ClientBourse.java • Classe principale du client graphique et fonction *main*

```
import java.lang.*;
import java.io.*;
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.table.*;
import javax.swing.event.*;
import javax.swing.border.*;


class ClientBourse extends JFrame implements ActionListener
{

  private static final int COMMAND_NOTHING=0;
  private static final int COMMAND_ADDUSER=1;
  private static final int COMMAND_LOGIN=2;
  private static final int COMMAND_LIST=3;
  private static final int COMMAND_LISTMARKET=4;
  private static final int COMMAND_BUYSELL=5;

  private class ClientGestionSouris extends MouseAdapter
  {
    // Gestion des evenements du type clic sur les boutons :
    public void mouseReleased(MouseEvent e)
    {
      if(e.getSource()==bAcheter)
      {
        actionAcheter();
      }
      if(e.getSource()==bVendre)
      {
        actionVendre();
      }
      if(e.getSource()==bConnect)
      {
        actionConnection();
      }
    }
  };


  // Controles de la fenetre :
  private JTable tBourse, tClient;
  private DefaultTableModel mBourse, mClient;
  private JButton bAcheter, bVendre, bConnect;
  private JTextArea tLog;
  private JPanel cadre;
  private JScrollPane dBourse, dClient, dLog;
  private Vector entetes;
  private GridBagContainLayout layout;
  private ClientGestionSouris gSouris;

  // Menus :
  private JMenuBar menuBar;
  private JMenu menuFichier,menuTransaction;
  private JMenuItem itemConnexion,itemDeconnexion,itemQuitter;
  private JMenuItem itemAcheter,itemVendre;

  // Client associe :
  ClientInterne client;

  // Membres donnees :
  private final String titreApplication= "Client Boursier";
  private boolean connecte;
  private String nomLogin;
  private int ecoute;


  public ClientBourse()
  {
    // Initie la fenetre :
    super();
    setTitle(titreApplication);
    nomLogin=new String("");

    // Cree les boutons et les place dans le cadre :
    bAcheter=new JButton("Acheter >>");
    bVendre=new JButton("<< Vendre");
    bAcheter.setEnabled(false);
    bVendre.setEnabled(false);
    bConnect=new JButton("Connexion");
    cadre=new JPanel();
    cadre.setLayout(new GridLayout(3, 1, 20, 20));
    cadre.add(bAcheter);
    bAcheter.setMaximumSize(new Dimension(100, 40));
    bVendre.setMaximumSize(new Dimension(100, 40));
    bConnect.setMaximumSize(new Dimension(100, 40));
    cadre.add(bVendre);
    cadre.add(bConnect);
    cadre.setPreferredSize(new Dimension(100, 200));


    // Cree les tables :
    entetes=new Vector();
    entetes.addElement("Action");
    entetes.addElement("Quantité");
    mBourse=new DefaultTableModel(entetes, 0);
    mClient=new DefaultTableModel(entetes, 0);
    tBourse=new JTable(mBourse);
    tClient=new JTable(mClient);
    dBourse=new JScrollPane(tBourse);
    dClient=new JScrollPane(tClient);
    Border lborder=BorderFactory.createEtchedBorder(EtchedBorder.LOWERED);
    TitledBorder tborder1=BorderFactory.createTitledBorder(lborder, "Bourse");
    tborder1.setTitleJustification(TitledBorder.CENTER);
    TitledBorder tborder2=BorderFactory.createTitledBorder(lborder, "Client");
    tborder2.setTitleJustification(TitledBorder.CENTER);
    dBourse.setBorder(tborder1);
    dClient.setBorder(tborder2);
    tBourse.setShowVerticalLines(false);
    tClient.setShowVerticalLines(false);
    dBourse.setPreferredSize(new Dimension(200, 200));
    dClient.setPreferredSize(new Dimension(200, 200));

    // Cree l'afficheur de log :
    tLog=new JTextArea();
    tLog.setEditable(false);
    tLog.setLineWrap(true);
    dLog=new JScrollPane(tLog);
    dLog.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
    dLog.setBorder(BorderFactory.createLoweredBevelBorder());
    dLog.setPreferredSize(new Dimension(300, 50));

    // Place tout ca dans le layout :
    layout=new GridBagContainLayout();
    getContentPane().setLayout(layout);
    layout.constrain(getContentPane(), dBourse, 0, 0, 1, 1, 5, 0.5);
    layout.constrain(getContentPane(), cadre, 1, 0, 1, 1, 10, 0);
    layout.constrain(getContentPane(), dClient, 2, 0, 1, 1, 5, 0.5);
    layout.constrain(getContentPane(), dLog, 0, 1, 3, 1, 5, 0.5);

    // Menus :
    menuBar=new JMenuBar();
    menuFichier=new JMenu("Fichier");
    menuTransaction=new JMenu("Transaction");
    itemConnexion=new JMenuItem("Connexion...");
    itemDeconnexion=new JMenuItem("Déconnexion");
    itemDeconnexion.setEnabled(false);
    itemConnexion.addActionListener(this);
    itemDeconnexion.addActionListener(this);
    itemQuitter=new JMenuItem("Quitter");
    itemQuitter.addActionListener(this);
    itemAcheter=new JMenuItem("Acheter...");
    itemAcheter.addActionListener(this);
    itemVendre=new JMenuItem("Vendre...");
    itemVendre.addActionListener(this);
    menuFichier.add(itemConnexion);
    menuFichier.add(itemDeconnexion);
    menuFichier.add(itemQuitter);
    menuTransaction.add(itemAcheter);
    menuTransaction.add(itemVendre);
    menuTransaction.setEnabled(false);
    menuBar.add(menuFichier);
    menuBar.add(menuTransaction);
    setJMenuBar(menuBar);

    // Ajoute les listeners au bouton :
    gSouris=new ClientGestionSouris();
    bAcheter.addMouseListener(gSouris);
    bVendre.addMouseListener(gSouris);
    bConnect.addMouseListener(gSouris);

    // Log :
    ecoute=0;
    connecte=false;
    addLog("Client Boursier prêt !");
    addLog("Cliquez sur le bouton 'Connexion' pour entrer votre login.");
  }


  // Gestion des evenements du type clic sur un item de menu :
  public void actionPerformed(ActionEvent e)
  {
    // Distribution de chaque clic sur item au gestionnaire competent :
    if(e.getSource()==itemConnexion)
      actionConnection();

    if(e.getSource()==itemDeconnexion)
      actionConnection();

    if(e.getSource()==itemAcheter)
      actionAcheter();

    if(e.getSource()==itemVendre)
      actionVendre();

    if(e.getSource()==itemQuitter)
      System.exit(0);
  }



  public void actionConnection()
  {
    if(connecte==true)
    {
      // Deconnexion :
      ecoute=COMMAND_NOTHING;
      client.sendLogout();
      client.close();
      connectionClose();
    }
    else
    {
      // Affiche la fenetre de connexion :
      LogScreen connection=new LogScreen(this);
      if(connection.estValide())
      {
        nomLogin=connection.getNom();
        String adresse=connection.getAdresse();
        int port=connection.getPort();

        // C'est parti !
        addLog("Connexion a " + adresse + " sur le port " + port);
        client=new ClientInterne(this);
        try
        {
          client.connect(adresse, port);
          addLog("Connexion etablie.");
        }
        catch(ClientInterne.UnableToConnectException except)
        {
          addLog("Connexion impossible.");
          return;
        }
        connecte=true;
        bConnect.setText("Déconnexion");
```

```
      // On se logge
      ecoute=COMMAND_LOGIN;
      client.sendLogin(nomLogin);

      bAcheter.setEnabled(true);
      bVendre.setEnabled(true);
      menuTransaction.setEnabled(true);
      itemConnexion.setEnabled(false);
      itemDeconnexion.setEnabled(true);
      setTitle(titreApplication+" - " + nomLogin);
    }
  }
}



public void connectionReceive(String chaine)
{
  switch(ecoute)
  {
    case COMMAND_NOTHING:
    {
      addLog(chaine);
      break;
    }
    case COMMAND_LISTMARKET:
    {
      if(chaine.equals(""))
      {
        ecoute=COMMAND_LIST;
        client.sendList();
      }

      else
      {
        StringTokenizer st0=new StringTokenizer(chaine,"\n");
        String ligne=new String("");
        if(st0.hasMoreTokens()) ligne=st0.nextToken();
        StringTokenizer st=new StringTokenizer(chaine," ");
        Vector data=new Vector();
        if(st.hasMoreTokens()) data.addElement(st.nextToken());
        if(st.hasMoreTokens()) data.addElement(st.nextToken());
        mBourse.addRow(data);
        mBourse.fireTableDataChanged();
      }

      break;
    }
    case COMMAND_LIST:
    {
      if(chaine.equals(""))
      {
        ecoute=COMMAND_NOTHING;
      }
      else
      {
        StringTokenizer st0=new StringTokenizer(chaine,"\n");
        String ligne=new String("");
        if(st0.hasMoreTokens()) ligne=st0.nextToken();
        StringTokenizer st=new StringTokenizer(chaine," ");
        Vector data=new Vector();
        if(st.hasMoreTokens()) data.addElement(st.nextToken());
        if(st.hasMoreTokens()) data.addElement(st.nextToken());
        mClient.addRow(data);
        mClient.fireTableDataChanged();
      }

      break;
    }
    case COMMAND_BUYSELL:
    {
      addLog(chaine);
      if(chaine.toLowerCase().startsWith("err"))
           addLog("Attention : la transaction n'a pas ete validee"
                  + " par le serveur.");
      ecoute=COMMAND_LISTMARKET;
      client.sendListMarket();
      mBourse.setRowCount(0);
      mClient.setRowCount(0);
      break;
    }
    case COMMAND_ADDUSER:
    {
      if(chaine.toLowerCase().startsWith("err"))
      {
        addLog("Vous etes deja connecte !");
        ecoute=COMMAND_NOTHING;
        client.sendLogout();
        client.close();
        connectionClose();
      }
      else
      {
        ecoute=COMMAND_LOGIN;
        client.sendLogin(nomLogin);
      }
      break;
    }
    case COMMAND_LOGIN:
    {
      addLog(chaine);

      // Voit si on est accepte par le serveur :
      if(!chaine.toLowerCase().startsWith("ok"))
      {
        ecoute=COMMAND_ADDUSER;
        client.sendAddUser(nomLogin);
      }
      else
      {
        ecoute=COMMAND_LISTMARKET;
        client.sendListMarket();
      }
      // Vide les tables :
      mBourse.setRowCount(0);
      mClient.setRowCount(0);
      break;
    }
  }
}
```

```
  public void connectionLost()
  {
    addLog("Deconnecte du serveur !");
    connectionClose();
  }


  public void connectionClose()
  {
    nomLogin=new String("");
    client=null;
    connecte=false;
    bConnect.setText("Connexion");
    bAcheter.setEnabled(false);
    bVendre.setEnabled(false);
    menuTransaction.setEnabled(false);
    itemConnexion.setEnabled(true);
    itemDeconnexion.setEnabled(false);
    setTitle(titreApplication);
    mBourse.setRowCount(0);
    mClient.setRowCount(0);
    mBourse.fireTableDataChanged();
    mClient.fireTableDataChanged();
  }



  public void actionAcheter()
  {
    int i=tBourse.getSelectedRow();
    if(i<0) {addLog("Sélectionnez une action à acheter dans la bourse.");
             return;}

    String action=(String)mBourse.getValueAt(i, 0);
    int qte=Integer.valueOf((String)mBourse.getValueAt(i, 1)).intValue();
    TransactionScreen transac=new TransactionScreen(this, action, qte,
                                      TransactionScreen.ACHETER);
    if(transac.estValide())
    {
      ecoute=COMMAND_BUYSELL;
      client.sendBuy(action, transac.getQuantite());
    }
  }



  public void actionVendre()
  {
    int i=tClient.getSelectedRow();
    if(i<0)
    {
      addLog("Sélectionnez une action à vendre dans votre portefeuille.");
      return;
    }

    String action=(String)mClient.getValueAt(i, 0);
    int qte=Integer.valueOf((String)mClient.getValueAt(i, 1)).intValue();
    TransactionScreen transac=new TransactionScreen(this, action, qte,
                                       TransactionScreen.VENDRE);
    if(transac.estValide())
    {
      ecoute=COMMAND_BUYSELL;
      client.sendSell(action, transac.getQuantite());
    }

  }



  // Ajoute une ligne de texte a l'ecran de log :
  public void addLog(String chaine)
  {
    tLog.setCaretPosition(tLog.getText().length());
    tLog.replaceSelection(chaine+"\n");
    tLog.select(tLog.getText().length()-1, tLog.getText().length()-1);
  }



  // Gestion centralisee des messages d'erreur - dans une boite de dialogue :
  public void boiteErreur(String message)
  {
    JOptionPane.showMessageDialog(this, message, "Erreur",
                                  JOptionPane.ERROR_MESSAGE);
  }



  public static void main(String args[])
  {
    // L'application essaie de prendre le Look And Feel du systeme courant :
    try
    {
      UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName() );
    }
    catch(Exception e)
    {}

    // Cree la fenetre principale de l'application :
    JFrame fenetre = new ClientBourse();

    // Lecture de la taille de l'écran :
    java.awt.Dimension tailleEcran =
        java.awt.Toolkit.getDefaultToolkit().getScreenSize();

    // Lorsque la fenetre sera fermee, l'application quittera :
    fenetre.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    // Verification d'integrite des controles de la fenetre :
    fenetre.pack();

    // Maintenant que la fenetre a sa taille finale, la centre sur l'ecran :
    fenetre.setLocation( (tailleEcran.width-fenetre.getWidth())/2,
                         (tailleEcran.height-fenetre.getHeight())/2 );

    // Affiche la fenetre pour notre cher utilisateur :
    fenetre.setVisible(true);
  }
};
```

TransactionScreen.java • Fenêtre de réglage d'une transaction

```
import java.lang.*;
import java.io.*;
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import java.beans.*;



class TransactionScreen extends JDialog implements ActionListener,
ChangeListener
{
  public static final int ACHETER=1;
  public static final int VENDRE=2;

  private class ScreenGestionSouris extends MouseAdapter
  {
    // Gestion des evenements du type clic sur les boutons :
    public void mouseReleased(MouseEvent e)
    {
      if(e.getSource()==bOK)
      {
        try
        {
          quantite=Integer.valueOf(tQuantite.getText()).intValue();

          // Controle des valeurs :
          if(quantite<=0) {boiteErreur("Quantite d'action invalide."); return;}

          valide=true;
          dispose();
        }
        catch(Exception except)
        {
          boiteErreur("Quantite d'action invalide.");
        }

      }
      if(e.getSource()==bCancel)
      {
        dispose();
      }
    }
  };



  // Controles de la fenetre :
  private JLabel aTextel, aTexte2;
  private JTextField tQuantite;
  private JButton bOK, bCancel;
  private JSlider gliss;
  private JPanel cadre;
  private GridBagContainLayout layout;
  private ScreenGestionSouris gSouris;

  private ClientBourse cb;

  // Membres a renvoyer :
  boolean valide;
  int quantite;


  public TransactionScreen(ClientBourse cb, String action, int qteMax,
                           int mode)
  {
    super(cb, true);
    setTitle("Transaction");
    this.cb=cb;
    valide=false;

    // Cree les boutons, les labels, et les textfields :
    aTextel=new JLabel();
    if(mode==ACHETER) aTextel.setText("Acheter");
    else aTextel.setText("Vendre");
    aTexte2=new JLabel("actions de " + action + ".");
    tQuantite=new JTextField(String.valueOf((int)qteMax/2));
    tQuantite.setMinimumSize(new Dimension(40,10));
    bOK=new JButton("OK");
    bCancel=new JButton("Annuler");

    // Slider :
    gliss=new JSlider();
    int mintick=qteMax/50;
    int maxtick=qteMax/10;
    if(maxtick<1) {mintick=2; maxtick=10;}
    gliss.setMinorTickSpacing(mintick);
    gliss.setMajorTickSpacing(maxtick);
    gliss.setMaximum(qteMax);
    gliss.setMinimum(0);
    gliss.setPaintTicks(true);

    // Les place dans un cadre intermediaire :
    cadre=new JPanel();
    GridBagContainLayout intra=new GridBagContainLayout();
    cadre.setLayout(intra);
    intra.constrain(cadre, aTextel, 0, 0, 1, 1, 1, 0.5);
    intra.constrain(cadre, tQuantite, 1, 0, 1, 1, 1, 0.5);
    intra.constrain(cadre, aTexte2, 2, 0, 1, 1, 1, 0.5);
    intra.constrain(cadre, gliss, 0, 1, 3, 1, 1, 0.5);

    // Place le tout dans le layout de la fenetre :
    layout=new GridBagContainLayout();
    getContentPane().setLayout(layout);
    layout.constrain(getContentPane(), cadre, 0, 0, 2, 1, 10, 0.5);
    layout.constrain(getContentPane(), bOK, 0, 1, 1, 1, 10, 0.5);
    layout.constrain(getContentPane(), bCancel, 1, 1, 1, 1, 10, 0.5);

    // Listeners des boutons :
    gSouris=new ScreenGestionSouris();
    bOK.addMouseListener(gSouris);
    bCancel.addMouseListener(gSouris);

    // Interaction slider et textfield :
    tQuantite.addActionListener(this);
    gliss.addChangeListener(this);

    // Centre la fenetre :
    pack();
    java.awt.Dimension tailleEcran =
          java.awt.Toolkit.getDefaultToolkit().getScreenSize();
    setLocation( (tailleEcran.width-getWidth())/2,
                 (tailleEcran.height-getHeight())/2 );

    setResizable(false);
    gliss.setValue((int)qteMax/2);
    setVisible(true);


  }


  // Gestion centralisee des messages d'erreur - dans une boite de dialogue :
  public void boiteErreur(String message)
  {
    JOptionPane.showMessageDialog(this, message, "Erreur",
                                  JOptionPane.ERROR_MESSAGE);
  }


  public void actionPerformed(ActionEvent e)
  {
    try
    {
      Integer value = Integer.valueOf(tQuantite.getText());
      gliss.setValue(value.intValue());
    }
    catch(Exception except)
    {}
  }


  public void stateChanged(ChangeEvent e)
  {
    JSlider source = (JSlider)e.getSource();
    int qte=(int)source.getValue();
    if(!source.getValueIsAdjusting())
    {
      //tQuantite.setText(new Integer(qte));
    }
    else
    {
      tQuantite.setText(String.valueOf(qte));
    }
  }


  // Accesseurs pour recuperer les donnees :
  boolean estValide() {return valide;}
  int getQuantite() {return quantite;}
}
```

LogScreen.java • Ecran de connexion à un serveur

```
import java.lang.*;
import java.io.*;
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;



class LogScreen extends JDialog
{

  public static final String FICHIER_SERVEURS="listeserveurs.cfg";

  private class ScreenGestionSouris extends MouseAdapter
  {
    // Gestion des evenements du type clic sur les boutons :
    public void mouseReleased(MouseEvent e)
    {
      if(e.getSource()==bConnect)
      {
        actionConnect();
      }
      if(e.getSource()==bCancel)
      {
        dispose();
      }
      if(e.getSource()==bAdd)
      {
        actionAdd();
      }
      if(e.getSource()==bSupp)
      {
        actionSupp();
      }
    }
  };

  // Controles de la fenetre :
  private JLabel aServeur, aLogin;
  private JTextField tLogin;
  private DefaultListModel mListe;
  private JList tListe;
  private JPanel cServeur, cLogin;
  private JButton bConnect, bCancel, bAdd, bSupp;
  private JScrollPane dListe;
  private GridBagContainLayout layout;
  private ScreenGestionSouris gSouris;


  private ClientBourse client;


  // Membres a renvoyer :
  boolean valide;
  String nom;
  String adresse;
  int port;
```

```
  public LogScreen(ClientBourse cb)
  {
    super(cb, true);
    setTitle("Connexion");
    valide=false;
    client=cb;


    // Cree les boutons et les labels :
    aServeur=new JLabel("Serveur : ");
    aLogin=new JLabel("Login :");
    bConnect=new JButton("Connexion");
    bCancel=new JButton("Annuler");
    bAdd=new JButton("Ajouter");
    bSupp=new JButton("Supprimer");
    tLogin=new JTextField();

    // Cree la liste :
    mListe=new DefaultListModel();
    loadServeurs();
    tListe=new JList(mListe);
    dListe=new JScrollPane(tListe);
    if(mListe.size()>0) tListe.setSelectedIndex(0);

    // Cree le cadre Login :
    cLogin=new JPanel();
    cLogin.setLayout(new GridLayout(2, 1, 10, 10));
    cLogin.add(aLogin);
    cLogin.add(tLogin);

    // Cree le cadre Serveur :
    cServeur=new JPanel();
    GridBagContainLayout intra=new GridBagContainLayout();
    cServeur.setLayout(intra);
    intra.constrain(cServeur, aServeur, 0, 0, 2, 1, 5, 0.5);
    intra.constrain(cServeur, dListe, 0, 1, 1, 2, 5, 0.5);
    intra.constrain(cServeur, bAdd, 1, 1, 1, 1, 5, 0.5);
    intra.constrain(cServeur, bSupp, 1, 2, 1, 1, 5, 0.5);

    // Place tout ca dans le layout :
    layout=new GridBagContainLayout();
    getContentPane().setLayout(layout);
    layout.constrain(getContentPane(), cLogin, 0, 0, 2, 1, 10, 0.0);
    layout.constrain(getContentPane(), cServeur, 0, 1, 2, 1, 10, 0.0);
    layout.constrain(getContentPane(), bConnect, 0, 2, 1, 1, 20, 0.5);
    layout.constrain(getContentPane(), bCancel, 1, 2, 1, 1, 20, 0.5);

    // Listeners des boutons :
    gSouris=new ScreenGestionSouris();
    bConnect.addMouseListener(gSouris);
    bCancel.addMouseListener(gSouris);
    bAdd.addMouseListener(gSouris);
    bSupp.addMouseListener(gSouris);

    // Centre la fenetre :
    pack();
    java.awt.Dimension tailleEcran =
                java.awt.Toolkit.getDefaultToolkit().getScreenSize();
    setLocation( (tailleEcran.width-getWidth())/2,
                 (tailleEcran.height-getHeight())/2 );
    setResizable(false);
    setVisible(true);

  }


  public void loadServeurs()
  {
    try
    {
      FileInputStream istream=new FileInputStream(FICHIER_SERVEURS);
      ObjectInputStream p=new ObjectInputStream(istream);

      // Recupere l'objet grace a la magie de la serialisation :
      mListe=(DefaultListModel)p.readObject();
      istream.close();
    }
    catch(ClassNotFoundException except)
    {}
    catch(FileNotFoundException except)
    {}
    catch(IOException except)
    {}
  }


  public void saveServeurs()
  {
    try
    {
      FileOutputStream ostream = new FileOutputStream(FICHIER_SERVEURS);
      ObjectOutputStream p=new ObjectOutputStream(ostream);

      // Grace a la magie de la serialisation, cette simple fonction enregistre
      // la totalite de l'objet dans le fichier specifie !
      p.writeObject(mListe);
      p.flush();
      ostream.close();
    }
    catch(FileNotFoundException except)
    {}
    catch(IOException except)
    {}
  }



  public void actionAdd()
  {
    LogAddScreen nouveau=new LogAddScreen(this);
    if(nouveau.estValide())
    {
      String nouvserveur=nouveau.getNomServeur() + " <" +
             nouveau.getAdresse() + ":" + nouveau.getPort() + ">";
      mListe.addElement(nouvserveur);
      tListe.setSelectedIndex(mListe.size()-1);
    }
    saveServeurs();
  }


  public void actionSupp()
  {
    int i=tListe.getSelectedIndex();
    if(i>=0)
    {
      mListe.removeElementAt(i);
    }
    saveServeurs();
  }


  public void actionConnect()
  {
    try
    {
      nom=tLogin.getText();
      String element=(String)mListe.getElementAt(tListe.getSelectedIndex());
      StringTokenizer st=new StringTokenizer(element, "<>");
      if(st.hasMoreTokens())
        st.nextToken();
      if(st.hasMoreTokens())
      {
        element=st.nextToken();
        StringTokenizer st2=new StringTokenizer(element, ":");
        if(st2.hasMoreTokens()) adresse=st2.nextToken();
        if(st2.hasMoreTokens())
           port=Integer.valueOf(st2.nextToken()).intValue();
      }

      // Controle :
      if(nom==null || nom.equals(""))
         {boiteErreur("Entrez un login."); return;}
      if(adresse==null || adresse.equals(""))
         {boiteErreur("Adresse IP invalide."); return;}
      if(port<=0)
         {boiteErreur("Numero de port invalide."); return;}

      valide=true;
      dispose();
    }
    catch(Exception except)
    {
      boiteErreur("Serveur invalide");
    }

  }


  // Gestion centralisee des messages d'erreur - dans une boite de dialogue :
  public void boiteErreur(String message)
  {
    JOptionPane.showMessageDialog(this, message, "Erreur",
                                  JOptionPane.ERROR_MESSAGE);
  }


  // Accesseurs pour recuperer les donnees :
  boolean estValide() {return valide;}
  String getNom() {return nom;}
  String getAdresse() {return adresse;}
  int getPort() {return port;}

};
```

LogAddScreen.java • Fenêtre d'ajout d'un serveur

```
import java.lang.*;
import java.io.*;
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;


class LogAddScreen extends JDialog
{
  private class ScreenGestionSouris extends MouseAdapter
  {
    // Gestion des evenements du type clic sur les boutons :
    public void mouseReleased(MouseEvent e)
    {
      if(e.getSource()==bOK)
      {
        try
        {
          nomServeur=tNom.getText();
          adresse=tIP.getText();
          port=Integer.valueOf(tPort.getText()).intValue();

          // Controle des valeurs :
          if(nomServeur==null || nomServeur.equals(""))
            {boiteErreur("Entrez un nom pour le serveur."); return;}
          if(adresse==null || adresse.equals(""))
            {boiteErreur("Veuillez entrer une adresse IP."); return;}
          if(port<=0)
            {boiteErreur("Numero de port invalide."); return;}

          valide=true;
          dispose();
        }
        catch(Exception except)
        {
          boiteErreur("Numero de port invalide.");
        }

      }
      if(e.getSource()==bCancel)
      {
        dispose();
      }
    }
  };



  // Controles de la fenetre :
  private JLabel aNom, aIP, aPort;
  private JTextField tNom, tIP, tPort;
  private JButton bOK, bCancel;
  private JPanel cadre;
  private GridBagContainLayout layout;
  private ScreenGestionSouris gSouris;


  // Membres a renvoyer :
  boolean valide;
  String nomServeur;
  String adresse;
  int port;
```

```
  public LogAddScreen(LogScreen ls)
  {
    super(ls, true);
    setTitle("Nouveau serveur");
    valide=false;

    // Cree les boutons, les labels, et les textfields :
    aNom=new JLabel("Nom :");
    aIP=new JLabel("IP :");
    aPort=new JLabel("Port :");
    tNom=new JTextField("Nouveau serveur");
    tIP=new JTextField("casimir");
    tPort=new JTextField();
    bOK=new JButton("OK");
    bCancel=new JButton("Annuler");

    // Les place dans un cadre intermediaire :
    cadre=new JPanel();
    GridBagContainLayout intra=new GridBagContainLayout();
    cadre.setLayout(intra);
    intra.constrain(cadre, aNom, 0, 0, 1, 1, 5, 0.5);
    intra.constrain(cadre, aIP, 0, 1, 1, 1, 5, 0.5);
    intra.constrain(cadre, aPort, 0, 2, 1, 1, 5, 0.5);
    intra.constrain(cadre, tNom, 1, 0, 1, 1, 5, 0.5);
    intra.constrain(cadre, tIP, 1, 1, 1, 1, 5, 0.5);
    intra.constrain(cadre, tPort, 1, 2, 1, 1, 5, 0.5);

    // Place le tout dans le layout de la fenetre :
    layout=new GridBagContainLayout();
    getContentPane().setLayout(layout);
    layout.constrain(getContentPane(), cadre, 0, 0, 2, 1, 10, 0.5);
    layout.constrain(getContentPane(), bOK, 0, 1, 1, 1, 10, 0.5);
    layout.constrain(getContentPane(), bCancel, 1, 1, 1, 1, 10, 0.5);

    // Listeners des boutons :
    gSouris=new ScreenGestionSouris();
    bOK.addMouseListener(gSouris);
    bCancel.addMouseListener(gSouris);


    // Centre la fenetre :
    pack();
    java.awt.Dimension tailleEcran =
        java.awt.Toolkit.getDefaultToolkit().getScreenSize();
    setLocation( (tailleEcran.width-getWidth())/2,
                 (tailleEcran.height-getHeight())/2 );
    setResizable(false);
    setVisible(true);

  }


  // Gestion centralisee des messages d'erreur - dans une boite de dialogue :
  public void boiteErreur(String message)
  {
    JOptionPane.showMessageDialog(this, message, "Erreur",
                                  JOptionPane.ERROR_MESSAGE);
  }


  // Accesseurs pour recuperer les donnees :
  boolean estValide() {return valide;}
  String getNomServeur() {return nomServeur;}
  String getAdresse() {return adresse;}
  int getPort() {return port;}
}
```

ClientInterne.java • Classe chargée de la communication asynchrone avec le serveur

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.net.UnknownHostException;


public class ClientInterne
{
  public class Recepteur extends Thread
  {
    public void run()
    {
             BufferedReader reader = null;

      try
      {
        reader=new BufferedReader(new
                    InputStreamReader(socket.getInputStream()));
        while(doitSortir==false)
        {
          String line = reader.readLine();
          if (line != null)
          {
            cb.connectionReceive(line);
          }
          else
          {
            cb.connectionLost();
            close();
          }
        }
      }
      catch (IOException e)
      {
        cb.connectionLost();
      }
      close();
    }
  };


  public class ClientException extends Exception {};
  public class UnableToConnectException extends ClientException {};
  public class HostNotFoundException extends UnableToConnectException {};

  private int numport;
  private String ip;
  private boolean connecte;
  private PrintWriter writer;
  private BufferedReader reader;
  private ClientBourse cb;
  private Recepteur recept;

  private Socket socket;
  private boolean doitSortir;


  public ClientInterne(ClientBourse cb)
  {
    this.cb=cb;
    ip=null;
    numport=0;
    connecte=false;
    doitSortir=false;
    writer=null;
    reader=null;
    socket=null;
  }


  public void connect(String ip, int numport) throws UnableToConnectException
  {
    if(connecte==true) throw new UnableToConnectException();
    this.ip=ip;
    this.numport=numport;

    try
    {
      socket=new Socket(ip, numport);
      connecte=true;
      doitSortir=false;
      recept=new Recepteur();
      recept.start();
      writer=new PrintWriter(socket.getOutputStream(),true);
      reader=new BufferedReader(new
                  InputStreamReader(socket.getInputStream()));
    }
    catch (UnknownHostException e)
    {
      throw new HostNotFoundException();
    }
    catch (IOException e)
    {
      throw new UnableToConnectException();
    }
  }



  public boolean send(String chaine)
  {
    cb.addLog(chaine);
    writer.println(chaine);
    writer.flush();
    return true;

  }

  public void sendLogin(String nom)
  {
    if(connecte) send("login " + nom);
  }

  public void sendAddUser(String nom)
  {
    if(connecte) send("add user " + nom);
  }

  public void sendList()
  {
    if(connecte) send("list");
  }

  public void sendListMarket()
  {
    if(connecte) send("list market");
  }

  public void sendBuy(String action, int qte)
  {
    if(connecte) send("buy "+action+" "+qte);
  }

  public void sendSell(String action, int qte)
  {
    if(connecte) send("sell "+action+" "+qte);
  }

  public void sendLogout()
  {
    if(connecte)
    {
      send("logout");
      close();
    }
  }

  public void close()
  {
    if(connecte)
    {
      doitSortir=true;
      connecte=false;
      try
      {
        socket.close();
      }
      catch(IOException except)
      {}
    }
  }

};
```