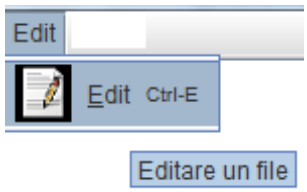
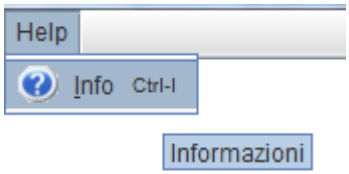
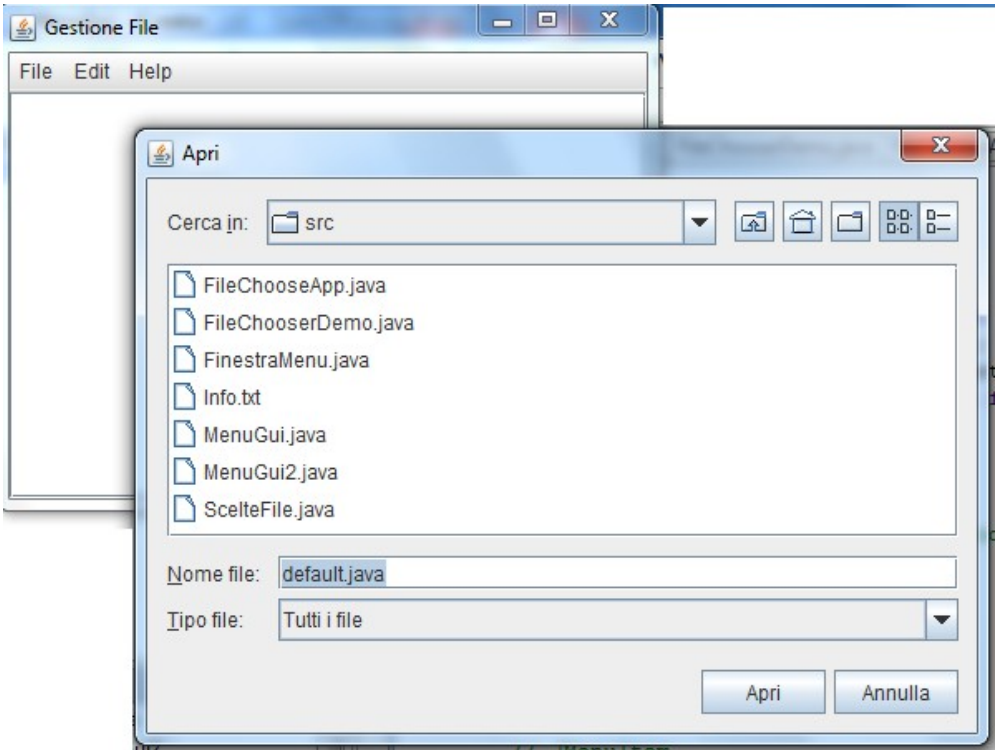
Progettare un'applicazione di tipo GUI con tre menu e voci di menu (come nelle figure sottostanti):

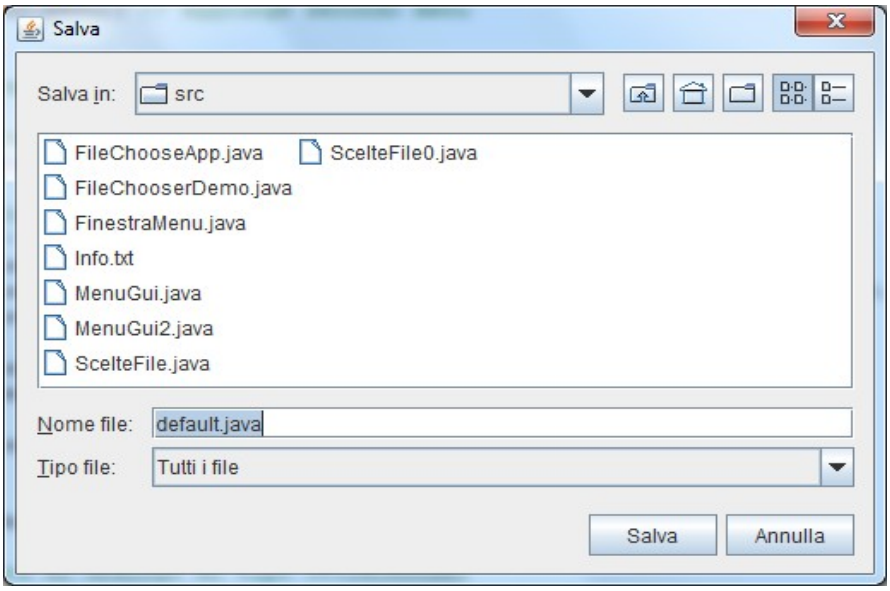




All'attivazione della voce Apri del menu File corrisponde la possibilità di aprire un file scelto con uso di finestra di tipo JFileChooser e visualizzarlo nell'area di testo (senza possibilità di editing)

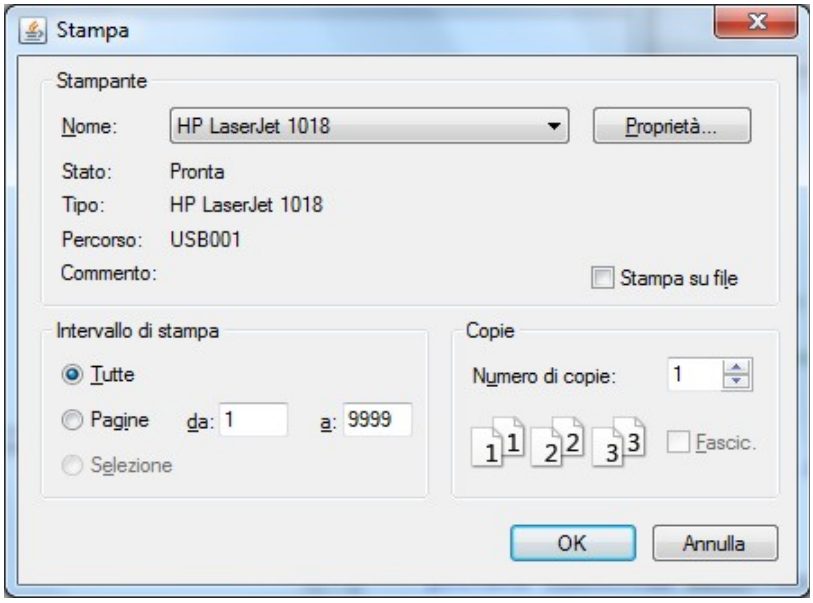


All'attivazione della voce Salva del menu File corrisponde la possibilità di salvare il testo visualizzato nell'area di testo su file con nome scelto usando una finestra di tipo JFileChooser



nome di default impostato

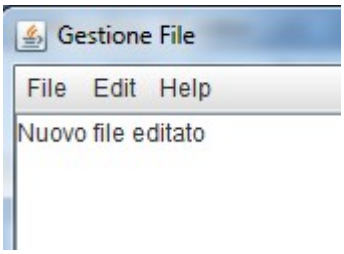
All'attivazione della voce Print del menu File corrisponde la stampa del testo visualizzato nell'area di testo:



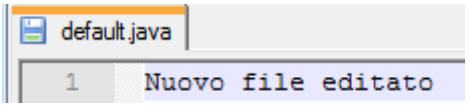
NB: si limita
ad unica pagina

All'attivazione della voce Quit del menu File corrisponde la chiusura della GUI con uso di funzione dispose()

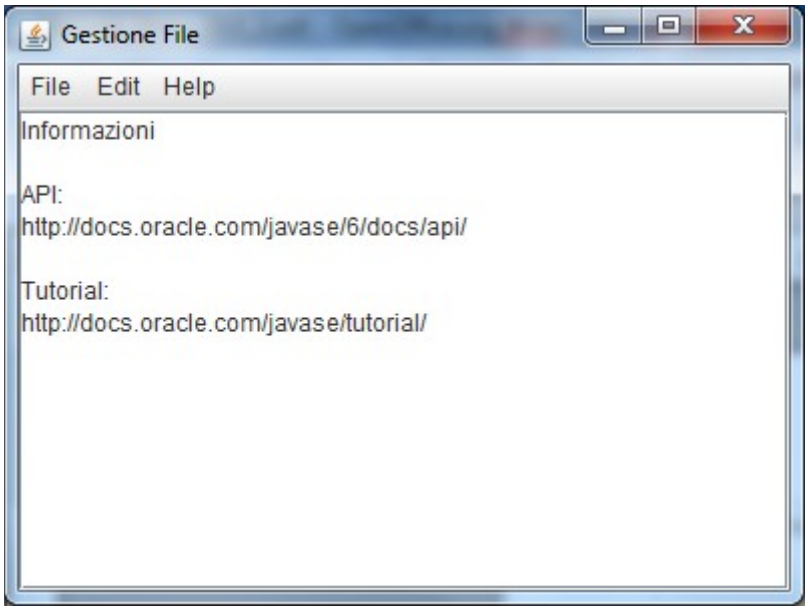
All'attivazione della voce Edit del menu Edit corrisponde la possibilità di editare il file (modificare un file esistente o crearlo)



nb: si potrebbe prevedere
un salvataggio automatico
(con nome del file di default)



Alla selezione della voce Info nel menu Help: deve essere visualizzato il contenuto di un file di testo come mostrato in figura:



NB: i componenti swing non sono *thread safe.* Si consiglia di creare la GUI come thread

```
public static void main(String[] args) {

    //Schedule a job for the event dispatch thread:
    //creating and showing this application's GUI.

    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            //Turn off metal's use of bold fonts
            UIManager.put("swing.boldMetal", Boolean.FALSE);

            new MenuGui();   // con nome applicazione  MenuGui
        }
    });
}
```

NB: per gestire le azioni di selezione annullate si ricorre alla costante JFileChooser.**CANCEL_OPTION**)

```
int returnVal = fc.showXxxDialog(<scelta>);
if (returnVal == JFileChooser.CANCEL_OPTION) {
        System.out.println("opzione annullata");  // per test
    }
```

Implementando l'interfacfcia **Printable**

```
public int print(Graphics g, PageFormat pf, int page)  throws PrinterException {
        if (page > 0) {  return NO_SUCH_PAGE;  }                    // unica pagina
        Graphics2D g2d = (Graphics2D)g;
        g2d.translate(pf.getImageableX(), pf.getImageableY());
        fTextArea.print(g);       /* per stampare il contenuto del TextArea  fTextArea*/
        return PAGE_EXISTS;
    }
private void gestisci_stampa(){
        PrinterJob job = PrinterJob.getPrinterJob();
        job.setPrintable(this);
        boolean ok = job.printDialog();
          if(ok) {
                try {    job.print();
            } catch (PrinterException ex) { }
        }
    } // fine gestisci_stampa()
```

Per selezionare in JFileChoose solo file con estensione .java

```
import javax.swing.*;
import java.io.*;

/** Filter to work with JFileChooser to select java file types. **/
class JavaFilter extends javax.swing.filechooser.FileFilter {
  public boolean accept (File f) {
    return f.getName ().toLowerCase ().endsWith (".java") || f.isDirectory ();
  }

  public String getDescription () {
    return "Java files (*.java)";
  }
} // class JavaFilter
```

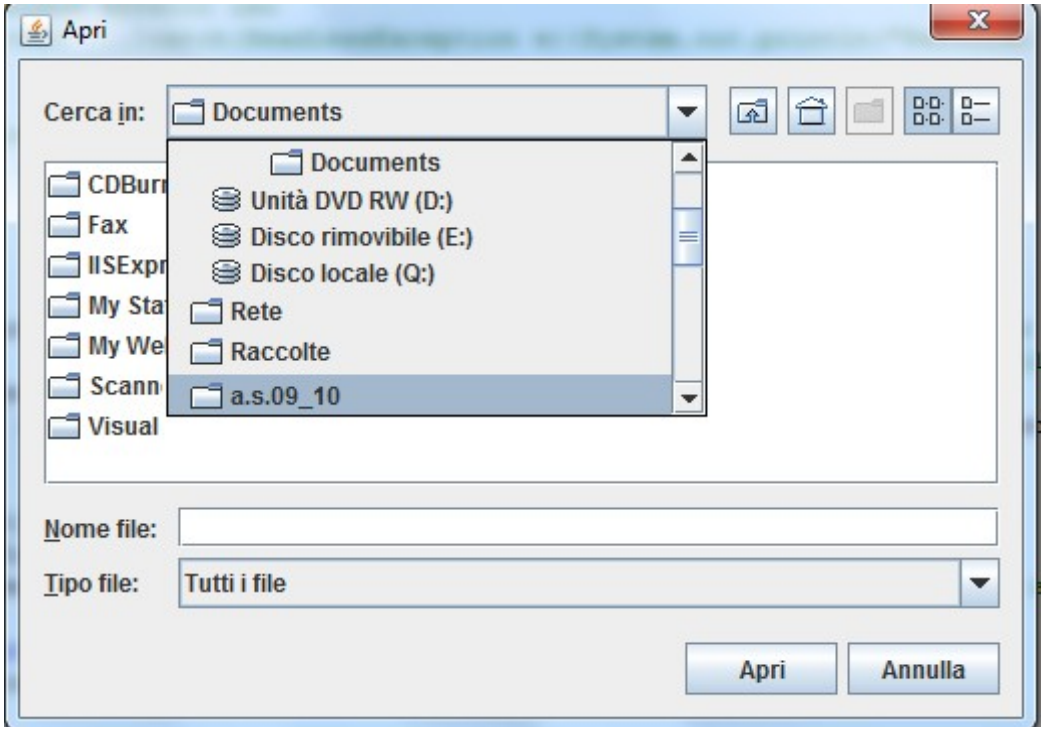
BUG con SO Windows http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=6672017

Si verifichi errore "Exception occurred during event dispatching" quando si tenta (in Windows 7) di usare un collegamento *short* senza impostare il percorso dalla cartella in cui è creato:

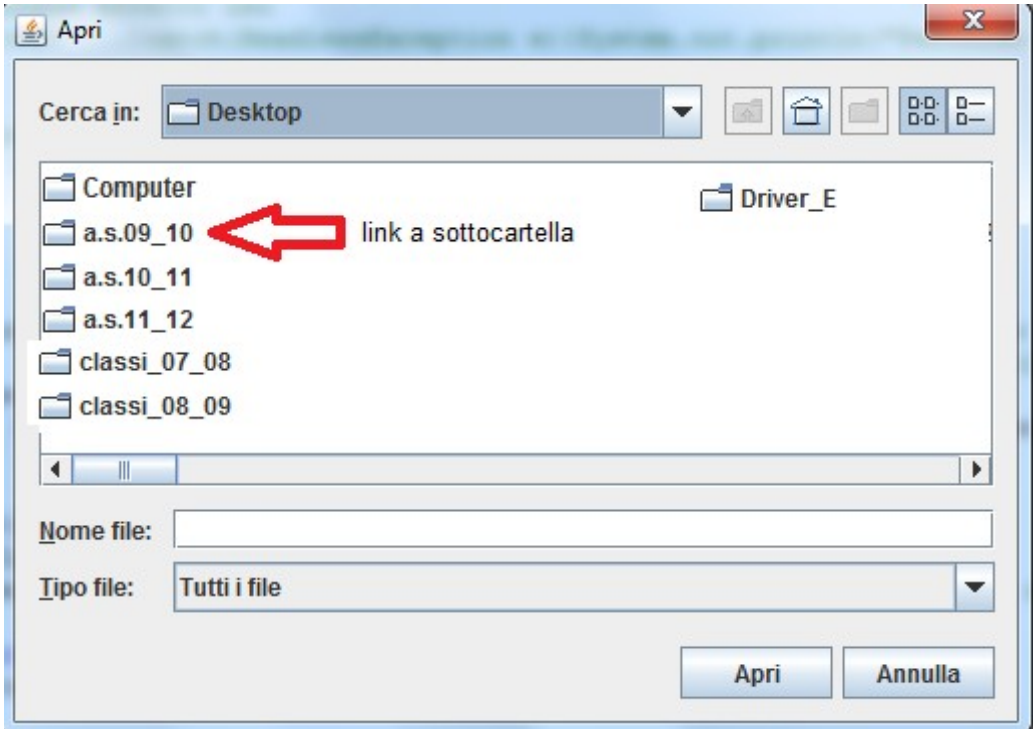
Ad esempio un collegamento creato su Desktop

C:\.....\Desktop\a.s.09_10.lnk

non può essere individuato con uso di JFileChooser se si tenta di selezionarlo dalla sotto-cartella Documents



ma è interpretato correttamente se si seleziona da Desktop:



nb: si potrebbe personalizzare il titolo della finestra avvertendo l'utente

Codice

```
/**
 * @(#)MenuGui.java
 *
 * MenuGui application
 *
 */
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;  // per File volendo anche settare directory di partenza in JFileChooser
import java.awt.print.*; // per la stampa

 public class MenuGui extends JFrame implements Printable, ActionListener{

    private JMenuBar menuBar;
    private JMenu menu;                // usato per più menu
    private JMenuItem menuItem;  // usato per più voci
    private JPanel p;
    private JTextArea fTextArea;
    private JScrollPane scrollPane;
    private JFileChooser fc;
    private File fFile = new File ("default.java");

   public MenuGui(){

            super("Gestione File");          // anche setTitle()

            //Create the menu bar
            menuBar = new JMenuBar();

            //Build the menu
            menu = new JMenu("File");

            // gruppo di JMenuItems
            menuItem = new JMenuItem("Apri", new ImageIcon("IMG/open.png") );
            menuItem.setMnemonic(KeyEvent.VK_A);
            menuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_1, ActionEvent.CTRL_MASK));
                                                                    //  anche InputEvent.CTRL_MASK
            menuItem.setToolTipText("Apre un file");
            menuItem.addActionListener(this);
            menu.add(menuItem);

            // seconda voce
            menuItem = new JMenuItem("Salva", new ImageIcon("IMG/save_as.png"));
            menuItem.setMnemonic(KeyEvent.VK_S);
            menuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_2, ActionEvent.CTRL_MASK));
            menuItem.setToolTipText("Salva su file");
            menuItem.addActionListener(this);
            menu.add(menuItem);

            // terza voce
            menuItem = new JMenuItem("Print", new ImageIcon("IMG/print.png") );
            menuItem.setMnemonic(KeyEvent.VK_P);
            menuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_3, ActionEvent.CTRL_MASK));
```

```
menuItem.setToolTipText("Stampa");
menuItem.addActionListener(this);
menu.add(menuItem);

// quarta voce
menuItem = new JMenuItem("Quit", new ImageIcon("IMG/exit.png") );
menuItem.setMnemonic(KeyEvent.VK_Q);
menuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_Q, ActionEvent.CTRL_MASK));
menuItem.setToolTipText("Chiudi");
menuItem.addActionListener(this);
menu.add(menuItem);

 menuBar.add(menu);   // aggiunge primo menu

// secondo menu

 //Build the menu
menu = new JMenu("Edit");

// JMenuItem
menuItem = new JMenuItem("Edit", new ImageIcon("IMG/edit.png") );
menuItem.setMnemonic(KeyEvent.VK_E);
menuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_E, ActionEvent.CTRL_MASK));
menuItem.setToolTipText("Editare un file");
menuItem.addActionListener(this);
menu.add(menuItem);

 menuBar.add(menu); // aggiunge secondo menu

// terzo menu

 //Build the menu
menu = new JMenu("Help");

// JMenuItem
menuItem = new JMenuItem("Info", new ImageIcon("IMG/help.gif") );
menuItem.setMnemonic(KeyEvent.VK_I);
menuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_I, ActionEvent.CTRL_MASK));
menuItem.setToolTipText("Informazioni");
menuItem.addActionListener(this);
menu.add(menuItem);

 menuBar.add(menu); // aggiunge terzo menu

setJMenuBar(menuBar);

// finestra di dialogo di tipo JFileChooser

fc = new JFileChooser();

// Start nella directory corrente
 fc.setCurrentDirectory (new File ("../src"));  // corrente se new File (".") di default Documents
 // Set to a default name for save
 fc.setSelectedFile (fFile);                    // nome di default scelto default.java
```

```
        // Pannello che costituirà il content-pane
        p = new JPanel(new BorderLayout());
        p.setOpaque(true);

        //Create a scrolled text area
        fTextArea = new JTextArea(5, 30);
        fTextArea.setEditable(false);    // si vuole scrivere solo se attivato comando Edit
                                         // per gestire a capo automatico, posizione cursore, inserire margini
        scrollPane = new JScrollPane(fTextArea);

        //Add the scrolled text area
        p.add(scrollPane, BorderLayout.CENTER);

        setContentPane(p);
        setSize(400,300);
        setVisible(true);
        setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
}

public void apri(){

        int returnVal = fc.showOpenDialog(this);            // apre finestra modale ...
                                                            // visualizza un JFileChooser per apertura file
        if(returnVal == JFileChooser.APPROVE_OPTION) {
            System.out.println("Apertura file: " + fc.getSelectedFile().getName());  // per test

            fTextArea.setEditable(false);    // alla nuova apertura si impedisce nuovamente di Editare
                                             // si vuole scrivere solo se attivato comando Edit
            apriFile();
        }
}

public void apriFile(){

        fFile = fc.getSelectedFile ();
        // Invoke the readFile method in this class
        String file_string = readFile (fFile);

        if (file_string != null)
            fTextArea.setText (file_string);
        else
            JOptionPane.showMessageDialog (null,"Errore nell'apertura del file!", "File Open Error",
                                           JOptionPane.ERROR_MESSAGE);
}

public void salva(){

        int returnVal = fc.showSaveDialog(null);            // apre finestra modale ...
                                                            // visualizza un JFileChooser per salvataggio file
        if(returnVal == JFileChooser.APPROVE_OPTION) {
            System.out.println("Salvataggio file: " + fc.getSelectedFile().getName()); // per test
            salvaFile();
        }
}
```

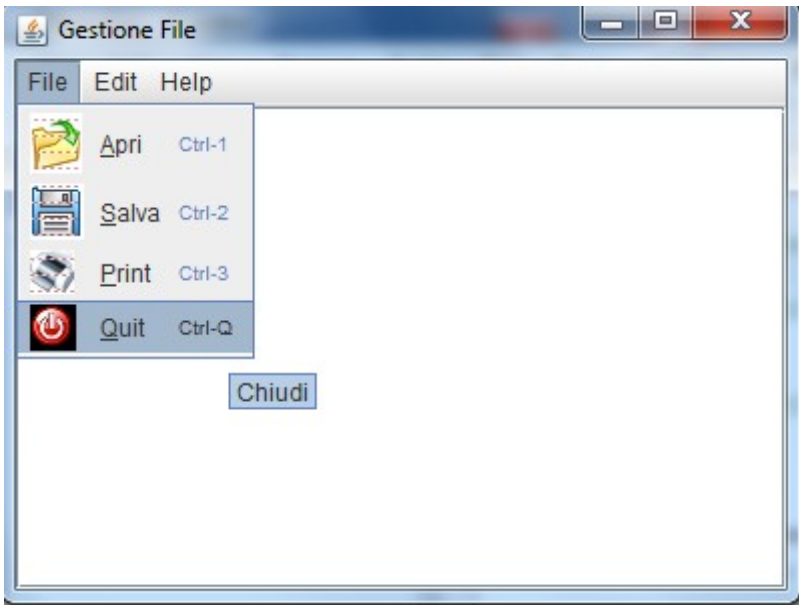
```
public void salvaFile(){
    fFile = fc.getSelectedFile ();
    if (fFile.exists ()) {
       int response = JOptionPane.showConfirmDialog (null,
                        "Vuoi sovrascrivere il file esistente?","Confirm Overwrite",
                        JOptionPane.OK_CANCEL_OPTION,  JOptionPane.QUESTION_MESSAGE);
       if (response == JOptionPane.CANCEL_OPTION) {

          JOptionPane.showMessageDialog ( null,  "Azione annullata", "File not Save",
                                                  JOptionPane.ERROR_MESSAGE );
       }
    }
   writeFile (fFile, fTextArea.getText ());
}

 public void edita(){
    fTextArea.setEditable(true);
}

 public void info(){

 File f = new File ("../src/Info.txt");
 String file_string = readFile (f);
 if (file_string != null)
       fTextArea.setText (file_string);
 else
    JOptionPane.showMessageDialog (null,"Errore nell'apertura del file!", "File Open Error",
                                          JOptionPane.ERROR_MESSAGE);
}

public int print(Graphics g, PageFormat pf, int page)
       throws PrinterException {

                    if (page > 0) {
                       return NO_SUCH_PAGE;      // unica pagina
                    }

                    Graphics2D g2d = (Graphics2D)g;
                    g2d.translate(pf.getImageableX(), pf.getImageableY());

                   fTextArea.print(g);    /*  stampa il contenuto della TextArea */
                    return PAGE_EXISTS;
          }
 private void gestisci_stampa(){

              PrinterJob job = PrinterJob.getPrinterJob();
              job.setPrintable(this);
              boolean ok = job.printDialog();
               if(ok) {
                        try {       job.print();
                         } catch (PrinterException ex) { }
             }
   } // fine gestisci_stampa
```

Info.txt

```
1  Informazioni
2
3  API:
4  http://docs.oracle.com/javase/6/docs/api/
5
6  Tutorial:
7  http://docs.oracle.com/javase/tutorial/
```

```
    public void actionPerformed(ActionEvent e){

          String s = e.getActionCommand();

          if( s.equals("Apri") )
                apri();

           if( s.equals("Salva") )
                salva();

           if( s.equals("Quit") )
                dispose();                  // chiude la GUI

           if( s.equals("Edit") )
                edita ();

           if( s.equals("Info") )
                info();

            if( s.equals("Print"))
                gestisci_stampa();
  }

  public static void main(String[] args) {
      //Schedule a job for the event dispatch thread:
     //creating and showing this application's GUI.
     SwingUtilities.invokeLater(new Runnable() {
        public void run() {
           UIManager.put("swing.boldMetal", Boolean.FALSE); // Turn off metal's use of bold fonts
                                                                  // per modifiche al look&feel
           new MenuGui();  }
     });
  }

 /** Use a BufferedReader wrapped around a FileReader to read
  *    the text data from the given file.
  **/
public String readFile (File file) {

  StringBuffer fileBuffer;
  String fileString=null;
  String line;
  try {
              FileReader in = new FileReader (file);
              BufferedReader dis = new BufferedReader (in);
              fileBuffer = new StringBuffer () ;
              while ((line = dis.readLine ()) != null) {
                  fileBuffer.append (line + "\n");
              }
              in.close ();
              fileString = fileBuffer.toString ();
  }    catch  (IOException e ) {      return null;   }
  return fileString;
} // fine readFile
```

```
  /**
   * Use a PrintWriter wrapped around a BufferedWriter, which in turn
   * is wrapped around a FileWriter, to write the string data to the
   * given file.
  **/
 public static void writeFile (File file, String dataString) {
  try {
                PrintWriter out =  new PrintWriter (new BufferedWriter (new FileWriter (file)));
                out.print (dataString);
                out.flush ();
                out.close ();
   }   catch (IOException e) { JOptionPane.showMessageDialog ( null,
                                    "Errore di IO nel salvare il file!!", "File Save Error",
                                    JOptionPane.ERROR_MESSAGE );
  }
 } // fine writeFile

} // fine applicazione
```



The Java Tutorial

Menu
http://docs.oracle.com/javase/tutorial/uiswing/components/menu.html

JFileChooser
http://docs.oracle.com/javase/tutorial/uiswing/components/filechooser.html

Stampare
*http://java.sun.com/docs/books/tutorial/**2d**/printing/index.html*

stampa facile (ver. 1.6)
http://www.mrwebmaster.it/java/articoli/java-modo-veloce-stampare-file_984.html

Appendice

Scelta di un file in Swing:

1. **Creare il file chooser**

   fc = new JFileChooser();

2. Chiamare uno dei seguenti metodi che mostrano un dialogo modale contenente il file chooser

   int scelta = fc.showOpenDialog(*finestra*);
   int scelta = fc.showSaveDialog(*finestra*);
   int scelta = fc.showDialog(*finestra,"**titolo**"*);

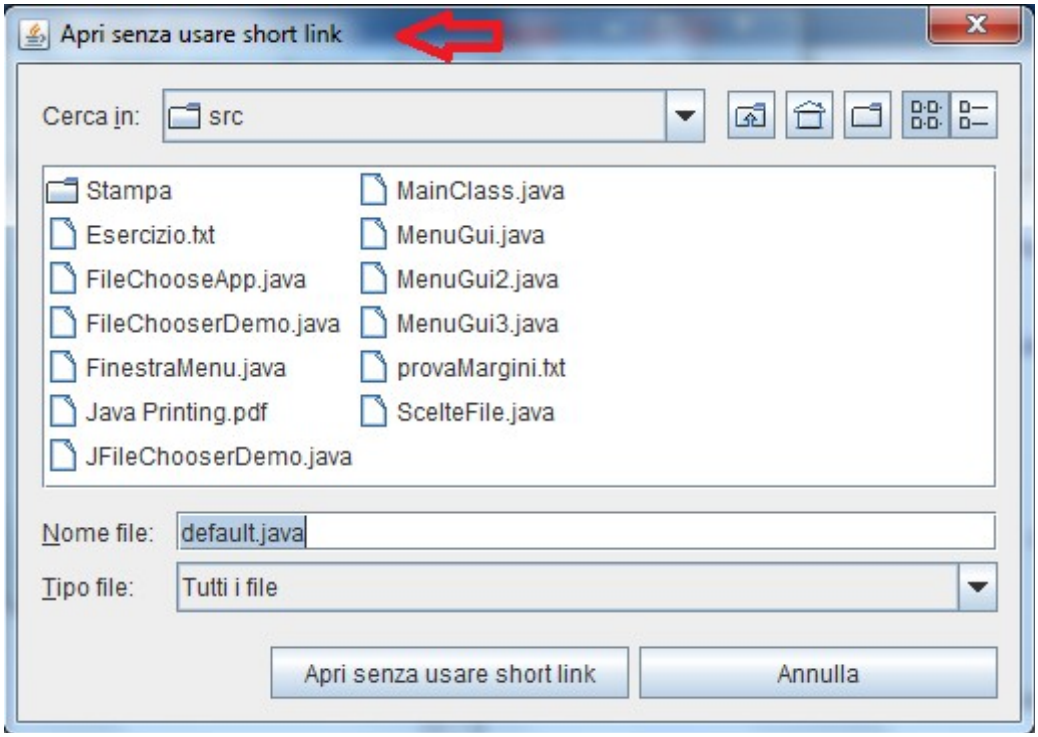
   dove finestra e' la componente da cui il dialogo deve dipendere.

int	**showDialog**(Component parent, String approveButtonText) Pops a custom file chooser dialog with a custom approve button.
int	**showOpenDialog**(Component parent) Pops up an "Open File" file chooser dialog.
int	**showSaveDialog**(Component parent) Pops up a "Save File" file chooser dialog.

3. Leggere il valore di ritorno, che può essere JFileChooser.APPROVE_OPTION o JFileChooser.CANCEL_OPTION e, in caso di approvazione, prendere il file

   if (scelta==JFileChooser.APPROVE_OPTION) {
   ...*aprire il file* fc.getSelectedFile().getPath()...
   }

Altri metodi consentono di assegnare directory corrente, definire filtri, personalizzare.



Gestione JTextArea

Nel gestire JTextArea:

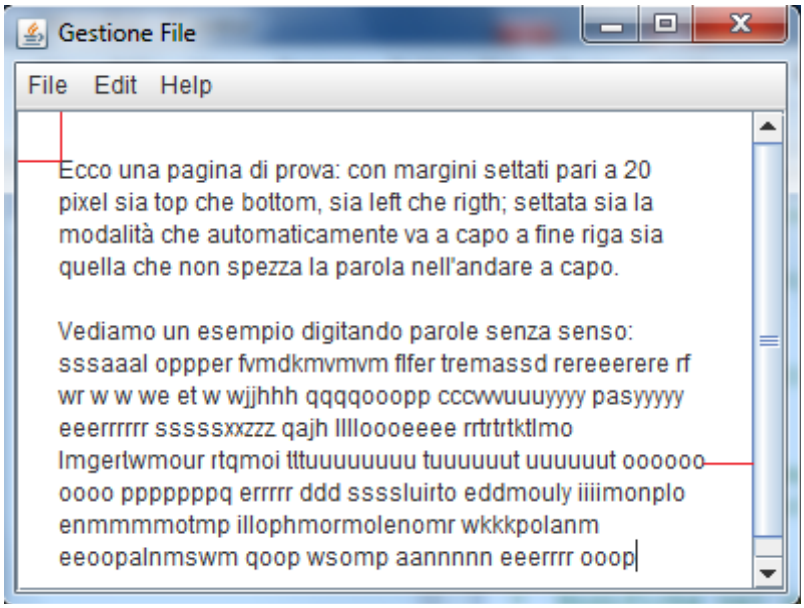
```
                                        // definizione di attributi
    private JTextArea fTextArea;
    private JScrollPane scrollPane;
```

segmento di programma per inizializzare i componenti swing:

```
//Create a scrolled text area
fTextArea = new JTextArea(5, 30);
fTextArea.setEditable(false);    //  readonly

fTextArea.setLineWrap(true); // specifica di andare a capo automaticamente a fine riga
                             // se "true"

fTextArea.setWrapStyleWord(true);    // va a capo con la parola se "true"
                                     // col singolo carattere se "false"

int marg = 20; // 5  consigliato

fTextArea.setMargin(new Insets(marg,marg,marg,marg));  // per settare margini

scrollPane = new JscrollPane(fTextArea);  // inserimento nel pannello a scorrimento
```

Effetto:

impostati i margini: 20 pixel



Volendo impostare la **posizione del cursore** alla fine del testo:

```
fTextArea.setCaretPosition(fTextArea.getDocument().getLength());  // posiziona cursore
```

```
JOptionPane.showMessageDialog ( null,fTextArea.getDocument().getLength(), "numero caratteri",JOptionPane.ERROR_MESSAGE ); // per test
```