

UE Conception Orientée Objet

Scrutins électoraux

On s'intéresse à la programmation de différents modes de scrutins électoraux.

Un scrutin consiste pour un électeur à *exprimer* son vote par un bulletin. Pour chaque scrutin, une liste de bulletins de votes possibles est proposée. Mais un électeur a en fait la possibilité de voter ce qu'il veut : il peut donc créer ses propres bulletins. Cependant lors de l'établissement des résultats du scrutin, les bulletins qui ne correspondent pas à l'un des votes possibles sont considérés comme des votes *nuls*. En plus des votes possibles un électeur a toujours à sa disposition un vote particulier : le vote *blanc*, qui permet d'exprimer que l'on refuse de choisir parmi les bulletins de votes possibles proposés. On peut en fait considérer que ce vote correspond au choix d'un bulletin de vote supplémentaire particulier, appelé blanc. De la même manière les bulletins *nuls* peuvent être considérés comme correspondant à un bulletin de vote particulier appelé nul.

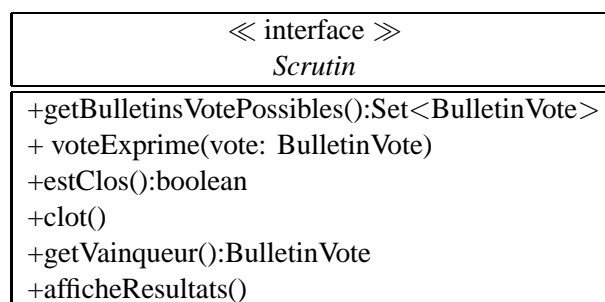
Il est possible de voter tant qu'un scrutin n'est pas clos. Une fois que celui-ci est clos, et pas avant, il est possible de déterminer le résultat de ce scrutin. La phase de dépouillement consiste alors à comptabiliser pour chaque bulletin (y compris les bulletins blanc et nul) le nombre de fois où il a été choisi.

Le calcul du résultat, et notamment du vainqueur, dépend du type de scrutin. Dans ce sujet 2 modes de scrutin seront considérés :

- le scrutin à la majorité des suffrages exprimés (type `ScrutinMajoritaire`) : dans ce cas est déclaré vainqueur le vote qui aura obtenu plus de 50% des suffrages autres que les bulletins blancs ou nuls.
- le scrutin à majorité relative (type `ScrutinRelatif`) : est déclaré vainqueur le vote (autre que les bulletins blancs ou nuls) qui a reçu le plus de suffrages pour peu qu'au moins 15% des inscrits au scrutin aient exprimé ce vote.

Il peut bien sûr n'y avoir aucun vainqueur, dans chacun de ces cas.

Un scrutin est défini par l'interface suivante du paquetage `scrutin` :



- La méthode `getBulletinsVotePossibles` retourne l'ensemble des bulletins de vote possibles pour ce scrutin.
- La méthode `voteExprime(vote)` permet d'indiquer qu'un électeur a choisi le le bulletin de vote `v`. Son bulletin vient donc s'ajoute aux votes exprimés pour le scrutin (on ne s'occupe pas ici des électeurs ni du fait que chacun ne peut voter qu'une fois, ceci est supposé géré par ailleurs).

Tant que le scrutin n'est pas clos, cette méthode accepte tous les bulletins de votes, qu'ils fassent ou non partie des votes "possibles". Elle lève une exception `ScrutinClosException` si le scrutin est clos et n'a alors aucun effet.

- Les méthodes `getVainqueur()` et `afficheResultats()` lèvent une exception `ScrutinNonClosException` si le scrutin n'est pas clos. Elles n'ont alors aucun autre effet.

On suppose les classes d'exception créées et définies dans le paquetage `scrutin`.

- La méthode `clot` clôt le scrutin **et** établit le résultat du scrutin (phase de dépouillement du scrutin). Les résultats sont mémorisés sous la forme d'une **table de hachage** associant à un objet `BulletinVote` le nombre de suffrages qu'il a reçus.
Rappelons que si l'un des votes exprimés ne fait pas partie des bulletins de votes possibles pour le scrutin, il est alors considéré comme un vote nul.
- La méthode `afficheResultats` affiche pour chaque bulletin de vote possible, ainsi que les bulletins nul et blanc, le nombre de suffrage reçus, une fois le scrutin clos.
- Le résultat de `getVainqueur` est `null` s'il n'y a aucun vainqueur.

La suite de ce sujet s'intéresse à divers aspects d'une telle application. Toutes les classes que vous définirez devront appartenir au paquetage `scrutin`.

Votes

Q 1. Donnez le diagramme UML puis le code d'un type `BulletinVote`.

Un bulletin de vote est défini par le texte (ou nom) représentant ce vote, évidemment non modifiable. Deux votes portant le même texte sont naturellement égaux. Pensez à prévoir qu'il faut gérer la table des résultats (voir description de la méthode `clot` ci-dessus). La méthode `toString` doit également être définie.

On définit également 4 constantes correspondant à des bulletins de votes particuliers : les votes `oui`, `non`, `blanc` et `nul`.

Scrutins

Q 2. Donnez un diagramme UML puis le code de la classe `ScrutinMajoritaire`, sachant que le nombre d'inscrits (électeurs) et l'ensemble des bulletins (ou votes) possibles sont communiqués à la création d'une instance. Aux bulletins possibles doivent systématiquement être ajoutés les bulletins `blanc` et `nul`.

Q 3. En supposant que le reste de la classe reste tout à fait similaire à `ScrutinMajoritaire`, donnez pour la classe `ScrutinRelatif` le code nécessaire à la méthode `getVainqueur`.

Q 4. Créez une classe contenant comme seule méthode une méthode `main` qui :

1. crée un référendum, qui est un *scrutin majoritaire* dans lequel les bulletins de votes possibles sont les bulletins `oui` et `non`, on fixera le nombre de votants à 110,
2. ajoute 100 bulletins de vote choisis aléatoirement entre `oui` et `non`,
3. clôt le scrutin et annonce le vainqueur.