Classe GraphDocuments

GraphDocuments.h

```
/**************************************************************************
GraphDocuments  -  fichier d'en-tête de la classe <GraphDocuments>
-------------------
début                : 27/11/2015 22:27:39
copyright            : (C) 2015 par Quentin SCHROTER, Nicolas GRIPONT
e-mail               : quentin.schroter@insa-lyon.fr , nicolas.gripont@insa-lyon.fr
**************************************************************************/

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

//--------------------------------------------------- Interfaces utilisées

#include <vector>

using namespace std;


#include "Document.h"

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

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

template class std::vector<Document*>;
template class std::vector<string>;

//------------------------------------------------------------------------
// Rôle de la classe <GraphDocuments>
// Classe permettant de definir un graph de documents pourun serveur
// (nom de domaine)
//
//------------------------------------------------------------------------

class GraphDocuments
{
        //------------------------------------------------------------------ PUBLIC

public:
        //----------------------------------------------------- Méthodes publiques
    void TraiterLogLine ( LogLine l );
    // Mode d'emploi :
    // Methode permettant de traiter une ligne de log.
    // Ajoute et/ou met a jour les documents en fonction de la ligne de log.
        //
    // Contrat : aucun.
    //

    void TrierParNombreDeHitsReussis ();
```

```
    // Mode d'emploi :
    // Methode permettant de trier le vecteur en fonction du nombre de hits totals
    // reussis de maniere decroissante.
    //
    // Contrat : aucun.
    //

    const vector<Document*> & Documents() const;
    // Mode d'emploi :
    // Accesseur en lecture sur l?attribut document.
    //
    // Contrat : aucun.
    //

    const string & NomDomaine() const;
    // Mode d'emploi :
    // Accesseur en lecture seule sur le nom de domaine.
    //
    // Contrat : aucun.
    //

        //------------------------------------------------- Surcharge d'opérateurs
    GraphDocuments & operator = (const GraphDocuments & unGraphDocuments);
    // Mode d'emploi :
    // Redefinition de l'operateur =
    //
    // Contrat : aucun.
    //

        //-------------------------------------------- Constructeurs - destructeur
        GraphDocuments(const GraphDocuments & unGraphDocuments);
    // Mode d'emploi : constructeur de copie
        //
    // Contrat : aucun.
        //

    GraphDocuments( string nD );
    // Mode d'emploi : Constructeur.
        //
    // Contrat : aucun.
        //

        virtual ~GraphDocuments();
    // Mode d'emploi : destructeur.
        //
    // Contrat : aucun.
        //

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

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

    Document * DocumentPresent(string nomDomaine, string cheminAccesFichier );
    // Mode d'emploi :
    // Methode permettant de definir si un document ayant pour nom de domaine
```

```
    // nomDomaine et commen chemin d'acces cheminAccesFichier est deja present
    // dans le vector documents.
    //
    // Contrat : aucun.
    //



        //------------------------------------------------------ Attributs protégés

    vector<Document*> documents;  // vecter contenant les documents du graph

    string nomDomaine;          // nom de domaine du serveur correspondant
                                // au graph.
};

//--------------------------- Autres définitions dépendantes de <GraphDocuments>

#endif // GraphDocuments_H
```

GraphDocuments.cpp

```
/*************************************************************************
GraphDocuments  -  fichier de réalisation de la classe <GraphDocuments>
-------------------
début                : 27/11/2015 22:27:56
copyright            : (C) 2015 par Quentin SCHROTER, Nicolas GRIPONT
e-mail               : quentin.schroter@insa-lyon.fr , nicolas.gripont@insa-lyon.fr
*************************************************************************/

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

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

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

#include <iostream>
using namespace std;

//------------------------------------------------------ Include personnel
#include "GraphDocuments.h"
#include "LogParser.h"
#include <algorithm>
//------------------------------------------------------------- Constantes

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

//----------------------------------------------------- Méthodes publiques




void GraphDocuments::TraiterLogLine ( LogLine l )
// Algorithme :
```

```
//
{
    Document * documentSource = DocumentPresent(l.domainName,l.sourceFile);
    Document * documentDemande = DocumentPresent(l.domainName,l.requestedURL);
    if ( documentSource == nullptr )
    {
        documentSource = new Document(l.domainName,l.sourceFile);
        documents.push_back(documentSource);
    }
    if ( documentDemande == nullptr )
    {
        documentDemande = new Document(l.domainName,l.requestedURL);
        documents.push_back(documentDemande);
    }
    documentDemande->MAJHits(l.status,l.date.heure);

    documentSource->MAJDocAtteignable(l.status,l.date.heure,documentDemande);
} //----- Fin de TraiterLogLine


void GraphDocuments::TrierParNombreDeHitsReussis()
// Algorithme :
//
{
    sort(documents.begin(), documents.end(), Document::CompareParNombreDeHitsReussis);
} //----- Fin de TrierParNombreDeHitsReussis

const vector<Document*> & GraphDocuments::Documents() const
// Algorithme :
//
{
    return documents;
} //----- Fin de Documents


const string & GraphDocuments::NomDomaine() const
// Algorithme :
//
{
    return nomDomaine;
} //----- Fin de NomDomaine
//------------------------------------------------- Surcharge d'opérateurs

//GraphDocuments & GraphDocuments::operator = (const GraphDocuments &
unGraphDocuments)
//// Algorithme :
////
//{
//} //----- Fin de operator =


  //-------------------------------------------- Constructeurs - destructeur
GraphDocuments::GraphDocuments(const GraphDocuments & unGraphDocuments) :
    documents(),
    nomDomaine(unGraphDocuments.nomDomaine)
// Algorithme :
```

```
//
{
#ifdef MAP
    cout << "Appel au constructeur de copie de <GraphDocuments>" << endl;
#endif
    for ( vector<Document*>::const_iterator it = unGraphDocuments.documents.begin(); it !=
unGraphDocuments.documents.end(); it++ )
    {
        Document *tmp = new Document(*(*it));
        documents.push_back(tmp);
    }
} //----- Fin de GraphDocuments (constructeur de copie)


GraphDocuments::GraphDocuments(string nD) :
    documents(),
    nomDomaine (nD)
// Algorithme :
//
{
#ifdef MAP
    cout << "Appel au constructeur de <GraphDocuments>" << endl;
#endif
} //----- Fin de GraphDocuments


GraphDocuments::~GraphDocuments()
// Algorithme :
//
{
#ifdef MAP
            cout << "Appel au destructeur de <GraphDocuments>" << endl;
#endif
    for ( vector<Document*>::iterator it = documents.begin(); it != documents.end(); it++ )
    {
        delete *it;
    }
} //----- Fin de ~GraphDocuments


  //------------------------------------------------------------------ PRIVE
  //---------------------------------------------------- Méthodes protégées

Document* GraphDocuments::DocumentPresent(string nomDomaine, string cheminAccesFichier )
{
    Document *document = nullptr;
    Document tmp(nomDomaine,cheminAccesFichier);
    for ( vector<Document*>::iterator it = documents.begin(); it != documents.end(); it++ )
    {
        if ( *(*it) == tmp )
        {
            document = *it;
        }
    }
    return document;
}
```