modifica");
    private static JMenu cerca = new JMenu ("Cerca");
    private JMenu help = new JMenu ("Help");
    //item del menu file
    private JMenuItem apri_ = new JMenuItem ("Apri");
    private JMenuItem nuovo_ = new JMenuItem ("Nuovo");
    private static JMenuItem salva_ = new JMenuItem ("Salva");
    private JMenuItem chiudi_ = new JMenuItem ("Chiudi");
    //item del menu modifica
    private JMenuItem inserisci_ = new JMenuItem ("Inserisci");
    private static JMenuItem rinomina_ = new JMenuItem ("Rinomina");
    private static JMenuItem elimina_ = new JMenuItem ("Elimina");
    //item del menu cerca
    private JMenuItem ricerca_ = new JMenuItem ("Ricerca");
    private JMenuItem visualizza_ = new JMenuItem ("Visualizza");
    //item del menu help
    private JMenuItem info_ = new JMenuItem("Info");

    private static JLabel messaggi = new JLabel(); //messaggi

    static DefaultTableModel model = null; //da utilizzare per gestire la JTable in modo dinamico
    private JTable tabella = null; //tabella che conterrà i dati per la visualizzazione
    private JScrollPane scrollPaneTabella; //scroll della tabella

    static String nomiColonne[] = {"ind.","Cognome","Nome","Telefono","E-Mail"}; //nomi delle colonne
della tabella



    public Finestra () //costruttore
        {
            //aggiunta degli item ai menu
            file.add(apri_);
            file.add(nuovo_);
            file.add(salva_);
            file.addSeparator();
            file.add(chiudi_);
            modifica.add(inserisci_);
            modifica.add(rinomina_);
            modifica.add(elimina_);
```

```
            cerca.add(ricerca_);
            cerca.add(visualizza_);
            help.add(info_);
            //aggiunta dei menu alla menubar
            mb.add(file);
            mb.add(modifica);
            mb.add(cerca);
            mb.add(help);
            //mnemonici per i tasti rapidi
            file.setMnemonic('f');
            modifica.setMnemonic('m');
            cerca.setMnemonic('c');
            help.setMnemonic('h');
            apri_.setMnemonic('a');
            nuovo_.setMnemonic('n');
            salva_.setMnemonic('s');
            chiudi_.setMnemonic('e');
            inserisci_.setMnemonic('i');
            rinomina_.setMnemonic('r');
            elimina_.setMnemonic('d');
            ricerca_.setMnemonic('t');
            visualizza_.setMnemonic('v');
            info_.setMnemonic('x');
            this.setJMenuBar(mb);   // aggiunta del menubar al pannello

            //associazione degli ascoltatori sugli item dei menu
            Ascoltatore oggettoAscoltatore = new Ascoltatore();
            apri_.addActionListener(oggettoAscoltatore);
            nuovo_.addActionListener(oggettoAscoltatore);
            salva_.addActionListener(oggettoAscoltatore);
            chiudi_.addActionListener(oggettoAscoltatore);
            inserisci_.addActionListener(oggettoAscoltatore);
            rinomina_.addActionListener(oggettoAscoltatore);
            elimina_.addActionListener(oggettoAscoltatore);
            ricerca_.addActionListener(oggettoAscoltatore);
            visualizza_.addActionListener(oggettoAscoltatore);
            info_.addActionListener(oggettoAscoltatore);

            salva_.setEnabled(false);
            modifica.setEnabled(false);
            cerca.setEnabled(false);

            String tabStringhe[][]= new String[elenco.size()][nomiColonne.length];
            caricaTabella(elenco,tabStringhe); //caricamento dei dati da inserire nella tabella

            //creazione della tabella con model
            model = new DefaultTableModel(tabStringhe,nomiColonne);
            tabella = new JTable( model);

            TableColumn column = tabella.getColumnModel().getColumn(0); // indice
            column.setPreferredWidth(20);
            column = tabella.getColumnModel().getColumn(2); // Cognome
            column.setPreferredWidth(200);
            column = tabella.getColumnModel().getColumn(1); // Nome
            column.setPreferredWidth(200);
            column = tabella.getColumnModel().getColumn(3); // Telefono
            column.setPreferredWidth(200);
            column = tabella.getColumnModel().getColumn(4); // email
            column.setPreferredWidth(200);

            scrollPaneTabella = new JScrollPane( tabella ); //aggiungo la tabella allo scroll

            messaggi.setText(msg + "Nessun file aperto!!"); //visualizzazione dei messaggi

            this.setVisible(true); //rendo visibile la finestra
            this.setTitle("Rubrica telefonica"); //titolo della finestra
            this.setBounds(20,40,1200,530); //posizionamento della finestrapos_x,pos_y, dim_x, dim_y
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  //aggiunta dei tasti di default per
chiudere o ridurre la finestra
            this.addWindowListener(new GestoreFinestra()); //aggiungo gli ascoltatori ai tasti della
finestra
```

```
        getContentPane().add( scrollPaneTabella, BorderLayout.CENTER);
        getContentPane().add(messaggi,  BorderLayout.SOUTH);


    } //fine costruttore


class Ascoltatore implements ActionListener //gestore degli ascolti sugli item
 {
  public void actionPerformed(ActionEvent oggettoEvento)
   {
     String scelta= oggettoEvento.getActionCommand(); //ricavo la stringa
     //eventi item del menu  file
     if (scelta.equals("Apri")) gestoreApri() ;
     if (scelta.equals("Nuovo")) gestoreNuovo();
     if (scelta.equals("Salva")) gestoreSalva();
     if (scelta.equals("Chiudi")) gestoreChiudi();
     //eventi item del menu modifica
     if (scelta.equals("Inserisci")) gestoreInserisci();
     if (scelta.equals("Rinomina")) gestoreRinomina();
     if (scelta.equals("Elimina")) gestoreElimina();
     //eventi item del menu cerca
     if (scelta.equals("Visualizza")) gestoreVisualizza();
     if (scelta.equals("Ricerca")) gestoreRicerca();
     //eventi item del menu help
     if (scelta.equals("Info")) gestoreInfo();
   }
 } //fine classe ascoltatore

  public static void gestoreChiudi() //chiude il programma e permette di salvare i dati
  {
      int scegli = 1;
        if (modificato)
            {
             scegli = Visual.conferma("Salvare le ultime modifiche ?");
            }
        // 0==> salvi ; 1==> non salvi

        if (scegli == 0 && nomeFile.length() > 0)
            {
                FileBin.salvaFile(nomeFile, elenco);
            }
        Visual.message("Programma terminato ! ");
        System.exit(0);
  }//fine gestoreEsci


  public void gestoreApri () //apre un file già esistente
    {
        if (modificato)
            {
             int scegli = 1;
             scegli = Visual.conferma("Salvare le ultime modifiche ?");
             if (scegli == 0)
                 gestoreSalva();
            }
        elenco.clear(); //svuoto l'elenco
        modificato = false;
        salva_.setEnabled(false);
        JFileChooser cercaFile = new JFileChooser();
        percorso = caricaUltimoPercorso(); //percorso di partenza
        cercaFile.setCurrentDirectory( new File(percorso) ); //imposto il percorso
        cercaFile.setDialogTitle("Cerca data - base"); //titolo della finestra
        cercaFile.setFileSelectionMode(JFileChooser.FILES_ONLY); //ammetto la selezione solo di file
        int result = cercaFile.showOpenDialog(this);
        if (result==JFileChooser.APPROVE_OPTION)
          {
             File f = cercaFile.getSelectedFile();
             String temp = "" + cercaFile.getCurrentDirectory();
             salvaUltimoPercorso(temp);
             nomeFile= "" + f; //assegno a nomeFile il nome del file da gestire
```

```
                nomeFileTemp = estraiNomeFile(nomeFile);
                String parteFinale = nomeFile.substring(nomeFile.length()-4);
                if (parteFinale.equals(estensioneFile))
                    {
                    FileBin.leggiFile(nomeFile, elenco);
                    aggiornaTabella("");
                    abilitaItem();
                    messaggi.setText(msg + " aperto il file : " + nomeFileTemp);
                    this.setTitle("Rubrica Telefonica : " + nomeFileTemp);
                    }
                else
                    {
                     messaggi.setText(msg + "errore durante l'aperturra del file");
                     Visual.message("File non compatibile !!");
                    }
            }

        }//fine gestoreApri

      public static void abilitaItem()
        {
            modifica.setEnabled(true);
            if (elenco.size() > 0)
                {
                 elimina_.setEnabled(true);
                 rinomina_.setEnabled(true);
                 cerca.setEnabled(true);
                }
            else
                {
                 cerca.setEnabled(false);
                 elimina_.setEnabled(false);
                 rinomina_.setEnabled(false);
                }
        }//fine abilitaItem

      public void gestoreNuovo() //crea un nuovo file
        {
          if (modificato)
            {
             int scegli = 1;
             scegli = Visual.conferma("Salvare le ultime modifiche ?");
             if (scegli == 0)
                 gestoreSalva();
            }
          elenco.clear();
          modificato = false;
          salva_.setEnabled(false);
          JFileChooser cercaFile = new JFileChooser();
          percorso = caricaUltimoPercorso(); //percorso di partenza
          cercaFile.setCurrentDirectory( new File(percorso) ); //imposto il percorso
          cercaFile.setDialogTitle("Nuova rubrica"); //titolo della finestra
          cercaFile.setApproveButtonText("salva");
          cercaFile.setFileSelectionMode(JFileChooser.FILES_ONLY); //ammetto la selezione solo di file
          int result = cercaFile.showOpenDialog(this);
          if (result==JFileChooser.FILES_ONLY)
              {
                String temp = "" + cercaFile.getCurrentDirectory();
                salvaUltimoPercorso(temp);
                File f = cercaFile.getSelectedFile();
                nomeFile = "" + f + estensioneFile; //assegno a nomeFile il nome del file da gestire e
l'estensione
                nomeFileTemp = estraiNomeFile(nomeFile);
                if (FileBin.creaFile(nomeFile))
                    {
                    messaggi.setText(msg + " Creato nuovo file " + nomeFileTemp);
                    aggiornaTabella("");
                    abilitaItem();
                    this.setTitle("Rubrica Telefonica : " + nomeFileTemp);
                    }
            }
        }//fine gestoreNuovo
```

```
      public void gestoreSalva()
        {
          if (modificato)
              if (FileBin.salvaFile(nomeFile , elenco))
                  modificato = false;
          else
              Visual.message("Nessuna modifica da salvare !");
          salva_.setEnabled(false);
          messaggi.setText(msg + " salvataggio effettuane");
        }//fine gestore salva

      public void gestoreInserisci()
        {
          if (nomeFile.length() == 0) Visual.message("Nessuna file aperto!!");
          else
              {                                                    //
titolo  nome    cognome tel email   nuovo  indice
              AggiungiContatto finestra_input = new AggiungiContatto("Nuovo contatto" , "" , "" , "" ,
"" , true , -1);
              }
        }//fine gestore inserisci

      public static void riceviOggetto(Contatto nuovo , int pos)
        {
          if (pos < 0)
              {
              Visual.message("Aggiunto nuovo contatto !");
               elenco.addElement(nuovo);
               aggiornaTabella("");
               messaggi.setText(msg + " Nuovo contatto inserito!");
              }
          else
            {
              Visual.message("Contatto modificato !");
              elenco.setElementAt(nuovo, pos);
              aggiornaTabella("");
              messaggi.setText(msg + " Contatto modificato!");
            }
          modificato = true;
          salva_.setEnabled(true);
          abilitaItem();
        } //fine riceviOggetto

      public static void gestoreRinomina()
       {
          //gestoreVisualizza();
          int indice = Visual.readInt("\n Indice del contatto da rinominare : ");
          if (indice < 0 || indice >= elenco.size())
                  Visual.message("Indice non corretto !");
          else
            {
              Contatto temp = (Contatto) elenco.elementAt(indice);
              AggiungiContatto modifica_input = new AggiungiContatto("Modifica contatto" ,temp.getNome()
, temp.getCognome() , temp.getTelefono() , temp.getEmail() , false , indice);
            }
       }//fine gestoreRinomina

      public static void gestoreElimina()
      {
          int indice= Visual.readInt("\n Indice del contatto da eliminare : ");
         try
          {
           elenco.removeElementAt(indice);
           aggiornaTabella("");
           messaggi.setText(msg + " Contatto eliminato!!");
           Visual.message("Eliminazione effettuata");
           modificato = true;
           salva_.setEnabled(true);
           abilitaItem();
         }
         catch (Exception e )
```

```
        {
         Visual.message("Eliminazione NON possibile !");
         messaggi.setText(msg + " Eliminazione non possibile!!");
        }
}  // fine gestoreElimina


public void gestoreRicerca()
{
  String cerca = Visual.readLine("Contatto da cercare : ");

   int dimensione =  model.getRowCount(); //svuoto la tabella
   if(dimensione>0)
    for(;;)
     {
      int riga= model.getRowCount()-1;
      model.removeRow(riga);
      if (riga==0) break;
     }
  if (elenco.size() > 0) //scansiono l'elenco per trovare il contatto con il cognome/nome cercato
   {
      boolean trovato = false;
      String tabStringhe[][]= new String[elenco.size()][nomiColonne.length];
      caricaTabella(elenco,tabStringhe);
      Contatto c = null;
      int conta = 0;
      for(int i=0 ; i < elenco.size() ; i++)
      {
        c = (Contatto) elenco.elementAt(i);
        if( cerca.equals( c.getCognome() ) || cerca.equals(c.getNome()))
         {
          trovato = true;
          model.insertRow(conta++, tabStringhe[i]);
         }
      }// fine for
      if (trovato)
          messaggi.setText(msg + " Trovate " + conta + " voci con termine " + cerca);
      else
          messaggi.setText(msg + " nessun contatto corrispondente a " + cerca);
   }
}   // fine gestoreRicerca


public static void gestoreVisualizza()
  {
    aggiornaTabella("");
  }//fine gestoreVisualizza

public void gestoreInfo() //visualizza una finestra con alcune informazioni
  {
    String info = "Programma : RubricaVisuale1_1\n" +
                  "Autore : Riontino Raffaele\nData : 01/05/2011\n" +
                  "ITIS Molinari - Milano\n" +
                  "4° informatici (corso serale)\n";
    JOptionPane nuovo = new JOptionPane();
    nuovo.showMessageDialog(null,info);
  }//fine gestoreInfo



static void caricaTabella(Vector elenco, String tabStringhe[][])
    {
         Contatto c;
          for(int i=0;i<elenco.size(); i++)
          { c = (Contatto) elenco.elementAt(i);

           tabStringhe[i][0]= "" + (new Integer(i)).toString();
           tabStringhe[i][1]= "" + c.getCognome();
           tabStringhe[i][2]= "" + c.getNome();
           tabStringhe[i][3]= "" + c.getTelefono();
           tabStringhe[i][4]= "" + c.getEmail();
          }// fine for
```

```
        }//fine caricaTabella

    public static void aggiornaTabella(String messaggio) //aggiornamento della tabella
        {
           int dimensione =  model.getRowCount(); //svuotamento della tabella
           if(dimensione > 0)
            for(;;)
            {
             int riga= model.getRowCount()-1;
             model.removeRow(riga);
             if (riga == 0) break;
            }
          ordinaElenco();  //richiamo il metodo che ordina i contatti in ordine alfabetico
          if (elenco.size()>0)
            {
              String tabStringhe[][]= new String[elenco.size()][nomiColonne.length];
              caricaTabella(elenco,tabStringhe);
              for(int i=0;i<elenco.size(); i++)
                  model.insertRow(i, tabStringhe[i]);
              messaggi.setText(msg + "Dati caricati");
            }
           else
              messaggi.setText(msg + "Elenco vuoto!");

        }   // fine aggiornaTabella

    public static void ordinaElenco() //riordino dell'elenco in ordine alfabetico
      {
        boolean flag = true;
           int k = elenco.size() - 1 ;
           while (flag && k > 0)
              {
               flag = false;
               for (int i = 0 ; i < k ; i++)
                  {
                   if
 (((Contatto)elenco.elementAt(i)).getCognome().compareTo(((Contatto)elenco.elementAt(i+1)).getCognome())
> 0  ||

((Contatto)elenco.elementAt(i)).getCognome().compareTo(((Contatto)elenco.elementAt(i+1)).getCognome())
== 0 &&

((Contatto)elenco.elementAt(i)).getNome().compareTo(((Contatto)elenco.elementAt(i+1)).getNome()) >= 0)
                        {
                         flag = true;
                         Contatto temp = new Contatto();
                         temp = ((Contatto)elenco.elementAt(i));
                         elenco.set(i, elenco.elementAt(i+1));
                         elenco.set(i+1, temp);
                        }
                  }
               k--;
              }
      }//fine ordinaElenco


    public static String caricaUltimoPercorso()
      { //caricamento dell'ultima path aperta dal programma
        String ultimoPercorso = "";
        try
            {
             FileReader fileFisico = new FileReader("percorso.txt");
             BufferedReader fileLogico = new BufferedReader(fileFisico);
             ultimoPercorso = fileLogico.readLine();
             fileLogico.close();
            }
        catch(IOException e)
            {
             //Visual.message("Problemi in lettura" + e);
             System.out.println("errore nel caricamento dell'ultima path " + e);
            }
        return ultimoPercorso;
```

```
      } //fine caricaUltimoPercorso

      public static void salvaUltimoPercorso(String path)
      {     //salvataggio dell'ultima path aperta nella ricerca dei file delle rubriche
          try
             {
              FileWriter fileFisico = new FileWriter("percorso.txt");
              PrintWriter fileLogico = new PrintWriter(fileFisico);
              fileLogico.println(path);
              fileLogico.close();
              System.out.println("SALVATO il file  " + nomeFile);
             }
          catch(Exception e)
                 {
                  //Visual.message("Eccezione in scrittura file: " + eccez.getMessage());
                    System.out.println("errore nel salvataggio del path " + e);
                 }
      } //fine salvaUltimoPercorso

      public static String estraiNomeFile(String temp)
        {  //estraggo solo il nome del file senza path e senza estensione
          return temp.substring(temp.lastIndexOf("\\") + 1 , temp.length() - 4);
        }
}//fine classe Finestra



//classe utilizzata per gestire gli eventi sui pulsanti della finestra
class GestoreFinestra implements WindowListener
{
  public void windowIconified(WindowEvent e) {}
  public void windowDeiconified(WindowEvent e) {}
  public void windowActivated(WindowEvent e) {}
  public void windowDeactivated(WindowEvent e) {}
  public void windowOpened(WindowEvent e) {}
  public void windowClosed(WindowEvent e) {}


  public GestoreFinestra()
  {

  }

  public void windowClosing(WindowEvent e)
  {
    Finestra.gestoreChiudi(); //quando viene premuto il tasto chiudi viene richiamato il metodo della
classe
                              //finestra che gestisce l'uscita dal programma
  }
} //fine GestoreFinestra
```