Université de Paris Cité **Langages à objets avancés**
M1 **[dépendances croisées]**

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.

On souhaite modéliser un scrutin pour des élections. (Un scrutin c'est la phase du vote qui se déroule sur une journée). Pour un scrutin donné on doit gérer plusieurs urnes disposées dans des bureaux de vote. En supposant qu'il n'y a qu'une urne par bureau de vote nous pourrons confondre ces 2 notions.

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" ; pour le premier tour d'une élection présidentielle il y a autant de choix que ce candidats etc ...

1. Commencez par déclarer une classe `Scrutin` qui encapsulera un vecteur de chaînes de caractères dénotant les choix possibles. Ce vecteur sera initialisé à la construction et restera invariant. Munissez la d'une méthode `void afficheChoix()` dont le rendu sera de la forme `1 - Oui, 2 - Non` etc . . .Modifie t'elle le scrutin considéré ? Faites en sorte que le destructeur affiche un message. Ecrivez un `Main.cpp`, un `Makefile` et faites un premier test.
2. Définissez à présent une classe pour les urnes en vous contentant du strict minimum suivant : les objets de cette classe encapsulent un numéro qui permet de les identifier ; ce numéro est automatiquement affecté lors de l'appel à un constructeur sans argument. Faites en sorte que le destructeur affiche un message.
3. En réalité, les déclarations des urnes et des scrutins sont liées et nous demandent un peu d'attention. Nous souhaitons stocker des objets urnes dans les scrutins à l'aide d'un vecteur d'objets (de sorte que ce soit une composition et pas d'une agrégation). Et nous souhaitons aussi que les urnes aient connaissance du scrutin auquel elles participent. Proposez une solution (sans pointeurs) ; modifier le constructeur de scrutin pour qu'il prenne aussi en compte le nombre d'urnes concernées et qu'il les construise [1] ; modifiez aussi naturellement le constructeur d'urne. Assurez vous que votre `Makefile` est bien écrit. Ajoutez une méthode `afficheUrnes()` qui permette de voir les identités des urnes qui composent un scrutin. Testez [2]
4. Introduisez dans `Urne` une méthode `bool voter(int choix)`, qui retournera `false` si l'option est impossible. Les urnes conserveront le décompte des votes.
5. Pour tester la méthode précédente, se pose la question des droits à accorder pour accéder aux données encapsulées par un scrutin. Pourquoi est-il hors de question d'écrire un getter qui retourne le vecteur d'urne ? On suggère d'écrire un getter qui retourne une urne lorsqu'on dispose de son id. Quel est son type retour ?

1. il vous faudra comparer `push_back` et `emplace_back`
2. si vous n'avez pas pensé à utiliser `reserve` pour vos vecteurs, vous devriez avoir des copies parasites qui apparaissent lorsque le vecteur est étendu ...

6. Vous ajouterez quelques lignes qui vous permettront de procéder à plusieurs votes de manière interactive. Vérifiez les résultats localement ainsi que le résultat du scrutin en entier.
7. Si dans `main` vous disposez de deux urnes `u1` et `u2`, que se passe t'il si vous faites l'affectation `u1=u2` ?
8. La copie est elle possible ?
9. On pourrait imaginer une forme de fraude où quelqu'un présenterait une "fausse" urne à un votant qu'il voudrait tromper : ce dernier penserait avoir voté, mais si l' urne est distincte de celle du scrutin alors son vote ne serait pas pris en compte... Procéder aux modification de sécurité qui permettent de s'en prémunir.
10. Si vous avez le temps, vous pouvez probablement réécrire la partie interactive en raisonnant avec des exceptions plutot que des tests.
11. Vous pouvez aussi imaginer un mécanisme pour qu'on ne puisse pas voter lorsqu'un scrutin est clos, ni qu'on puisse consulter les résultats avant la fin.