Chat con interfaccia grafica

Realizzare l'interfaccia grafica per un programma di chat, cioè un'applicazione Client/Server che consente di scambiare messaggi tra un client e un server. Sia il programma server che il programma client possono terminare la connessione facendo click sul bottone "Fine".

Il problema rappresenta un ampliamento del programma presentato nel Paragrafo 3 con l'aggiunta dell'interfaccia grafica.

Il programma server può essere organizzato con due classi. La prima viene usata per avviare il programma, per aprire e chiudere la connessione. La seconda classe viene usata per gestire l'interfaccia grafica, l'invio e la ricezione dei messaggi.

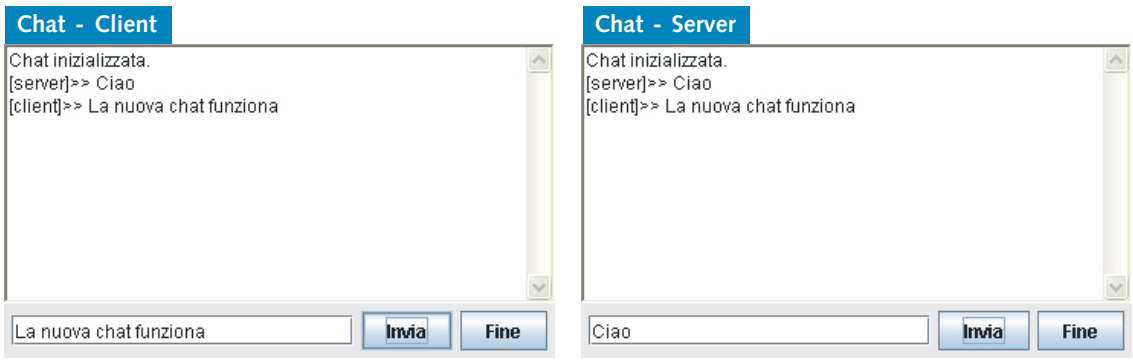
La prima classe (*GrafServerChat*) può essere impostata nel seguente modo:

```
// import . . .

public class GrafServerChat
{
   private ServerSocket sSocket;
   private Socket connessione = null;
   public BufferedReader sIN;
   public PrintWriter sOUT;

   public GrafServerChat()
   {
      int port = 2345;
      InputStreamReader in;
      OutputStream out;

      try
      {
         // crea connessione
         sSocket = new ServerSocket(port);
         connessione = sSocket.accept();

         // apre il flusso in uscita su socket . . .

         // apre il flusso in ingresso da socket . . .
      }
      catch (Exception e)
      {
         System.exit(-1);
      }
   }

   public void chiudi()
   {
      // chiude le socket . . .
   }
```

```
    public static void main(String args[])
    {
       // crea la finestra
       FinestraServer f = new FinestraServer();
       f.setTitle("Chat - Server");
       f.pack();
       f.setVisible(true);

       // avvia la chat
       GrafServerChat serverChat = new GrafServerChat();
       f.gestisciConnessione(serverChat);
    }
 }
```

Gli elementi principali che compongono l'interfaccia grafica sono:
- un'area di testo per visualizzare i messaggi ricevuti e inviati
- una casella di testo in cui inserire il messaggio da inviare
- un bottone per inviare il messaggio
- un bottone per chiudere la chat.

La finestra può essere organizzata con un *BorderLayout*. Nella zona centrale, la più ampia della finestra, si può posizionare l'area di testo mentre nella zona in basso le altre componenti grafiche.



La seconda classe (*FinestraServer*) può essere impostata nel seguente modo:

```
// import . . .

public class FinestraServer extends JFrame implements ActionListener
{
   // JPanel panel, comandi
   // TextArea area. JTextField messaggio
   // JButton invia, fine
   private GrafServerChat chat;

   public FinestraServer()
   {
      // creazione interfaccia . . .
   }
```

```
    public void gestisciConnessione(GrafServerChat chat)
    {
       this.chat = chat;

       // imposta la stringa di comando per l'evento sui bottoni
       invia.setActionCommand("invia");
       fine.setActionCommand("fine");

       // registra gli ascoltatori per i bottoni
       invia.addActionListener(this);
       fine.addActionListener(this);

       area.append("Chat inizializzata.\n");
       while (true)
       {
          try
          {
             // legge dalla socket e stampa il messaggio
             String msg = chat.sIN.readLine();
             if (msg == null)
             {
                chat.chiudi();
                System.exit(0);
             }

             area.append("[client]>> " + msg + "\n");
          }
          catch (IOException e)
          {
             chat.chiudi();
             System.exit(-1);
          }
       }
    }

    public void actionPerformed(ActionEvent e)
    {
       if ("invia".equals(e.getActionCommand()))
       {
          // invia il messaggio letto dalla casella di testo . . .
       }

       if ("fine".equals(e.getActionCommand()))
       {
          chat.chiudi();
          System.exit(0);
       }
    }
 }
```

Il metodo *gestisciConnessione* riceve come parametro un riferimento *chat* alla classe *GrafServerChat* che gestisce la connessione. Questo riferimento viene usato per ricevere i messaggi (*chat.sIN.readLine*), per inviare i messaggi (*chat.sOUT.println*) e per chiudere la connessione (*chat.chiudi*).
Il programma client è molto simile al programma server e può essere realizzato con due classi simili a quelle utilizzate per il server. L'unica differenza è che il flusso in ingresso da socket deve essere aperto prima di quello in uscita.