```
/*************************************************************************
                    Catalogue.h  -  Interface de Catalogue.cpp
                                   La seule classe auquel le Main a accès
                             -------------------
    début                : 15/11/2017
    auteurs              : B3405 - Etienne et Grégoire
*************************************************************************/

//-------- Interface du module <Catalogue> (fichier Catalogue.h) ---------
#if ! defined ( Catalogue_H )
#define Catalogue_H

//------------------------------------------------------------------------
// Rôle du module <Catalogue>
// Gere le catalogue :
//      Ajouter des trajets
//      Afficher le catalogue
//      Rechercher les parcours (simple ou avancée)
//------------------------------------------------------------------------

//////////////////////////////////////////////////////////////////  INCLUDE
//--------------------------------------------------- Interfaces utilisées
#include "Trajet.h"
#include "TabTrajet.h"


class Catalogue {
//------------------------------------------------------ Fonctions publiques

public:

    void Afficher() const;
    // Contrat :
    //      Affiche le catalogue des Trajets

    void AjouterTrajet(const Trajet * T);
    // Contrat :
    //      Ajoute un trajet (simple ou composé) au catalogue
    //      en utilisant la fonction Add de la classe TabTrajet
    // Mode d'emploi
    //      T est un Trajet alloué dans le tas,
    //      non détruit avant la fin de vie du Catalogue

    void RechercherSimple(const char * dep, const char *arr) const;
    // Contrat :
    //      Affiche les trajets (simples ou composés) dont le départ
    //      et l'arrivée correspondent aux paramètres

    void RechercherAvancee(const char *dep, const char *arr) const;
    // Contrat :
    //      Affiche les parcours réalisables à partir des trajets du
    //      catalogue, dont le départ et l'arrivée correspondent
    //      aux paramètres
```

```
54
55  //------------------------------------------ Constructor and Desctructor
56
57      Catalogue();
58      virtual ~Catalogue();
59      // Mode d'emploi :
60      //      Fait appel au destructeur de TabTrajet
61
62
63  protected:
64      TabTrajet * catalog;
65
66      int max(int tab[]) const;
67      // Contrat :
68      //      Renvoie l'entier maximal parmis les éléments
69      //      du tableau passé en paramètre
70
71      void AfficherParcours(int utilise[]) const;
72      // Contrat :
73      //      Affiche le parcours composé des trajets décrits
74      //      par le tableau passé en paramètre
75      // Mode d'emploi:
76      //      utilise est un int[] représentant le parcours à afficher,
77      //      les valeurs correspondant à l'ordre d'affichage
78
79      void recure(int utilise[], int numeroTrajet, int trajetPrecedent , const
           char * arriveeFinale) const;
80      // Contrat :
81      //      Fonction récursive qui permet de tester tous les parcours
82      //      réalisables à partir des trajets du catalogue, partant du
83      //      départ demandé et qui appelle l'affichage lorsque le
84      //      parcours arrive à destination
85      // Mode d'emploi :
86      //      utilise : Tableau représentant le parcours courant
87      //      numeroTrajet : Numéro du trajet dans le parcours courant
88      //      trajetPrecedent : Indice du trajet précédent dans le Catalogue
89      //      arriveeFinale : Arrivee demandée par le client
90  };
91
92  #endif // Catalogue_H
93
94  /*************************************************************************
95                              Catalogue.cpp
96                               -------------------
97      début                : 15/11/2017
98      auteurs              : B3405 - Etienne et Grégoir
99  *************************************************************************/
100
101 //---------- Réalisation de la classe <Catalogue> (fichier Catalogue.cpp)
102
103 //------------------------------------------------------------- INCLUDE
104 //-------------------------------------------------- Include système
105 using namespace std;
```

```
106 #include <iostream>
107 #include <cstring>
108
109 //------------------------------------------------------ Include personnel
110 #include "Catalogue.h"
111
112 //-------------------------------------------------------------------- PUBLIC
113 //----------------------------------------------------- Méthodes publiques
114 void Catalogue::Afficher() const
115 {
116     unsigned int size = catalog->GetUtilise();
117     if (size > 0)
118     {
119         cout << endl << "Le catalogue contient " << size << " trajets :" << endl;
120         for (unsigned int i = 0; i < size; i++)
121         {
122             cout << i + 1 << " - ";
123             if (catalog->Get(i) != nullptr)
124             {
125                 catalog->Get(i)->Afficher();
126             }
127             else
128             {
129                 cout << "null pointer..." << endl;
130             }
131         }
132         cout << endl;
133     }
134     else
135     {
136         cout << "Le catalogue est vide..." << endl;
137     }
138 }
139
140 void Catalogue::AjouterTrajet(const Trajet * T)
141 {
142     catalog->Add(T);
143 }
144
145 void Catalogue::RechercherSimple(const char * depart, const char * arrivee) const
146 {
147     cout << endl << "*****************************************" << endl;
148     cout << "Parcours directs disponibles pour faire " << depart << " --> " << arrivee << " :" << endl;
149     for (unsigned int i = 0; i < catalog->GetUtilise(); i++)
150     {
151         if (!strcmp(depart, catalog->Get(i)->getDepart()) && !strcmp(arrivee, catalog->Get(i)->getArrivee()))
152         {
153             catalog->Get(i)->Afficher();
154         }
```

```
155     }
156     cout << "****************************************" << endl << endl;
157 }
158
159 void Catalogue::RechercherAvancee(const char * depart, const char * arrivee) ↵
       const
160 {
161     cout << endl << "****************************************" << endl;
162     cout << "Parcours disponibles pour faire " << depart << " --> " <<      ↵
          arrivee << " :" << endl << endl;
163     int * utilise = new int[catalog->GetUtilise()];
164     for (unsigned int i = 0; i < catalog->GetUtilise(); i++)
165     {
166         for (unsigned int j = 0; j < catalog->GetUtilise(); j++)
167         {
168             utilise[j] = 0;
169         }
170         if (!strcmp(depart, catalog->Get(i)->getDepart()))
171         {
172             utilise[i] = 1;
173             this->recure(utilise, 2, i, arrivee);
174         }
175     }
176     cout << "****************************************" << endl << endl;
177     delete[] utilise;
178 }
179
180
181 //------------------------------------------ Constructeurs - destructeur
182 Catalogue::Catalogue()
183 {
184 #ifdef MAP
185     cout << "Appel au constructeur de <Catalogue>" << endl;
186 #endif
187     catalog = new TabTrajet();
188 } //----- Fin de Catalogue
189
190 Catalogue::~Catalogue()
191 {
192 #ifdef MAP
193     cout << "Appel au destructeur de <Catalogue>" << endl;
194 #endif
195     delete catalog;
196 } //----- Fin de ~Catalogue
197
198
199   //------------------------------------------------------------------ PRIVE
200   //---------------------------------------------------- Méthodes protégées
201
202 void Catalogue::recure(int utilise[], int numeroTrajet, int trajetPrecedent, ↵
       const char * arriveeFinale) const
203 {
204     if (!strcmp(catalog->Get(trajetPrecedent)->getArrivee(), arriveeFinale))
```

```
205     {
206         this->AfficherParcours(utilise);
207         return;
208     }
209     for (unsigned int i = 0; i < catalog->GetUtilise(); i++)
210     {
211         if (!strcmp(catalog->Get(i)->getDepart(), catalog->Get
              (trajetPrecedent)->getArrivee()) && utilise[i] == 0)
212         {
213             utilise[i] = numeroTrajet;
214             this->recure(utilise, numeroTrajet + 1, i, arriveeFinale);
215             utilise[i] = 0;
216         }
217     }
218 }
219
220 void Catalogue::AfficherParcours(int utilise[]) const
221 {
222     int indiceMax = this->max(utilise);
223     cout << "--------------------------" << endl;
224     for (int j = 1; j <= indiceMax; j++)
225     {
226         for (unsigned int i = 0; i < catalog->GetUtilise(); i++)
227         {
228             if (j == utilise[i])
229             {
230                 catalog->Get(i)->Afficher();
231             }
232         }
233     }
234 }
235
236 int Catalogue::max(int utilise[]) const // On sait que utilise est de la
      taille de catalog
237 {
238     int maxi = 0;
239     for (unsigned int i = 0; i < catalog->GetUtilise(); i++)
240     {
241         if (utilise[i] > maxi) maxi = utilise[i];
242     }
243     return maxi;
244 }
245
```