Université Paris Diderot **Langages à objets avancés**
M1 **Année 2019-2020**

TP n° 12 : Design Pattern

Dans ce TP, on va reprendre le thème des élections de l'exercice 4 du TP 5, mais en utilisant une partie des motifs (patterns) vus en cours.

Exercice 1 Les élections sont contrôlées par une commission électorale. Cette commission est unique, on utilisera donc le motif Singleton pour la créer. Pour l'instant, cette classe n'aura pas d'attribut, on les ajoutera petit à petit. Regardez le cours, pour voir comment mettre en œuvre ce motif.

Écrivez cette classe et vérifiez que la méthode qui retourne un pointeur sur une instance de commission électorale, retourne bien toujours le même objet quelque soit le nombre d'appels. (Affichez l'adresse contenue dans le pointeur).

Exercice 2

1. Nous reprenons en partie ce qui a été fait dans le TP5 pour les urnes (qui se confondent plus ou moins avec les bureaux de vote).

   À savoir : elles contiennent un tableau d'entiers comptabilisant les votes pour chaque option ; c'est la commission électorale qui connaît le nombre d'options disponibles. Une méthode `bool voter(int choix)` qui retournera `true` pour l'instant et vous ajouterez les méthodes nécessaires pour pouvoir obtenir les résultats d'un bureau de vote et d'afficher ces résultats.

   Cependant, on veut pouvoir contrôler la manière dont sont créées et détruites les urnes.

   On va donc faire une Fabrique en utilisant "factory method". La fabrique aura une méthode qui fabriquera une urne numérotée par un nombre fourni en argument, avec un nombre d'options lui aussi donné en argument, on retournera un pointeur sur cette urne.

   Pensez à l'amitié pour donner accès au constructeur et destructeur de Urne. La copie d'Urne sera bien sûre interdite ainsi que l'affectation.

2. Une fois vos tests effectués, on rendra la méthode de fabrication privée ; elle ne sera plus utilisable que par le commission électorale via l'amitié. La commission aura une méthode `lanceFabricationUrnes(int nbUrnes)` qui fera fabriquer le nombre d'urnes prévues, le nombre d'options étant un attribut de `CommissionElectorale`. Il faudra aussi prévoir une méthode qui met à jour ce nombre d'options. Prévoyez les méthodes d'affichage nécessaires pour pouvoir tester que ces méthode marchent.

Exercice 3

1. On suppose que la commission électorale ne s'occuppe que d'un Scrutin à la fois. On veut savoir dans quelle phase du scrutin on est, on va définir une enum `phaseScrutin` ayant quatre valeurs :
   — `prepare` : (le nombre d'options est décidé et les urnes sont construites),
   — `enCours` : le vote a commencé mais n'est pas fini.

— `fini` : le vote est fini, mais les urnes ne sont pas encore détruites, car on veut pouvoir calculer les résultats.
— `neant` : on a "effacé" les données du scrutin précédent, urnes,..

On va dans Commission électorale, faire des méthodes qui permettent de passer d'une étape à l'autre, à chaque fois, si la phase actuelle ne permet pas ce passage, on envoie une exception.

— `preparerScrutin(int nbUrnes, int nbOptions)` qui met à jour le nombre d'options et crée les urnes.
— `debuterVote()` : se réduit au changement de phase.
— `finirVote()` : se réduit au changement de phase pour l'instant.
— `effacerScrutin()` : on détruit ou réinitialise les attributs.

Faites en sorte de pouvoir saisir des votes : il faut pouvoir récupérer l'urne numéro n et voter. La méthode voter de Urne, vérifiera qu'on est bien dans la bonne phase du scrutin. Testez.

2. Une fois le vote fini, il faut dépouiller.

   On décide de confier cette opération à une classe `Depouillement` via le motif "visitor". `Depouillement` contiendra un tableau de resultats qui contient le résultat cumulé des urnes déjà dépouillées. (À chaque vote, on crée un nouvel objet Depouillement.)

   La classe `Depouillement` aura une méthode `depouille(Urne*)` (la "visite" de Urne). C'est dans la classe Commission électorale au moment de la fin du vote qu'on "obligera" chaque urne à accepter le dépouillement. Une méthode `getResultats` privée dans `Depouillement` permettra à une méthode `afficheResultats` de la commission électorale de fonctionner.

Exercice 4

1. Regardez votre code : quelles méthodes doivent être privées ? Quelles amitiés sont nécessaires ? Quelles méthodes n'ont servi que pour debugguer et peuvent être éliminées ? Qu'est-ce qui peut (ou doit) être constant ?

   Faut-il ajouter des mécanisme de contrôle pour éviter que certaines choses soit faites au mauvais moment ? (On part de l'idée que la commission électorale ne fait pas de bêtise dans ses méthodes).

2. Pour aller plus loin, on peut remarquer qu'il serait sans doute logique de dissocier la commission électorale du scrutin proprement dit pour permettre par exemple d'avoir plusieurs élections en même temps.