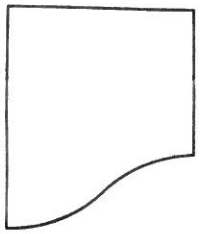
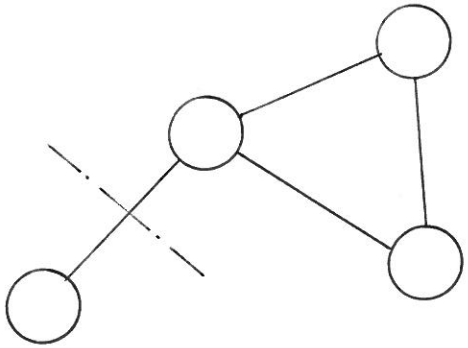
Une réalisation pour TSI ?

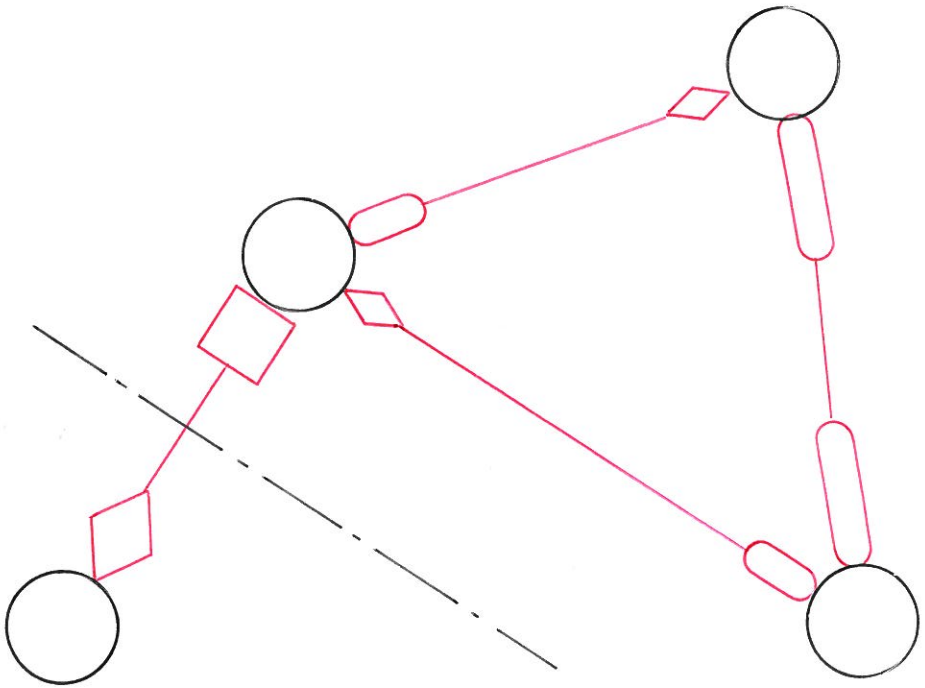
définition XML
du "réseau"

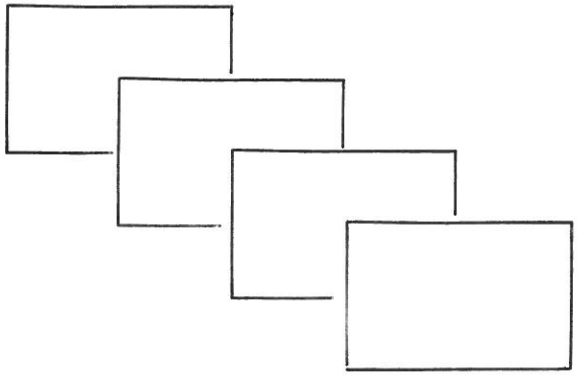


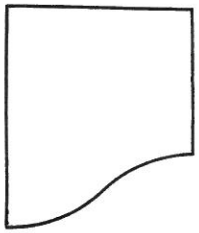


- mémoire partagée sémaphorisée
- file de messages
- socket udp
- socket tcp
- pipe
- pipe nommé
- driver
- telecom

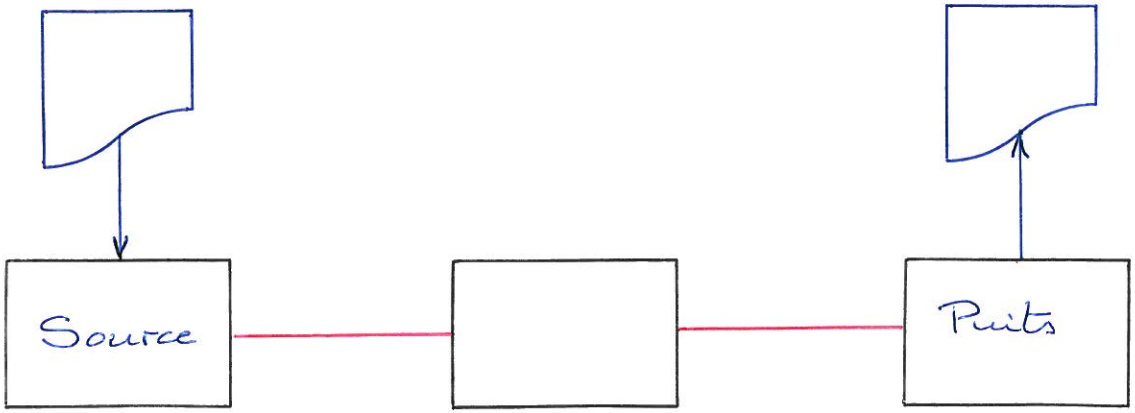
- Ordinateur
- Processeur
- Port / Interface
- Liaison
- Technologie







Chaque noeud connaît
la configuration totale

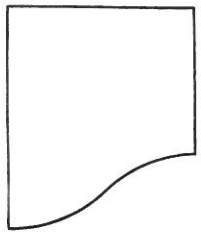


Les nœuds communiquent par les Plaisous

Certains nœuds sont des sources
(lisent de l'extérieur)

Certains nœuds sont des puits
(écrivent à l'extérieur)

Chaque message est émis avec la liste
des noeuds qu'il doit traverser



1 ligne de texte
1 ligne pour le compteur de noeud
1 ligne pour chaque destination, dans l'ordre

Un jeu d'essai c'est une succession de messages
qui entrent par une source

```
echo  "preparation du re-formateur d'entree"
flex  prep.l
gcc   lex.yy.c    -ll   -oPrep

echo  "preparation du logiciel 'chargeur de dom light'"
g++   -I.   -c    Vrac.cc
g++   -I.   -c    Node.cc
g++   -I.   -c    ItfExits.cc
g++   -I.   -c    TrueExits.cc
g++   -I.   -c    newTrueExits.cc
g++   -shared     Vrac.o Node.o ItfExits.o      \
          TrueExits.o newTrueExits.o    \
                                  -olibtest.so

g++   -I.   MyDOM.cc    -L. -ltest

echo  "test sur fichier adapte"
a.out < Node.in
a.out < Node.in2

echo  "test sur fichier re-formatte"
Prep  < test.in   | a.out

#echo "version un peu plus C++-STL"
#gcc  -c    Node.new.cc
```

```
%{
#include    <stdio.h>

int         node  = 0;
int         key   = 1;
%}
PREM   [a-zA-Z0-9]
DEUX   [a-zA-Z0-9\.]
MOT    {PREM}{DEUX}*


%start              inComment
%%
<inComment>"-->"   { BEGIN 0; }
<inComment>{MOT}   { }
<inComment>.               { }
<inComment>\n              { }

"<!--"             { BEGIN inComment; }

"\""          { }
"<"           { node = 1; }
"="           { key = 0; printf(" "); }
"/"{MOT}      { ECHO; }
"/"           { printf(" / "); }
">"           { }
{MOT}         {
              if    (node == 1) { node = 0; ECHO; }
              else if (key  == 1)     { printf(" attr "); ECHO; }
              else if (key  == 0)     { ECHO; key = 1; }
              }
%%
```

```
//---------------------------------------------------------------------
---
//---------------------------------------------------------------------
---
//
//          ------------------
//           Un Node du DOM ?
//          ------------------
//
//---------------------------------------------------------------------
---
//---------------------------------------------------------------------
---

#include    <stdio.h>
#include    <stdlib.h>

#include    <iostream>

#include    <utility>
#include    <string>
#include    <map>

using       namespace   std;

//---------------------------------------------------------------------
---
//---------------------------------------------------------------------
---
//---------------------------------------------------------------------
---
//---------------------------------------------------------------------
---

#include    <Node.h>

/**
 *    Le document (simili-DOM !) est une arborescence de 'Node'.
 *    Il n'existe pas de 'Node' particuliers ('#document', '#text').
 */

Vrac        Node::      getAttributes     ()
      {
      return      theAttributes;
      }

Vrac        Node::      getInnerNodes     ()
      {
      return      theInnerNodes;
      }

//---------------------------------------------------------------------
---
// constructeurs
//---------------------------------------------------------------------
---

// constructeur   sans parametre:

            Node::      Node        ()
      {
      strcpy      (dummy, "anonymous");
      nom   = string("anonymous");
      //cout << "construction du Node (1) '" << nom << "'\n";
      }
```

```
// constructeur   naturel:

            Node::      Node        (char * s)
      {
      strcpy      (dummy, s);
      nom   = string(s);
      //cout << "construction du Node (2) '" << nom << "'\n";
      }

//-------------------------------------------------------------------------
---
//    destructeur
//-------------------------------------------------------------------------
---

            Node::      ~Node       ()
      {
      cout << "destruction du Node '" << nom << "'\n";
      }

//-------------------------------------------------------------------------
---
//-------------------------------------------------------------------------
---
//-------------------------------------------------------------------------
---
//-------------------------------------------------------------------------
---
```

```
//-------------------------------------------------------------------
---
//-------------------------------------------------------------------
---
//
//          ---------------------------------------
//           PreEtude de mise en .cpp de 'Forward':
//          ---------------------------------------
//
//-------------------------------------------------------------------
---
//-------------------------------------------------------------------
---

#include    <iostream>

using       namespace   std;

//-------------------------------------------------------------------
---
//-------------------------------------------------------------------
---

#include    <TrueExits.h>

//-------------------------------------------------------------------
---
//-------------------------------------------------------------------
---

//-------------------------------------------------------------------
---
//-------------------------------------------------------------------
---

/**
 *    Chargement dynamique: on invente 'newInstance()' statique.
 */
TrueExits * newTrueExits      ()
      {
      cout << "passage dans newTrueExits \n";
      TrueExits * temp = new TrueExits();
      return      temp;
      }

//-------------------------------------------------------------------
---
//-------------------------------------------------------------------
---
//-------------------------------------------------------------------
---
//-------------------------------------------------------------------
---
```

```
//-----------------------------------------------------------------
---
//-----------------------------------------------------------------
---
//
//          -------------------------
//           Classe:    ItfExits
//          -------------------------
//
////-----------------------------------------------------------------
-----
//-----------------------------------------------------------------
---

#ifndef     _ItfExits
#define     _ItfExits

#include    <Node.h>

using       namespace   std;

/**
 *    Les traitements particuliers sont pr&eacute;par&eacute;s
 *    dans les fonctions pr&eacute;vues par le parcours,
 *    qu'on place dans un objet applicatif distinct.
 */
class       ItfExits
      {
// debut de classe
//-----------------------------------------------------------------
---

public      :

virtual
void        nodeStart   (Node * n)
      {}

virtual
void        nodeEnd           (Node * n)
      {}

virtual
void        attrListStart     ()
      {}

virtual
void        attrListEnd ()
      {}

virtual
void        attrTrt           (char * k, char * v)
      {}

virtual
void        innerStart  ()
      {}

virtual
void        innerEnd    ()
      {}

//-----------------------------------------------------------------
---
// fin de classe
      };
```

```
#endif

//-----------------------------------------------------------------------
---
//-----------------------------------------------------------------------
---
//-----------------------------------------------------------------------
---
//-----------------------------------------------------------------------
---
```

```
//-------------------------------------------------------------------
---
//-------------------------------------------------------------------
---
//
//           -------------------------
//            Classe:    ItfExits
//           -------------------------
//
//-------------------------------------------------------------------
---
//-------------------------------------------------------------------
---

#include    <iostream>
#include    <ItfExits.h>


//-------------------------------------------------------------------
---
//-------------------------------------------------------------------
---
//-------------------------------------------------------------------
---
//-------------------------------------------------------------------
---

//-------------------------------------------------------------------
---
//-------------------------------------------------------------------
---
//-------------------------------------------------------------------
---
//-------------------------------------------------------------------
---
```

```
//-----------------------------------------------------------------
---
//-----------------------------------------------------------------
---
//
//          --------------------------
//           Classe:    TrueExits
//          --------------------------
//
////-----------------------------------------------------------------
-----
//-----------------------------------------------------------------
---

#ifndef     _TrueExits
#define     _TrueExits

#include    <ItfExits.h>
#include    <Node.h>

using       namespace   std;

/**
 *    Les traitements particuliers sont pr&eacute;par&eacute;s
 *    dans les fonctions pr&eacute;vues par le parcours,
 *    qu'on place dans un objet applicatif distinct.
 */
class       TrueExits
      :     public            ItfExits
      {
// debut de classe
//-----------------------------------------------------------------
---

public      :

virtual
void        nodeStart   (Node * n)
      ;

virtual
void        nodeEnd           (Node * n)
      ;

virtual
void        attrListStart     ()
      ;

virtual
void        attrListEnd ()
      ;

virtual
void        attrTrt           (char * k, char * v)
      ;

virtual
void        innerStart  ()
      ;

virtual
void        innerEnd    ()
      ;

//-----------------------------------------------------------------
---
```

```
// fin de classe
      };

#endif

//-------------------------------------------------------------------------
---
//-------------------------------------------------------------------------
---
//-------------------------------------------------------------------------
---
//-------------------------------------------------------------------------
---
```

```
//-----------------------------------------------------------------
---
//-----------------------------------------------------------------
---
//
//          -------------------------
//           Classe:    TrueExits
//          -------------------------
//
//-----------------------------------------------------------------
---
//-----------------------------------------------------------------
---

#include    <iostream>
#include    <TrueExits.h>

using       namespace   std;

/**
 *    Les traitements particuliers sont pr&eacute;par&eacute;s
 *    dans les fonctions pr&eacute;vues par le parcours,
 *    qu'on place dans un objet applicatif distinct.
 */

void        TrueExits::
            nodeStart   (Node * n)
      {
      cout << "\n...nodeStart\n";
      }

void        TrueExits::
            nodeEnd           (Node * n)
      {
      cout << "\n...nodeEnd\n";
      }

void        TrueExits::
            attrListStart     ()
      {
      cout << "\n...attrListStart\n";
      }

void        TrueExits::
            attrListEnd ()
      {
      cout << "\n...attrListEnd\n";
      }

void        TrueExits::
            attrTrt           (char * k, char * v)
      {
      cout << "\n...attrTrt\n";
      cout << string(k) << " = " << string(v) << '\n';
      }

void        TrueExits::
            innerStart  ()
      {
      cout << "\n...innerStart\n";
      }

void        TrueExits::
            innerEnd    ()
      {
      cout << "\n...innerEnd\n";
```

```
    }
//------------------------------------------------------------------------
---
//------------------------------------------------------------------------
---
//------------------------------------------------------------------------
---
//------------------------------------------------------------------------
---

//------------------------------------------------------------------------
---
//------------------------------------------------------------------------
---
//------------------------------------------------------------------------
---
//------------------------------------------------------------------------
---
```

README:

J'ai hésité à le faire tout en C.
Finalement, c'est un C++ modéré.

Un tout petit programme de pre-traitement, généré
par lex, facilite le chargement d'un DOM, suivi de son parcours.

Restrictions sur le DOM:
- valeurs d'attributs sans blanc intermédiaire.
- aucune forme en "<?"
- pas de texte intersticiel entre ">" et "<"

Un tel DOM convient à une description GML.

Lors du parcours du DOM, on passe par les exits très semblables
à la version Java, et on peut les charger dynamiquement dans un
objet de la classe TrueExits.

J'ai fait cela pour vous en revenant de l'hôpital:
il n'y a pas de raison que je sois le seul embêté !

Bonne lecture !
B. M. G.
bg@eisti.fr

```
//-----------------------------------------------------------------------
---
//-----------------------------------------------------------------------
---
//
//          --------------------------------
//           Chargement XML + Parcours DOM:
//          --------------------------------
//
//-----------------------------------------------------------------------
---
//-----------------------------------------------------------------------
---

#include    <iostream>

#include    <Node.h>
#include    <ItfExits.h>
#include    <TrueExits.h>

using       namespace   std;

#define     tabu  {cout << "\n"; for(w = 0; w < level; w++) {cout <<
"\t";}}

ItfExits *  theTrueExits;

//-----------------------------------------------------------------------
---
//-----------------------------------------------------------------------
---

/**
 *    Fonction recursive de saisie d'un noeud.
 */
void        rec_enter   (int level, Node * current, char * ch)
      {
      char  s[20];
      char  aN[20];
      char  aV[20];

      int   w;

      tabu; cout << ch;

      Node *      theNode = new Node(ch);
      tabu; cout << "'" << theNode->nom
            << "' est un subordonne de '" << current->nom << "'";

      //
      // .../... mise en place du noeud comme subordonne
      //

      (current->theInnerNodes).setKeyPointer(ch, (char *)theNode);

      scanf ("%s", s);
      tabu; cout << "lu: " << string(s); //sleep(1);
      while (s[0] != '/')
            {
            // est-ce un noeud attribut ?
            /* */if     (!strcmp(s, "attr"))
                  {
                  scanf ("%s", aN);
                  scanf ("%s", aV);
                  tabu; cout << aN << "=" << aV ;
                  //
```

```
                    // .../... mise en place d'un attribut
                    //
                    (theNode->theAttributes).setKeyValue(&aN[0], &aV
[0]);
                    //
                    scanf ("%s", s);
                    tabu; cout << "lu: " << s;
                    continue;
                    }
              // est-ce un noeud subordonne ?
              else  {
                    //tabu;     cout << "entree en recursion\n";
                    rec_enter   (level+1, theNode, &s[0]);
                    //tabu;     cout << "sortie de recursion\n";
                    }
              //
              scanf ("%s", s); tabu; cout << string(s); // sleep(1);
              }
        /*
        tabu; cout << "subordonnes de '" << current->nom << "': ";
        (current->theInnerNodes).show();
        tabu; cout << "sortie de rec_enter par !" << s << "!";
        */
        }

//-------------------------------------------------------------------
---
//-------------------------------------------------------------------
---

/**
 *    Parcours recursif d'un noeud.
 */
void        parcourir   (int level, Node * n)
        {
        int   w;

        //cout << "exit: nodeStart\n";
        theTrueExits->nodeStart(n);

        tabu; cout << "(" << level << ")\t" << n->nom;
        tabu; cout << "attributs de '" << n->nom << "': ";

        //cout << "\nexit: attrListStart\n";
        theTrueExits->attrListStart();
        (n->theAttributes).show();
        for   (w = 0; w < SizeMax; w++)
              {
              if    (n->theAttributes.used[w])
                    {
                    //cout << "exit: attrTrt\nexit: ";
                    //cout << n->theAttributes.keys[w] << " = ";
                    //cout << n->theAttributes.values[w] << '\n';
                    theTrueExits->attrTrt
                          (n->theAttributes.keys[w],
                          n->theAttributes.values[w]);
                    }
              }
        //cout << "exit: attrListEnd\n";
        theTrueExits->attrListEnd();

        tabu; cout << "subordonnes de '" << n->nom << "':\n";
        //cout << "exit: innerStart\n";
        theTrueExits->innerStart();
        (n->theInnerNodes).show();
        for   (w = 0; w < SizeMax; w++)
```

```
            {
            if    (n->theInnerNodes.used[w])
            parcourir (level+1, (Node *)(n->theInnerNodes.values
[w]));
            }
      //cout << "exit: innerEnd\n";
      theTrueExits->innerEnd();

      tabu; cout << "fin Noeud '" << n->nom << "' level " << level
<< '\n';
      //cout << "exit: nodeEnd\n";
      theTrueExits->nodeEnd(n);
      //sleep(1);
      }

//-----------------------------------------------------------------------
---
//-----------------------------------------------------------------------
---

/**
 *    Indicateur de mise au point vs production.
 */
bool        debug       = false;

/**
 *    Nom du document.
 */
char        docname[40] = "test 5-2-2010";

/**
 *    Nom de la classe des exit's de traitement.
 */
char        classname[40]     = "TrueExits";

/**
 *    Procedure principale.
 */
int         main        (int argc, char * argv[])
      {
      cout << "\n";
      cout << "-------------------------------------------\n";
      cout << " Chargement d'un document X.M.L. + parcours: \n";
      cout << "-------------------------------------------\n";
      cout << "\nB.M.G. version Hiver 2010\n";
      cout << "\nligne de commande:";
      cout << "$ MyDOM [-debug][-docname=...]" << '\n';
      for   (int w = 0; w < argc; w++)
            {
            cout << '\t' << w << '\t' << argv[w] << '\n';
            if    (! strcmp(argv[w], "-debug"))
                  {
                  debug = true;
                  }
            if    (! strncmp(argv[w], "-docname=", 9))
                  {
                  strcpy(docname, &argv[w][9]);
                  }
            if    (! strncmp(argv[w], "-trueExits=", 11))
                  {
                  strcpy(classname, &argv[w][11]);
                  }
            }

      // .../... prealable:
      //
```

```
        testVrac(0, (char **)0);

        // .../... creation et chargement d'un document:
        //
        Node *      document = new Node(docname);
        Node& doc = *document;
        int   level = 1;
        //
        //
        cout << '\n';

        char  ch[20];
        scanf ("%s", ch);
        while (! feof(stdin))
              {
              rec_enter(level, document, ch);
              cout << "\n";
              scanf ("%s", ch);
              }

        //
        // .../... creation d'un objet de traitement (exit's):
        //
        theTrueExits = new TrueExits();     // objet par defaut (vide)

/*
          void *  entrypt;
          ItfExits * theTrueExits;
          ItfExits * (* p)();
          void *  handle  = dlopen("libPLUGIN.so", RTLD_LAZY);
          if      (handle == 0)   cout << "handle est nul\n";
          else                    cout << "handle trouve \n";

          cout << "\n\tnewDragon ";
          //entrypt       = dlsym (handle, "_Z9newDragonv");
          entrypt = dlsym (handle, scramble("Dragon"));
          if      (entrypt == 0)  cout << "entrypt est nul\n";
          else                    cout << "entrypt trouve \n";
          theTrueExits = samer(p, entrypt);

*/

        //
        // .../... parcours du document: avec appel dynamique\n";
        //
        parcourir(1, document);

        //
        //
        cout << "\n";
        return      0;
        }

//-----------------------------------------------------------------------
---
//-----------------------------------------------------------------------
---
//-----------------------------------------------------------------------
---
//-----------------------------------------------------------------------
---
```

```
//-----------------------------------------------------------------------
--
//-----------------------------------------------------------------------
--
/*
                    ---------------------
                     Utilisation de DOM:
                    ---------------------
*/
//-----------------------------------------------------------------------
--
//-----------------------------------------------------------------------
--

//    Java de base
//
import          java.io.*;
import            java.util.*;

//    Traitement XML
//
import          javax.xml.parsers.*;
import          org.w3c.dom.*;

/* 1.5
import            javax.xml.xpath.*;
*/

//-----------------------------------------------------------------------
--
//-----------------------------------------------------------------------
--

/**
 *    Definition des 'points de sortie' ('exits') de l'application.
 */
interface   DOMExits
      {
// debut de classe
//-----------------------------------------------------------------------
--

public
void        nodeStart          (Node n);

public
void        nodeEnd                  (Node n);

public
void        attrListStart            ();

public
void        attrListEnd        ();

public
void        attrTrt                  (Node n);

public
void        innerListStart           ();

public
void        innerListEnd             ();

//-----------------------------------------------------------------------
--
// fin de classe
```

```
      }

//---------------------------------------------------------------------
--
//---------------------------------------------------------------------
--
//---------------------------------------------------------------------
--
//---------------------------------------------------------------------
--

/**
 *    'Adaptateur' des exits de l'application.<br>
 *    (Traitements par d&eacute;faut)
 */
abstract
class       DOMExitsAdapter                implements  DOMExits
      {
// debut de classe
//---------------------------------------------------------------------
--

public
void        nodeStart          (Node n)
      {}
public
void        nodeEnd                  (Node n)
      {}
public
void        attrListStart            ()
      {}
public
void        attrListEnd        ()
      {}
public
void        attrTrt                  (Node n)
      {}
public
void        innerListStart           ()
      {}
public
void        innerListEnd             ()
      {}

/**
 * Fonction utilitaire d'impression:
 */
void        a                  (String p)
      {
      System.out.println(p);
      }

/**
 * Fonction utilitaire d'impression:
 */
void        b                  (String p)
      {
      if    (! DOM.debug)      return;
      System.err.println(p);
      }

/**
 * Fonction utilitaire d'impression:
 */
void        c                  (String p)
      {
```

```
        if      (! DOM.debug)       return;
        System.err.print(p);
        }

/**
 * Fonction utilitaire d'impression:
 */
void            ta                  (int wi)
        {
        for     (int i = 0; i < wi; i++)
                {
                System.out.print("    ");
                }
        }

/**
 * Fonction utilitaire d'impression:
 */
void            tb                  (int wi)
        {
        if      (! DOM.debug)       return;
        for     (int i = 0; i < wi; i++)
                {
                System.out.print("    ");
                }
        }

//-------------------------------------------------------------------------
--
// fin de classe
        }

//-------------------------------------------------------------------------
--
//-------------------------------------------------------------------------
--
//-------------------------------------------------------------------------
--
//-------------------------------------------------------------------------
--

/**
 *    Traitement effectif des exits de l'application.
 */
class       DefaultTrueExits  extends           DOMExitsAdapter
        {
// debut de classe
//-------------------------------------------------------------------------
--

//    ---/---

//-------------------------------------------------------------------------
--
// fin de classe
        }

//-------------------------------------------------------------------------
--
//-------------------------------------------------------------------------
--
//-------------------------------------------------------------------------
--
//-------------------------------------------------------------------------
--
```

```
/**
 *    Lecture d'un document X.M.L. en DOM.
 */
public
class           DOM
        {
// debut de classe
//---------------------------------------------------------------------
--

public
static
DOM             theAppli;

public
static
boolean           debug       = false;

public
static
void            main            (String[] args)
                        throws            Exception
        {
      a(" ");
      a("-----------------------------------");
      a(" Lecture d'un document XML en DOM: ");
      a("-----------------------------------");
      a("\nB.M.G. version Automne 2008");
      a("\nLigne de commande:");
      a("\tjava\tDOM\t[-debug]\t[-exits=...]");
      for   (int w = 0; w < args.length; w++)
            {
            a("\t"+w+"\t"+args[w]);
            if    (args[w].startsWith("-debug"))
                  {
                  debug = true;
                  }
            else
            if    (args[w].startsWith("-exits="))
                  {
                  Class c = Class.forName(args[w].substring(7));
                  theTrueExits = (DOMExits)c.newInstance();
                  }
            }
      a("\n"+(new Date()).toString()+" demarre...");
      theAppli = new DOM();
      theAppli.myGo(args);
      a("\n"+(new Date()).toString()+" ...termine !");
        }

/**
 *    Constructeur sans param&egrave;tre.
 */
public
            DOM             ()
                                throws          Exception
        {
        super();
        }

/**
 *    Fonction utilitaire d'impression:
 */
static
void        a                   (String p)
      {
```

```
        System.out.println(p);
        }

/**
 *    Fonction utilitaire d'impression:
 */
static
void        b                   (String p)
        {
        if    (! debug)    return;
        System.err.println(p);
        }

/**
 *    Fonction utilitaire d'impression:
 */
static
void        ta                  (int wi)
        {
        for   (int i = 0; i < wi; i++)
              {
              System.out.print("   ");
              }
        }

/**
 *    Fonction utilitaire d'impression:
 */
static
void        tb                  (int wi)
        {
        if    (! debug)    return;
        for   (int i = 0; i < wi; i++)
              {
              System.err.print("   ");
              }
        }

/**
 *    Usine &agrave; lecteur de documents.
 */
DocumentBuilderFactory  theFactory;

/**
 *    Lecteur de documents.
 */
DocumentBuilder         theParser;

/**
 *    Le document sous format d'arbre DOM.
 */
Document                theDocument;

/**
 *    Les traitements appel&eacute;s aux points d'exit.
 */
static
DOMExits          theTrueExits      = new DefaultTrueExits();

/**
 *    Fonction principale de l'application.
 */
public
void            myGo            (String[] args)
                                throws          Exception
       {
```

```
        b("start of generation\n");

        theFactory      = DocumentBuilderFactory.newInstance();
        theFactory.setIgnoringComments(true);
      theFactory.setIgnoringElementContentWhitespace(true);

      b("Caracteristiques du parser fabrique:");
        b("isNamespaceAware: "+theFactory.isNamespaceAware());
        b("isValidating: "+theFactory.isValidating());
        b("isCoalescing: "+theFactory.isCoalescing());
        b("isExpandEntityReferences:
"+theFactory.isExpandEntityReferences());
        b("isIgnoringComments: "+theFactory.isIgnoringComments());
        b("isIgnoringElementContentWhitespace:
"+theFactory.isIgnoringElementContentWhitespace());

        theParser       = theFactory.newDocumentBuilder();
        theDocument     = theParser.parse(System.in);

      b("\nAffichage sequentiel:");
      b("---------------------\n");

      affiche(0, theDocument);
/*
      b("\nAffichage en acces direct:");
      b("-------------------------\n");

      NodeList inside = theDocument.getElementsByTagName("corps");
        for     (int wi = 0; wi < inside.getLength(); wi++)
                {
                affiche(0, inside.item(wi));
                }
*/
/* 1.5 XPath:

      XPath theXPath = XPathFactory.newInstance().newXPath();
      String      theXPathExpr = "nodename";
      Node  theNodeThruXPath = (Node)theXPath.evaluate
                  (theXPathExpr, theDocument, XPathConstants.NODE);

1.5 XPath */

        b("end of generation\n");
        }

/**
 *    Texte sans contenu significatif ?
 */
boolean           jetable           (String s)
      {
      if    (s == null) return      true;
      if    (s.equals(""))    return      true;
      for   (int i = 0; i < s.length(); i++)
            {
            String      l = s.substring(i, i+1);
            if    (" \t\n".indexOf(l) == -1)    return false;
            }
      return      true;
      }

/**
 *    Affichage de l'arborescence du document.
 */
public
void            affiche         (int level, Node noeud)
       {
```

```
        // on elimine les textes intercalaires
        //
        if    (noeud.getNodeName().equals("#text"))     return;

        boolean     bbb;
        //
        //    appel d'exit
        //
        theTrueExits.nodeStart(noeud);

        tb(level); b("name:  "+noeud.getNodeName());
          tb(level); b("type:  "+noeud.getNodeType());
        bbb = jetable(noeud.getNodeValue());
        if    (! bbb)
              {
              tb(level); b("value: <"+noeud.getNodeValue()+">");
              }

        // attributs (de la balise d'entree)

        NamedNodeMap      nnm   = noeud.getAttributes();
        if    (nnm != null)
              {
              //
              //    appel d'exit
              //
              theTrueExits.attrListStart();

              tb(level); b("nbAttr:"+nnm.getLength());
              Node  theAttr;
              for(int wj=0; wj < nnm.getLength(); wj++)
                    {
                    boolean     bb2;
                    theAttr = nnm.item(wj);

                    // on elimine les textes intercalaires
                    if    (theAttr.getNodeName().equals("#text"))
                          continue;

                    bb2 = jetable(theAttr.getNodeValue());
                    tb(level); b("type:  "+theAttr.getNodeType()+"\t"
                    +"name:  "+theAttr.getNodeName()+"\t");
                    tb(level+1);
                    if (bb2) b("jetable");
                    else  {
                          b("value: "+theAttr.getNodeValue());
                          //
                          //    appel d'exit
                          //
                          theTrueExits.attrTrt(theAttr);
                          }
                    }
              //
              //    appel d'exit
              //
              theTrueExits.attrListEnd();
              }

      // noeuds subordonn&eacute;s: appel r&eacute;cursif (en
profondeur)

      //
      //    appel d'exit
      //
      theTrueExits.innerListStart();
```

```
      NodeList        inside  = noeud.getChildNodes();
        for     (int wi = 0; wi < inside.getLength(); wi++)
                {
                affiche(level+1, inside.item(wi));
                }
      //
      //    appel d'exit
      //
      theTrueExits.innerListEnd();

      //
      //    appel d'exit
      //
      theTrueExits.nodeEnd(noeud);
        }

//-------------------------------------------------------------------------
--
// fin de classe
      }

//-------------------------------------------------------------------------
--
//-------------------------------------------------------------------------
--
//-------------------------------------------------------------------------
--
//-------------------------------------------------------------------------
--
```