Cours 420-203-RE ***Développement de programmes dans un environnement graphique*** Automne 2022 Cégep Limoilou Département d'informatique	Tp1 Première programme (8 %) • **JavaFX**

OBJECTIFS

- Analyser et reproduire une interface graphique.
- Utiliser les composants *JavaFX* pour créer une interface graphique;
- Comprendre et utiliser les gestionnaires de disposition.
- Programmer le comportement d'un bouton.

ACTIVITÉS À RÉALISER

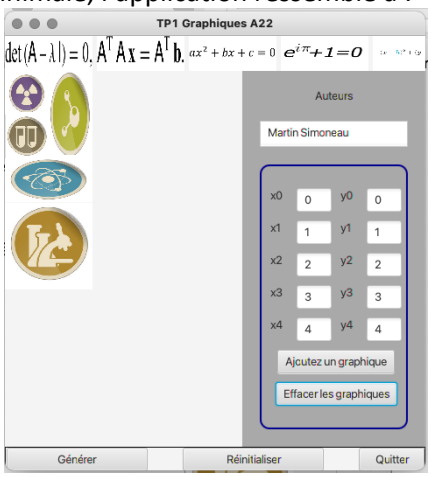
Vous avez à concevoir une interface graphique en respectant l'image du prototype fourni.

DURÉE DE L'ACTIVITÉ :

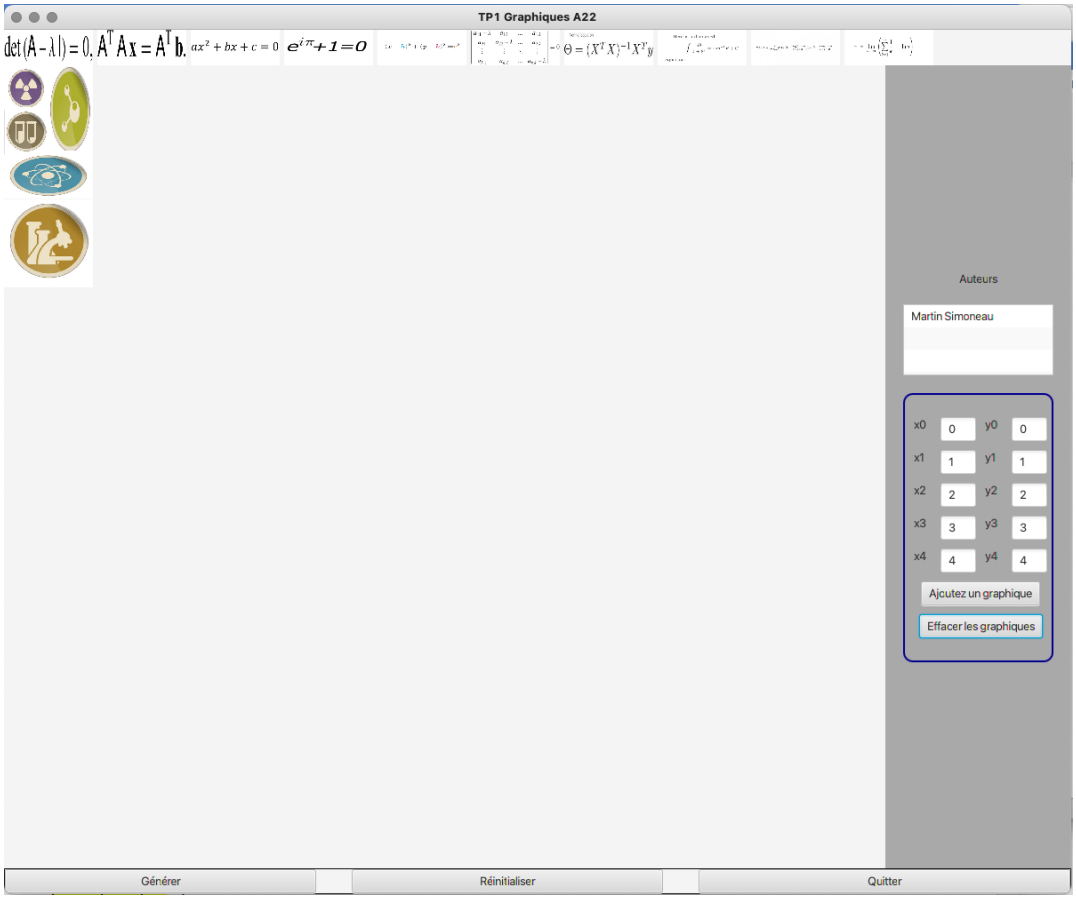
- ~1 semaine
- Ce travail peut être réalisé en équipe de 2 maximum.

PROTOTYPE 1 :

- Lorsque la fenêtre est à sa taille minimale, l'application ressemble à :



- Lorsque la fenêtre est à sa taille maximale, l'application ressemble à :



Notez que :

- Vous avez reçu une vidéo nommée ***tp1-a.mov***, qui vous montre le comportement de la fenêtre lorsqu'on modifie sa taille.
- La fenêtre n'a pas de hauteur maximale, mais elle a une largeur maximale de **1200px**

ÉTAPE 1, RÉFLEXION SUR LE DESING

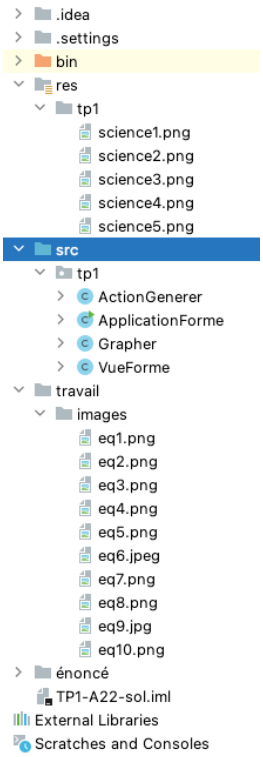
Votre première tâche consiste à identifier sur le prototype d'interface graphique présentée ci-dessus chacune des parties de vos interfaces, les gestionnaires de disposition proposés et les composants requis. Mettre vos réponses dans un document comprenant des textes d'identification des parties et les dessins pour illustrer votre réflexion (illustrer les interfaces). Voici un exemple de ce que j'aimerais:

- Faites valider votre croquis avant de commencer la programmation.
- Vous pouvez aussi faire le croquis à la main et le prendre en photo.

Informations utiles

- Dans cette scène , on retrouve au moins
  - 1 GridPane
  - 1 *TilePane*
  - Plusieurs *VBox* et *HBox*
  - 1 *BorderPane*

ÉTAPE 2 : PROGRAMMATION DES INTERFACES GRAPHIQUES



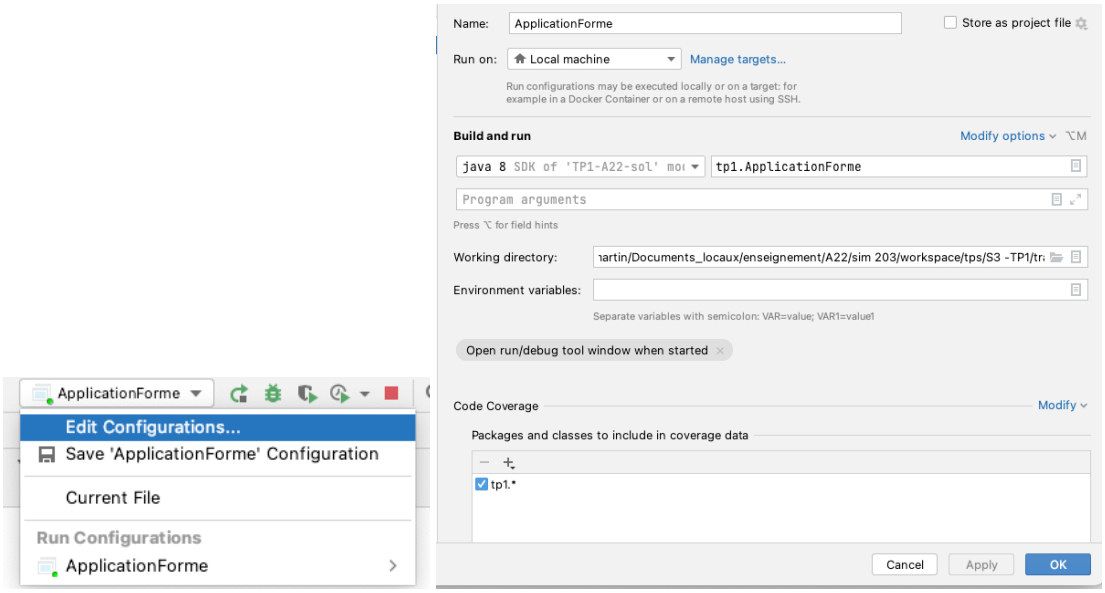
Votre interface doit être codée en *JavaFX* directement. Disposez votre code en *packages* et vos ressources dans un dossier de ressources, telle que présentée précédemment. L'application ne fait que présenter ce que *VueForme* crée comme *scène*. La classe « ***VueForme*** » prépare l'ensemble de l'interface, gardez le code comme présenté dans les exemples.

Informations utiles

- Dimensions
  - Estimez la taille minimale de la fenêtre
  - La taille maximale de la fenêtre est de 1200 px.
  - Les graphiques de sciences ont :
    - Largeur max de 200
    - Hauteur max de 150
  - Les images de science ont uniquement 100px ou 200px comme hauteur ou largeur (référez-vous à l'image)
  - La taille des 3 boutons change avec la taille de la fenêtre, observez attentivement leur comportement.
    - Taille maximale 350 pour les boutons *Générer* et *Réinitialiser*
    - Taille maximum illimitée pour le *Quitter*
    - Tous les boutons ont une taille minimum de 50 et une taille préférée de 75
- Pour vous aider à dimensionner les éléments graphiques, on a laissé plusieurs constantes provenant de la solution dans la classe *VueForme*.
- ListView
  - Pour mettre des éléments dans un *ListView* il faut faire :
    `listView.getItems().addAll("item1", "item2");`
  - Le *ListView* est générique, paramétrez-le avec un String. *ListView<String>*
- File
  - Objet qui permet de rechercher tous les fichiers dans un dossier:

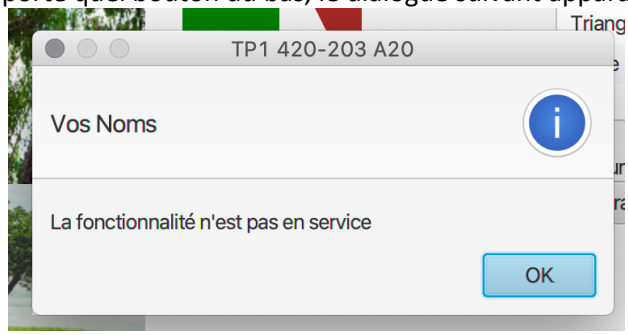
```
File dossier = new File("./path");//relatif au working
dossier.listFiles()
```

- Pour régler le working d'une application dans IntelliJ:



Exigences

- Votre application doit reproduire le plus fidèlement possible les comportements de la fenêtre lorsqu'on la redimensionne (taille des boutons, positionnement des éléments…). Consultez le vidéo *tp1a.mov* pour les détails.
- Toute la fabrication de l'interface graphique doit être dans la classe *VueGraph*.
- L'application *ApplicationGraph* utilise la classe *VueGraph* pour créer la scène et elle règle les dimensions de la fenêtre.
- Dans le haut de chacune des classes, vous devez mettre les noms des auteurs.
- Lorsqu'on appuie sur n'importe quel bouton du bas, le dialogue suivant apparaît.



- Dans le nom du projet (ar*tifact ID* avec Maven), vous devez restructurer le nom TP1_A22-***vos-initiales*** en remplaçant ***vos-initiales*** par vos propres initiales.
- Découpez vos classes en plusieurs méthodes de façon adéquate. Évitez les méthodes de plus de 50 lignes (incluant la documentation et les commentaires) , c'est difficile à maintenir.
- La barre du haut doit présenter toutes les équations du dossier *travail/images* peu importe leur nombre. Ce dossier ne doit pas être dans les ressources du programme, car son contenu peut être changé par l'utilisateur.
- La classe ***Grapher*** sert à créer un graphique avec les données reçues de l'interface. Les paramètres sont transmis dans un objet de type ***Grapher.Parameters*** qui contient les informations pertinentes (cette classe interne vous est déjà fournie). Les seuls objets *JavaFX* que devrait voir *Grapher* sont les classes liées au *LineChart*.
- Faites en sorte que lorsqu'on appuie sur le bouton *Ajouter un graphique* :
    - L'interface envoie les informations de la section gauche de l'UI à la classe *Grapher* pour que ce dernier puisse fabriquer le *LineChart* correspondant.

    - Vous pouvez utiliser la classe ***ActionGenerer*** ou utiliser une autre méthode de *callback* vue en classe.
    - Le nombre de données (x,y) doit pouvoir être modifié simplement en changeant une constante dans votre code.
    - Le UI doit ajouter le *LineChart* dans la section centrale de votre application. Chaque fois qu'on appuie sur le bouton, un nouveau *LineChart* est créé avec les dernières données saisies.
- Faites en sorte que lorsqu'on appuie sur le bouton *Effacer les graphiques* :
    - Tous les graphiques de la section centrale sont effacés.
- Mettez des commentaires appropriés dans le code et faites la *javadoc*.
- Le vidéo ***tp1-b.mov*** montre le comportement de l'application lorsqu'on appuie sur le bouton *Générer*.

- **Barème D'évaluation:**

- Le croquis de l'UI **(10%)**
    - Clair et **propre**
    - cohérent
    - Complet
- La présentation de L'UI **(30%)**
    - Conformité statique (15%)
        - Espacements
        - Marges
        - Choix des composants
        - fidélité
    - Gestion du redimensionnement (15%)
- Le code de l'application **(35%)**
    - Code de fabrication des graphiques
        - *Grapher*
        - Action du bouton *générer*
    - Qualité du code
        - Respect de l'architecture imposée.
        - Gestion des ressources
        - Séparation du code en méthode
        - Méthodes bien découpées avec un maximum de 50 lignes.
        - Commentaires (*javadoc* et commentaires)
- Fonctionnement **(25%)**
    - Fonctionnement de l'application principale
    - Apparition des dialogues.
    - Création des *LineChart* qui respectent les paramètres de l'UI
    - Absence de StackTrace dans la console.

À REMETTRE :

- Remettez votre projet complet, incluant votre croquis, dans une archive .**zip** sur Léa à La date indiquée sur Omnivox/Léa.