CliniPlug : Client

mise à jour : 02/11/2014

1 - Présentation

a) CliniPlug

CliniPlug est un logiciel permettant à tout logiciel métier d'utiliser les fonctionnalités de freeSAM.

CliniPlug associe :
- un serveur TCP
- le système de gestion des connaissances de freeSAM (système expert, arbres décisionnels ...)

Le serveur de CliniPlug est un lien entre un logiciel métier et les services de freeSAM.

Le logiciel métier doit simplement incorporer un client TCP.

Le client TCP se connecte au serveur à l'aide d'une socket. Le numéro du port utilisé est 6000.

Le serveur est automatiquement lancé lors de l'exécution de CliniPlug et prêt à établir des connexions avec des clients.
Un client se comporte comme un navigateur internet qui envoie des requêtes à un serveur Web.

La communication entre un client et le serveur utilise des messages prédéfinis qui seront détaillés par la suite.

Avertissement :

CliniPlug ne doit pas être utilisé à des fins commerciales sans l'autorisation de ses concepteurs.

Cette version est limitée à 1 seul client possible par serveur.
Pour une utilisation commerciale prière d'envoyer un message aux concepteurs : page contact du site freeSAM.

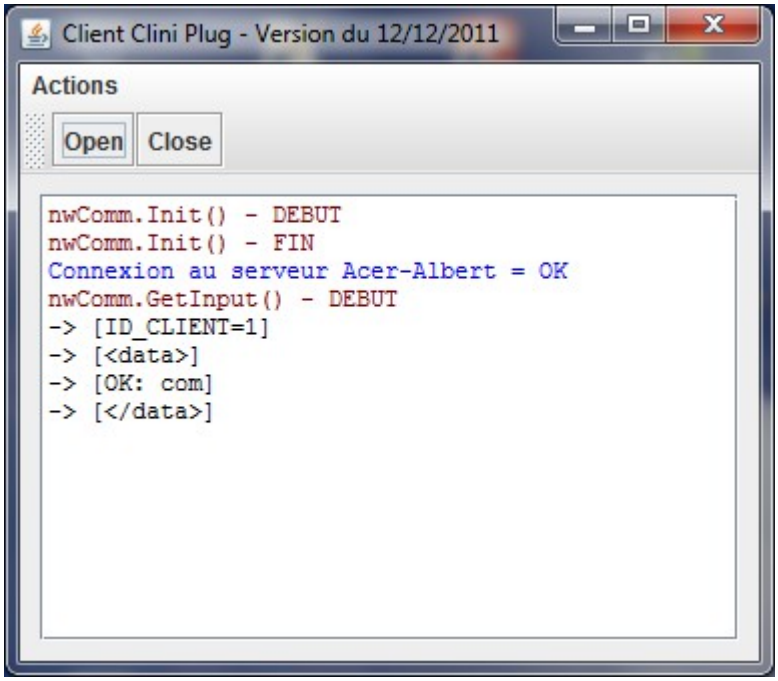
b) Client CliniPlug

Le client CliniPlug est un client TCP de démonstration à l'utilisation du serveur CliniPlug.
Il simule un logiciel métier utilisant freeSAM.

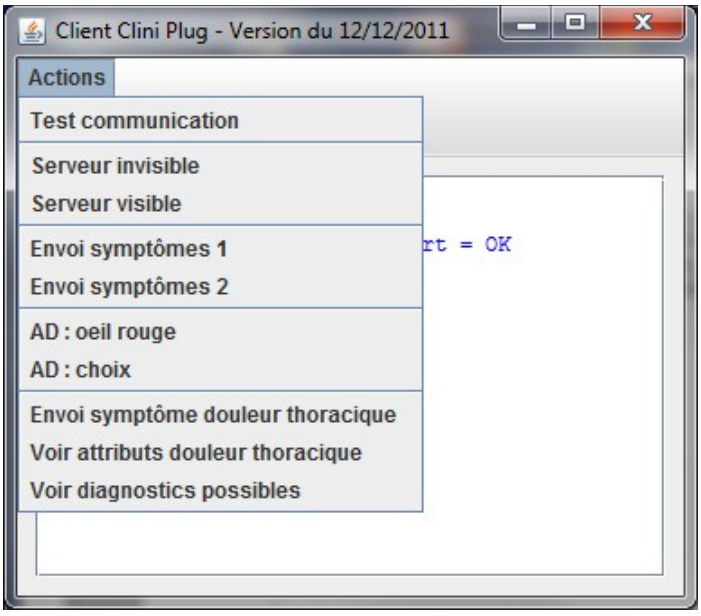
La suite du document est consacrée à la gestion pratique du client d'un logiciel métier.

Pour tester CliniPlug à l'aide du client CliniPlug :

Lancer CliniPlug puis le client CliniPlug.
Cliquer sur le bouton 'Open' du client pour établir la connexion avec le serveur.
Quand cette connexion est établie, le client et le serveur peuvent échanger des messages.
Quand tous les tests sont finis, cliquer sur 'Close' pour fermer définitivement le serveur et mettre fin à la communication.



Le client peut communiquer avec le serveur après un clic sur le bouton 'Open'
Un test de communication a été fait à l'aide du menu 'Actions/Test communication'.



2 - Les messages envoyés par le client

Les messages circulant entre le serveur et son client sont des lignes de texte contenues dans des blocs délimités par la balise <data>
Ils permettent d'effectuer des requêtes auprès du serveur et de recueillir des informations en retour.

► **Ex : test de communication avec le serveur (Menu 'Test communication')**

Le client envoie	Le serveur retourne
<data> sys: test com </data>	<data> OK: com </data>

Le balisage <data> permet de vérifier que tous les messages sont bien arrivés à destination.

Un message est un ligne de texte comportant :
- un identificateur de type de message
- le séparateur ':'
- le message lui même

1) Messages 'sys'

```
-----------------------------------------------------
sys: charge BC
-----------------------------------------------------
```

Demande au serveur de charger la base de connaissances (dictionnaire, règles).
Cette demande est à faire une seule fois avant de pouvoir utiliser le système expert.

```
-----------------------------------------------------
sys: unités=anciennes/SI
-----------------------------------------------------
```

Définit le système d'unité utilisé pour entrer les données.
anciennes → unités anciennes
SI → unités récentes du système international

```
-----------------------------------------------------
sys: nouvelle consultation
-----------------------------------------------------
```

Initialise le système expert pour une nouvelle consultation.

```
-----------------------------------------------------
sys: set visible=true/false
-----------------------------------------------------
```

Rend la fenêtre serveur visible ou non.

Test : Menu 'Serveur visible / Serveur invisible'

---

sys: close

Ferme CliniPlug définitivement avec disparition de la fenêtre serveur si elle était visible.

Test : Clic bouton 'Close'

---

sys: test com

Teste la communication avec le serveur de CliniPlug.
Le serveur retourne le message client 'OK: com' si la communication est bien établie.
Test : Menu 'Test communication'

---

sys: getBoxAttribut(nom symptome)

Demande au serveur d'afficher une boite de dialogue permettant la saisie des attribut du symptôme dont le nom est précisé.

Voir sys: getDiagnosticsPossibles()

---

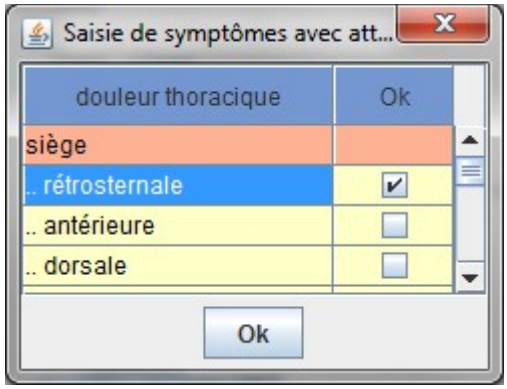
sys: getBoxDiagnosticsPossibles()

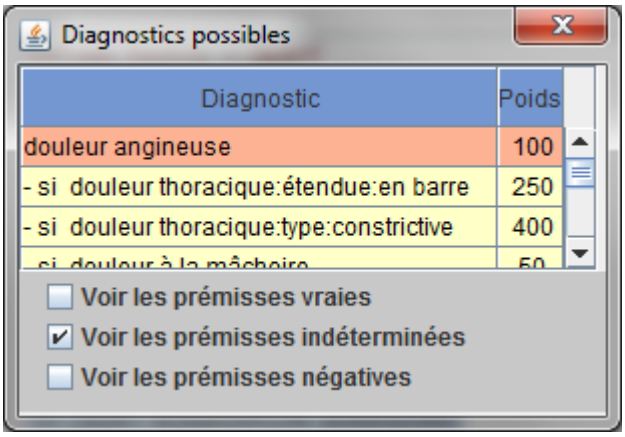
Demande au serveur de montrer la boite des diagnostics possibles.

► **Ex : Menu ' Envoi symptôme douleur thoracique ' + ' Voir attributs douleur thoracique ' + ' Voir diagnostics possibles '**

Le client envoie	Le serveur retourne

`<data>`
`st: douleur thoracique`
`</data>`
	`<data>]`
	`sys: attribut(douleur thoracique)`
	`</data>`

`<data>`
`sys: getAttribut(douleur thoracique)`
`</data>`
	
	
	Après clic sur bouton Ok :

	<data> sys: diagnostics possibles </data>
<data> sys: getDiagnosticsPossibles() </data>	 Après fermeture de la boite : <data> OK: to you </data>

2) Messages 'st'

----

st: symptome

Envoi de symptômes vers le système expert de CliniPlug
Le serveur retourne les éventuelles déductions faites.

► Ex : Menu ' Envoi symptômes 1 '

Le client envoie	Le serveur retourne
<data> sys: charge BC sys: unités=anciennes st: Hb=10 st: age=25 st: sexe=f st: mal à la tête st: cholestérol LDL=1,90 st: toto </data>	<data> OK: charge BC OK: unités se: anémie lienAD: AD_anemie se: céphalées err: mot inconnu=toto </data>

C'est la première consultation demandée au système expert de cliniPlug.
Il faut donc charger la base de connaissances.
Le système d'unité utilisé pour entrer les données est précisé.
Les symptômes sont entrés – Ex : Hb=10 → hémoglobine=10 g/dl

Le serveur retourne des messages :

- la base de connaissance est bien chargée
- le système d'unités est pris en compte
- 'anémie' est déduit par le système expert (d'après le taux d'hémoglobine)
- un arbre de décision d'aide au diagnostic d'anémie est signalé
- 'céphalées' est déduit de 'mal à la tête'
- le symptôme 'toto' est signalé comme inconnu

► Ex : Menu ' Envoi symptômes 2 '

Le client envoie	Le serveur retourne
<data> sys: nouvelle consultation st: poids=150 st: taille=1,50 st: age=50 </data>	<data> OK: nouvelle consultation se: IMC = 66,67 se: obésité se: obésité morbide info: Une chirurgie de l'obésité peut être envisagée. </data>

Pour la 2ème consultation, il n'est plus nécessaire de charger la base de connaissances mais il faut réinitialiser le système expert pour annuler les symptômes précédents (sys: nouvelle consultation)
Les nouveaux symptômes sont entrés.

Le serveur retourne des messages :

- la nouvelle consultation est bien initialisée
- déductions : calcul IMC, obésité
- informations : chirurgie obésité possible.

3) Messages 'AD'

AD: exe=nom fichier

Exécute un arbre de décision.
Voir 'AD: charge'

AD: montre

Montre la fenêtre gérant les arbres de décision
Voir 'AD: charge'

AD: charge

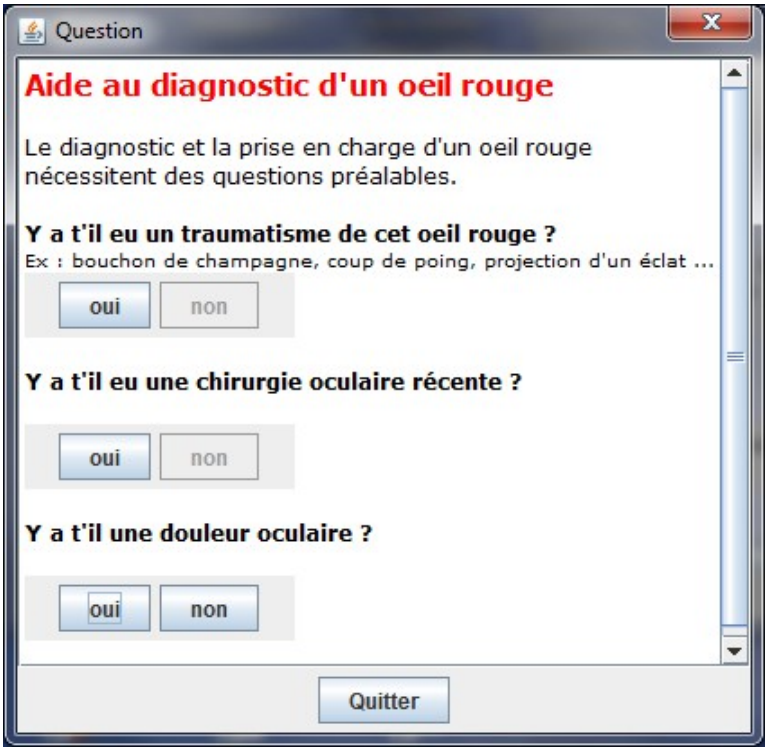
Ouvre une boite permettant de choisir un arbre qui sera chargé et exécuté.
► Ex : Menu ' AD : oeil rouge '

Le client envoie :

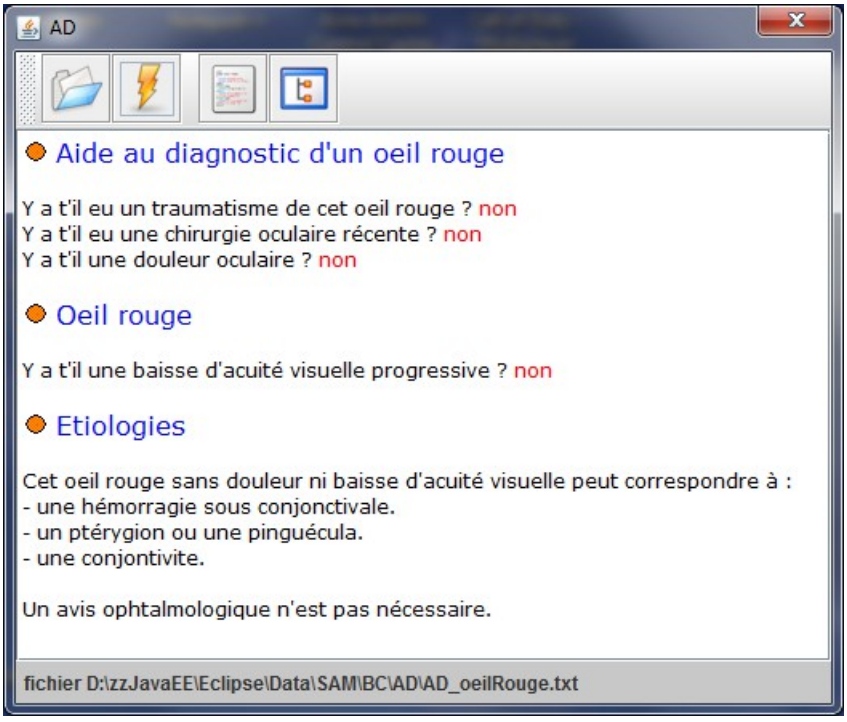
```
<data>
AD: exe=AD_oeilRouge
AD: montre
</data>
```

Le serveur affiche 2 fenêtres :

Une fenêtre qui exécute les formulaires de l'arbre de décision.



Une fenêtre de gestion de l'arbre de décision.

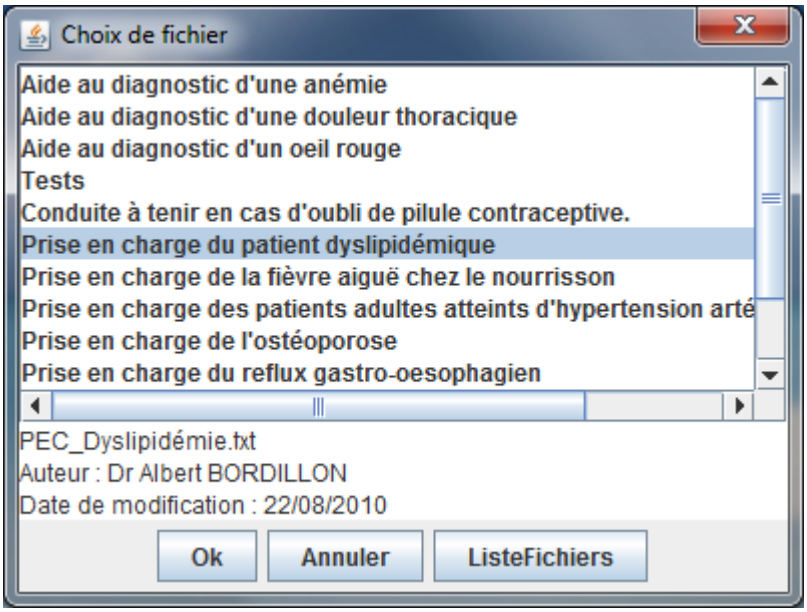


► **Ex : Menu ' AD : choix '**

Le client envoie :

```
<data>
AD: charge
</data>
```

Le serveur affiche une boite de choix de fichier AD à charger puis à exécuter.



3 - Les messages reçus par le client

Ces messages ont été vus lors des exemples précédents.
En voici tout de même la liste.

OK: message

Indique le bon déroulement d'une action demandée auparavant au serveur.

OK: charge BC	La base de connaissances est bien chargée.
OK : unités	Le système d'unité a bien été défini.
OK : set visible	La modification de visibilité du serveur a bien été prise en compte.
OK: to you	L'opération demandée est finie – On peut envoyer une nouvelle requête.
OK: com	La communication avec le serveur est établie.

---

mod: nom symptôme

Le symptôme spécifié a été modifié – Ex : âge = 20 puis âge = 50
Une telle modification réinitialise tout le système expert avec la nouvelle donnée.

---

se: déduction

Signale une déduction.
Ex - se: obésité

---

info: information

Information pour l'utilisateur.
Ex - info: chirurgie de l'obésité possible

La distinction déduction / information permet d'afficher différemment ces données.
Les déduction signalent plutôt des diagnostics.

---

lienAD: nom fichier AD

Signale qu'un arbre de décision dont le nom est précisé peut être consulté.
Ceci peut arriver quand cet arbre peut faciliter un diagnostic.

Ex - lienAD: AD_anémie
→ on peut charger le fichier 'AD_anémie' par l'envoi du message 'AD: exe=AD_anémie'
→ aide au diagnostic d'une anémie.

---

err: mot inconnu=nom mot

Signale un mot inconnu du système expert.

Ex - err: mot inconnu=toto → indique que 'toto' est un mot inconnu.

---

err: symptôme inconnu=symptôme

Signale un symptôme inconnu, en particulier lorsqu'il est muni d'attributs.
Les attributs sont séparés du symptôme par ':'
Ex – 'douleur thoracique:augmentée par:effort (2 attributs pour 'douleur thoracique')

Ex - err: symptôme inconnu=douleur thoracique:verte
→ signale que 'verte' est un attribut qui n'est pas associé à 'douleur thoracique'

---

sys: attribut(nom symptôme)

Signale qu'un symptôme possède des attributs servant à le décrire

Ex - sys: attribut(douleur thoracique)
→ signale que le symptôme 'douleur thoracique' possède des attributs.
→ demander l'affichage de la boite de saisie des attributs par l'envoi du message
' sys: getAttribut(douleur thoracique) '

---

sys: diagnostics possibles

Signale que des diagnostics sont possibles.

Ex - sys: diagnostics possibles
→ demander d'afficher ces diagnostics par l'envoi du message 'sys: getDiagnosticsPossibles()'

4 – Code source du client CliniPlug

Ce code utilise une bibliothèque non fournie permettant :

- d'afficher du texte enrichi dans une fenêtre
- de gérer des menus, barres d'outils ...
- de programmer les sockets d'un client TCP

L'intérêt de ces sources est le code des fonctions action0, action1 ...

```
/*************************************************************************
 * Fenêtre principale de l'application ClientCP.
 *
 * <pre>
 * Simule un client relié à CliniPlug.
 * Le menu action -> envoi de messages au serveur CliniPlug
 * Nécessite :
 *   - Papou </pre>
 *
 * Créé le 12/12/2011 - Modifié le 07/06/2014 - Imprimé le 16/03/2013
 * @author Albert BORDILLON
 *************************************************************************/

public class FenPrincClientCP extends JFrame implements ActionListener, WindowListener,
                                                         IObjetTraiteInput
{
	// ------------------------------------------------------------------
	// Propriétés
	// ------------------------------------------------------------------

	private static final long serialVersionUID = -38893737165829360L;

	private CLayout                    layoutPrinc ;
	private JPanel                     panelPrinc ;

	/** Zone d'affichage des informations.*/
	private CTextPane                  tpInfo ;

	/** Client TCP */
	private nwClient                   client ;

	/** Barre de menu.*/
	private CMenubar                   menubar ;

	/** Barre d'outils.*/
	private CToolbar                   toolbar ;

	/** Boutons Open et Close.*/
	public  JButton                    bOpen, bClose ;

	/** true si la base de connaissances est chargée.*/
	private boolean                    chargeBC=false ;

	// ------------------------------------------------------------------
	// Constructeur
	// ------------------------------------------------------------------

	public FenPrincClientCP(String Titre)
	{
		super(Titre) ;
		addWindowListener(this) ;

		// ------------------------------------------------------------
		// Fermeture de l'application à la fermeture de la fenêtre
		//
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE) ;

		// ------------------------------------------------------------
		// Création de l'interface visuelle
```

```
        //
        // Création du panel principal
        panelPrinc=new JPanel() ;
        // Création du layout du panel principal
        layoutPrinc=new CLayout(panelPrinc) ; panelPrinc.setLayout(layoutPrinc) ;
        // Création du CTextPane d'informations
        tpInfo=new CTextPane() ;
        layoutPrinc.Put(tpInfo.GetScrollPane(), "x=10, y=10, w=WC-20, h=HC-20") ;
        //
        this.add(panelPrinc, BorderLayout.CENTER) ;

        // ------------------------------------------------------------
        // Création et initialisation de la barre de menu
        //
        menubar = new CMenubar(this) ;

        menubar.addMenu("Actions") ;
        menubar.addSousMenu("Test communication", "m_action0") ;
        menubar.addSeparator() ;
        menubar.addSousMenu("Serveur invisible", "m_action1") ;
        menubar.addSousMenu("Serveur visible", "m_action2") ;
        menubar.addSeparator() ;
        menubar.addSousMenu("Envoi symptômes 1", "m_action3") ;
        menubar.addSousMenu("Envoi symptômes 2", "m_action4") ;
        menubar.addSeparator() ;
        menubar.addSousMenu("AD : oeil rouge", "m_action5") ;
        menubar.addSousMenu("AD : choix", "m_action6") ;
        menubar.addSeparator() ;
        menubar.addSousMenu("Envoi symptôme douleur thoracique", "m_action7") ;
        menubar.addSousMenu("Voir attributs douleur thoracique", "m_action8") ;
        menubar.addSousMenu("Voir diagnostics possibles", "m_action9") ;
        menubar.addSeparator() ;
        menubar.addSousMenu("Voir attributs déficit moteur", "m_action10") ;

        menubar.setEnabledMenu("Actions", false) ;
        setJMenuBar(menubar.menuBar) ;

        toolbar=new CToolbar(this) ;
        bOpen=toolbar.addBouton("Open") ;
        bClose=toolbar.addBouton("Close") ; bClose.setEnabled(false) ;
        this.add(toolbar, BorderLayout.NORTH) ;

        // ------------------------------------------------------------
        // Création du client
        //
        client=new nwClient() ;
        client.setTPinfo(tpInfo) ;
        client.setObjetTraiteInput(this) ;
        client.setDecorationMessages(false) ;
        client.setAffInput(false) ;
}

/*****************************************************************
 * Action0 : test de communication avec le serveur.
 *****************************************************************/

private void action0()
{
        client.Send("<data>") ;
        client.Send("sys: test com") ;
        client.Send("</data>") ;
}

/*****************************************************************
 * Action1 : le serveur devient invisible.
 *****************************************************************/

private void action1()
{
        client.Send("<data>") ;
        client.Send("sys: set visible=false") ;
        client.Send("</data>") ;
}
```

```
	/*********************************************************************
	 * Action2 : le serveur devient visible.
	 *********************************************************************/

	private void action2()
	{
		client.Send("<data>") ;
		client.Send("sys: set visible=true") ;
		client.Send("</data>") ;
	}

	/*********************************************************************
	 * Action3 : envoi de symptômes vers le serveur.
	 *********************************************************************/

	private void action3()
	{
		tpInfo.ajouteInfoS("Envoi symptômes 1\n", "bold") ;

		client.Send("<data>") ;
		if (! chargeBC) client.Send("sys: charge BC") ;
		client.Send("sys: nouvelle consultation") ;
		client.Send("sys: unités=anciennes") ;
		client.Send("st: Hb=10") ;
		client.Send("st: age=25") ;
		client.Send("st: sexe=f") ;
		client.Send("st: mal à la tête") ;
		client.Send("st: cholestérol LDL=1,90") ;
		client.Send("st: toto") ;
		client.Send("</data>") ;
		chargeBC=true ;
	}

	/*********************************************************************
	 * Action4 : envoi de symptômes vers le serveur.
	 *********************************************************************/

	private void action4()
	{
		tpInfo.ajouteInfoS("Envoi symptômes 2\n", "bold") ;

		client.Send("<data>") ;
		if (! chargeBC) client.Send("sys: charge BC") ;
		client.Send("sys: nouvelle consultation") ;
		client.Send("st: poids=150") ;
		client.Send("st: taille=1,50") ;
		client.Send("st: age=50") ;
		client.Send("</data>") ;
		chargeBC=true ;
	}

	/*********************************************************************
	 * Action5 : demande au serveur d'exécuter l'arbre de décision 'oeil rouge'
	 *********************************************************************/

	private void action5()
	{
		tpInfo.ajouteInfoS("AD : oeil rouge\n", "bold") ;

		client.Send("<data>") ;
		client.Send("AD: exe=AD_oeilRouge") ;
		client.Send("AD: montre") ;
		client.Send("</data>") ;
	}

	/*********************************************************************
	 * Action6 : demande au serveur de charger un arbre de décision qui sera choisi par
l'utilisateur.
	 *********************************************************************/

	private void action6()
	{
		tpInfo.ajouteInfoS("AD : choix\n", "bold") ;
```

```
	client.Send("<data>") ;
	client.Send("AD: charge") ;
	client.Send("</data>") ;
}

/*********************************************************************
 * Action7 : envoi du symptôme 'douleur thoracique' au serveur.
 *********************************************************************/

private void action7()
{
	tpInfo.ajouteInfoS("Envoi symptôme douleur thoracique\n", "bold") ;

	client.Send("<data>") ;
	if (! chargeBC) client.Send("sys: charge BC") ;
	client.Send("sys: nouvelle consultation") ;
	client.Send("st: douleur thoracique") ;
	client.Send("</data>") ;
	chargeBC=true ;
}

/*********************************************************************
 * Action8 : demande au serveur d'afficher la boite de saisie des attributs
 * du symptôme 'douleur thoracique'
 *********************************************************************/

private void action8()
{
	tpInfo.ajouteInfoS("Saisie attributs douleur thoracique\n", "bold") ;

	client.Send("<data>") ;
	if (! chargeBC) client.Send("sys: charge BC") ;
	client.Send("sys: nouvelle consultation") ;
	client.Send("sys: getBoxAttribut(douleur thoracique)") ;
	client.Send("</data>") ;
	chargeBC=true ;
}

/*********************************************************************
 * Action9 : demande au serveur d'afficher la boite des diagnostics possibles.
 *********************************************************************/

private void action9()
{
	tpInfo.ajouteInfoS("Voir diagnostics possibles\n", "bold") ;

	client.Send("<data>") ;
	client.Send("sys: getBoxDiagnosticsPossibles()") ;
	client.Send("</data>") ;
}

/*********************************************************************
 * Action10 : demande au serveur d'afficher la boite de saisie des attributs
 * du symptôme 'déficit moteur'
 *********************************************************************/

private void action10()
{
	tpInfo.ajouteInfoS("Saisie attributs déficit moteur\n", "bold") ;

	client.Send("<data>") ;
	if (! chargeBC) client.Send("sys: charge BC") ;
	client.Send("sys: nouvelle consultation") ;
	client.Send("sys: getBoxAttribut(déficit moteur)") ;
	client.Send("</data>") ;
	chargeBC=true ;
}

/*********************************************************************
 * Open : initialisation du client
 * -> ouverture de la communication avec le serveur.
 *********************************************************************/

private void actionOpen()
```

```
    {
          client.Init() ;
          client.Start() ;
          menubar.setEnabledMenu("Actions", true) ;
          bClose.setEnabled(true) ;
    }

    /*****************************************************************
     * Close : fermeture de la communication avec le serveur.
     *****************************************************************/

    private void actionClose()
    {
          client.Send("<data>") ;
          client.Send("sys: close") ;
          client.Send("</data>") ;
          client.Stop() ;
    }

    // ---------------------------------------------------------------
    // Gestion des évènements
    // (les menus et boutons de la barre d'outils ont des noms communs)
    // ---------------------------------------------------------------

    @Override
    public void actionPerformed(ActionEvent e)
    {
          String cmd = e.getActionCommand() ;

          if (cmd.equals("m_action0"))          action0() ;
          if (cmd.equals("m_action1"))          action1() ;
          if (cmd.equals("m_action2"))          action2() ;
          if (cmd.equals("m_action3"))          action3() ;
          if (cmd.equals("m_action4"))          action4() ;
          if (cmd.equals("m_action5"))          action5() ;
          if (cmd.equals("m_action6"))          action6() ;
          if (cmd.equals("m_action7"))          action7() ;
          if (cmd.equals("m_action8"))          action8() ;
          if (cmd.equals("m_action9"))          action9() ;
          if (cmd.equals("m_action10"))         action10() ;

          if (cmd.equals("Open"))             actionOpen() ;
          if (cmd.equals("Close"))            actionClose() ;
    }


    // ---------------------------------------------------------------
// Gestion des évènements Windows
// ---------------------------------------------------------------

    @Override
    public void windowActivated(WindowEvent arg0) {}

    @Override
    public void windowClosed(WindowEvent arg0) {}

    @Override
    public void windowClosing(WindowEvent arg0)
    {
          actionClose() ;
          System.exit(0) ;
    }

    @Override
    public void windowDeactivated(WindowEvent arg0) {}

    @Override
    public void windowDeiconified(WindowEvent arg0) {}

    @Override
    public void windowIconified(WindowEvent arg0) {}

    @Override
    public void windowOpened(WindowEvent arg0) {}
```

```
	// ---------------------------------------------------------------------
	// Fonction de traitement des entrées réseau
	// ---------------------------------------------------------------------

	@Override
	public void fctTraiteInput(String Input)
	{
		if (Input==null) return ;
		tpInfo.ajouteInfoln("-> [" + Input + "]") ;
	}

	// ---------------------------------------------------------------------
	// Fonction main
	// ---------------------------------------------------------------------

	public static void main(String[] args)
	{
		FenPrincClientCP fenPrinc = new FenPrincClientCP("Client Clini Plug - Version du
07/06/2014") ;
		fenPrinc.pack() ;
		fenPrinc.setSize(600, 600) ;
		fenPrinc.setVisible(true) ;
	}

}
```