```cpp
/*************************************************************************
                          Collection  -  description
                             -------------------
    début                : 2015/10/09
    copyright            : (C) 2015 par cespeute & brenault
*************************************************************************/

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

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

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

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

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

//----------------------------------------------------- Méthodes publiques
void Collection::Afficher () const
{
#ifdef DEBUG
    printNbElements();
    printAlloue();
#endif
    unsigned int i;
    for (i = 0; i < nbElements; i++)
    {   cout << tableau[i];
        if (i < nbElements - 1)
        { cout << ","; // Afficher la virgule si on n'est pas au dernier élément
        }
    }
    cout << "\n\n";
}

void Collection::Ajouter (const int valeur)
{   // Couvre le cas ou la collection est déjà pleine.
    if (alloue == nbElements)
    {   Ajuster(alloue+1);
    }
    tableau[nbElements] = valeur;
    nbElements++;
}

void Collection::Retirer (const int valeur, const int occurencesNb)
{   // On remplace chaque occurence de la valeur donnée par la dernière valeur
    // du tableau et on décrémente nbElements pour signifier le retrait d'une
    // valeur. On réduit finalement la taille allouee du tableau en conséquence.
    int occurences = 0;
    for (unsigned int i = 0; i < nbElements; i++) {
        if (tableau[i] == valeur) {
            occurences++;
            if (occurences <= occurencesNb || occurencesNb < 0) {
                if (i != nbElements - 1) {
                    tableau[i] = tableau[nbElements - 1];
```

```
                }
                i--;
                nbElements--;
            }
        }
    }
    Ajuster(nbElements);
}

int Collection::Ajuster (const unsigned int uneTaille)
{
    // Erreur si la nouvelle taille est trop petite
    if (uneTaille < nbElements)
    {   return ERR_TAILLE;
    }
    // Recréer un tableau si sa taille est differente que celle actuelle
    else if (uneTaille != alloue)
    {   // Nouveau tableau qui sera alloué
        int *nouveauTableau = new int[uneTaille];
        // Copie de l'ancien tableau vers le nouveau
        for (unsigned int i = 0;i<nbElements;i++)
        {   nouveauTableau[i] = tableau[i];
        }

        delete tableau;
        tableau = nouveauTableau;
    }
    alloue = uneTaille;
    return PAS_ERR;
}

void Collection::Reunir (const Collection &uneCollection)
{
    // Cas où la collection courante n'est pas assez grande pour accueillir tous
    // les éléments de la seconde.
    if (alloue < nbElements + uneCollection.nbElements)
    {   Ajuster(nbElements + uneCollection.nbElements);
    }

    // Copie les valeurs de la seconde collection dans la courante
    unsigned int i;
    for (i = nbElements; i < alloue; i++)
    {   tableau[i] = uneCollection.tableau[i - nbElements];
    }

    nbElements += uneCollection.nbElements;
}

//------------------------------------------- Constructeurs - destructeur
Collection::Collection (const unsigned int uneTaille)
{
#ifdef MAP
    cout << "Appel au premier constructeur de <Collection>" << endl;
#endif

    alloue = uneTaille;
    nbElements = 0;
```

```
115      tableau = new int[uneTaille];
116  }
117
118  Collection::Collection (const unsigned int uneTaille, const int *unTableau)
119  {
120  #ifdef MAP
121      cout << "Appel au second constructeur de <Collection>" << endl;
122  #endif
123
124      alloue = uneTaille;
125      tableau = new int[uneTaille];
126
127      // Copie les éléments du tableau passé en paramètre un à un
128      unsigned int i;
129      for (i = 0; i < alloue; i++)
130      {   tableau[i] = unTableau[i];
131      }
132      nbElements = alloue;
133  }
134
135
136  Collection::~Collection ()
137  {
138  #ifdef MAP
139      cout << "Appel au destructeur de <Collection>" << endl;
140  #endif
141
142      delete tableau;
143  }
144
145  void Collection::printNbElements() const
146  {   cout << "NbElements : " << nbElements << '\n';
147  }
148
149  void Collection::printAlloue () const
150  {   cout << "Alloue : " << alloue << '\n';
151  }
152
```