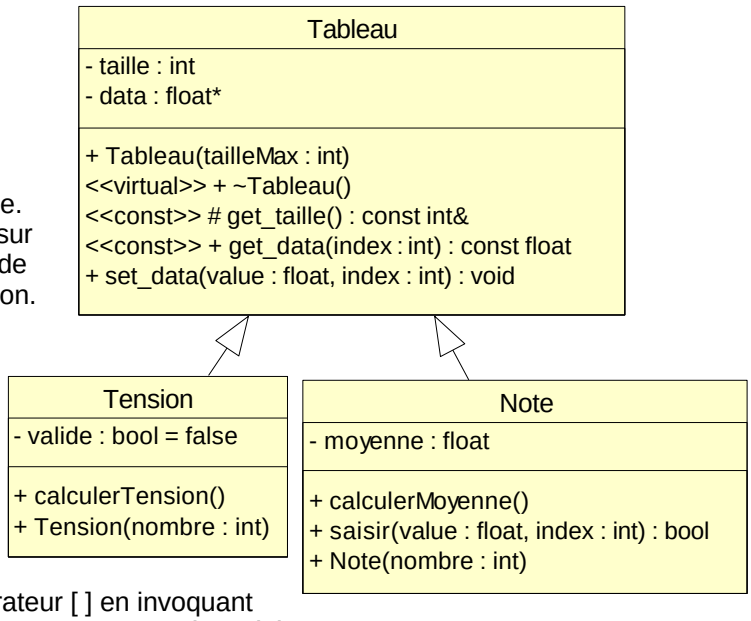
# TD : Héritage simple pour constituer une hiérarchie de classes

## *Sujet*

Soit le modèle de la hiérarchie de classes «Tableau» présenté ci-contre.




Un tableau de flottants est créé de manière dynamique lors de l'exécution du programme. Cette technique permet de créé un tableau sur mesure à l'exécution du programme et non de fixer sa taille lors de l'opération de compilation.

Ce tableau est représenté dans la classe par la donnée membre ***data*** qui est un pointeur sur un type float qui pointe sur le premier objet du tableau.
La donnée membre *tailleMax* représente la taille du tableau

Dans le programme, on peut accéder à la valeur d'un élément du tableau par l'opérateur [ ] en invoquant l'index de l'élément : data[0] premier élément, data[1] deuxième élément, …

Le constructeur, grâce à l'instruction *new*, est chargé de créer dynamique le tableau de flottants. L'instruction produit un pointeur (une adresse) qui est affecté à la donnée membre *data*.
Principe de programmation : Tout objet créé dynamiquement doit être détruit.
Donc, le destructeur détruit le tableau grâce à l'instruction *delete[].*

**La classe Note**

Elle spécialise la classe *Tableau* pour stocker des notes.

- *moyenne* : représente la valeur moyenne des notes contenues dans le tableau qui est renseignée par *calculerMoyenne()* :
- *saisir() :* affecte la valeur de la note (value) à Tableau : :*data[index]* si la valeur appartient à l'intervalle 0 - 20 et retourne la valeur *true*. (Si la valeur n'appartient pas à l'intervalle, elle n'est pas affectée et la valeur de retour est *false*)

**La classe Tension**

Elle spécialise la classe *Tableau* pour stocker des valeurs de tension.
Principe d'utilisation :

- Le tableau est renseigné au départ par des valeurs de ddp.
- La fonction *calculerTension()* est appelée. Elle parcourt tout le tableau pour rendre chaque valeur positive et met *valide* à *true*. (tension = | ddp|)

## *Travail demandé*

**Utilisation de la classe Note**

1. On dispose de 4 notes en informatique : 16, 8, 12 et 11. Proposez un extrait de un programme qui crée une instance de la classe *Note* nommée *informatique*, appelle 4 fois la méthode saisie() pour renseigner le tableau et calcule la moyenne.

**La classe Note**

2. Etablissez la déclaration de la classe.
3. Etablissez le corps du constructeur.
4. Etablissez le corps de la fonction membre *saisie().*

**La classe Tension**

5. Etablissez la déclaration de la classe.
6. Etablissez le corps du constructeur. ( Il initialise la donnée membre *valide* à *false* et chaque élément du tableau avec la valeur '0'.
7. Etablissez le corps de la fonction membre *calculerTension().*

**Reflexion :**

Il n'apparaît pas explicitement de destructeur des classes *Tension* et *Note*. Le programmeur C++ n'implémente pas de destructeur pour ces classes lorsqu'il écrit les programmes. Des instances de ces classes sont crées lors de l'exécution d'un programme. On quitte ce programme.

8. Comment sont détruits les tableaux créés dynamiquement ?

On souhaite que le programmeur utilisateur des classes *Note* et *Tension* dispose de 2 fonctions : *size()* - retourne le nombre d'éléments stockés dans le tableau et *capacity()* - retourne la taille du tableau.

9. Proposez dans la hiérarchie de classes les modifications nécessaires pour mettre à disposition du programmeur ces 2 fonctions ?
10. Etablissez le corps des 2 fonctions.
11. Comparer la qualité des services offerts par la nouvelle hiérarchie de classes avec celle de la hiérarchie de classes d'origine. Quel est l'intérêt de ces 2 fonctions.

## *Extraits de la programmation de la classe Tableau*

```
#ifndef  _TABLEAU_H
#define _TABLEAU_H
   1. class Tableau {
   2. public:
   3.       Tableau(int tailleMax);
   4.       virtual ~Tableau();
   5.       const float get_data(int index) const;
   6.       void set_data(float value, int index);
   7. protected:
   8.       const int& get_taille() const;
   9. private:
   10.           int taille;
   11.           float* data;
   12.     };
#endif  // _TABLEAU_H
```

```
1. #include "Tableau.h"
2.
3. Tableau::Tableau(int tailleMax) {
4.    data = new float[tailleMax];
5. }
6.
7. Tableau::~Tableau() {
8.    delete[] data;
9. }
10.
11.   const float Tableau::get_data(int index) const {
12.         return data[index];
13.   }
14.
15.   void Tableau::set_data(float value, int index) {
16.         data[index] = value;
17.         return;
18.   }
19.
20.   const int& Tableau::get_taille() const {
21.         return taille;
22.   }
```