Cours 420-4A6-LI Applications monopostes I Hiver 2020 Cégep Limoilou Département d'informatique	Tp 2 Les événements JavaFX et l'architecture MVC 8 %

Nom : _____ Gr : 1

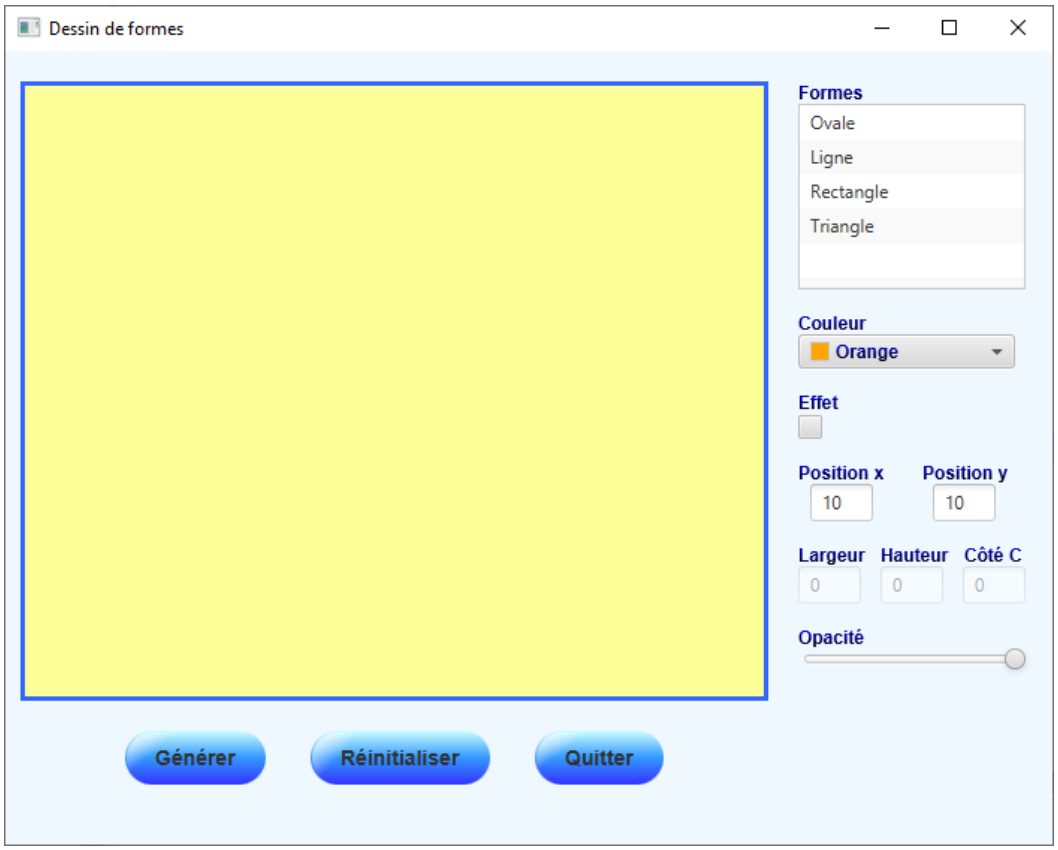
Nom : _____

Objectif

- Créer, un modèle (diagramme) de classes pour illustrer les composants de votre application.
- Utiliser les composants JavaFx pour créer une interface graphique.
- Comprendre et utiliser les gestionnaires de disposition.
- Utiliser les écouteurs pour pouvoir répondre à des événements.
- Respecter le modèle MVC vue dans le cours (code version 3 ou « LeChasseur »).
- Utiliser les possibilités des propriétés et du « databinding ».

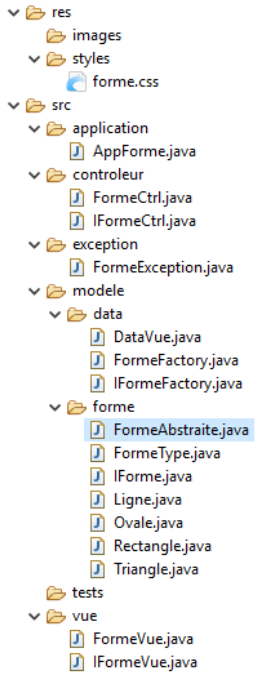
Activités à réaliser

- Vous avez à programmer l'application de dessin de formes que vous avez construit l'interface dans le Tp1.



À remettre et/ou à présenter

- Si vous faites un diagramme de classe, ce qui est fortement recommandé, je peux le valider avec vous. Cette validation peut se faire soit à la période de laboratoire du 6 mars ou dans la semaine du 16 au 20 mars.
- Pour le code source et les ressources, il doit respecter obligatoirement l'arbre de packages suivant et il doit respect aussi obligatoirement les noms des classes pour les packages « application », « controleur » et « vue ». Pour les classes du modèle, vous avez une certaine l'attitude, mais celles proposées donnent de bons indices sur la stratégie utilisée. Voici donc ce que l'on veut :



- Pour la disposition des contrôles et leur mise en forme, il faut tenir compte de l'image écran et des consignes du présent travail.
- Tout votre code doit-être documenté avec de la JavaDoc.
- Des tests unitaires JUnit doivent être faits pour toutes les classes métiers du package « modele ».
- Me remettre avant la fin de la période de laboratoire du **27 mars**, le projet « Eclipse » complet, compressé (.zip) dans un fichier du nom « noms des équipiers – Tp2.zip » en le déposant sur le réseau dans le dossier « Q:\!Dépôt\JGoulet\420-4A6-H20\Tp 2 – Code ».

Durée de l'activité

- Ce travail doit être réalisé en équipe de 2. Si le nombre d'étudiants est impair, je proposerai une solution.
- Vous avez du 6 mars au 27 mars (fin du laboratoire) pour réaliser ce travail.

Avant de commencer

- Il faut avoir fait au moins un des cas au complet du formatif 5 et/ou un des problèmes de la préparation à l'examen de la semaine 7.
- Il faut avoir lu les documents des notes de cours des semaines 4, 5 et 6 inclusivement.
- Il faut avoir compris les exemples fournis par le professeur pour les semaines 5 et 6 et les solutions partielles données par le professeur au cas 2 et 3 du formatif 5.

Étape 1 : Stratégie et diagramme de classe (non obligatoire, mais fortement recommandé)

- Votre première tâche consiste à fabriquer un diagramme de classes de toutes vos classes impliquées dans votre architecture MVC. Utilisez l'outil « **Visual Paradigm** » ou « **ObjectAid** » installé dans Eclispe (revoir, dans le formatif 2, les références à la fin du cas 1).

  Cette étape est fortement recommandée pour prendre pour vous permettre de prendre plusieurs décisions. En plus de vous permettre de réfléchir à votre stratégie de conception, il vous permettra de déterminer les interfaces, les classes abstraites et les héritages. Bien entendu, il n'y a rien de parfait, faites de votre mieux. Je peux, si vous me le demandez, regarder votre diagramme avec vous. Commencez par lire le présent travail au complet, ce qui va vous permettre de faire un premier diagramme papier de haut niveau.
- Comme la section « À remettre ou à présenter » le précise, il n'y a rien à remettre à cette étape.

Étape 2 : Programmation et validation des classes métiers

- Programmer vos classes de la partie « modèle » du MVC, les classes de formes (Forme, Ovale, Rectangle, Triangle, Ligne) et le « pseudo factory ».

  Pour les formes, ayez une référence sur le coin supérieur gauche pour la position et une taille (largeur, hauteur et côté C) qui peut varier entre 10 et 1000.

  Pour l'ovale, le rectangle et la ligne, on prend les valeurs dans les champs « Largeur » et « Hauteur ».

  Pour le triangle « Largeur », « Hauteur » et le « Côté C » sont la longueur de chacun de ses côtés. **Faire attention pour le triangle, il faut vérifier si c'est possible de faire un triangle avec les longueurs reçues.**
- Astuce : Pour toutes les formes, j'utilise l'interface « IForme » qui contient la méthode « getSommets() » qui retournera les sommets sous forme de points pour chaque forme. Très utile, dans mon cas, pour vérifier si la forme entre bien dans la zone de dessins.
- **IMPORTANT :** La dimension de la zone de dessins de l'application est fixée 400 pixels pour la hauteur et 500 pixels pour la largeur. Vous devez prévoir une classe usine (pseudo factory) qui produira, validera et livrera des objets formes valides qui respecteront les dimensions de la zone de dessins sans **débordement**. **Voir le diagramme de fonctionnement de haut niveau dans la dernière page.**
- Des tests unitaires JUnit complets doivent obligatoirement être faits pour toutes les classes du package « modele ».

Étape 3 : Programmation des événements

- À partir de votre vue déjà programmée dans le travail 1 et des classes modèles produites précédemment, adaptez le code et programmez la gestion des événements en respectant l'architecture MVC présentée dans le cours (code version 3 ou « LeChasseur »).
- Le bouton « **Générer** » ajoute la forme avec les paramètres choisis, dans la zone de dessins de l'application.
- Le bouton « **Réinitialiser** », réinitialise toutes les valeurs et vide la zone de dessins. Pour les valeurs par défaut et l'état des champs, référez-vous à l'image d'écran présentée au début de ce document.
- Pour bien utiliser les classes métiers (modèle dans le MVC), il est important que vos formes soient créées par un objet comme un « factory » et non directement en JavaFX. L'utilisation des classes métiers est très importante pour la validation de la forme avant d'être créée en JavaFX. (Les métiers permettent plus facilement de recréer la même application dans un autre module d'interface graphique différent comme Android ou Swing). Pour dessiner dans la vue, vous allez utiliser des objets « Shape », voir le document « **Diapo - ihm2_fx_01 (Graphics 2D Shapes, Canvas, Charts).pdf** » fourni dans les notes de

cours de la semaine 7. Les formes, avec des paramètres valides, s'ajoutent au fur et à mesure que vous appuyez sur le bouton **Générer**.

- La case à cocher « **Effet** » et la glissière « **Opacité** » affectent l'ensemble des objets de la zone de dessins.

  La case à cocher « **Effet** », applique un effet **<u>à votre goût</u>** sur les formes dessinées (dans l'exemple c'est « dropShadow »).

  Pour l'opacité vous devez utiliser du « databinding », à vous de découvrir le principe. Référez-vous au document « **S07 - Diapo - ihm1_fx_02 - partie 2 - (Compléments de programmation).pdf** » disponible dans le dossier des notes de cours semaine 7. On va revenir sur ce sujet dans les prochains cours. L'effet et l'opacité sont appliqués directement dans la vue, sur le vecteur de nœuds « Shape » de la zone de dessins.
- Il est aussi très important de signaler à l'utilisateur, à l'aide de boîtes de dialogue, **tous les problèmes** qui surviennent lors de la validation des données ou autres. Voir les infos sur les boîtes de dialogues dans le dossier des notes de cours de la semaine 6.

Diagramme de fonctionnement pour le bouton « Générer »