Projet APO : Modélisation de systèmes électoraux et dynamiques de vote

Table des matières

[Introduction 2](#_Toc94472542)

[Faire fonctionner le programme : 2](#_Toc94472543)

[Comment compiler le projet : 2](#_Toc94472544)

[Compiler en ligne de commande : 2](#_Toc94472545)

[Compiler avec ant : 2](#_Toc94472546)

[Comment executer le programme : 2](#_Toc94472547)

[Comment générer la javadoc : 2](#_Toc94472548)

[Avec ant (https://ant.apache.org/) : 2](#_Toc94472549)

[Sans ant : 2](#_Toc94472550)

[Déroulement du programme 3](#_Toc94472551)

[Choix de l’action à réaliser 3](#_Toc94472552)

[Affichage des candidats 3](#_Toc94472553)

[Election 3](#_Toc94472554)

[Interactions socio politiques 4](#_Toc94472555)

[Sondages 5](#_Toc94472556)

[Méthodologie 6](#_Toc94472557)

[Conception du programme 6](#_Toc94472558)

[Diagramme de cas d’utilisation : 7](#_Toc94472559)

[Description des cas d’utilisation : 7](#_Toc94472560)

[Diagramme des classes : 7](#_Toc94472561)

[Extensions 8](#_Toc94472562)

# Introduction

Dans ce projet, nous avons modélisé et analysé différentes manières de voter. Nous avons également implémenté des interactions politiques telles que les sondages ou l’interaction entre deux personnes.

Nous avons modélisé différents les scrutins suivants afin d’analyser les différences entre les systèmes électoraux :

1. Scrutin majoritaire à un tour
2. Scrutin majoritaire à deux tours
3. Vote par approbation
4. Vote alternatif
5. Méthode de Borda

Notre programme permet de simuler des élections en utilisant une liste d’électeurs et une liste de candidats.

Chaque personne possède une opinion sur 6 axes politiques différents comme la Justice, le Capitalisme ou encore le Réformisme. En plus de la simulation d’élection, notre programme permet de faire des interactions socio-politiques entre personnes et permet également de faire des sondages pour faire varier les opinions des électeurs.

# Faire fonctionner le programme :

## Comment compiler le projet :

Pour compiler le projet, il faut avoir java JDK, qui fournit le programme javac pour compiler le code.

### Compiler en ligne de commande :

Depuis un terminal :

$> javac -d bin src/Personnes/\*.java src/Scrutin/\*.java src/Dynamique/\*.java src/\*.java

-Copier le fichier config.properties dans le dossier bin

### Compiler avec ant :

-Installer ant (https://ant.apache.org/)

-Depuis un terminal : $> ant

## Comment executer le programme :

Pour exécuter le programme, il faut déjà l'avoir compilé. Ensuite, depuis un terminal :

$> java -cp bin Main

## Comment générer la javadoc :

### Avec ant (https://ant.apache.org/) :

Depuis un terminal : $> ant documentation

### Sans ant :

Depuis un terminal à la racine du projet :

$>javadoc -d doc -sourcepath src/\*.java src/Personnes/\*.java src/Scrutin/\*.java src/Dynamique/\*.java

# Déroulement du programme

Lors de l’exécution du programme, les paramètres de la simulation (nombre de candidats et nombre d’électeurs) sont définis en lisant le fichier « config.properties ». On peut donc changer les paramètres de la simulation en modifiant les valeurs dans ce document.

## Choix de l’action à réaliser

A l’exécution du programme, on arrive sur une boite de dialogue permettant de choisir l’action à réaliser. A la fin de chaque action, l’utilisateur peut fermer le programme ou bien revenir à ce menu.

Une image contenant texte

Description générée automatiquement

Figure - Choix Action

### Affichage des candidats

Le programme affiche les informations des candidats de la simulation :

Une image contenant texte

Description générée automatiquement

Figure - Affichage des candidats

### Election

L’utilisateur peut ensuite choisir le mode de scrutin de l’élection :

Une image contenant texte

Description générée automatiquement

Figure - Choix du mode de scrutin

Après que l’utilisateur a choisi son mode de scrutin, le résultat est affiché :

Une image contenant texte

Description générée automatiquement

Figure - Sortie d'une élection

### Interactions socio politiques

L’utilisateur peut choisir un nombre entre 1 et 5. Il s’agit du nombre de fois que chaque électeur va rencontrer une autre personne pour changer ses opinions

Une image contenant texte, plaque

Description générée automatiquement

Figure - Exemple d'interactions socio politiques

### Sondages

Cette partie va créer une élection avec un nombre d’électeurs réduit (ici, la moitié de l’électorat total). Une fois sur la catégorie de sondage, l’utilisateur choisit l’effet du sondage. Un sondage va modifier les opinions des électeurs selon cet effet.

Une image contenant texte

Description générée automatiquement

Figure - Choix de l'effet du sondage

Une fois l’effet choisi, l’utilisateur choisit le type d’élection simulé par le sondage.

Une image contenant texte

Description générée automatiquement

Figure - Choix du type d'élection pour le sondage

On voit ensuite les résultats du sondage ainsi que l’indication stipulant que les opinions de tous les électeurs ont bien été modifiés.

Une image contenant texte

Description générée automatiquement

Figure - Résultats du sondage

# Méthodologie

Tout d’abord, nous avons réfléchi sur la manière d’organiser le code et de concevoir notre application. Nous avons commencé par faire le diagramme des cas d’utilisations et une première ébauche du diagramme des classes. Grâce à ces travaux, nous avions une idée de comment réaliser ce projet et avons pu commencer à coder.

Nous avons créé le dossier du projet sur la forge de Lyon 1 pour permettre un meilleur partage du code. *Lien du projet git :* https://forge.univ-lyon1.fr/p1907453/simulationelection

Nous n’avons pas utilisé d’IDE pour coder ce projet. Nous trouvions plus intéressant le fait de comprendre comment fonctionnait la compilation manuelle etc… Ainsi, nous avons commencé par installer ANT et créer le fichier de build « build.xml » qui est devenu l’outil utilisé pour compiler le code.

Nous avons d’abord codé le package Personnes, qui comprenait les informations des candidats et des électeurs, ainsi que la définition des axes politiques. Ensuite, nous avons créé le fichier de configuration et la classe ReadConfig qui permet d’en extraire les informations, à savoir le nombre de candidats et d’électeurs de la simulation. Nous avons ensuite implémenté les différentes élections avec le package Scrutin, puis avons fini avec le package Dynamique comprenant les interactions socio politiques et les sondages. Tout au long du développement de l’application, nous avons maintenu la classe de simulation pour permettre de tester nos méthodes et le bon fonctionnement de l’application.

Nous avons réalisé la plupart des tâches ensemble à l’aide de discord et du fait que nous soyons colocataires. Ainsi, nous avons tous les deux contribué à toutes les parties du projet.

# Conception du programme

Nous avons réalisé deux diagrammes qui illustrent les choix de conception et le fonctionnement de notre programme :

## Diagramme de cas d’utilisation :

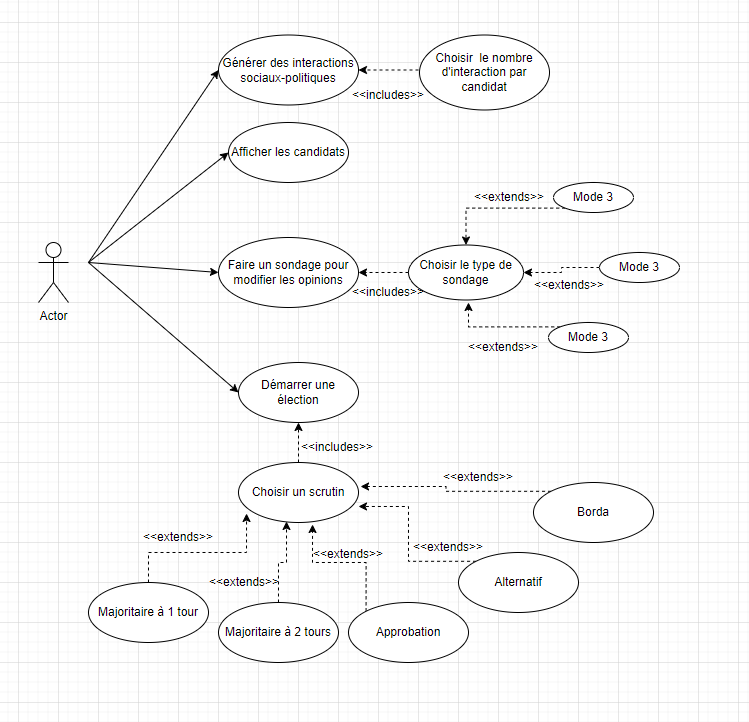


Figure - Diagramme des cas d'utilisations

### Description des cas d’utilisation :

-Générer des interactions socio-politiques : Les opinions des électeurs se changent en fonction de leurs interactions sociales.

Afficher les candidats : Affichage des informations des candidats

Faire un sondage : Effectuer un sondage en choisissant le type de sondage et le type d’élection du sondage.

Démarrer une élection : On va lancer une élection

Choisir un scrutin : On choisit le scrutin sur lequel l’élection va se baser.

## Diagramme des classes :

# Extensions

Comme expliqué dans les parties méthodologie et déroulement du programme, nous avons implémenté trois des extensions possibles.

-Nous avons créé un projet git que nous avons maintenu à jour : <https://forge.univ-lyon1.fr/p1907453/simulationelection>

-Nous avons créé le fichier build.xml interprété par ANT. Ce dernier permet de nettoyer les fichiers de compilation, de compiler le code et de générer la javadoc du projet.

-Nous avons créé un fichier de configuration « config.properties », qui contient les paramètres de la simulation à savoir les nombres de candidats et d’électeurs. La manière dont le code est organisé rend facile l’amélioration de ce fichier de configuration. Le fichier étant un ensemble de paires de chaines de caractère (key :value), il est facile d’enrichir le fichier de configuration pour avoir des candidats avec des noms et des idées politiques réalistes.

# Conclusion

Ce projet aura été très enrichissant car il s’agissait de notre premier projet en JAVA. Le résultat nous donne envie de continuer à faire d’autres projet de la sorte. Certains points ont été particulièrement intéressants à programmer comme le vote alternatif ou le troisième type de sondage.

Il est facile d’imaginer comment on pourrait améliorer ce programme, notamment en réalisant les autres extensions telles que l’interface graphique, la sauvegarde ou l’analyse des scrutins. Bien que le projet soit maintenant rendu, il n’est pas exclu que nous continuions à travailler dessus car il nous a beaucoup plu.