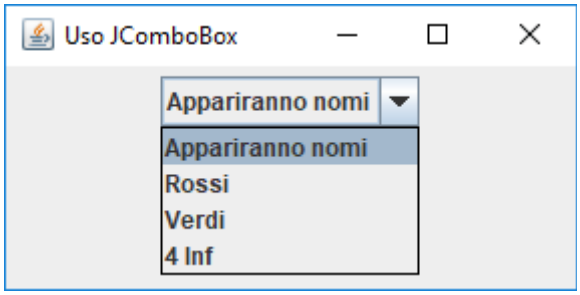
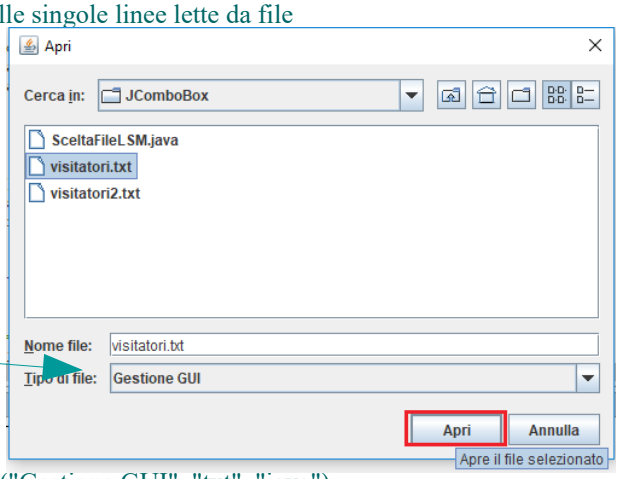
JFileChooser per visualizzare **finestre modali** per **scegliere percorso** di file

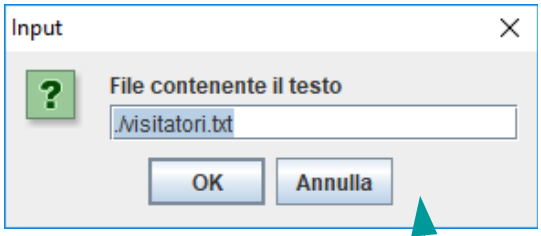
Progetto con uso JComboBox<String> ed ArrayList<String>



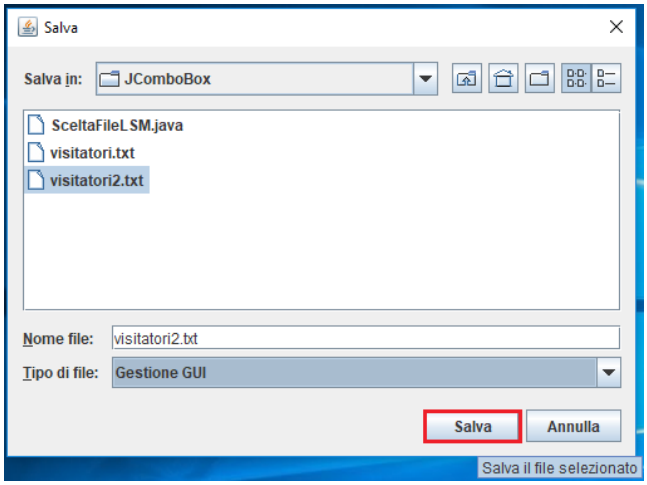

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import javax.swing.filechooser.*;  // per FileNameExtensionFilter
import java.io.*;
import java.util.*;

class ScelteFileLSM extends JFrame implements ActionListener {

   String dir="./";
   String nome = "visitatori.txt";// di default
   String path = dir + nome ; // di default per aprire

   String nomeS = "visitatori2.txt";// di default
   String pathS = dir + nomeS ; // di default per salvare

   JFileChooser fc;
   String sF = "";                                          // stringa in cui viene memorizzato il file letto
   ArrayList<String> elenco = new ArrayList<String>();    //  elenco delle singole linee lette da file

   // per GUI
   private Container c;
   private JPanel p;
   private JComboBox <String> jcb;

   /**
    * Metodo costruttore di default
    * che inizializza il componente di tipo JFileChooser
    * filtrando la visualizzazione di file con estensione .txt e .java
    * con categoria personalizzata "Gestione GUI"
    */
    public  ScelteFileLSM(){

            fc = new  JFileChooser(dir);   // directory corrente
            FileNameExtensionFilter filter = new FileNameExtensionFilter("Gestione GUI", "txt", "java");
            fc.setFileFilter(filter); // se si desidera filtrare
   }
```

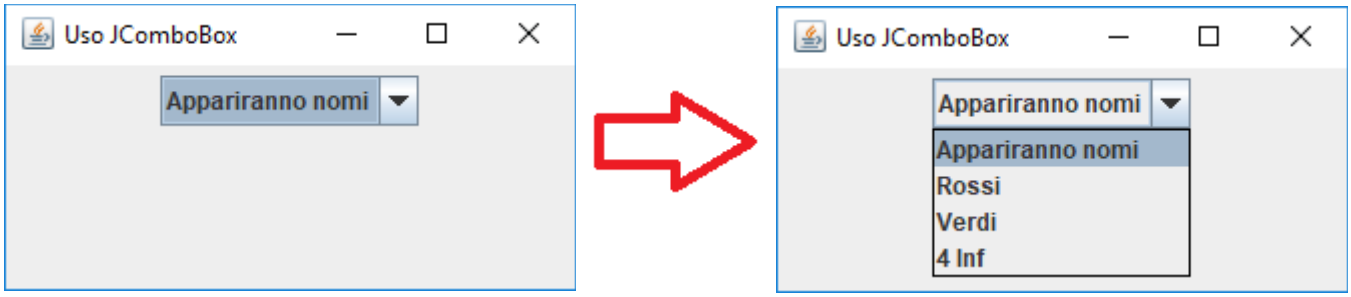


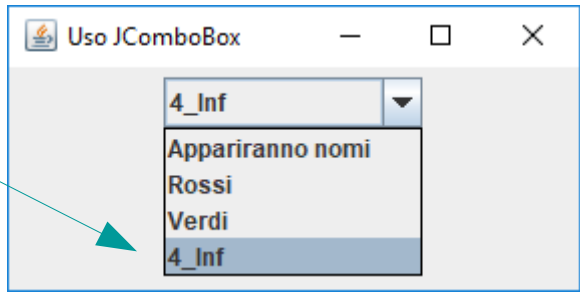

```
   /**
   * Metodo che permette di selezionare percorso
   * del file da cui sarà letto il testo
   * gestendo la pressione di Annulla
   * proponendo prima l'uso di componente JFIleChooser  per apertura file
   * poi l'uso di finestra di dialogo
   */
   public void sceltaNomeA(){

                  int returnVal = fc.showOpenDialog(this);      // apre finestra modale visualizzando un JFileChooser
                                                                // per apertura file
                   if(returnVal == JFileChooser.CANCEL_OPTION) {
                                          System.out.println("non hai effettuato la scelta");     // per test
                   }
                  if(returnVal == JFileChooser.APPROVE_OPTION) {
                      nome=fc.getSelectedFile().getName();
                      System.out.println("Hai scelto di aprire il file: " + nome);
                      path=fc.getCurrentDirectory()+"\\"+nome;   // doppio backslach in ambienteWindows
                  }
                  else {                                              /// gestendo pressione Annulla
                     String s=JOptionPane.showInputDialog("File contenente il testo", path);              //  altra finestra modale
                     if(s!=null)
                           path =  s;
                  }
   }// fine metodo sceltaNomeA
```



```
/**
* Metodo che permette di selezionare percorso
* del file dove sarà salvato testo,  gestendo la pressione di Annulla
*  proponendo prima l'uso di componente JFIleChooser per salvataggio  file
*  poi l'uso di finestra di dialogo
*/
public void sceltaNomeS(){

            int returnVal = fc.showSaveDialog(this);
            if(returnVal == JFileChooser.APPROVE_OPTION)
                        pathS=fc.getSelectedFile().getPath();
            else {                                                  /// gestendo pressione Annulla
                        String s=JOptionPane.showInputDialog("File per salvare il testo",pathS);  // altra finestra modale
                        if(s!=null)
                                    pathS =  s;
             }

 } // fine metodo sceltaNomeS

/**
* Metodo che inizializza la GUI  ed imposta il flusso
*/
 public void iniGUI(){

                setTitle("Uso JComboBox");
                c = getContentPane();
                p =new JPanel();
                jcb = new JComboBox <String>();
                jcb.addItem("Appariranno nomi");
                sceltaNomeA();
                 sF = leggiFile(path);       // per leggere il testo del file scelto salvandolo in stringa
                sceltaNomeS();
                salvaSuFile(pathS);   // scrivere potendo scegliere altro percorso
                jcb = aggiunge();                 // aggiunge in JComboBox<String> e visualizza
                jcb.addActionListener(this);  // nb: non se vuoto jcb
                p.add(jcb);
                c.add(p);
                pack();
                setVisible(true);
                setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
}

/**
* Metodo che popola il componente JComboBox <String>
* aggiungendo le linee lette e memorizzate come elementi di un ArrayList<String>
*/
 public JComboBox <String> aggiunge( ){

        for(int i =0; i<elenco.size(); i++)
                    jcb.addItem (elenco.get(i));
        return jcb;
       }
```








```
/**
* Metodo di risposta all'evento
* per rimpiazzare spazi bianchi con underline
* nell'elenco nominativo
* @param ev => evento generato di tipo ActionEvent
*/
 public void actionPerformed(ActionEvent ev){
                    Object o = ev.getSource();
                    if(o == jcb) {
                      int i = jcb.getSelectedIndex();
                      String s =((String)jcb.getSelectedItem()).replace(" ","_");  // casting da Object
                      jcb.insertItemAt(s,i);  // aggiorna JComboBox - in realtà aggiunge inserendo prima del vecchio
                      jcb.removeItemAt(i+1);
                      elenco.add("nuovo");     // per evitare IndexOutOfBoundsException si deve aggiungere un elemento
                      try {
                                elenco.set(i,s);    // poi settare che - in realtà - aggiunge
                                elenco.remove(i-1); // poi eliminare il precedente
                      }catch (Exception e){ System.out.println("Eccezione " + e);  }
                    }
 }

 /**
 * Metodo per visualizzare a terminale
 * il numero e gli elementi dell'elenco
 */
 public void vediElementi(){

        System.out.println("");
        System.out.println("n_elementi " + elenco.size() );
        for(int i =0; i<elenco.size(); i++)
                        System.out.println(elenco.get(i) );
}

/**
* Metodo che legge da File con percorso passato come parametro
* @param nome => stringa che è il percorso del file
* @return sF => stringa che è il testo contenuto nel file letto
*/
 public String leggiFile(String nome){
        String sF="";

         try{
                FileReader fr = new FileReader(nome);
                Scanner in = new Scanner(fr);
                while (in.hasNextLine()) { // se l'input termina hasNextLine() ritorna falso
                    String line = in.nextLine(); // lettura di una riga alla volta
                    elenco.add(line)   ; // aggiungo ad ArrayList <String>
                    System.out.println(line);  // per test
                    sF = sF + line + "\n";  // salvo in attributo con gestione dell' a capo
                }
        fr.close();
        }catch (FileNotFoundException fe){  System.out.println("File non trovato");
         }catch (IOException e){  }

        return sF;
 }
```

```
    /**
    * Metodo che salva su File una stringa data come parametro
    * @param nome => stringa che è il percorso del file
    */
    public void salvaSuFile(String nome) { // quello che viene scritto è contenuto nell'attributo sF

          try {

                BufferedWriter outputWriter = new BufferedWriter(new FileWriter(nome)); // non accoda
                outputWriter.write(sF);  // bufferizzato (unico accesso per flusso di caratteri)
                outputWriter.flush();
                outputWriter.close();// salva solo alla chiusura dello stream

           } catch(IOException ioe)  {  //Gestico eccezioni I/O
                                        System.out.println("Errore: " + ioe.getMessage());
                                       }
    }

    /* metodo principale */
    public static void main (String arg[]){
              ScelteFileLSM o = new ScelteFileLSM();
              o.iniGUI();
    }
} // fine applicazione
```

Output a terminale

senza scegliere percorsi