Université de Paris Cité **Langages à objets avancés**
M1 **TP n° 6**

Cet exercice traite des difficultés que vous pouvez rencontrer dans le cas de définitions de classes croisées (l'ordre de leur définition), pose la question de l'usage de pointeurs ou de références, s'assure de la destruction des objets construits, illustre les amitiés de classes.

Exercice 1 On souhaite modéliser un scrutin pour des élections. Pour un scrutin donné, on gère plusieurs bureaux de vote (avec dans chacun une urne). Le nombre d'options de vote possibles change à chaque scrutin : par exemple, pour un référendum, il y a 3 options interprétées par exemple comme "oui", "non" et "vote nul ou blanc".

Commencez à déclarer une classe `Scrutin` qui encapsulera le nombre de bureaux de vote (et donc d'urnes), le nombre d'options de vote et un tableau de pointeurs sur des urnes.

Débutez aussi la déclaration d'une classe `Urne` qui contiendra une référence sur un `Scrutin`, un entier représentant le numéro du bureau de vote (utilisez un compteur « static »pour qu'il soit attribué automatiquement) et un tableau d'entier comptabilisant les votes pour chaque option.

De plus, prévoyez dans `Urne` une méthode `bool voter(int choix)`, qui retournera `false` si l'option est impossible. Vous ajouterez rapidement des méthodes qui vous permettrons de faire des vérifications et tests : obtenir les résultats d'un bureau de vote, celui du scrutin entier et afficher ces résultats.

Indication : Nous sommes dans un cas où les dépendances sont croisées : une urne contient une référence à un scrutin qui contient un tableau d'urnes. Vous ne pouvez pas vous contenter de mettre `#include "Scrutin.hpp"` dans le fichier Urne.hpp, et `#include "Urne.hpp"` dans le fichier Scrutin.hpp,

Pour résoudre le problème, il vous faut faire une déclaration préalable. Par exemple dans Urne.hpp déclarez `class Scrutin;` ou vice versa (voir cours).

Exercice 2 Écrivez les destructeurs des classes `Urne` et `Scrutin`. À la fin d'un scrutin, les urnes seront détruite. Vérifiez avec des sorties écran appropriées que les destructions attendues ont bien lieu.

Exercice 3 Pour éviter qu'on puisse fabriquer des urnes et les rattacher à un scrutin indûment, nous allons rendre les constructeurs et destructeurs d'`Urne` privées. Pour que `Scrutin` puisse construire des Urnes et les détruire, nous allons déclarer la classe `Scrutin` amie de `Urne`.

Exercice 4 Ajoutez des `const` partout où c'est possible : attributs, méthodes, ...

Exercice 5 Comment éviter que l'on puisse voter après la fin du scrutin et que l'on puisse afficher les résultats avant la fin du scrutin ? Proposez un mécanisme, et développez en les conséquences sur les autres opérations.

Exercice 6 Modifiez votre travail pour utiliser un type énuméré au lieu d'entiers pour modéliser les choix. (On renoncera au fait que les options de vote changent à chaque scrutin)