```
1  /*************************************************************************
2                            Collection  -  description
3                              -------------------
4      début                : 2015/10/09
5      copyright            : (C) 2015/10/092015 par cespeute & brenault
6  *************************************************************************/
7
8  //---------- Interface de la classe <Collection> (fichier Collection.h) ------
9  #if ! defined ( COLLECTION_H )
10 #define COLLECTION_H
11 #define DEBUG
12 //#define MAP
13
14 //------------------------------------------------------------ Enumérations
15 enum CodesRetour
16 {   ERR_TAILLE  = 0,    // Erreur de retour si la nouvelle taille est
17                         //plus petite que le nombre d'éléments du tableau
18     PAS_ERR     = 1     // Si la fonction s'est correctement exécutée
19 };
20 // Enumération des codes de retour pour la fonction Ajuster
21
22 //-------------------------------------------------------------------------
23 // Rôle de la classe <Collection>
24 // Gère le stockage, la modification et l'affichage d'un ensemble de nombres
25 // entiers. Le stockage est réalisé dans un tableau d'entiers (int) géré de
26 // façon dynamique simple. Les redondances sont acceptées. Les valeurs ne sont
27 // pas triées.
28 //-------------------------------------------------------------------------
29 class Collection
30 {
31 //------------------------------------------------------------------ PUBLIC
32
33 public:
34 //----------------------------------------------------- Méthodes publiques
35     void Afficher () const;
36     // Mode d'emploi :
37     // Imprime dans un terminal chaque élément de la Collection un a un, dans l'
38     // ordre de leur apparition dans le tabelau, suivis par une virgule. Si
39     // DEBUG est déclaré, alors le nombre d'éléments et la taille allouée sont
40     // affichés avant.
41
42     void Ajouter (const int  valeur);
43     // Mode d'emploi :
44     // Ajoute la valeur entière donnée à l'extrêmité du tableau contenant la
45     // Collection et ajuste celle-ci si nécessaire.
46
47     void Retirer (const int valeur, const int occurencesNb);
48     // Mode d'emploi :
49     // Supprime au plus un nombre occurencesNb d'occurences de la valeur donnée
50     // en paramètre si elle est présente, et ajuste a chaque fois la quantité de
51     // mémoire utilisée. Si occurencesNb < 0, alors toutes les occurences sont
52     // supprimées. Chaque valeur supprimée est remplacée par une valeur en fin de
53     // la collection.
54
55     int Ajuster (const unsigned int uneTaille);
56     // Mode d'emploi :
57     // Réajuste le tableau pour que sa taille soit égale à uneTaille.
```

```
58      // Si la nouvelle taille est plus petite que le nombre d'éléments
59      // présents dans le tableau, l'ajustement n'a pas lieu et la fonction
60      // retourne ERR_TAILLE.
61      // Si l'ajustement du tableau s'est bien déroulé, la fonction renvoie
62      // PAS_ERR (Équivalent à vrai).
63
64      void Reunir (const Collection &Collection);
65      // Mode d'emploi : Ajoute les éléments de la collection donnée après ceux de
66      // la courante en réajustant la taille du tableau de la courante si
67      // nécessaire.
68
69  //------------------------------------------- Constructeurs - destructeur
70      Collection (const unsigned int uneTaille);
71      // Mode d'emploi : Créé une collection pouvant accueillir uneTaille éléments
72      // sans avoir besoin d'être redimensionnée. Affiche "Appel au premier
73      // constructeur de <Collection>" si MAP est définie.
74
75      Collection (const unsigned int uneTaille, const int *unTableau);
76      // Mode d'emploi : Crée une collection pouvant accueillir uneTaille éléments
77      // sans avoir besoin d'etre redimensionnée, et lui ajoute chacun des
78      // elements du tableau donné. Affiche "Appel au second constructeur de
79      // <Collection>" si MAP est définie.
80      // Contrat : On assume que la taille donnée en paramètre est égale à la
81      // taille réelle du tableau donné.
82
83      virtual ~Collection ();
84      // Mode d'emploi : Détruit la collection courante. Affiche "Appel au
85      // destructeur de <Collection>" si MAP est définie.
86
87  //------------------------------------------------------------------ PRIVE
88
89  private:
90  //------------------------------------------------------ Attributs privés
91      unsigned int nbElements;
92      unsigned int alloue;
93      int *tableau;
94
95      /////// CLASSES NON DEMANDEES MAIS UTILES POUR LE TEST ////////////
96      void printNbElements () const;
97      // Affiche le nombre d'éléments de la Collection
98
99      void printAlloue () const;
100     //Affiche la taille allouée de la Collection
101 };
102
103 #endif // COLLECTION_H
104
```