Institut des sciences du Digital Management & Cognition
INSTITUT UNIVERSITAIRE DU NUMÉRIQUE

# Rapport
# Projet Baignoire

Marcus Richier

2 Juin 2024

suivant les cours de Azim Roussalany

Sources du projet :
https://github.com/Nimarcs/projet-POO

# Table des matières

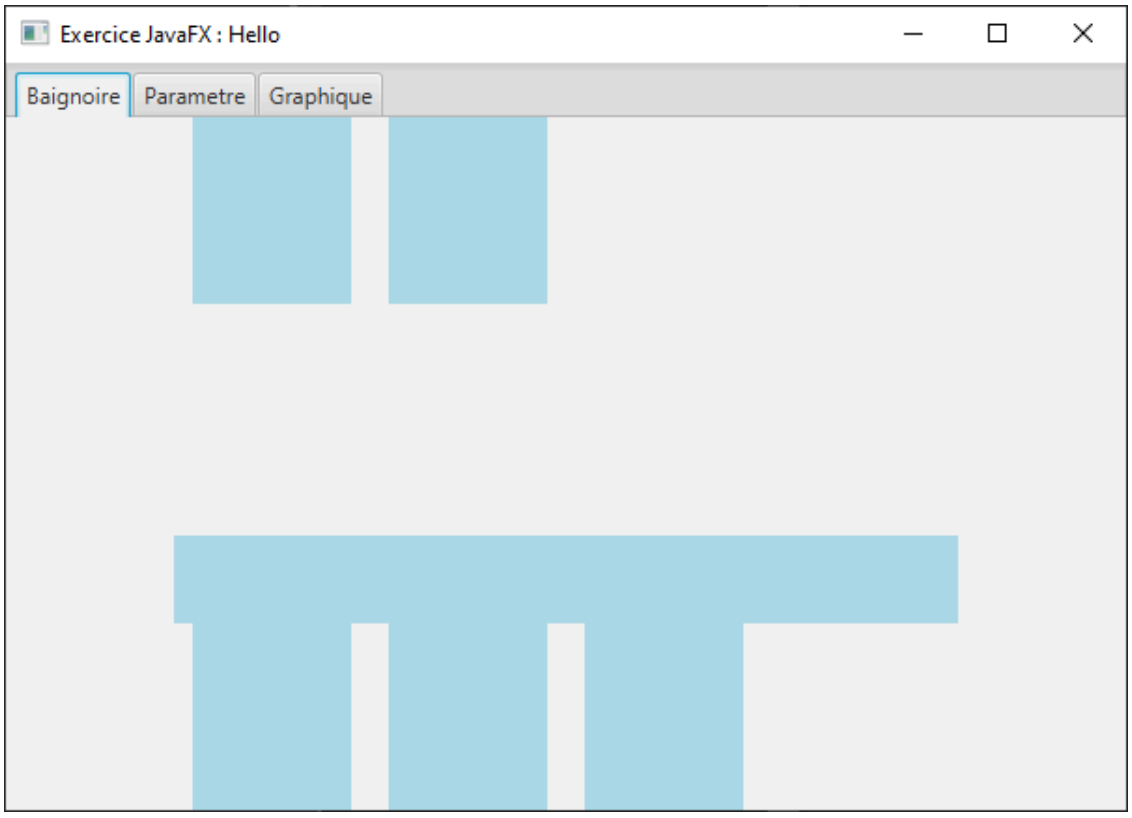


Note :

Une version de toute les images présentés sur ce rapport est disponible dans la partie /ressources du projet

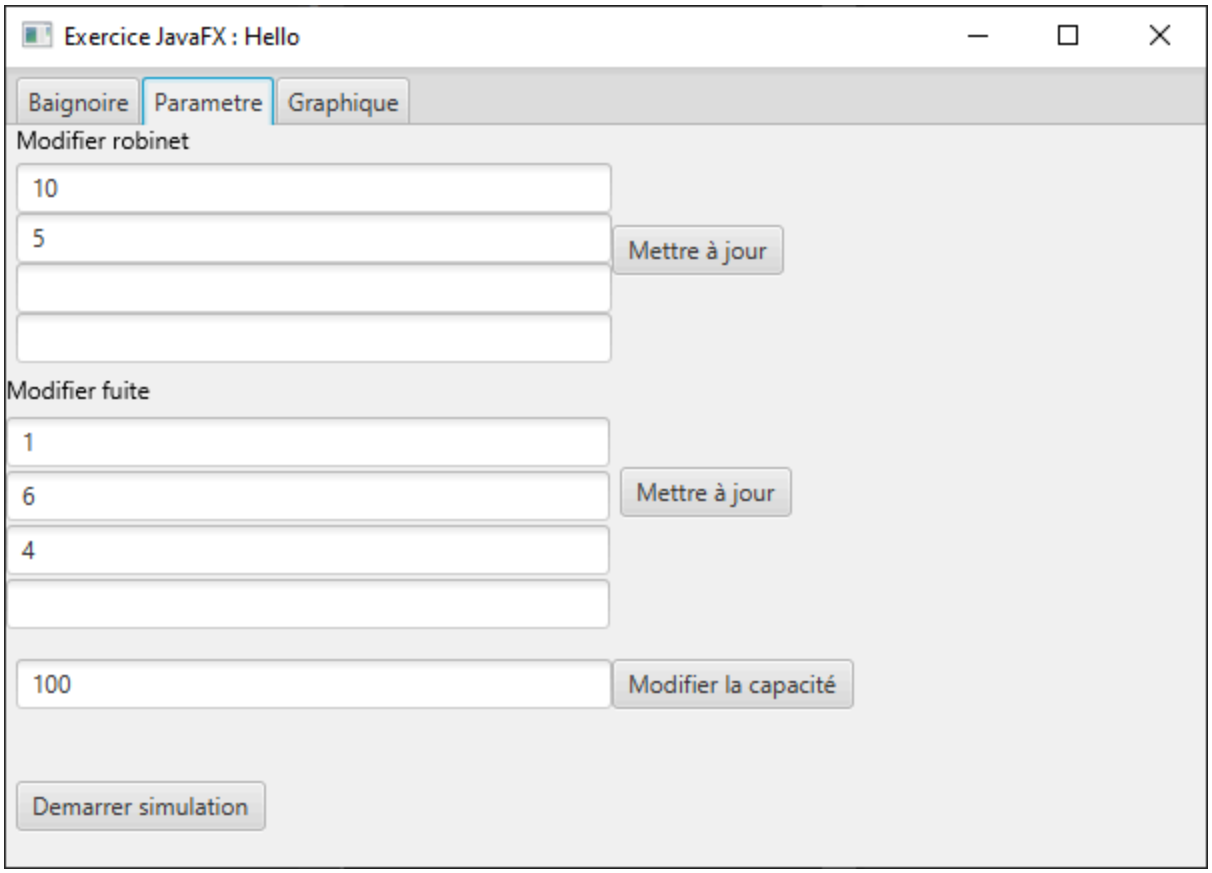
## Description du projet

Ce projet est un projet scolaire qui permet d'expérimenter avec Maven et javafx.
L'idée est de représenter une baignoire avec des fuites et des robinets qui la remplissent
On définit un nombre de robinets et de fuite < 4 puis on lance la simulation
On peut changer le débit de chaque robinet et supprimer certaines fuites
Lorsque la baignoire est remplie, on arrête la simulation et affiche le temps pris.

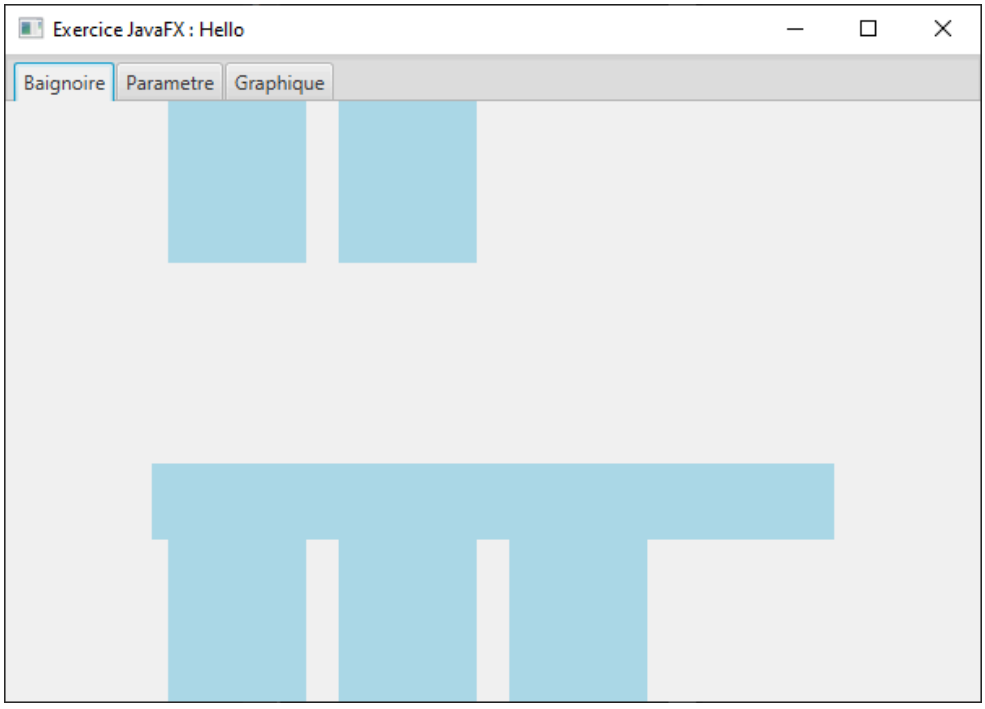


## Les fonctionnalitées

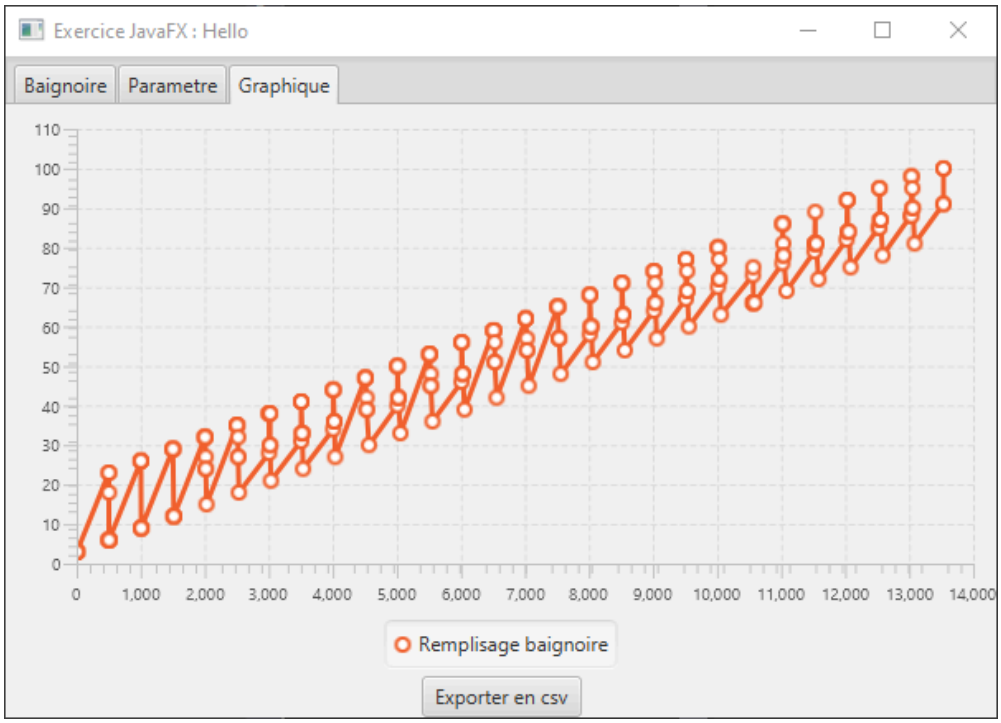
L'application permet de simuler une baignoire avec un nombre de robinet et de fuite allant de 1 à 4. Elle permet de changer le débit de toutes ces fuites et robinets indépendamment et sans avoir à redémarrer l'application. Elle permet aussi de changer la capacité de la baignoire. Durant la simulation on peut également reboucher des fuites et changer les débits des robinets. On peut voir tous ces paramètres sur un onglet dédié de l'application.




L'application dispose également d'un visuel de la baignoire avec des robinets et des fuites visualisé par, respectivement, de l'eau qui rentre au-dessus et de l'eau qui sort en dessous. Cette visualisation permet d'avoir une vue d'ensemble de ce qu'il se passe dans le simulateur.




L'application dispose également d'un affichage de graphique dans l'application et d'un export du même graphique en CSV.

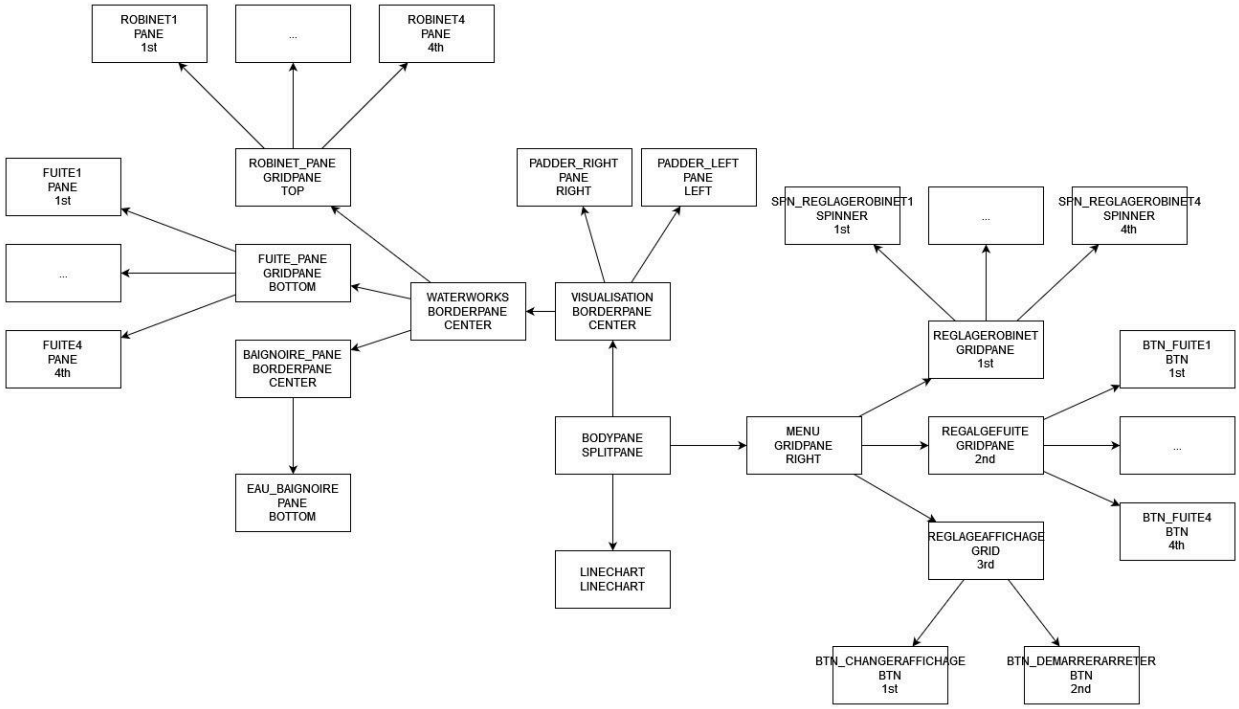


# Conception

Lors de la réalisation du projet j'ai réalisé une conception initiale de l'interface graphique et j'ai utilisé un autre projet que j'ai réalisé en cours (Verre) comme base.

## Conception Initiale

J'ai tout d'abord imaginé une interface se basant sur les SplitPane pour la séparation du graphique et de l'affichage. Bien que cela présente un certain avantage en termes de modularité d'interface, j'ai fini par l'abandonner car cette solution était peu intuitive et demandait plus de travail car elle n'était pas entièrement adaptée à mon usage.




Lors du TD Verre, nous avons utilisé ScheduledService pour mettre en parallèle les threads. J'ai donc décidé de réutiliser cette implémentation avec laquelle j'étais déjà familière même si j'ai pensé à utiliser un ExecutorService.

Je savais donc que le programme s'axera en 5 grande classe :

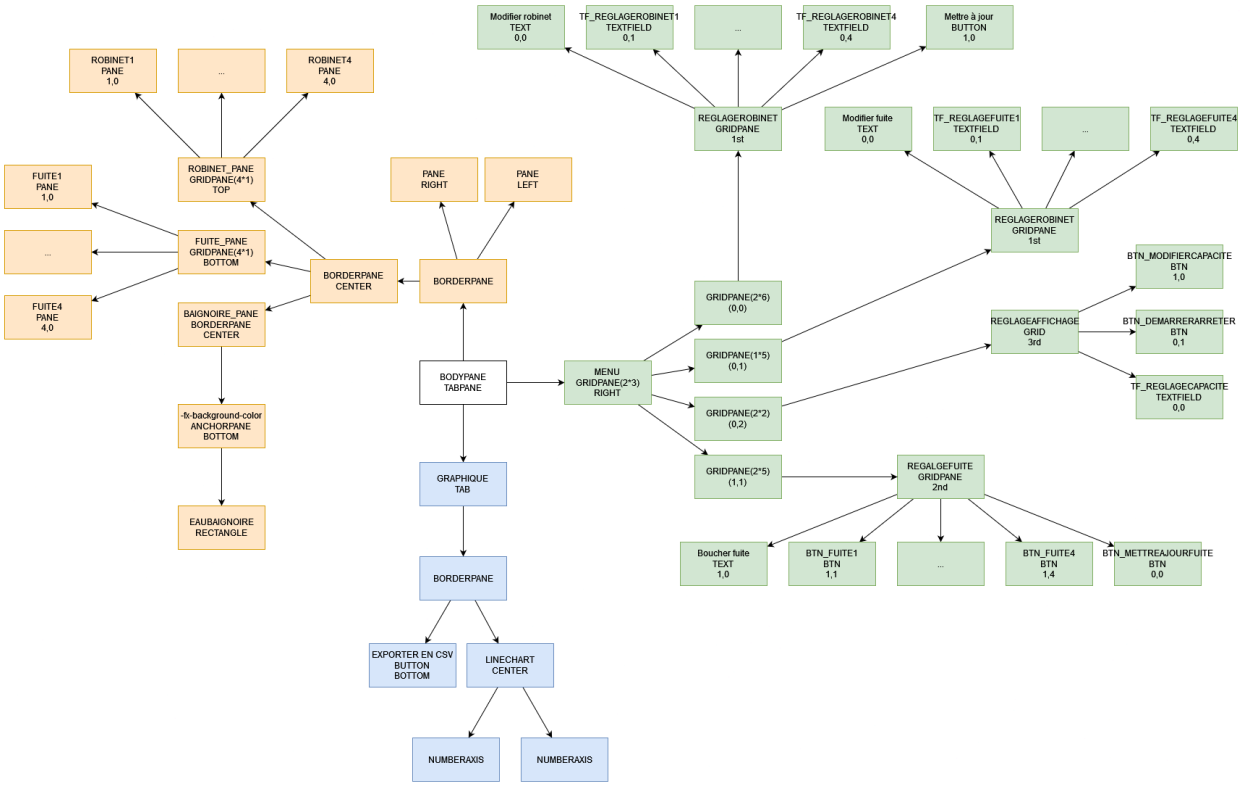
- App qui s'occupe de lancer l'application
- Baignoire qui représente la baignoire
- Fuite qui représente une fuite

- Robinet qui représente un robinet
- BaignoireController qui gère l'interactivité de l'interface

## Conception finale

Après avoir réalisé l'application, la conception finale à dévié en certain point de celle originelle. Dans cette partie je vais expliquer ces différents changements.

### Conception graphique




On peut voir que l'architecture à gardé ces 3 grandes partie :

- Affichage de la baignoire
- Affichage du graphique
- Affichage du menu de paramètre

Même si désormais ces 3 sont séparés par un TabPanel qui permet d'avoir des onglets qui compartimentent très facilement ces 3 parties. Un autre changement global est que tous les panel ne sont cette fois pas nommés, en effet si cela ne sont pas utilisé cela à pour effet d'encombrer le .fxml sans raison et est donc une mauvaise pratique.

- Dans la partie de l'affichage de la baignoire :

Le changement principal est l'utilisation d'un ensemble BorderPane [AnchorPane [Rectangle]]] qui permet d'avoir une baignoire entièrement responsive. L'idée est que l'on a un rectangle invisible qui sert à définir la hauteur que prend notre AnchorPane qui a un fond de couleur. L'AnchorPane lui-même est coincé en bas de l'espace maximal qu'il peut prendre par le BorderPane.

- Dans la partie du Graphique :

On a ajouté un bouton pour l'export de CSV. Avec plus de temps j'aurai eu l'occasion de faire en sorte que l'on puisse choisir où le fichier est sauvegardé mais j'ai n'ai malheureusement pas eu ce luxe.

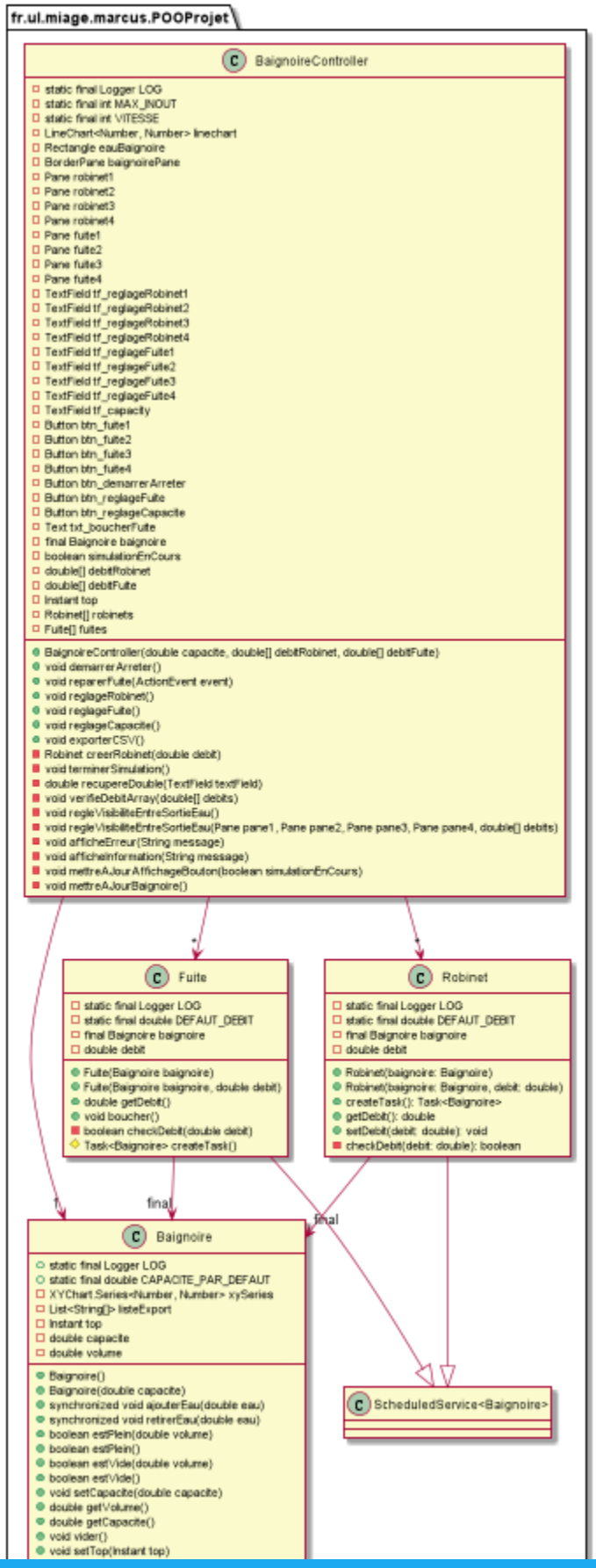
- Dans le menu paramètre :

C'est un changement complet suite au fait que le menu fait l'entièreté de l'application au lieu d'un bord de celle- ci. De plus, des labels permettant plus de lisibilité ont été ajoutés ainsi que des menus qui avaient été oubliés suite à une mauvaise interprétation du sujet. On peut notamment noter que les Spinner ont été abandonnés car mal intégrés dans SceneBuilder, pour des question d'homogénéité j'ai donc décidé d'utiliser un TextField et de vérifier si l'entrée est un nombre.

## Conception globale

La conception prévue grâce au projet similaire qu'à été Verre à permis d'avoir très peu de changement sur ce point de vue.

Pour des questions de lisibilité App et Launcher ne sont pas sur le diagramme de classe suivant.


fr.ul.miage.marcus.POOProjet
BaignoireController
static final Logger LOG
static final int MAX_INOUT
static final int VITESSE
LineChart<Number, Number> linechart
Rectangle eauBaignoire
BorderPane baignoirePane
Pane robinet1
Pane robinet2
Pane robinet3
Pane robinet4
Pane fuite1
Pane fuite2
Pane fuite3
Pane fuite4
TextField tf_reglageRobinet1
TextField tf_reglageRobinet2
TextField tf_reglageRobinet3
TextField tf_reglageRobinet4
TextField tf_reglageFuite1
TextField tf_reglageFuite2
TextField tf_reglageFuite3
TextField tf_reglageFuite4
TextField tf_capacity
Button btn_fuite1
Button btn_fuite2
Button btn_fuite3
Button btn_fuite4
Button btn_demarrerArreter
Button btn_reglageFuite
Button btn_reglageCapacite
Text txt_boucherFuite
final Baignoire baignoire
boolean simulationEnCours
double[] debitRobinet
double[] debitFuite
Instant top
Robinet[] robinets
Fuite[] fuites
BaignoireController(double capacite, double[] debitRobinet, double[] debitFuite)
void demarrerArreter()
void reparerFuite(ActionEvent event)
void reglageRobinet()
void reglageFuite()
void reglageCapacite()
void exporterCSV()
Robinet creerRobinet(double debit)
void terminerSimulation()
double recupereDouble(TextField textField)
void verifieDebitArray(double[] debits)
void regleVisibiliteEntreSortieEau()
void regleVisibiliteEntreSortieEau(Pane pane1, Pane pane2, Pane pane3, Pane pane4, double[] debits)
void afficheErreur(String message)
void afficheInformation(String message)
void mettreAJourAffichageBouton(boolean simulationEnCours)
void mettreAJourBaignoire()
Fuite
static final Logger LOG
static final double DEFAUT_DEBIT
final Baignoire baignoire
double debit
Fuite(Baignoire baignoire)
Fuite(Baignoire baignoire, double debit)
double getDebit()
void boucher()
boolean checkDebit(double debit)
Task<Baignoire> createTask()
Robinet
static final Logger LOG
static final double DEFAUT_DEBIT
final Baignoire baignoire
double debit
Robinet(baignoire: Baignoire)
Robinet(baignoire: Baignoire, debit: double)
createTask(): Task<Baignoire>
getDebit(): double
setDebit(debit: double): void
checkDebit(debit: double): boolean
final
final
Baignoire
static final Logger LOG
static final double CAPACITE_PAR_DEFAUT
XYChart.Series<Number, Number> xySeries
List<String[]> listeExport
Instant top
double capacite
double volume
Baignoire()
Baignoire(double capacite)
synchronized void ajouterEau(double eau)
synchronized void retirerEau(double eau)
boolean estPlein(double volume)
boolean estPlein()
boolean estVide(double volume)
boolean estVide()
void setCapacite(double capacite)
double getVolume()
double getCapacite()
void vider()
void setTop(Instant top)
ScheduledService<Baignoire>

## Dépendances

L'application utilise des librairies pour faciliter certaines tâches. On peut donc y voir :

- commons-cli 1.6.0
- JavaFX 17
- opencsv 5.3

De plus l'application utilise différent plugins dont en voici les plus notable:

- JavaFX
- maven-assembly-plugin (pour zipper le projet)
- appassembler-maven-plugin (pour créer des binaries)
- maven-jar-plugin (Permet de générer un jar avec un manifest)
- maven-javadoc-plugin (Permet de générer la javadoc directement avec Maven)

## Autres

J’ai volontairement décidé de ne pas intégrer un plugin permettant la création d’une page Web a partir de mon ReadMe sachant que mon projet est partagé uniquement via GitHub qui propose déjà une mise en forme en ligne du dit README. Cela aurait ajouté au poids et dépendance du projet sans raison. Il faut toujours garder en tête que bien que les librairies soit bien pratique une sur dépendance à celle ci sans raison n’est pas saine car cela reste une dépendance supplémentaire au projet.

Pour générer la Javadoc utiliser la commande mvn javadoc:javadoc ou utiliser les commandes par défaut de java.

# Conclusion

Ce projet est dans la continuité des cours et demande l'usage des connaissances acquises tout au long de celui-ci. J'ai pu dans ce projet travailler et utiliser Maven pour un projet d'une ampleur supérieure à 12h heure de travail.