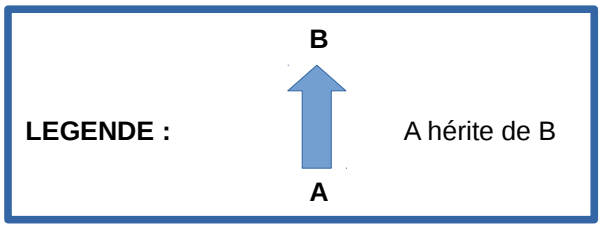
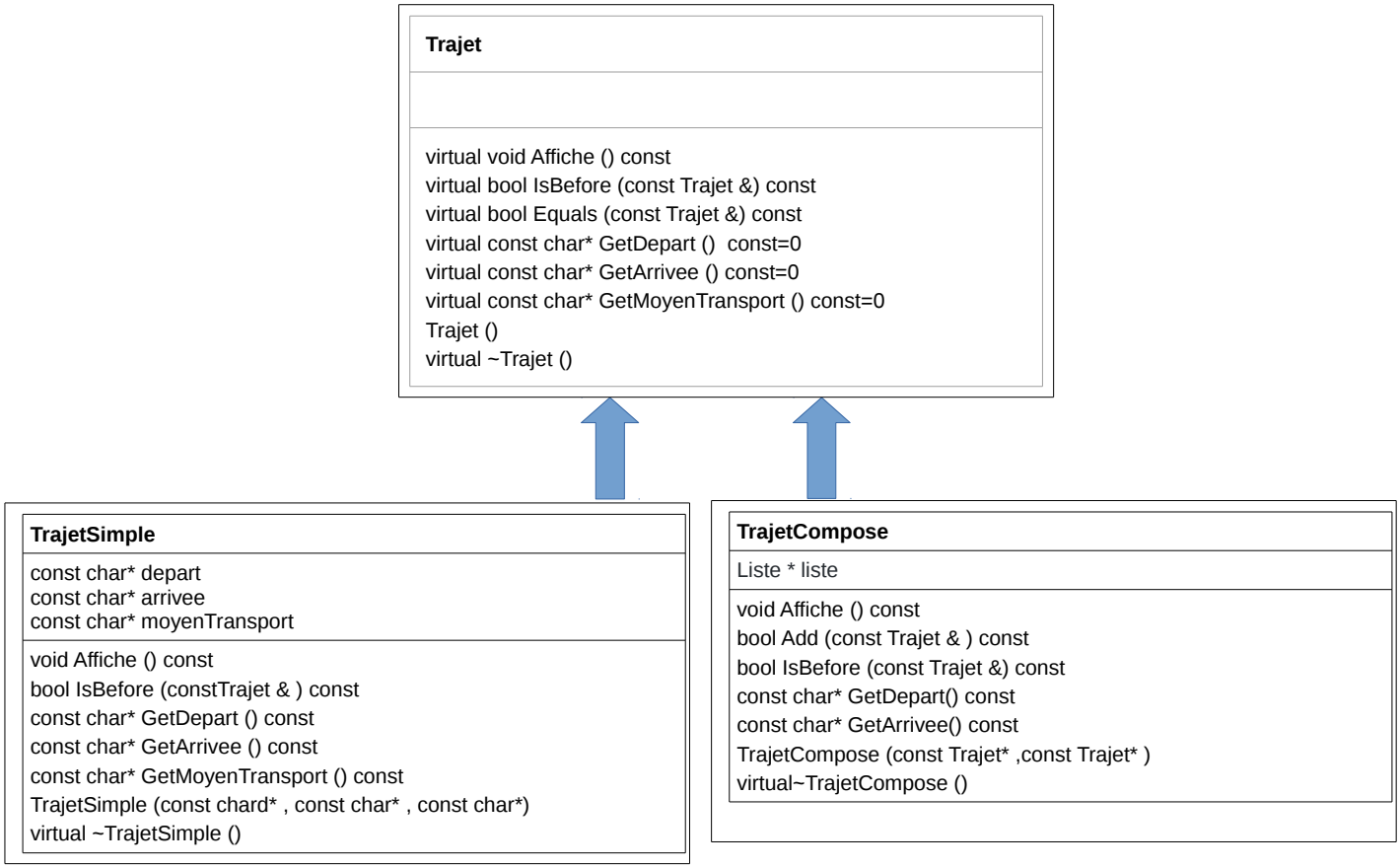
Florian MUTIN 3IF4 binôme B3414
Loïc CASTELLON

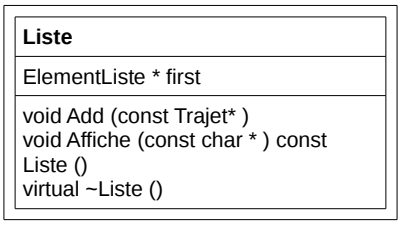
Compte rendu TP C++ n°2 : Héritage - Polymorphisme

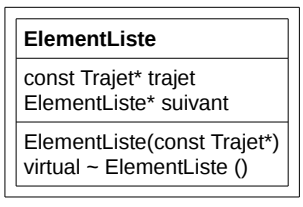
SOMMAIRE :

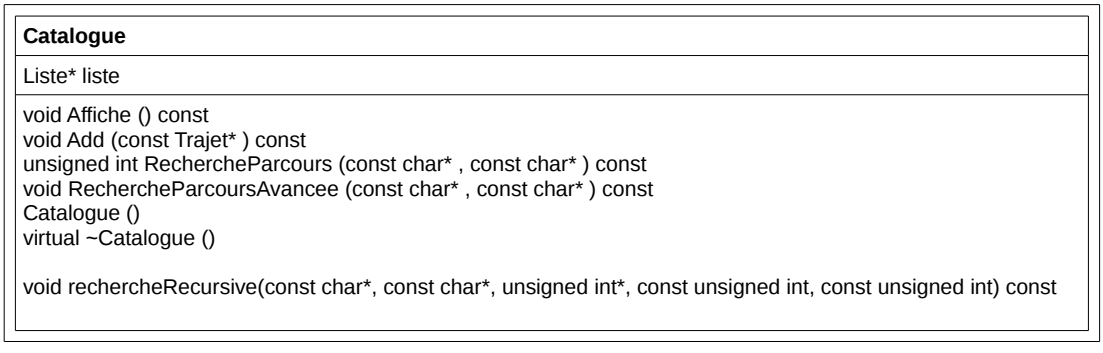
I) Description des classes – graphe d'héritage











- Nous avons choisi de ne mettre aucun attribut dans la classe Trajet car les départs et arrivées se retrouvent à partir des TrajetSimples.
- Les attributs de la classe TrajetSimple sont en private, on peut y accéder uniquement en lecture grâce à des méthodes Get spécifiques.
- Les classes TrajetCompose et Catalogue reposent fortement sur la classe Liste, cela nous permet d'éviter une redondance de code entre ces deux premières.
- Les classes Liste et ElementListe sont expliquées ci-dessous.

II) Description de la structure de donnée – dessin mémoire

Un catalogue possède un pointeur sur une Liste. Liste est une classe qui implémente une liste chaînée d'ElementListe, quand on ajoute un ElementListe il est inséré en fin de liste. Un ElementListe possède un pointeur sur un Trajet et un pointeur sur un ElementListe suivant. Un TrajetCompose possède un pointeur sur une Liste , il fonctionne de la même manière.

Le dessin mémoire donné en annexe illustre cela.

III) Conclusion

Nous avons rencontré plusieurs problèmes lors de ce projet. Le premier est que nous avions au départ mal implémenté la façon de faire une Liste, nous avions essayer de gérer ça avec de l'héritage entre une classe Liste et les classes de Trajets. Nous nous sommes ensuite souvenu qu'il fallait séparer tout cela et que de l'héritage entre les deux n'était pas la bonne solution.

Ensuite, nous avons eu du mal à implémenter la méthode RechercheParcoursAvancee. Ce n'était pas tant le fait de combiner les trajets qui nous a paru difficile mais plutôt de trouver comment cette méthode pouvait afficher tous les trajets possibles. Pour résoudre cela nous avons dû faire une recherche récursive avec parcours en profondeur de la Liste du catalogue.

Le constructeur de TrajetCompose n'est pas optimal lorsque l'on veut créer un TrajetCompose dans notre menu. Nous nous en sommes rendu compte sur le fait. Nous avons décidé d'assumer notre choix de constructeur et d'adapter notre menu.

La méthode IsBefore nous a servi en début de projet pour assurer la continuité des trajets dans un TrajetCompose. Cependant notre menu assure la continuité. Cette méthode est maintenant obsolète mais nous l'avons conservée pour effectuer des tests et afficher d'éventuelles erreurs.

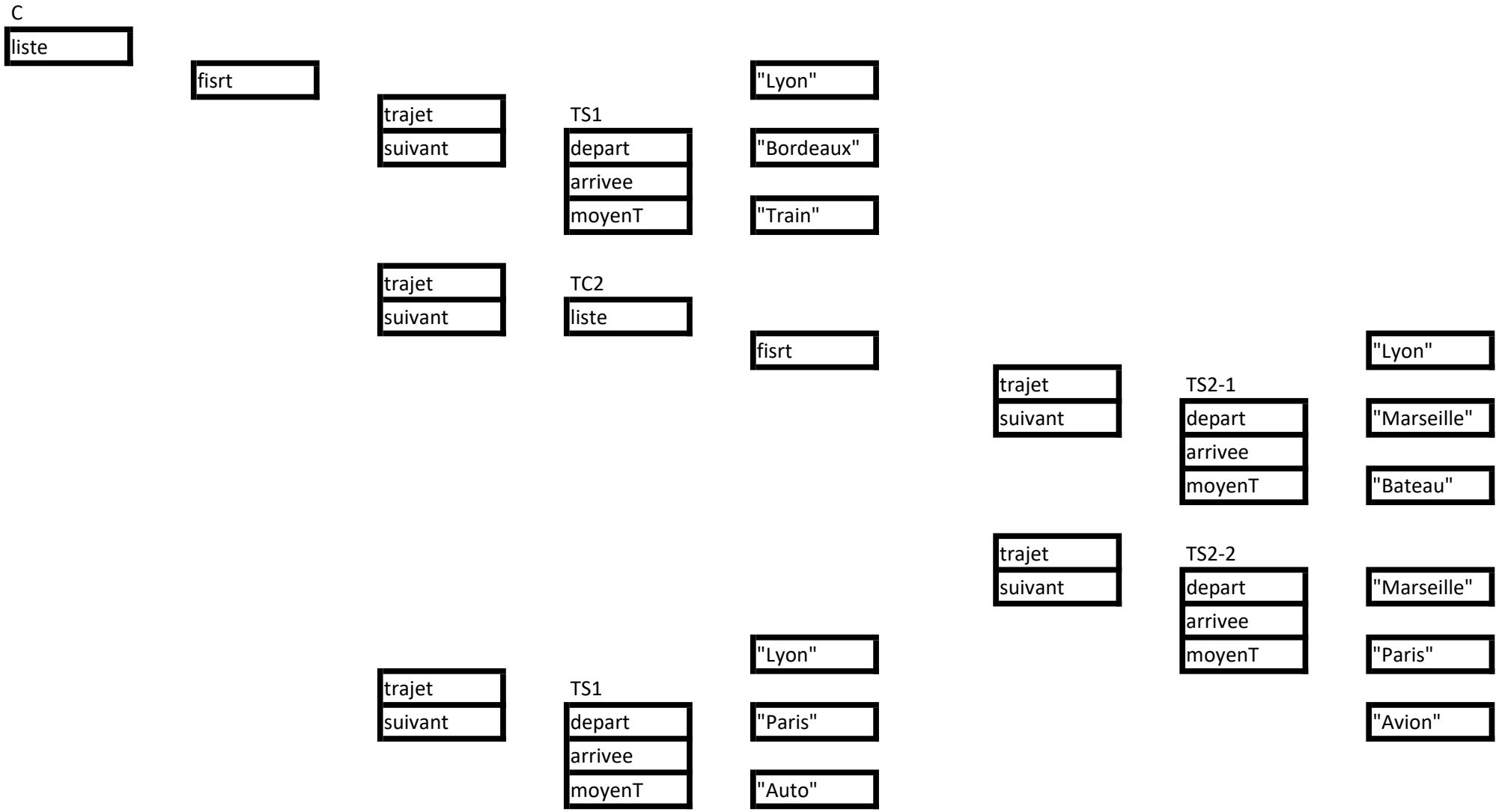
Quand on renseigne les attributs de Trajet ils sont transformés en majuscules. Cela nous permet d'assurer qu'une ville ne sera pas écrite en minuscule et en majuscule. Mais cela ne fonctionne pas pour les caractères spéciaux tels les accents ou la cédille. Cela pourrait donc faire l'objet d'une amélioration.

De plus quand on ajoute un TrajetSimple on vérifie s'il existe mais nous ne vérifions pas cela pour l'ajout de TrajetComposé. Cela n'a pas été implémenté par manque de temps mais peut faire l'objet d'une amélioration.

Dans notre menu, si l'utilisateur saisit autre chose qu'un entier le programme se ferme. De la même manière s'il saisit autre chose qu'un entier lors de l'ajout d'étapes pour un TrajetComposé le programme entre dans une boucle infinie. Ces constatations rendent l'expérience utilisateur difficile. Le menu est donc améliorable.

IV) Annexes : Dessin mémoire et listing des classes

avec moyenT = moyenTransport



```
/*************************************************************************
                                Main
                            -------------------
    début                : 18/11/2017
    copyright            : (C) 2017 par Loïc CASTELLON & Florian MUTIN
    e-mail               : loic.castellon@insa-lyon.fr
                           florian.mutin@insa-lyon.fr
*************************************************************************/
using namespace std;
#include <iostream>
#include "TrajetCompose.h"
#include "Catalogue.h"
#include <cstring>

void application();
// Mode d'emploi : lancement de l'application
// Contrat : aucun

void majuscule(char *chaine);
// Mode d'emploi : change les minuscules en majuscules (hors caractères
//     spéciaux, accents, cédilles)
// Contrat : chaine est non nul

int main (){
    application();
    return 0;

}

void application()
// Algorithme : un switch dans un while permet d'évoluer dans le menu
//      puis de revenir au menu principal autant de fois que souhaité.
{
    const unsigned int NB_MAX_CHAR = 100;

    cout<<endl;
    cout<<"*******************************"<<endl;
    cout<<"* Ouverture de l'application *"<<endl;
    cout<<"*******************************"<<endl<<endl;
    cout<<"-------------------------------------------------------------";
    cout<<endl<<endl;

    Catalogue catalogue;
    unsigned int saisieMenu;
    do
    {
        cout<<"Menu :"<<endl;
        cout<<"0) quitter l'application"<<endl;
        cout<<"1) ajouter un trajet simple"<<endl;
        cout<<"2) ajouter un trajet composé"<<endl;
        cout<<"3) afficher le catalogue"<<endl;
        cout<<"4) recherche de parcours"<<endl;
        cout<<"5) recherche avancée de parcours"<<endl;
        cout<<endl;
        cout <<"Saisissez votre choix : ";
        cin>>saisieMenu;
        cout<<endl;
        switch(saisieMenu)
        {

            // AJOUT TRAJET SIMPLE
            case 1:
            {
                cout<<"-- ajouter un trajet simple --"<<endl;
                cout <<"Saisissez la ville de départ : ";
                char* depart = new char[NB_MAX_CHAR];
                cin >> depart;
                majuscule(depart);
                cout <<"Saisissez la ville d'arrivée : ";
```

```
        char* arrivee = new char[NB_MAX_CHAR];
        cin >> arrivee;
        majuscule(arrivee);
        cout <<"Saisissez le moyen de transport : ";
        char* moyenTransport = new char[NB_MAX_CHAR];
        cin >> moyenTransport;
        majuscule(moyenTransport);

        Trajet * t = new TrajetSimple(
                                depart,arrivee,moyenTransport);
        if(!catalogue.Add(t))
        {
            cout<<"Ce trajet est déjà présent dans le catalogue.";
            cout<<endl;
            delete t;
        }

        break;
    }

    // AJOUT TRAJET COMPOSE
    case 2:
    {
        //saisie d'au moins un départ et une étape
        cout<<"-- ajouter un trajet composé --"<<endl;
        cout <<"Saisissez la ville de départ : ";
        char* depart = new char[NB_MAX_CHAR];
        cin >> depart;
        majuscule(depart);
        cout <<"Saisissez une ville étape : ";
        char* etape = new char[NB_MAX_CHAR];
        cin >> etape;
        majuscule(etape);
        cout <<"Saisissez le moyen de transport : ";
        char* moyenTransport = new char[NB_MAX_CHAR];
        cin >> moyenTransport;
        majuscule(moyenTransport);

        Trajet * td = new TrajetSimple(
                                    depart,etape,moyenTransport);

        char* etapeCopie = new char[NB_MAX_CHAR];
        strcpy(etapeCopie,etape);

        TrajetCompose * tc;
        bool creation = true;

        //saisie et ajout d'autant d'étape que souhaité
        bool continuer;
        cout << "--" <<endl;
        cout << "0) Saisir la ville d'arrivee" <<endl;
        cout << "1) Saisir une ville étape" <<endl;
        cout << "Votre choix : ";
        cin >> continuer;
        cout << "--" <<endl;

        while(continuer)
        {
            //saisie de l'étape
            cout <<"Saisissez une ville étape : ";
            etape = new char[NB_MAX_CHAR];
            cin >> etape;
            majuscule(etape);
            cout <<"Saisissez le moyen de transport : ";
            moyenTransport = new char[NB_MAX_CHAR];
            cin >> moyenTransport;
            majuscule(moyenTransport);

            Trajet * t = new TrajetSimple(
```

```
                                etapeCopie,etape,moyenTransport);
            //ajout de l'étape (création du trajet composé si
            //nécessaire)
            if(creation)
            {
                tc = new TrajetCompose(td,t);
                creation = false;
            }
            else
            {
                tc->Add(t);
            }

            etapeCopie = new char[NB_MAX_CHAR];
            strcpy(etapeCopie,etape);

            cout << "--" <<endl;
            cout << "0) Saisir la ville d'arrivee" <<endl;
            cout << "1) Saisir une ville étape" <<endl;
            cout << "Votre choix : ";
            cin >> continuer;
            cout << "--" <<endl;
        }

        //saisie de l'arrivée
        cout <<"Saisissez la ville d'arrivée : ";
        char* arrivee = new char[NB_MAX_CHAR];
        cin >> arrivee;
        majuscule(arrivee);
        cout <<"Saisissez le moyen de transport : ";
        moyenTransport = new char[NB_MAX_CHAR];
        cin >> moyenTransport;
        majuscule(moyenTransport);

        Trajet * ta = new TrajetSimple(
                            etapeCopie,arrivee,moyenTransport);
        //ajout de l'arrivée (création du trajet composé si
        //nécessaire)
        if(creation)
        {
            tc = new TrajetCompose(td,ta);
            creation = false;
        }
        else
        {
            tc->Add(ta);
        }

        catalogue.Add(tc);

        break;
    }


    // AFFICHAGE CATALOGUE
    case 3:
        cout<<"-- afficher le catalogue --"<<endl;
        catalogue.Affiche();
        break;


    // RECHERCHE PARCOURS
    case 4:
    {
        cout<<"-- recherche de parcours --"<<endl;
        cout <<"Saisissez la ville de départ : ";
        char* depart = new char[NB_MAX_CHAR];
        cin >> depart;
        majuscule(depart);
```

```
                cout <<"Saisissez la ville d'arrivée : ";
                char* arrivee = new char[NB_MAX_CHAR];
                cin >> arrivee;
                majuscule(arrivee);

                catalogue.RechercheParcours(depart,arrivee);

                delete[] depart;
                delete[] arrivee;

                break;
            }

            // RECHERCHE AVANCEE PARCOURS
            case 5:
            {
                cout<<"-- recherche avancée de parcours --"<<endl;
                cout <<"Saisissez la ville de départ : ";
                char* depart = new char[NB_MAX_CHAR];
                cin >> depart;
                majuscule(depart);
                cout <<"Saisissez la ville d'arrivée : ";
                char* arrivee = new char[NB_MAX_CHAR];
                cin >> arrivee;
                majuscule(arrivee);

                catalogue.RechercheParcoursAvancee(depart,arrivee);

                delete[] depart;
                delete[] arrivee;

                break;
            }
        }
    cout<<endl;
    cout<<"------------------------------------------------------------";
    cout<<endl<<endl;
    }
    while(saisieMenu != 0);


    cout<<"*******************************"<<endl;
    cout<<"* Fermeture de l'application *"<<endl;
    cout<<"*******************************"<<endl<<endl;
} //----- Fin de application

void majuscule(char *chaine)
// Algorithme : aucun
{
    unsigned int i = 0;
    while (chaine[i] != '\0')
    {
        if (chaine[i]  >= 97 &&  chaine[i] <= 122)
            chaine[i] = chaine[i] - 32;
        i++;
    }
} //----- Fin de majuscule

/*************************************************************************
                                Liste
                            -------------------
    début                : 22/11/2017
    copyright            : (C) 2017 par Loïc CASTELLON & Florian MUTIN
    e-mail               : loic.castellon@insa-lyon.fr
                           florian.mutin@insa-lyon.fr
*************************************************************************/

//---------- Interface de la classe <Liste> (fichier Liste.h) ----------
#if ! defined ( Liste_H )
```

```
#define Liste_H

//---------------------------------------------------- Interfaces utilisées
#include "ElementListe.h"

//------------------------------------------------------------------------
// Rôle de la classe <Liste>
// Cette classe implémente une liste chainée de Trajets
//------------------------------------------------------------------------

class Liste
{
//----------------------------------------------------------------- PUBLIC

public:
//---------------------------------------------------- Méthodes publiques
    void Add(const Trajet* t);
    // Mode d'emploi : ajout de t en fin de liste
    // Contrat : t est non nul

    void Affiche(const char * texte) const;
    // Mode d'emploi : affiche tout les élements du contexte appelant en
    //     affichant texte devant
    // Contrat : texte est non nul

//------------------------------------------- Constructeurs - destructeur
    Liste ();
    // Mode d'emploi : aucun
    // Contrat : aucun

    virtual ~Liste ( );
    // Mode d'emploi : aucun
    // Contrat : aucun

//------------------------------------------------- Attributs publiques
    ElementListe * first;

};

#endif // Liste_H

/*************************************************************************
                                Liste
                            -------------------
    début                : 22/11/2017
    copyright            : (C) 2017 par Loïc CASTELLON & Florian MUTIN
    e-mail               : loic.castellon@insa-lyon.fr
                           florian.mutin@insa-lyon.fr
*************************************************************************/

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

//------------------------------------------------------------ INCLUDE
//-------------------------------------------------------- Include système
using namespace std;
#include <iostream>

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

//----------------------------------------------------------------- PUBLIC
//---------------------------------------------------- Méthodes publiques
void Liste::Affiche(const char * texte) const
// Algorithme : aucun
{
    ElementListe *cur=first;
    unsigned int i=1;
    while(cur!=nullptr)
    {
```

```
        cout<<texte<<i<<") ";
        cur->trajet->Affiche();
        cout<<endl;
        cur = cur->suivant;
        ++i;
    }
} //----- Fin de Affiche

void Liste::Add(const Trajet* t)
// Algorithme : aucun
{
    ElementListe* e = new ElementListe(t);
    if(first==nullptr){
        first=e;
        return;
    }
    ElementListe* cur = first;
    ElementListe* next = cur->suivant;
    while(next!=nullptr){
        cur = next;
        next = cur->suivant;
    }
    cur->suivant = e;
} //----- Fin de Add

//------------------------------------------- Constructeurs - destructeur
Liste::Liste ()
// Algorithme : aucun
:first(nullptr)
{
#ifdef MAP
    cout << "Appel au constructeur de <Liste>" << endl;
#endif
} //----- Fin de Liste


Liste::~Liste ( )
// Algorithme : aucun
{
#ifdef MAP
    cout << "Appel au destructeur de <Liste>" << endl;
#endif
    ElementListe *cur=first;
    ElementListe *next;
    while(cur!=nullptr)
    {
        next = cur->suivant;
        delete cur;
        cur = next;
    }
} //----- Fin de ~Liste

/*************************************************************************
                                ElementListe
                            -------------------
    début                : 22/11/2017
    copyright            : (C) 2017 par Loïc CASTELLON & Florian MUTIN
    e-mail               : loic.castellon@insa-lyon.fr
                           florian.mutin@insa-lyon.fr
*************************************************************************/

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

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

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

```
// Rôle de la classe <ElementListe>
// Cette classe implémente les éléments de la classe Liste.
//------------------------------------------------------------------------

class ElementListe
{
//------------------------------------------------------------------ PUBLIC
public:
//----------------------------------------- Constructeurs - destructeur
    ElementListe (const Trajet* t);
    // Mode d'emploi : aucun
    // Contrat : t est non nul

    virtual ~ElementListe ( );
    // Mode d'emploi : aucun
    // Contrat : aucun

//------------------------------------------------- Attributs publiques
    const Trajet * trajet;
    ElementListe * suivant;

};

#endif // ElementListe_H

/*************************************************************************
                                ElementListe
                            -------------------
    début                : 22/11/2017
    copyright            : (C) 2017 par Loïc CASTELLON & Florian MUTIN
    e-mail               : loic.castellon@insa-lyon.fr
                           florian.mutin@insa-lyon.fr
*************************************************************************/

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

//------------------------------------------------------------ INCLUDE
//-------------------------------------------------------- Include système
using namespace std;
#include <iostream>

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

//------------------------------------------------------------------ PUBLIC
//----------------------------------------- Constructeurs - destructeur
ElementListe::ElementListe (const Trajet* t)
// Algorithme : aucun
:trajet(t),suivant(nullptr)
{
#ifdef MAP
    cout << "Appel au constructeur de <ElementListe>" << endl;
#endif
} //----- Fin de ElementListe

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

/*************************************************************************
                                Trajet
                            -------------------
    début                : 15/11/2017
    copyright            : (C) 2017 par Loïc CASTELLON & Florian MUTIN
```

```
    e-mail               : loic.castellon@insa-lyon.fr
                           florian.mutin@insa-lyon.fr
*************************************************************************/

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

//------------------------------------------------------------------------
// Rôle de la classe <Trajet>
// Cette classe est construite comme la classe mère des classes
// TrajetSimple et TrajetCompose.
//------------------------------------------------------------------------

class Trajet
{
//------------------------------------------------------------------ PUBLIC
public:
//----------------------------------------------------- Méthodes publiques
    virtual void Affiche() const;
    // Mode d'emploi : affiche les attributs du contexte appelant
    // Contrat : aucun

    virtual bool IsBefore(const Trajet& t) const;
    // Mode d'emploi : retourne true si le départ de t correspond à
    //     l'arrivée du contexte appelant. Sinon retourne false
    // Contrat : aucun

    virtual bool Equals(const Trajet& t) const;
    // Mode d'emploi : retourne true si l'on compare un trajet simple à
    //     un autre trajet simple identique, sinon retourne false
    // Contrat : aucun

    virtual const char* GetDepart() const = 0;
    // Mode d'emploi : renvoi le depart du trajet
    // Contrat : aucun

    virtual const char* GetArrivee() const = 0;
    // Mode d'emploi : renvoi l'arrivée du trajet
    // Contrat : aucun

    virtual const char* GetMoyenTransport() const;
    // Mode d'emploi : renvoi le moyen de transport s'il existe sinon nul
    // Contrat : aucun

//-------------------------------------------- Constructeurs - destructeur
    Trajet ( );
    // Mode d'emploi : aucun
    // Contrat : aucun

    virtual ~Trajet ( );
    // Mode d'emploi : aucun
    // Contrat : aucun

};

#endif // Trajet_H

/*************************************************************************
                                Trajet
                             -------------------
    début                : 15/11/2017
    copyright            : (C) 2017 par Loïc CASTELLON & Florian MUTIN
    e-mail               : loic.castellon@insa-lyon.fr
                           florian.mutin@insa-lyon.fr
*************************************************************************/

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

```cpp
//------------------------------------------------------------------ INCLUDE
//-------------------------------------------------------- Include système
using namespace std;
#include <iostream>
#include <cstring>

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

//------------------------------------------------------------------ PUBLIC
//---------------------------------------------------- Méthodes publiques
void Trajet::Affiche () const
// Algorithme : aucun
{
    cout<<GetDepart()<<" -> "<<GetArrivee();
} //----- Fin de Affiche

bool Trajet::IsBefore (const Trajet & t) const
// Algorithme : aucun
{
    const char* departT = t.GetDepart();
    const char* arriveeThis = this->GetArrivee();

    if(strlen(departT) != strlen(arriveeThis))
        return false;
    else
    {
        for(unsigned int i=0 ; i<strlen(departT) ; i++)
        {
            if(departT[i] != arriveeThis[i])
                return false;
        }
    }
    return true;
} //----- Fin de IsBefore

bool Trajet::Equals (const Trajet & t) const
// Algorithme : aucun
{
    //comparaison des moyens de transport
    const char* moyenTransport = t.GetMoyenTransport();
    const char* moyenTransportThis = this->GetMoyenTransport();
    if(moyenTransportThis == nullptr || moyenTransport == nullptr)
    {
        return false;
    }
    else
    {
        if(strlen(moyenTransport) != strlen(moyenTransportThis))
            return false;
        else
        {
            for(unsigned int i=0 ; i<strlen(moyenTransport) ; i++)
            {
                if(moyenTransport[i] != moyenTransportThis[i])
                    return false;
            }
        }
    }

    //comparaison des départs
    const char* depart = t.GetDepart();
    const char* departThis = this->GetDepart();
    if(strlen(depart) != strlen(departThis))
        return false;
    else
    {
        for(unsigned int i=0 ; i<strlen(depart) ; i++)
        {
```

```
            if(depart[i] != departThis[i])
                    return false;
        }
    }

    //comparaison des arrivées
    const char* arrivee = t.GetArrivee();
    const char* arriveeThis = this->GetArrivee();
    if(strlen(arrivee) != strlen(arriveeThis))
        return false;
    else
    {
        for(unsigned int i=0 ; i<strlen(arrivee) ; i++)
        {
            if(arrivee[i] != arriveeThis[i])
                    return false;
        }
    }

    return true;
} //----- Fin de Equals

const char* Trajet::GetMoyenTransport() const
// Algorithme : aucun
{
    return nullptr;
} //----- Fin de GetMoyenTransport

//-------------------------------------------- Constructeurs - destructeur
Trajet::Trajet ( )
// Algorithme : aucun
{
#ifdef MAP
    cout << "Appel au constructeur de <Trajet>" << endl;
#endif
} //----- Fin de Trajet

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

/*************************************************************************
                                TrajetSimple
                            -------------------
    début                : 15/11/2017
    copyright            : (C) 2017 par Loïc CASTELLON & Florian MUTIN
    e-mail               : loic.castellon@insa-lyon.fr
                           florian.mutin@insa-lyon.fr
*************************************************************************/

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

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

//------------------------------------------------------------------------
// Rôle de la classe <TrajetSimple>
// Cette classe implémente les trajets simples
//------------------------------------------------------------------------

class TrajetSimple : public Trajet
{
//----------------------------------------------------------------- PUBLIC
```

```
public:
//---------------------------------------------------- Méthodes publiques
    void Affiche () const;
    // Mode d'emploi : affiche les attributs du contexte appelant
    // Contrat : aucun

    bool IsBefore(const Trajet& t) const;
    // Mode d'emploi : retourne true si le départ de t correspond à
    //     l'arrivée du contexte appelant. Sinon retourne false.
    // Contrat : aucun

    const char* GetDepart() const;
    // Mode d'emploi : renvoi le départ
    // Contrat : aucun

    const char* GetArrivee() const;
    // Mode d'emploi : renvoi l'arrivée
    // Contrat : aucun

    const char* GetMoyenTransport() const;
    // Mode d'emploi : renvoi le moyen de transport
    // Contrat : aucun

//------------------------------------------- Constructeurs - destructeur
    TrajetSimple (const char* d, const char* a, const char* mT);
    // Mode d'emploi : d le départ du trajet, a l'arrivée du trajet et mT
    //     le moyen de transport du trajet
    // Contrat : d, a et mT sont non nuls

    virtual ~TrajetSimple ( );
    // Mode d'emploi : aucun
    // Contrat : aucun

//------------------------------------------------------------------ PROTEGE
protected:
//----------------------------------------------------- Attributs protégés
    const char *depart, *arrivee, *moyenTransport;

};

#endif // TrajetSimple_H

/*****************************************************************************
                                TrajetSimple
                             -------------------
    début                : 15/11/2017
    copyright            : (C) 2017 par Loïc CASTELLON & Florian MUTIN
    e-mail               : loic.castellon@insa-lyon.fr
                           florian.mutin@insa-lyon.fr
*****************************************************************************/

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

//---------------------------------------------------------------- INCLUDE
//-------------------------------------------------------- Include système
using namespace std;
#include <iostream>

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

//------------------------------------------------------------------ PUBLIC
//---------------------------------------------------- Méthodes publiques
void TrajetSimple::Affiche () const
// Algorithme : aucun
{
    Trajet::Affiche();
    cout << " | " << moyenTransport;
} //----- Fin de Affiche
```

```cpp
bool TrajetSimple::IsBefore (const Trajet& t) const
// Algorithme : aucun
{
    return Trajet::IsBefore (t);
} //----- Fin de IsBefore

const char* TrajetSimple::GetDepart() const{
// Algorithme : aucun
    return depart;
} //----- Fin de GetDepart

const char* TrajetSimple::GetArrivee() const{
// Algorithme : aucun
    return arrivee;
} //----- Fin de GetArrivee

const char* TrajetSimple::GetMoyenTransport() const
// Algorithme : aucun
{
    return moyenTransport;
} //----- Fin de GetMoyenTransport

//------------------------------------------- Constructeurs - destructeur
TrajetSimple::TrajetSimple (const char* d, const char* a, const char* mT)
// Algorithme : aucun
: depart(d),arrivee(a),moyenTransport(mT)
{
#ifdef MAP
    cout << "Appel au constructeur de <TrajetSimple>" << endl;
#endif
} //----- Fin de TrajetSimple


TrajetSimple::~TrajetSimple ( )
// Algorithme :
{
    delete[] depart;
    delete[] arrivee;
    delete[] moyenTransport;
#ifdef MAP
    cout << "Appel au destructeur de <TrajetSimple>" << endl;
#endif
} //----- Fin de ~TrajetSimple

/*************************************************************************
                            TrajetCompose
                          -------------------
    début                : 15/11/2017
    copyright            : (C) 2017 par Loïc CASTELLON & Florian MUTIN
    e-mail               : loic.castellon@insa-lyon.fr
                           florian.mutin@insa-lyon.fr
*************************************************************************/

//--- Interface de la classe <TrajetCompose> (fichierTrajetCompose.h)----
#if ! defined ( TrajetCompose_H )
#define TrajetCompose_H

//------------------------------------------------- Interfaces utilisées
#include "Liste.h"
#include "TrajetSimple.h"

//------------------------------------------------------------------------
// Rôle de la classe <TrajetCompose>
// Cette classe implémente les trajets composés
//------------------------------------------------------------------------

class TrajetCompose : public Trajet
{
```

```
//------------------------------------------------------------------ PUBLIC
public:
//----------------------------------------------------- Méthodes publiques
    void Affiche()const;
    // Mode d'emploi : affiche les attributs du contexte appelant
    // Contrat : aucun

    bool Add(const Trajet* t) const;
    // Mode d'emploi : si le trajet t est une suite du contexte appelant,
    //     alors il est ajouté au contexte appelant et on retourne true,
    //     sinon on ne fait rien et retourne false
    // Contrat : t est non nul

    bool IsBefore(const Trajet& t) const;
    // Mode d'emploi : retourne true si le départ de t correspond à
    //     l'arrivée du contexte appelant. Sinon retourne false.
    // Contrat : aucun

    const char* GetDepart() const;
    // Mode d'emploi : retourne le départ
    // Contrat : aucun

    const char* GetArrivee() const;
    // Mode d'emploi : retourne l'arrivée
    // Contrat : aucun

//------------------------------------------- Constructeurs - destructeur
    TrajetCompose (const Trajet* t1, const Trajet* t2);
    // Mode d'emploi : t1 et t2 sont respéctivement l'étape 1 et 2 du
    //     trajet composé créé
    // Contrat : t1 et t2 sont non nuls et t2 est une suite de t1

    virtual ~TrajetCompose ( );
    // Mode d'emploi :
    // Contrat :

//------------------------------------------------------------------ PROTEGE
protected:
//----------------------------------------------------- Attributs protégés
    Liste * liste;

};

#endif // TrajetCompose_H

/*************************************************************************
                                TrajetCompose
                             -------------------
    début                : 15/11/2017
    copyright            : (C) 2017 par Loïc CASTELLON & Florian MUTIN
    e-mail               : loic.castellon@insa-lyon.fr
                           florian.mutin@insa-lyon.fr
*************************************************************************/

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

//------------------------------------------------------------------ INCLUDE
//-------------------------------------------------------- Include système
using namespace std;
#include <iostream>

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

//------------------------------------------------------------------ PUBLIC
//----------------------------------------------------- Méthodes publiques
 void TrajetCompose::Affiche() const
// Algorithme : aucun
{
```

```
    if(liste->first == nullptr)
    {
        cout << "Erreur : TrajetCompose vide"<<endl;
        return;
    }
    Trajet::Affiche();
    cout << endl;
    liste->Affiche("            ");

} //----- Fin de Affiche

 bool TrajetCompose::Add(const Trajet* t) const
// Algorithme : on ajoute le trajet *t en fin de liste
 {
    if(liste->first == nullptr)
    {
        cout<<"Erreur : TrajetCompose vide"<<endl;
        return false;
    }
    ElementListe* cur = liste->first;
    ElementListe* next = cur->suivant;
    while(next!=nullptr)
    {
        cur = next;
        next = cur->suivant;
    }
    if(cur->trajet->IsBefore(*t))
    {
        liste->Add(t);
        return true;
    }
    else
    {
        cout<<"Erreur: Les trajets ne se suivent pas"<<endl;
        return false;
    }
 } //----- Fin de Add

 bool TrajetCompose::IsBefore(const Trajet& t) const
// Algorithme : aucun
 {
     return Trajet::IsBefore(t);
 } //----- Fin de IsBefore

 const char* TrajetCompose::GetDepart() const
// Algorithme : aucun
 {
    if(liste->first == nullptr)
    {
        cout<<"Erreur TrajetCompose::GetDepart"<<endl;
        return "";
    }
    return liste->first->trajet->GetDepart();
} //----- Fin de GetDepart

const char* TrajetCompose::GetArrivee() const
// Algorithme : aucun
 {
    if(liste->first == nullptr)
    {
        cout<<"Erreur TrajetCompose::GetArrivee"<<endl;
        return "";
    }
    ElementListe* cur = liste->first;
    ElementListe* next = cur->suivant;
    while(next!=nullptr)
    {
        cur = next;
        next = cur->suivant;
```

```
    }
    return cur->trajet->GetArrivee();
 } //----- Fin de GetArrivee

//------------------------------------------- Constructeurs - destructeur
TrajetCompose::TrajetCompose (const Trajet* t1, const Trajet* t2)
// Algorithme : aucun
{
    if(t1->IsBefore(*t2)) {
        liste = new Liste();
        liste->Add(t1);
        liste->Add(t2);
    }
    else {
        liste = nullptr;
        cout<<"Erreur: Les trajets ne se suivent pas"<<endl;
    }
#ifdef MAP
    cout << "Appel au constructeur de <TrajetCompose>" << endl;
#endif
} //----- Fin de TrajetCompose

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

/**************************************************************************
                                Catalogue
                             -------------------
    début                : 18/11/2017
    copyright            : (C) 2017 par Loïc CASTELLON & Florian MUTIN
    e-mail               : loic.castellon@insa-lyon.fr
                           florian.mutin@insa-lyon.fr
**************************************************************************/

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

//------------------------------------------------- Interfaces utilisées
#include "Liste.h"

//------------------------------------------------------------------------
// Rôle de la classe <Catalogue>
// Cette classe permet de gérer de stocker des trajets et effectuer des
// recherches
//------------------------------------------------------------------------

class Catalogue
{
//----------------------------------------------------------------- PUBLIC
public:
//---------------------------------------------------- Méthodes publiques
    void Affiche() const;
    // Mode d'emploi : affiche les trajets du contexte appelant
    // Contrat : aucun

    bool Add(const Trajet* t) const;
    // Mode d'emploi : retourne false et ne fait rien si le trajet et
    //      déjà présent, sinon ajoute le trajet et retourne true
    // Contrat : aucun

    unsigned int RechercheParcours(const char* depart,
                                   const char* arrivee ) const;
```

```
    // Mode d'emploi : recherche de trajets qui vont de depart à arrivee
    //      et renvoie le nombre de solution
    // Contrat : depart et arrivee sont non nuls

    void RechercheParcoursAvancee(const char* depart,
                                  const char* arrivee ) const;
    // Mode d'emploi : recherche les compositions de trajets qui vont de     //
depart à arrivee et affiche toutes les solutions
    // Contrat : depart et arrivee sont non nuls

//------------------------------------------ Constructeurs - destructeur
    Catalogue ( );
    // Mode d'emploi : aucun
    // Contrat : aucun

    virtual ~Catalogue ( );
    // Mode d'emploi : aucun
    // Contrat : aucun

//------------------------------------------------------------------ PROTEGE
protected:
//---------------------------------------------------- Attributs protégés
    Liste* liste;

//-------------------------------------------------------------------- PRIVE
private:
//------------------------------------------------------ Methodes privées
    void rechercheRecursive (const char* depart, const char* arrivee,
                        unsigned int* tab,const unsigned int lengthTab,
                const unsigned int position) const;
    // Mode d'emploi : sous methode de RechercheParcoursAvancee qui
    //     recherche les compositions de trajets qui vont de depart à
    //     arrivee et affiche toutes les solutions. Tab est un tableau de
    //     même longueur que le nombre de trajets du catalogue. Il est
    //     remplit de 0 pour les trajets qui ne sont pas utilisés et de
    //     nombres strictement positifs représentant l'ordre
    //     d'utilisation pour les trajets utilisés. Position est alors la
    //     position qui va être occupée par le prochain trajet qui
    //     satisfait la recherche.
    // Contrat : depart, arrivee, tab et position sont non nuls et
    //     lengthTab est la longueur du tableau tab

};

#endif // Catalogue_H

/*************************************************************************
                                Catalogue
                            -------------------
    début                : 18/11/2017
    copyright            : (C) 2017 par Loïc CASTELLON & Florian MUTIN
    e-mail               : loic.castellon@insa-lyon.fr
                           florian.mutin@insa-lyon.fr
*************************************************************************/
//----- Réalisation de la classe <Catalogue> (fichier Catalogue.cpp) -----

//---------------------------------------------------------------- INCLUDE
//-------------------------------------------------------- Include système
using namespace std;
#include <iostream>
#include <cstring>

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

//-------------------------------------------------------------------- PUBLIC
void Catalogue::Affiche () const
// Algorithme : aucun
{
```

```
    cout<<"Catalogue de trajets"<<endl<<"{"<<endl;
    if(liste == nullptr)
    {
        cout << "Erreur : TrajetCompose vide"<<endl;
        return;
    }
    liste->Affiche("    ");
    cout<<"}"<<endl;
} //----- Fin de Affiche

bool Catalogue::Add (const Trajet* t) const
// Algorithme : aucun
{
    ElementListe* cur = liste->first;
    while(cur != nullptr)
    {
        if(cur->trajet->Equals(*t))
        {
            return false;
        }
        cur=cur->suivant;
    }
    liste->Add(t);
    return true;
} //----- Fin de Add

unsigned int Catalogue::RechercheParcours(const char* depart,
                                          const char* arrivee) const
// Algorithme : aucun
{
    unsigned int cpt = 0;
    cout<<"resultat :"<<endl;
    cout<<"{"<<endl;

    ElementListe* cur = liste->first;
    //parcours des trajets
    while(cur != nullptr)
    {
        //test sur le départ du trajet en cours
        bool ok = true;
        const char* departCur = cur->trajet->GetDepart();
        if(strlen(departCur) != strlen(depart))
            ok = false ;
        else
        {
            for(unsigned int i=0 ; i<strlen(departCur) ; i++)
            {
                if(departCur[i] != depart[i])
                    ok = false;
            }
        }

        //test sur l'arrivée du trajet en cours
        const char* arriveeCur = cur->trajet->GetArrivee();
        if(ok && strlen(arriveeCur) != strlen(arrivee))
            ok = false ;
        else
        {
            for(unsigned int i=0 ; i<strlen(arriveeCur) ; i++)
            {
                if(arriveeCur[i] != arrivee[i])
                    ok = false;
            }
        }

        //conclusion
        if(ok)
        {
            cpt++;
```

```
            cur->trajet->Affiche();
            cout<<endl;
        }
        cur = cur->suivant;
    }
    cout<<"{"<<endl;
    return cpt;
} //----- Fin de RechercheParcours

void Catalogue::RechercheParcoursAvancee(const char* depart,
                                         const char* arrivee) const
// Algorithme : aucun
{
    // initialisation
    unsigned int nbTrajet = 0;
    ElementListe* cur = liste->first;
    while(cur != nullptr)
    {
        ++nbTrajet;
        cur=cur->suivant;
    }
    unsigned int * tab = new unsigned int;
    for(unsigned int i = 0 ; i < nbTrajet ; i++)
    {
        tab[i] = 0;
    }

    //récursivité et affichage
    cout<<"resultat :"<<endl;
    cout<<"{"<<endl;
    rechercheRecursive(depart,arrivee,tab,nbTrajet,1);
    cout<<"{"<<endl;

    delete tab;
} //----- Fin de RechercheParcoursAvancee

//------------------------------------------ Constructeurs - destructeur
Catalogue::Catalogue ()
// Algorithme : aucun
{
#ifdef MAP
    cout << "Appel au constructeur de <Catalogue>" << endl;
#endif
    liste = new Liste();
} //----- Fin de Catalogue

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


//------------------------------------------------------------------ PRIVE
//------------------------------------------------------- Methodes privées
void Catalogue::rechercheRecursive(const char* depart,
                                   const char* arrivee, unsigned int* tab,
                                   const unsigned int lengthTab,
                                   const unsigned int position) const
// Algorithme : parcours en profondeur du catalogue de trajet
{
    //test fin de récursivité
    bool compFin=true;
    if(strlen(arrivee) != strlen(depart))
        compFin = false ;
    else
```

```
    {
        for(unsigned int i=0 ; i<strlen(arrivee) ; i++)
        {
            if(depart[i] != arrivee[i])
                compFin = false;
        }
    }
    if(compFin)
    {
        //affichage d'une solution
        cout<<endl<<"*"<<endl;
        for(unsigned int p = 1 ; p<position ; p++)
        {
            unsigned int numTrajet = 0;
            ElementListe* cur = liste->first;
            while(tab[numTrajet]!=p)
            {
                cur=cur->suivant;
                ++numTrajet;
            }
            cur->trajet->Affiche();
            cout<<endl;
        }
        cout<<endl;
        return;
    }

    //recursivité
    unsigned int numTrajet = 0;
    ElementListe* cur = liste->first;
    //parcours des trajets
    while(cur != nullptr)
    {
        //test sur le depart du trajet en cours
        bool compDepart = true;
        const char* departCur = cur->trajet->GetDepart();
        if(strlen(departCur) != strlen(depart))
            compDepart = false ;
        else
        {
            for(unsigned int i=0 ; i<strlen(departCur) ; i++)
            {
                if(departCur[i] != depart[i])
                    compDepart = false;
            }
        }

        //appel récursif si le depart convient et que le trajet n'est pas
        //utilisé
        if(compDepart && tab[numTrajet]==0)
        {
            tab[numTrajet]=position;
            rechercheRecursive(cur->trajet->GetArrivee(), arrivee, tab,
                                                lengthTab, position+1);
            tab[numTrajet]=0;
        }
        //on passe au trajet suivant
        ++numTrajet;
        cur = cur->suivant;
    }
} //----- Fin de rechercheRecursive
```