INFO 3A : APO

Compte rendu du TP final

INFO 3A

30/01/2022

LIEVRE Hugo

BEAL Clara

SOMMAIRE :

I)	Introduction	p.3
II)	Simulation	p.3
III)	Conception	p.6
IV)	Méthodologie	p.10
V)	Extensions	p.11
VI)	Conclusion	p.11

I) Introduction :

Pour le projet final en Algorithmique Programmation Objet, nous avons dû modéliser et analyser différents systèmes de votes et certaines dynamiques « politiques » avec des modèle très simplifiés.

Pour ce faire, nous avons tout d'abord utilisé différents modèles afin de simuler les systèmes de vote d'une élection de manière théorique. Il s'agit des scrutins :

> Majoritaire à un tour (MUT) :
> Majoritaire à deux tours (MDT)
> Vote par approbation (Approbation)
> Vote alternatif (Alternatif)
> Méthode de Borda (Borda)

Ici, chaque personne possède une représentation, classée en 2 axes. Ces axes permettent de représenter les préférences sur différents axes « sociétaux ». Nous pourrons ainsi mesurer l'appétence entre un électeur et un candidat par une distance entre leurs axes respectifs.

Pour se rapprocher d'un modèle qui pourrait ressembler à la réalité, nous avons simulé des sondages et des interactions entre personnes. Ces évènements peuvent modifier l'opinion des électeurs. Pour cela, nous avons ajouté différents types de sondages, qui ont des impacts différents sur l'électeur.

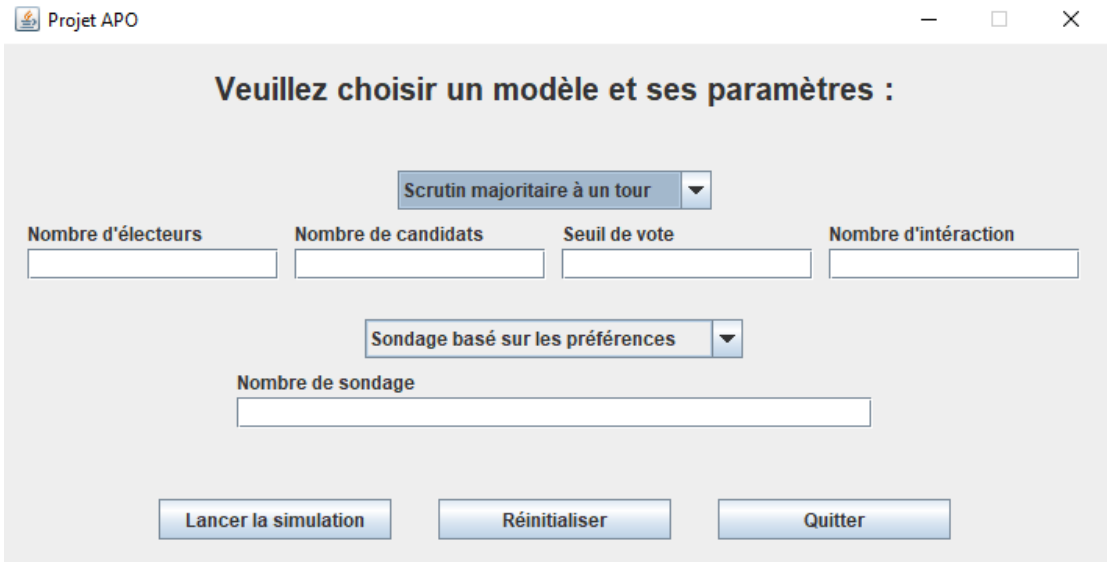
II) Simulation

a) Présentation générale

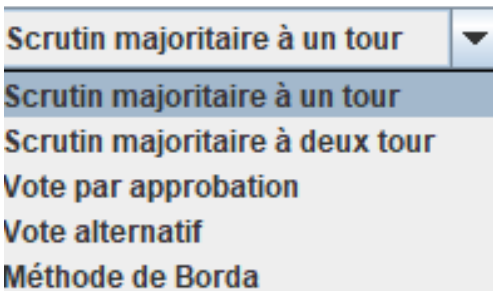
IMPORTANT : Pour éviter tout problème, pour chaque simulation, il est préférable de relancer le programme !

Afin de permettre à l'utilisateur de simuler rapidement une élection, sans avoir besoin de connaissances en JAVA, nous avons créé une interface où nous pouvons rentrer les différents paramètres.

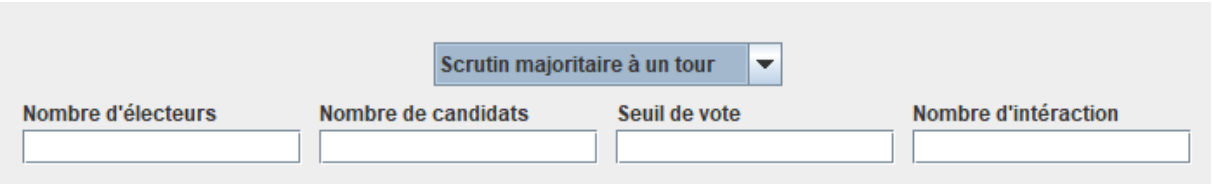
Voici à quoi ressemble l'interface lorsque l'utilisateur lance le programme.



Tout d'abord, il est possible de choisir le modèle voulu :



L'utilisateur peut ensuite rentrer les différents paramètres en fonction du modèle choisi :



Affichage choix « Scrutin majoritaire à un tour »

Affichage choix « Vote alternatif »

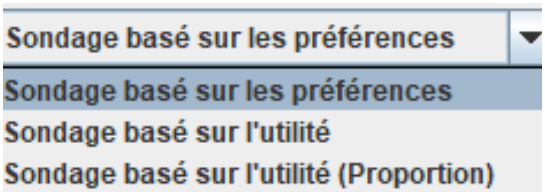
Dans les cases « Nombre d'électeur », il faut rentrer le nombre de personnes désiré (nombre entier). C'est-à-dire le nombre de personnes qui vont participer à l'élection.

Dans les cases « Nombre de candidats », il faut rentrer le nombre de personnes désiré (nombre entier). C'est-à-dire le nombre de personnes qui vont participer à l'élection en tant que candidat.

Pour certains modèles, vous aurez une case « Seuil de vote ». Il s'agit tout simplement, d'un seuil de distance. Si un électeur a une distance trop élevée, et donc supérieure au seuil, avec l'ensemble des candidats, alors il votera blanc (en fonction du mode de scrutin). A l'inverse, plus cette distance est faible, plus l'électeur sera susceptible de voter pour ce candidat.

La case « Nombre d'interaction » permet de choisir le nombre d'interaction que va avoir un électeur. Nous appelons interaction la rencontre entre un électeur et une personne, électeur ou candidat, tirée au hasard. Si l'électeur et la personne ont une distance inférieure au seuil donné, alors l'électeur verra ses axes se rapprocher de la personne tirée aléatoirement.

Pour chaque type de scrutin, il est aussi possible pour l'utilisateur de choisir le type de sondage voulu. Pour ce faire, nous tirons au hasard des électeurs parmi la totalité des électeurs, et nous regardons leur vote.



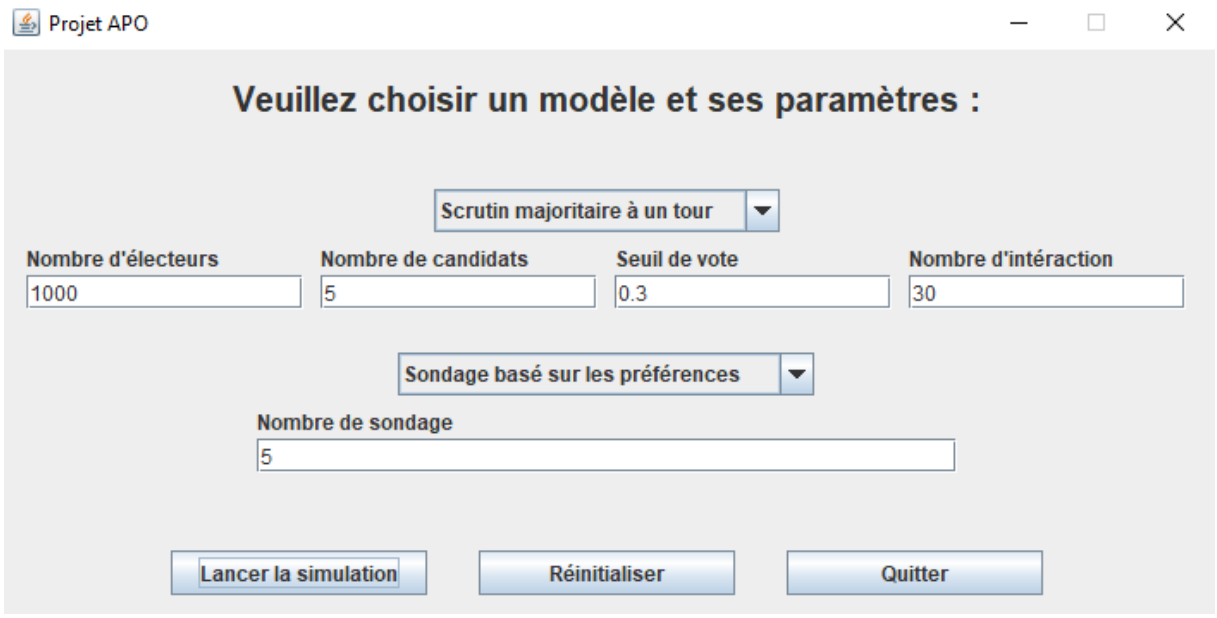
L'utilisateur a le choix entre 3 types de sondage :

- ➔ Préférence : L'électeur va voir sa distance se réduire avec le candidat se rapprochant le plus de ses préférences, parmi les n premiers du sondage.
- ➔ Utilité : Ici, l'électeur va calculer une utilité pour chaque candidat. Pour cela, nous allons multiplier la préférence de l'électeur (inverse de la distance au candidat) par le pourcentage de voix du candidat dans le sondage. L'électeur se rapprochera ainsi du candidat ayant l'utilité la plus élevée.
- ➔ Utilité (Proportion) : C'est le même principe que pour le sondage basé sur l'utilité. A la différence qu'ici, on rapproche l'électeur de manière proportionnelle.

L'interface dispose aussi d'une case « Nombre de sondage » où l'utilisateur doit saisir un nombre entier. Il s'agit du nombre de fois où le sondage sera réalisé.

b) Exemple et affichage

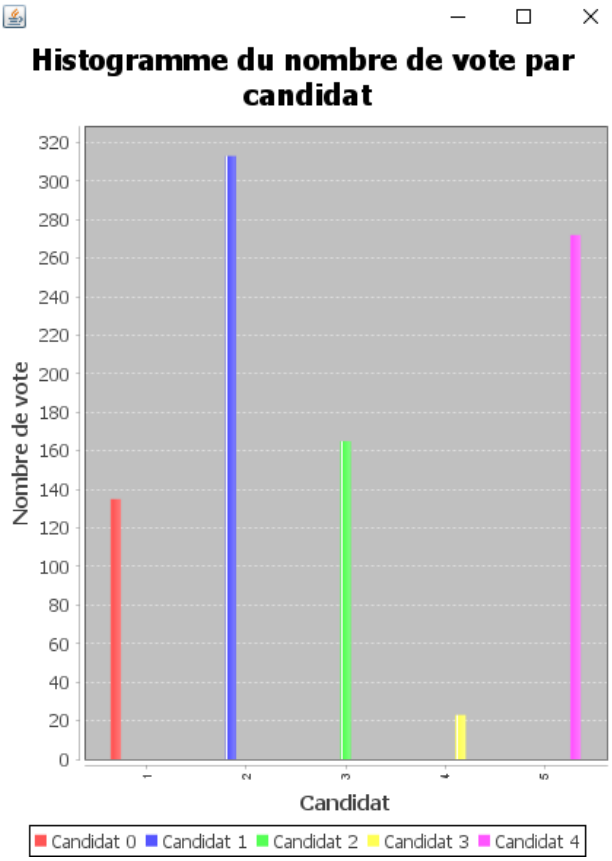
Exemple de simulation :



Après avoir appuyé sur « Lancer la simulation », nous obtenons un histogramme (ou plusieurs selon le modèle choisi), tracé grâce à la librairie JFreeChart qui sera fournie dans les docs (fichier .jar).

Il faut penser à l'ajouter pour que le code fonctionne !

Nous obtenons alors ceci :




```
Résultats : 2 5 3 1 4
Le gagnant est : Candidat 2
```

```
Nombre de voix :
Abstention : 92
Candidat 1 :135
Candidat 2 :313
Candidat 3 :165
Candidat 4 :23
Candidat 5 :272
```

Cet histogramme représente le modèle « scrutin majoritaire à un tour » avec les différents paramètres rentrés précédemment dans l'interface. Si le modèle choisit est « scrutin majoritaire à deux tours » ou « vote alternatif », vous obtiendrez plusieurs histogrammes, représentant chacun un tour de l'élection.

Nous obtenons aussi un affichage textuel, où nous pouvons voir les résultats de l'élection. Ici, le gagnant est le candidat numéro 2, qui a obtenu 313 votes. Ensuite, le deuxième est le candidat 5 avec 272 votes, etc.

Notre simulation dispose aussi d'un affichage pour les sondages ou les interactions. Celui-ci va, en fonction du sondage/interaction souhaité, faire avec un listing, avant et après, de chaque électeur avec son ID, sa représentation et son vote, qui sera vide s'il n'a pas encore voté.

Voici un exemple :

Avant le sondage

```
Electeur AVANT SONDAGE

Id : Electeur 1
Représentation : [0.79,0.18000000000000002]
```

Résultats du sondage

```
Résultat vote sondage : [9, 4, 2]
Classement du sondage : 1 2 3
```

Après le sondage

```
Electeur APRES SONDAGE

Id : Electeur 1
Représentation : [0.78,0.19000000000000003]
```

Nous pouvons alors voir que la représentation de l'électeur a été modifiée.

III) Conception

Diagramme de cas d'utilisation :

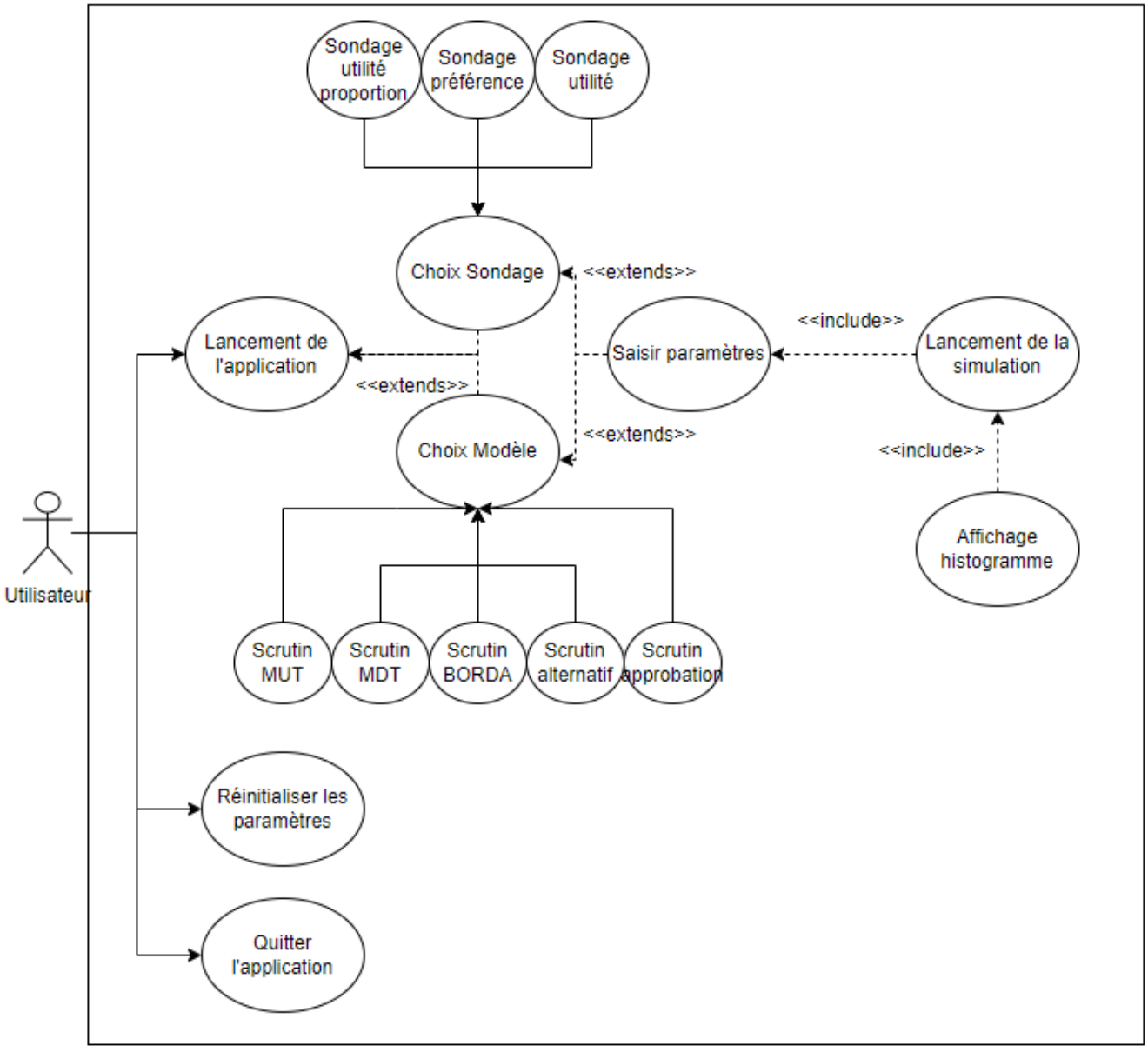


Diagramme de classes :

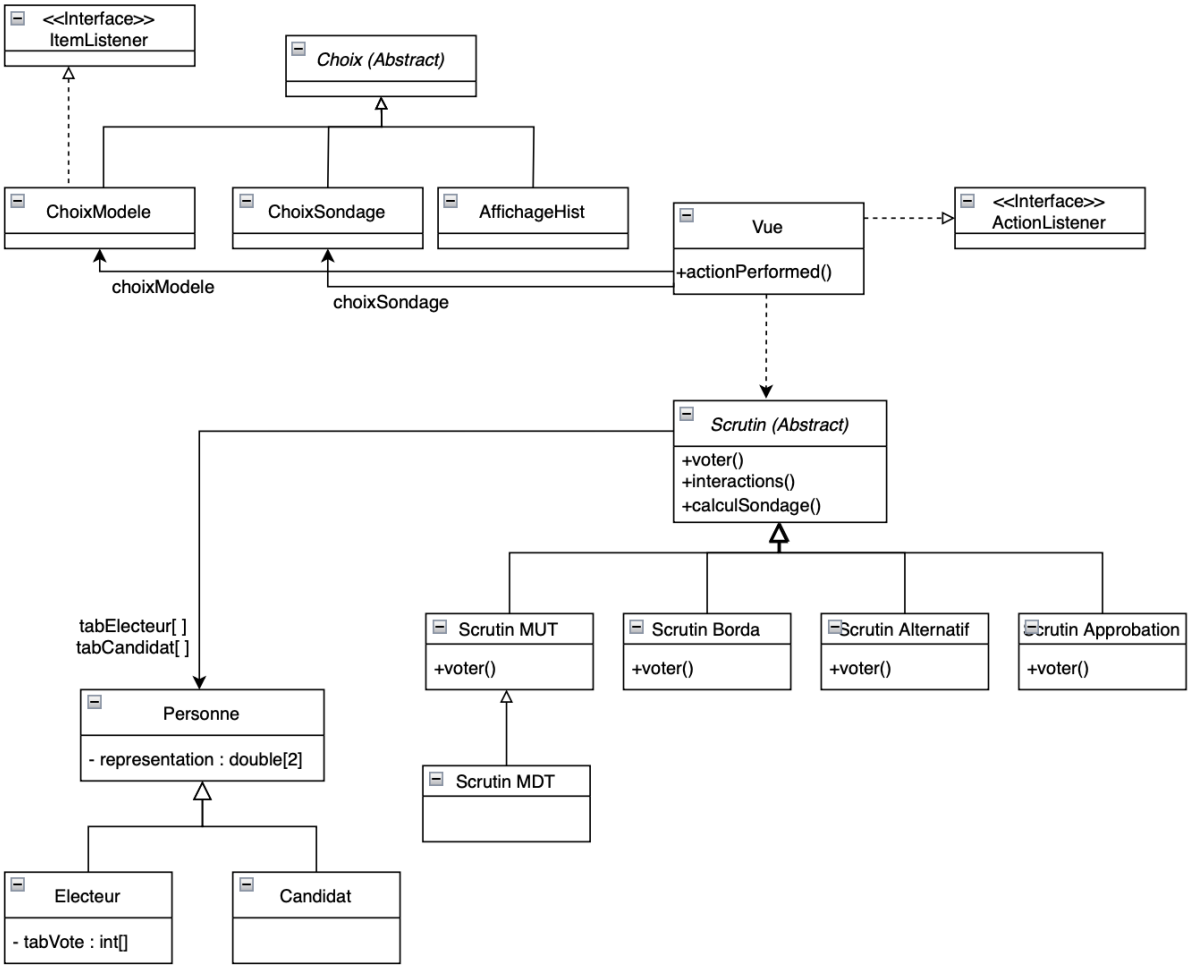


Diagramme d'activité :

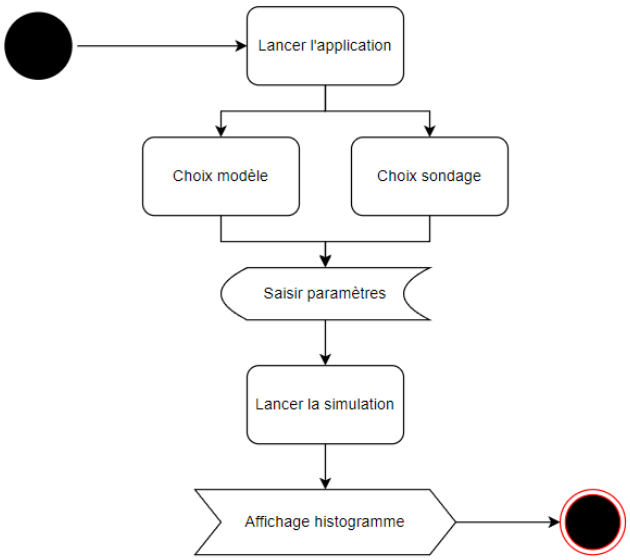
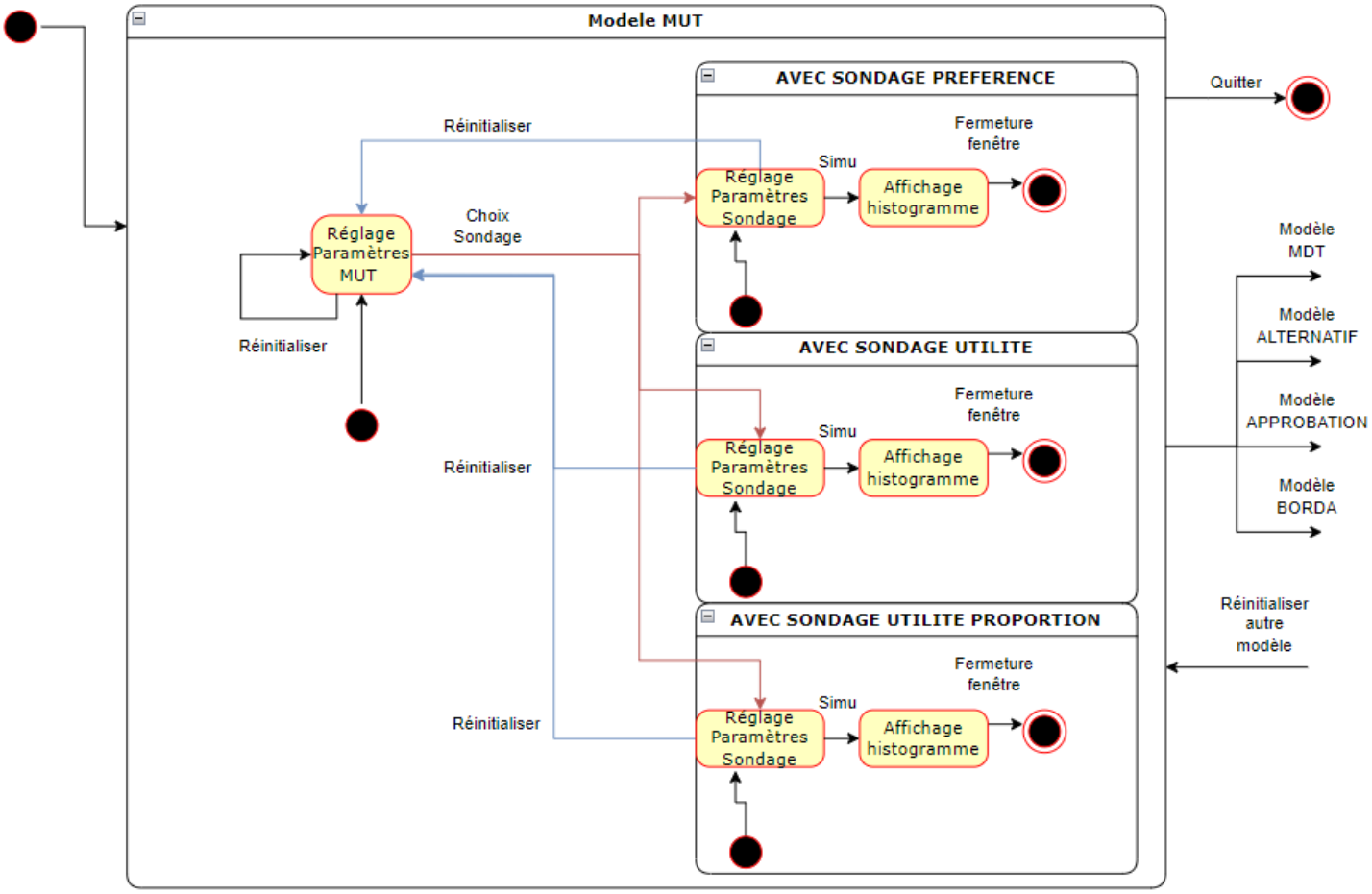


Diagramme états-transitions :



Simu = Lancement de la simulation

Pour éviter les répétitions, nous avons seulement modéliser le diagramme d'états transitions du Modèle MUT (Majoritaire à Un Tour). En effet, chaque modèle aura la même représentation que celle-ci-dessus. La seule modification est que si l'on est dans un autre modèle et que l'utilisateur choisit de réinitialiser ses valeurs, alors, on retourne au modèle MUT.

Il est aussi possible pour l'utilisateur de changer de modèle à tout moment.

IV) Méthodologie :

Nous avons commencé par faire différents diagrammes tel que le diagramme de cas d'utilisation, de classes, qui nous ont permis de développer nos premières idées concernant ce projet. Nous avons ainsi commencé à coder avec des idées claires et précises sur le modèle à réaliser. Nous avons ensuite réalisé un diagramme d'état transition et d'activité.

Nous avons essayé d'organiser notre projet de manière modulaire avec les packages : Personne, Scrutin et Affichage. Nous aurions pu créer un package Interaction qui aurait contenu les interactions et les sondages mais il était plus simple pour nous de l'inclure directement dans la classe Scrutin dans laquelle nous avions accès à tous les paramètres.

Ainsi, nous avons tout d'abord codé les différentes modèles/types de scrutin. Puis une fois que ceux-ci marchaient, nous avons directement codé les interactions et les sondages.

Concernant le projet et l'affichage, nous pensions faire une interface textuelle via la console. Cependant, le code est vite devenu lourd et nous trouvions que cette interface n'était pas adaptée pour un utilisateur ayant peu d'expérience en java. C'est pourquoi nous avons décidé de créer un GUI à la place, bien plus facile à utiliser pour nous dans l'avancement du projet, mais aussi pour l'utilisateur qui doit uniquement exécuter le code.

Pour tracer les histogrammes, nous avons utilisé la librairie JFreeChart.

Répartition des tâches :

Clara : Simulation des personnes, modèles et interactions/sondage, javadoc, rapport, GIT, correction code et mise au propre

Hugo : GUI, simulation des modèles et sondages, affichage courbe JFreeChart, rapport, fichier CSV

Bien que nous nous soyons divisé les tâches, nous avons globalement essayé de travailler à deux. En effet, étant donné que nous avons réalisé chacun des scrutins et des sondages, nous mettions régulièrement en commun sur les méthodes que nous utilisions ce qui pouvait donner des idées à l'autre personne, et ce qui permettait donc de gagner en efficacité.

V) Extensions :

5.1 Environnement de travail :

Voici le lien vers le GIT :
https://forge.univ-lyon1.fr/p1907112/projetapo

5.5 Sauvegarde :

Grâce à la méthode getCSV(), il est possible de récupérer les valeurs de la simulation dans un fichier CSV qui peut être utiliser sur Excel. Ce fichier se crée s'il n'est pas déjà créé dans le dossier du projet, ou écrasera les données s'il y en a déjà un du même nom.

5.6 Interface graphique :

Ici, nous avons séparé les classes d'affichages des classes du modèle/scrutin. Pour afficher des courbes lors de la simulation, nous avons utilisé la librairie JFreeChart. Vous retrouverez les fichiers .jar, dans le dossier du compte-rendu, à ajouter à votre bibliothèque pour pouvoir exécuter le programme.

VI) Conclusion :

La réalisation de ce projet a été pour nous l'occasion d'utiliser les notions que nous avons appris tout au long du semestre en les mettant en pratique dans le cadre de ce projet. Certains points ont été plus durs à réaliser que d'autres, nous n'avons donc pas eu le temps de réaliser plus d'extensions mais aussi de régler les quelques petits défauts de notre programme.