Classe Document

Document.h

```
/**************************************************************************
Document  -  fichier d'en-tête de la classe <Document>
-------------------
début                : 23/11/2015 15:43:24
copyright            : (C) 2015 par Quentin SCHROTER, Nicolas GRIPONT
e-mail               : quentin.schroter@insa-lyon.fr , nicolas.gripont@insa-lyon.fr
**************************************************************************/

//---------- Interface de la classe <Document> (fichier Document.h) ------
#if ! defined ( Document_H )
#define Document_H

//-------------------------------------------------- Interfaces utilisées
#include <map>

using namespace std;

#include "NombreDeHits.h"
#include "LogLine.h"
//------------------------------------------------------------ Constantes

//------------------------------------------------------------------ Types
class Document;
template struct std::pair<Document*, NombreDeHits>;
template class std::map<string,pair<Document*, NombreDeHits>>;

//--------------------------------------------------------------------------
// Rôle de la classe <Document>
// Classe permettant de définir un document d'un serveur.
//
//--------------------------------------------------------------------------

//------------------------------------------------------------ Fonctions

class Document
{
        //------------------------------------------------------------- PUBLIC

public:

    //------------------------------------------------------------------ Types

        //-------------------------------------------------- Méthodes publiques
        void MAJHits (int status, int heure);
        // Mode d'emploi :
        // Met à jour le nombre de hits du document en fonction du code de retour de la requête
        //
        // Contrat : aucun
        //
```

```
static bool CompareParNombreDeHitsReussis(Document* a, Document* b);
// Mode d'emploi :
// Methode permettant de comparer deux documents en fontion du nombre de
// hits. Return true si le nombre De Hits Reussis de a est superieur au
// nombre De Hits Reussis de b, false sinon.
//
// Contrat : aucun
//

void MAJDocAtteignable(int status, int heure, Document* unDoc);
// Mode d'emploi :
	// Met à jour les documents atteignables depuis ce document.
	//
// Contrat : Aucun.
	//

int NombreDeHitsAPartirDeCeDocument ( bool uniquementReussis = true );
// Mode d'emploi :
// Méthode permettant de recuperer le nombre de hits reussi (uniquementReussis = true)
// ou le nombre totals de hits provenant de ce document.
//
// Contrat : Aucun.
//

const map<string,pair<Document*, NombreDeHits>> & DocumentsAtteignables() const;
// Mode d'emploi :
// Retourne l'attribut documentsAtteignable
//
// Contrat : Aucun.

const NombreDeHits & NbHits() const;
// Mode d'emploi :
// Accesseur en lecture sur le nombre de Hits;
//
// Contrat : Aucun.

const string & CheminAccesRessource() const;
	// Mode d'emploi :
	// Retourne l'attribut cheminAccesRessource
	//
	// Contrat : Aucun.

	//------------------------------------------------- Surcharge d'opérateurs
Document & operator = (const Document & unDocument);
	// Mode d'emploi :
	// Surchage de l'opérateur =, copie profonde
	//
	// Contrat : Aucun
	//

bool operator < (const Document & unDocument) const;
// compare sur les attributs cheminAccesRessource (non utilisé)

	bool operator == (const Document & unDocument) const;
	// Mode d'emploi :
	// Compare les attributs cheminAccesRessource et nomDomaine des documents
```

```
        //
        // Contrat : Aucun.

        bool operator > (const Document & unDocument) const;
        // compare sur les attributs cheminAccesRessource (non utilisé)


        //------------------------------------------- Constructeurs - destructeur
        Document(const Document & unDocument);
        // Mode d'emploi (constructeur de copie) :
        // Constructeur par copie, copie profonde
        //
        // Contrat :
        //

    Document(string nD, string cAR);
    // Mode d'emploi :
    // Créé un document et initialise les attributs nomDomaine et cheminAccesRessource
    //
    // Contrat : aucun.
    //

        virtual ~Document();
    // Mode d'emploi : Destructeur.
        //
    // Contrat : aucun.
        //

        //------------------------------------------------------------------ PRIVE

protected:
        //---------------------------------------------------- Méthodes protégées

        //------------------------------------------------------ Attributs protégés
    string nomDomaine;                          // nom de domaine du
                                                // serveur ou le fichier
                                                // se trouve.

    string cheminAccesRessource;                // chemin d'acces
                                                // a la ressource
                                                // de puis le serveur

    NombreDeHits nbHits;                        // nombre de hits

    map<string,pair<Document*, NombreDeHits>> documentsAtteignables;
                                                // Map contenant les paires
                                                // documents accessible depuis
                                                // celui-ci et le nombre de
                                                // hits à partir de ce document
                                                // (valeur) et le chemin d?accès
                                                // au document atteignable
                                                // (clé) de type string.

};

//--------------------------- Autres définitions dépendantes de <Document>
```

```
#endif // Document_H
```

Document.cpp

```
/*************************************************************************
Document  -  fichier de réalisation de la classe <Document>
-------------------
début                : 23/11/2015 15:43:10
copyright            : (C) 2015 par Quentin SCHROTER, Nicolas GRIPONT
e-mail               : quentin.schroter@insa-lyon.fr , nicolas.gripont@insa-lyon.fr
*************************************************************************/

//---------- Réalisation de la classe <Document> (fichier Document.cpp) --

//------------------------------------------------------------------ INCLUDE

//-------------------------------------------------------- Include système

#include <iostream>
using namespace std;

//------------------------------------------------------ Include personnel
#include "Document.h"

//------------------------------------------------------------------ Constantes

//------------------------------------------------------------------- PUBLIC

//----------------------------------------------------- Méthodes publiques
void Document::MAJHits ( int status, int heure )
//Algorithme :
//
{
    nbHits.MAJHits(status/100 == 2,heure);
} //----- Fin de MAJHits

bool Document::CompareParNombreDeHitsReussis(Document* a, Document* b)
//Algorithme :
//
{
    return a->nbHits.NombreDeHitsTotal() > b->nbHits.NombreDeHitsTotal();
} //----- Fin de CompareParNombreDeHitsReussis


void Document::MAJDocAtteignable(int status, int heure, Document* unDoc)
//Algorithme :
//
{
```

```
    NombreDeHits unNombreDeHits;

    map<string,pair<Document*, NombreDeHits>>::iterator it = documentsAtteignables.begin();

    it = documentsAtteignables.find(unDoc->CheminAccesRessource());

    if ( it != documentsAtteignables.end())
    {
        it->second.second.MAJHits(status/100 == 2,heure);
    }
    else
    {
        unNombreDeHits.MAJHits(status/100 == 2,heure);
        documentsAtteignables.insert(make_pair(unDoc-
>CheminAccesRessource(),make_pair(unDoc, unNombreDeHits)));
    }
} //----- Fin de MAJDocAtteignable

int Document::NombreDeHitsAPartirDeCeDocument ( bool uniquementReussis )
//Algorithme :
//
{
    int somme = 0;
    for ( map<string,pair<Document*, NombreDeHits>>::const_iterator it =
documentsAtteignables.begin(); it != documentsAtteignables.end(); it++ )
    {
        somme += it->second.second.NombreDeHitsTotal(uniquementReussis);
    }
    return somme;
} //----- Fin de NombreDeHitsAPartirDeCeDocument

const string & Document::CheminAccesRessource() const
//Algorithme :
//
{
    return cheminAccesRessource;
} //----- Fin de CheminAccesRessource

const map<string,pair<Document*, NombreDeHits>> & Document::DocumentsAtteignables() const
//Algorithme :
//
{
    return documentsAtteignables;
} //----- Fin de DocumentsAtteignables

const NombreDeHits & Document::NbHits() const
//Algorithme :
//
{
    return nbHits;
} //----- Fin de NbHits

 //------------------------------------------------- Surcharge d'opérateurs
//Document & Document::operator = (const Document & unDocument)
//// Algorithme :
////
```

```
//{
//          nomDomaine = unDocument.nomDomaine;
//          cheminAccesRessource = unDocument.cheminAccesRessource;
//    documentsAtteignables = unDocument.documentsAtteignables;
//          nbHits = NombreDeHits(unDocument.nbHits);
//          return *this;
//} //----- Fin de operator =


bool Document::operator<(const Document & unDocument) const
//Algorithme :
//
{
          return cheminAccesRessource < unDocument.cheminAccesRessource;
} //----- Fin de operator<

bool Document::operator==(const Document & unDocument) const
//Algorithme :
//
{
   return cheminAccesRessource == unDocument.cheminAccesRessource;// && nomDomaine ==
unDocument.nomDomaine;
} //----- Fin de operator==

bool Document::operator>(const Document & unDocument) const
//Algorithme :
//
{
          return cheminAccesRessource > unDocument.cheminAccesRessource;
} //----- Fin de operator>

//------------------------------------------- Constructeurs - destructeur
Document::Document(const Document & unDocument) :
nomDomaine(unDocument.nomDomaine),
   cheminAccesRessource(unDocument.cheminAccesRessource), nbHits(unDocument.nbHits),
   documentsAtteignables(unDocument.documentsAtteignables)

// Algorithme :
//
{
#ifdef MAP
          cout << "Appel au constructeur de copie de <Document>" << endl;
#endif
} //----- Fin de Document (constructeur de copie)


Document::Document(string nD, string cAR) : nomDomaine(nD),
 cheminAccesRessource(cAR), nbHits(), documentsAtteignables()
// Algorithme :
//
{
#ifdef MAP
 cout << "Appel au constructeur de <Document>" << endl;
#endif

} //----- Fin de Document
```

```
Document::~Document()
// Algorithme :
//
{
#ifdef MAP
        cout << "Appel au destructeur de <Document>" << endl;
#endif
} //----- Fin de ~Document


  //------------------------------------------------------------------ PRIVE

  //------------------------------------------------------- Méthodes protégées
```