Package java.net

Classes

— class ContentHandler
— class DatagramPacket
— class DatagramSocket
— class InetAddress
— final class ServerSocket
— final class Socket
— class SocketImpl
— final class URL
— abstract class URLConnection
— class URLEncoder
— class URLStreamHandler

Interfaces

Interface ContentHandlerFactory
Interface SocketImplFactory
Interface URLStreamHandlerFactory

Exceptions

class MalformedURLException
class ProtocolException
class SocketException
class UnknownHostException
class UnknownServiceException

Identifier le contenu d'un URL

```
import java.io.IOException;
import java.net.*;
public class ContenuURL {

  public static void afficheInfos(URLConnection u)
    {// affiche l'URL et les infos s'y rattachant
     System.out.println(u.getURL().toExternalForm() + ":");
     System.out.println("Contenu :"+ u.getContentType());
     System.out.println("Longueur :"+ u.getContentLength());
    }

  static public void main(String[] args)
  throws MalformedURLException, IOException
   {if (args.length==0) return;
    URL url=new URL(args[0]);
    URLConnection connection=url.openConnection();
    afficheInfos(connection);
   }
}
```

exemple d'exécution

```
>java affURL http://cuiwww.unige.ch/db-research/Research/
http://cuiwww.unige.ch/db-research/Research/:
Contenu :text/html
Longueur :3495
```

vérifier des URL

```
import java.io.*;
import java.net.*;

public class VerifieURL {

 public static void verifie(URLConnection u)
 throws IOException
    {if (!u.getContentType().equals("text/html"))
       {System.out.println(
              "seuls les text/html sont verifies.");
        System.exit(0);};
     DataInputStream dis =new
                        DataInputStream(u.getInputStream());
     String s=dis.readLine();
     String motif1="<A HREF=";
     String motif2=">";
     String ref;
     int i,j;
     while (s!=null)
       {i=s.indexOf(motif1);
        if (i!=-1)
          {j=s.indexOf(motif2);
           ref=s.substring(i+motif1.length()+1,j-1);
           try
             {ValiditeURLConnexion.test(ref);}
           catch (IOException e)
             {System.out.println("Wrong URL : "+ref);}
           }
        s=dis.readLine();}
     dis.close();
    }
```

vérifier des URL

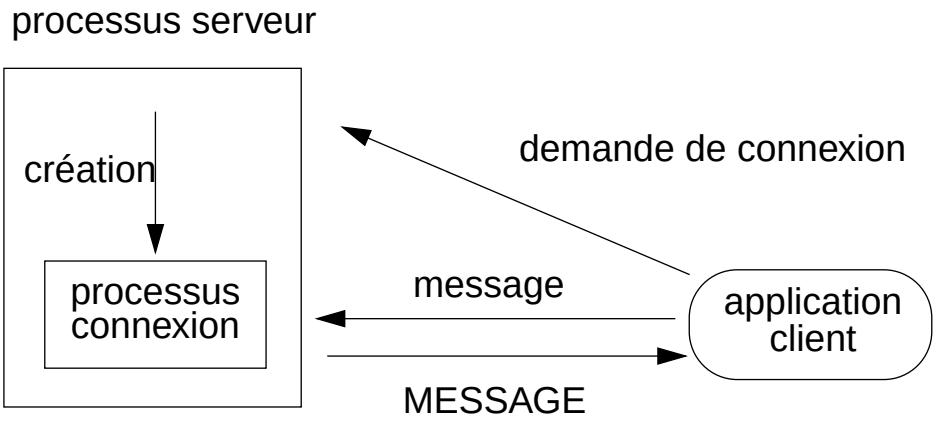
(suite)

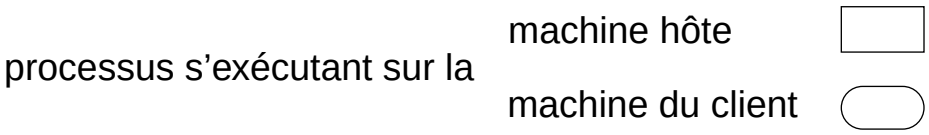
```
public static void main(String args[])
  throws MalformedURLException, IOException
    {if (args.length!=1)
      {System.out.println("Usage : java verifieURL ");
       System.exit(0);}
     URLConnection c=null;
     try
        {URL url =new URL(args[0]);
         ValiditeURLConnexion.test(url);
         c= url.openConnection();}
     catch (IOException e)
         {System.out.println("Wrong URL : "+args[0]);
          System.exit(0);}
     verifie(c);
    }
}
```

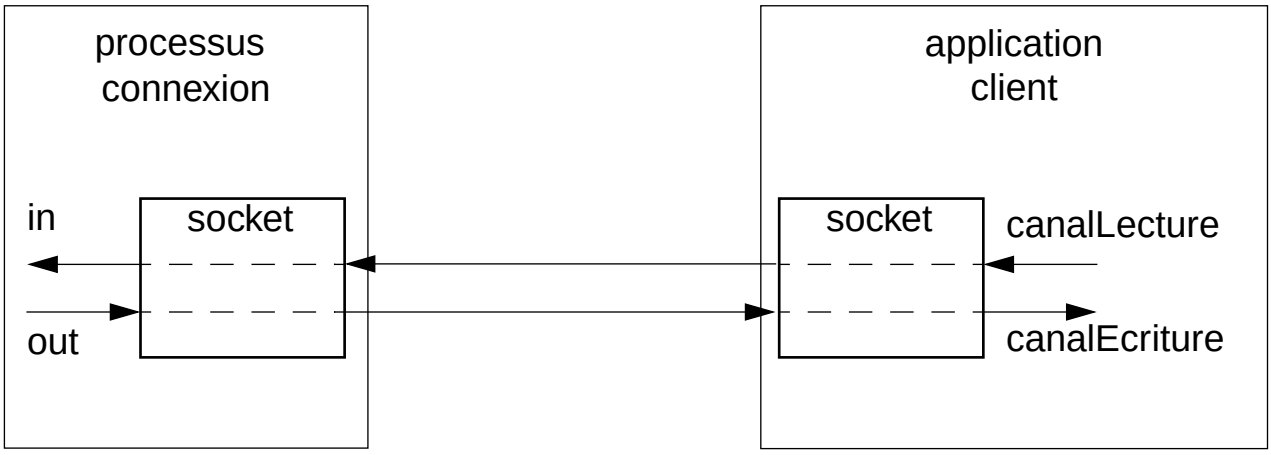
classe ValiditeURLConnexion

```
import java.io.*;
import java.net.*;
public class ValiditeURLConnexion {
   public static void test(URL url) throws IOException
     {URLConnection c=url.openConnection();
      DataInputStream dis =new
                          DataInputStream(c.getInputStream());
      dis.close();
     }
   public static void test(String s)
   throws MalformedURLException,IOException
     {test(new URL(s));
     }
}
```

un exemple client-serveur








```
>java Client cuisuna
Connexion : cuisuna/129.194.12.27 port : 45678
?toto
!TOTO
?lulu
!LULU
?FIN
Connexion terminee
```

Serveur

```
import java.io.*;
import java.net.*;

public class Serveur extends Thread {
    protected static final int PORT=45678;
    protected ServerSocket ecoute;

  public Serveur ()
   { try
       {ecoute=new ServerSocket(PORT);}
     catch (IOException e)
           {System.err.println(e.getMessage());
            System.exit(1);}
     System.out.println(
                     "Serveur en ecoute sur le port:"+PORT);
     this.start();
   }

  public void run()
  {try
     {while (true)
           {Socket client=ecoute.accept();
            Connexion c= new Connexion(client);}}
    catch (IOException e)
        {System.err.println(e.getMessage());
         System.exit(1);}
  }

  public static void main(String[] args)
  {new Serveur();}
}
```

Services du serveur

```
import java.io.*;
import java.net.*;

class Connexion extends Thread{
  protected Socket client;
  protected DataInputStream in;
  protected PrintStream out;
  public Connexion(Socket client_soc)
    {client=client_soc;
     try
        {in =new DataInputStream(client.getInputStream());
         out =new PrintStream(client.getOutputStream());}
     catch (IOException e)
           {try {client.close();} catch (IOException e1) {}
            System.err.println(e.getMessage());
            return;}
     this.start();
    }
  public void run()
  { String ligne;
    try
      {while(true)
       {ligne=in.readLine();
        if (ligne.toUpperCase().compareTo("FIN")==0)
              break;
        out.println(ligne.toUpperCase());}
      }
    catch (IOException e)
       {System.out.println("connexion:"+e.toString());}
    finally
       {try {client.close();} catch (IOException e){};}
  }
}
```

client du serveur

```
import java.io.*;
import java.net.*;
public class Client {
   protected static final int PORT=45678;
public static void main(String[] args)
{Socket s=null;
 if (args.length!=1)
   {System.err.println("Usage: java Client <hote>");
    System.exit(1);}
 try
  {s=new Socket(args[0],PORT);
   DataInputStream canalLecture =
              new DataInputStream(s.getInputStream());
   DataInputStream console=new DataInputStream(System.in);
   PrintStream canalEcriture=
                    new PrintStream(s.getOutputStream());
   System.out.println("Connexion etablie: "+
               s.getInetAddress()+" port: "+s.getPort());
    String ligne;
   while (true)
    {System.out.print("?");System.out.flush();
     ligne=console.readLine();
     canalEcriture.println(ligne);
     ligne=canalLecture.readLine();
     if (ligne==null)
       {System.out.println("Connexion terminee");break;}
    System.out.println("!"+ ligne); } // while
 } // try
 catch (IOException e) {System.err.println(e);}
 finally {try {if (s!=null) s.close();}
          catch (IOException e2){}}
 } // main
} // class
```

processus de la télé-discussion.

Un serveur en attente de connexions :

— attend une demande
— crée une connexion et l'insère dans la liste

Une connexion pour chaque client :

— attend un message de son client
— demande un accès à la liste des connexions afin de leur envoyer le message pour qu'elles le transmettent à leur client correspondant.
— en fin de connexion, avertit le processus nettoyeur chargé de la retirer de la liste des connexions.

Le processus nettoyeur :

— se réveille périodiquement ou est réveillé par une connexion.
— testee l'activité d'une connexion.

L'application client :

saisit des messages de l'utilisateur, les envoie à la connexion
affiche des messages provenant d'autres clients.

l'écoute et la réception sont donc asynchrones.

Le processus écouteur :

— affiche tous les messages qu'il reçoit de la connexion.

télé-discussion : le serveur

```
import java.io.*;
import java.net.*;
import java.util.*; // pour utiliser la classe Vector

public class Serveur2 extends Thread {
    protected static final int PORT=45678;
    protected ServerSocket ecoute;
    protected Vector connexions;
    protected Nettoyeur nettoyeur;

 public Serveur2 ()
 {try {ecoute=new ServerSocket(PORT);}
  catch (IOException e)
    {System.err.println(e.getMessage());System.exit(1);}
  System.out.println("Serveur en ecoute sur le port:"+PORT);
  connexions=new Vector();
  nettoyeur=new Nettoyeur(this);
  this.start();
 }
 public void run()
  {try
    {while (true)
      {// attente  d'une demande
       Socket client=ecoute.accept();
       System.out.println("Demande de connexion...");
       Connexion2 c= new Connexion2(client,nettoyeur,this);
        synchronized (connexions) {connexions.addElement(c);}
    }}
   catch (IOException e)
    {System.err.println(e.getMessage());System.exit(1);}
   } // run
  public static void main(String[] args)
  {new Serveur2(); }}
```

télé-discussion: la connexion

```
import java.util.*;

class Connexion2 extends Thread{
  protected Socket client;
  protected Serveur2 serveur;
  protected DataInputStream in;
  protected PrintStream out;
  protected Nettoyeur nettoyeur;

    public Connexion2(Socket client_soc, Nettoyeur n,
                      Serveur2 s)
    {client=client_soc;
     nettoyeur=n;
     serveur=s;
     try
       {in =new DataInputStream(client.getInputStream());
        out =new PrintStream(client.getOutputStream());}
     catch (IOException e)
      {try {client.close();} catch (IOException e1){};
            System.err.println(e.getMessage());
            return;}
     this.start();
    } // constructeur

....
```

télé-discussion: la connexion
(suite du listing)

```
public void run()
{String ligne;
 Connexion2 c;

try
{while(true)
  {ligne=in.readLine();
     synchronized(serveur.connexions)
     {for (int i=0;i<serveur.connexions.size();i++)
        {c=(connexion2) serveur.connexions.elementAt(i);
         c.out.println(ligne);}
     if (ligne.endsWith("FIN")) break;
  }
}
catch (IOException e){}
finally
      {try {client.close();} catch (IOException e){};
       System.out.println("Fin de connexion...");
       synchronized(nettoyeur) nettoyeur.notify();
      }
} // run
} // class
```

télé-discussion : le nettoyeur

```
import java.io.*;
import java.net.*;
import java.util.*;

class Nettoyeur extends Thread{
  protected Serveur2 serveur;

  protected Nettoyeur(Serveur2 serveur)
   {this.serveur=serveur;
    this.start();
   }

  public synchronized void run()
  {while(true)
    {try { // attention different de sleep
            this.wait(5000);}
     catch (InterruptedException e){}
     synchronized(serveur.connexions)
       {for (int i=serveur.connexions.size()-1;i>=0;i--)
         {Connexion2 c= (Connexion2)
                          serveur.connexions.elementAt(i);
          if (!c.isAlive())
            {serveur.connexions.removeElementAt(i);
             System.out.println("Fin de connexion: OK");}
         } // for
        } // synchro
      } // while
  }
}
```

télé-discussion : AppliClient

```
import java.io.*;
import java.net.*;
import java.awt.*;

public class AppliClient extends Frame{
    public static final int PORT=45678;
    Socket s;
    PrintStream canalEcriture;
    TextField entree;
    TextArea  visu;
    Button envoi,stop;
    Panel boutons;
    String Nom;


   public boolean action(Event e,Object obj)
   {if ((e.target==entree) || (e.target==envoi))
      {canalEcriture.println(Nom+">"+entree.getText());
       entree.setText(""); //efface le texte
       return true;
      }
    if (e.target==stop)
     {canalEcriture.println(Nom+">FIN");
      System.exit(0);}
    return false;
   } // action

 public static void main(String[] args)
   { Frame f= new AppliClient(args[0]); }
```

télé-discussion : AppliClient
(suite du listing)

```
public AppliClient(String n)
    {super("client"+ " "+n);
     try
        {Nom=n;
       // demande de connexion sur le serveur
       // le socket est renvoye lorsqu'elle est etablie
       s=new Socket("mycpu.bigco.ch",PORT);
        canalEcriture=new PrintStream(s.getOutputStream());

        // construction de l'interface graphique
        entree=new TextField();
        visu=new TextArea();
        visu.setEditable(false);
        this.setLayout(new BorderLayout());
        this.add("North",visu);
        this.add("Center",entree);
        boutons=new Panel();
        envoi=new Button("envoi");
        stop =new Button("stop");
       boutons.add(envoi);
       boutons.add(stop);
        this.add("South",boutons);
        this.pack();
        this.show();
       visu.setText("Connexion: "+ s.getInetAddress()+
              " port: "+s.getPort());
        Ecouteur ecoute=new Ecouteur(s,visu);
    }
     catch  (IOException e)
        {visu.setText(e.toString());}
   } //init
} //class
```

télé-discussion : l'écouteur

```
import java.io.*;
import java.net.*;
import java.awt.*;

class Ecouteur extends Thread{
 DataInputStream entree;
 TextArea visu;

 public Ecouteur(Socket s,TextArea out)
 { entree= new DataInputStream(s.getInputStream());
   visu=out;
   this.start();
  }

 public void run()
 { String ligne;
   try
      {while (true)
       {ligne=entree.readLine();
        if (ligne==null) break;
        visu.appendText("\n"+ligne);
       }}
    catch (IOException e) {visu.setText(e.toString());}
    finally
     {visu.setText("connexion interrompue par le serveur");}
 }
}
```