Rapport Projet C++

« Elections piège à cons ? »

EI-SE4

Année universitaire 2016-2017

BERNARD Alexandre

BALLET Alexis

Table des matières

1 - Présentation du projet & description de l'application

Dans le cadre de notre module de C++, nous devions réaliser une application avec comme thème :
« Elections, piège à cons ?».
Les contraintes à respecter sont :
— 8 classes minimum
— 3 niveaux de hiérarchie minimum
— 2 fonctions virtuelles différentes et utilisées à bon escient
— 2 surcharges d'opérateurs
— 2 conteneurs différents de la STL
— diagramme de classe UML complet
— commentaire du code
— pas d'erreurs avec Valgrind
— pas de méthodes/fonctions de plus de 30 lignes
— utilisation d'un Makefile avec une règle "all" et une règle "clean"

Nous avons décidez de réaliser un petit jeu textuel où le but est de se faire élire avec son parti contre deux AI qui représentent les partis opposés. Le joueur choisi son parti et participe à une suite d'évènements qui lui permette de gagner en fonds ou en popularité. Il faut atteindre une popularité de 100 avant les AIs.

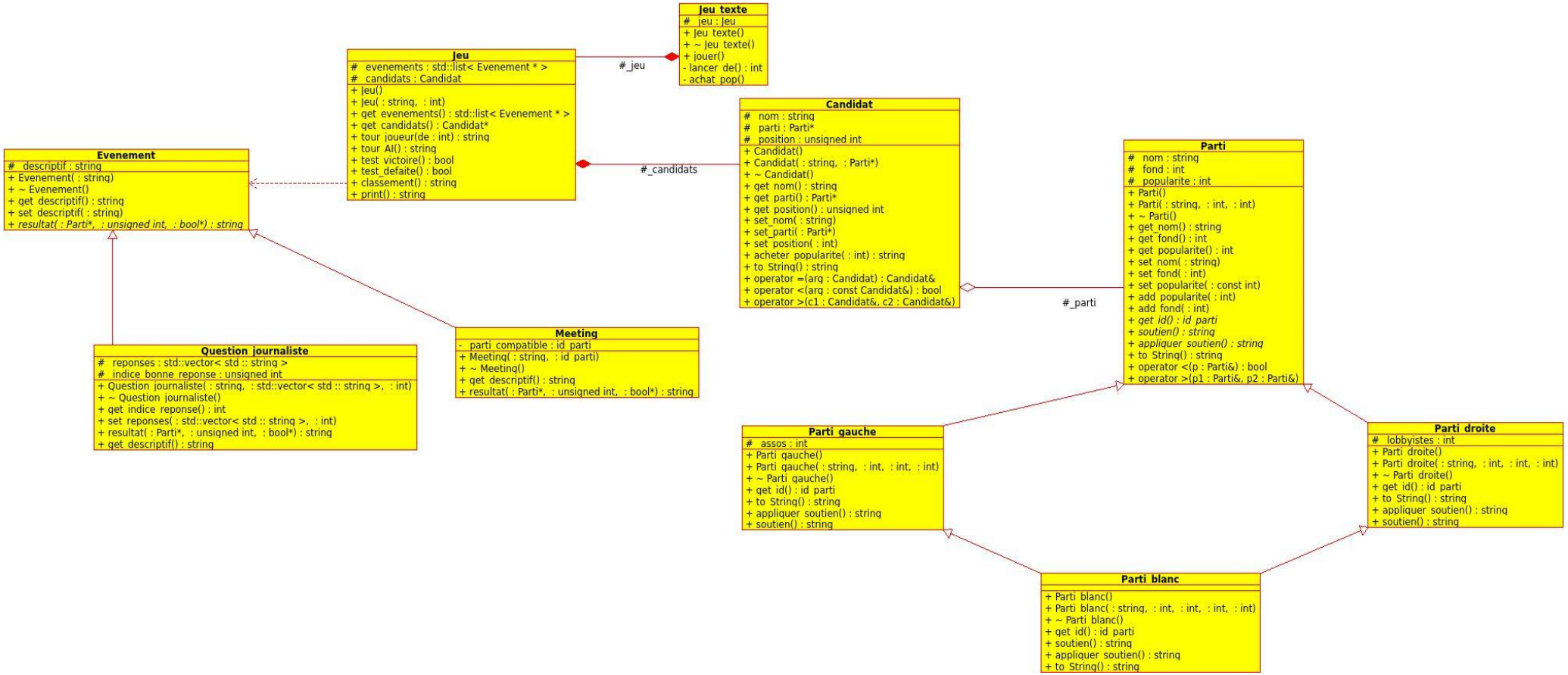
Il y a trois partis : « les Bleus », « Les Blancs » et « Les Rouges ». Dans cette version, les évènements sont des meetings ou des questions à choix multiples. Chaque meeting est rattaché à un parti, et permet pour « Les Bleus » de gagner un lobbyiste. Les lobbyistes permettent de gagner des fonds à chaque tour et ainsi permettre aux candidats de transformer ces fonds en popularité, en achetant tout simplement de la popularité. Pour « Les Rouges », les meetings permettent de gagner le soutien d'une association. Les associations permettent de gagner de la popularité à chaque tour. « Les Blancs » peuvent gagner des lobbyistes et des associations.

Les questions à choix multiples permettent de gagner directement de la popularité si vous y répondez correctement. Ces questions sont basées sur les meilleures boulettes des politiques. Enfaite, notre jeu a pour but principal de faire découvrir ces boulettes de manière interactive. La victoire est plus ou moins aléatoire par intention, et les bonnes réponses sont souvent complètement fausses en réalité. De plus, vous pouvez à tout moment acheter de la popularité, et donc être élu rapidement.

L'esprit de ce jeu nous ait venu des récents évènements politiques, qui nous laissent l'amer impression que dire assez d'âneries à la télévision et avoir assez d'argent, suffisent pour être élu.

Pour compiler, rendez-vous dans le répertoire et utiliser la commande « make ». « make test » vous permet de lancer le jeux. Vous pouvez aussi lancer l'exécutable depuis le répertoire avec « ./test1.out »

2 - Diagramme UML



3 - Difficultés rencontrées

L'utilisation de container avec des classes abstraites nous à poser quelques soucis. En effet, vu que nous ne pouvons pas instancier de classe abstraite, et que même sur des classes non abstraites, les classes filles n'ont pas toutes les mêmes membres, nous ne pouvons pas utiliser de container sans passer par des pointeurs, car nous risquerions de « couper » nos objets (la classe mère n'ayant pas les membres de la classe filles, ceux-ci serait « couper » en quelque sorte). Utiliser des containers de classes abstraites nous a aussi fait réfléchir à nos méthodes virtuelles pour pouvoir utiliser les éléments de ces containers de manière standardisé.

De plus, un problème qui concernait plus particulièrement Alexandre nous a beaucoup ralentis; Il a eu beaucoup de problème pour compiler du C++ sur son ordinateur personnel, et cela nous a beaucoup ralentis, surtout pendant les vacances car il n'avait pas accès aux ordinateurs de l'école. Pour résumé le problème, du code qui compiler parfaitement sur d'autres machines ne compiler pas du tout sur la sienne (peu importe le standard C++ utilisé). Mêmes sur machine virtuelle, impossible. Nous avons téléchargé plusieurs IDE avec les extensions, et même réinstaller le compilateur, mais rien à faire. Cela l'a forcé a coder « à l'aveugle » la majeur partie du temps ; Cela est vraiment faisable que pour quelques parties simple du code, et donc nous avons perdu beaucoup de temps.

4 - Parties de l'implémentation dont nous sommes fiers

Nous nous sommes mis comme objectif de rendre le code le plus modulable et granulaire possible, afin de travailler chacun de notre côté de la manière la plus efficace possible. Cela permet également de passer facilement d'une version texte à une version graphique. Malheureusement, nous n'avons pas eu le temps nécessaire pour développer la partie graphique.

Nous avons donc essayé de nous servir à bon escient de méthodes virtuelles, nous avons aussi dû faire attention à ne pas faire de cin<< autre part que dans la classe Jeu_texte, ce qui a demandé un peu de réflexion pour contourner d'éventuels problèmes. Par exemple, la fonction résultat d'Evenement nécessite une entrée de l'utilisateur, et cette entrée peut être erronée, et dans ce cas il faut redemander à l'utilisateur sa réponse. Nous avons donc décidé de faire resultat() retourner un booléen passé en argument par adresse, en plus d'une chaine de caractère décrivant le résultat, pour pouvoir gérer plus facilement les entrées erronées depuis Jeu_texte.