Université Paris Diderot INTERFACES GRAPHIQUES 29 juin 2018
9h30-11h30 durée 2h

Les seuls supports autorisés :
documentation API java `http://docs.oracle.com/javase/8/docs/api/index.html`,
documentation API JavaFX : `http://docs.oracle.com/javase/8/javafx/api/toc.htm`
les tutoriels de javaFX d'Oracle :
`http://docs.oracle.com/javase/8/javase-clienttechnologies.htm` et
`https://docs.oracle.com/javase/8/javase-books.htm`
et les transparents et programmes étudiés en cours :
`https://moodlesupd.script.univ-paris-diderot.fr`
L'utilisation de toute autre ressource web (en particulier des forums de développeurs), de moteurs de recherche (google et autre), de mail ou d'autres moyens de communication sera traitée comme une fraude. Les documents papier autorisés : vos notes de cours ou TP, les photocopies de transparents de cours, notes personnelles. Livres et photocopies de livres interdits. Vos mobiles doivent être impérativement éteints. L'écran de votre voisin n'est pas une ressource autorisée.
Le sujet comporte 4 pages.

Consignes

Votre classe principale doit porter votre nom, par exemple pour un dénommé Mathias Dupont la classe principale s'appellera `DupontMathias`.
Il y a un seul programme à développer mais le développement est découpé en plusieurs questions. Certaines questions sont indépendantes, d'autres non.
Il est possible de déposer plusieurs versions numérotées (DupontMathias1, DupontMathias2 etc.) du programme. Cependant c'est la dernière version qui sera examinée pour noter votre travail. Si pour une raison quelconque vous voulez que j'examine aussi d'autres versions faites un fichier README avec un message approprié.

Si vous developpez sur NetBeans le plus simple c'est exporter le programme vers zip (File → Export Projet → to Zip) et déposer le fichier zip avec le projet complet. Sinon vous pouvez aussi faire zip ou tar contenant juste les fichiers sources.
Le fichier archive doit être déposé sur moodle, cours Interfaces graphiques, dans dépôt d'examen session rattrapage.
N'attendez pas la fin de l'examen pour déposez vos programme, le dépôt sur moodle ferme à 11h30 pile.

Vos programmes peuvent être examinés par un logiciel de détection de plagiat.

Dans toutes les questions, si vous avez à implémenter une interface avec une seule méthode vous devez utiliser une lambda expression. L'utilisation de classes anonymes (ou non anonymes) là où une lambda expression est suffisante entraînera une diminution de la note.

Exercice 1

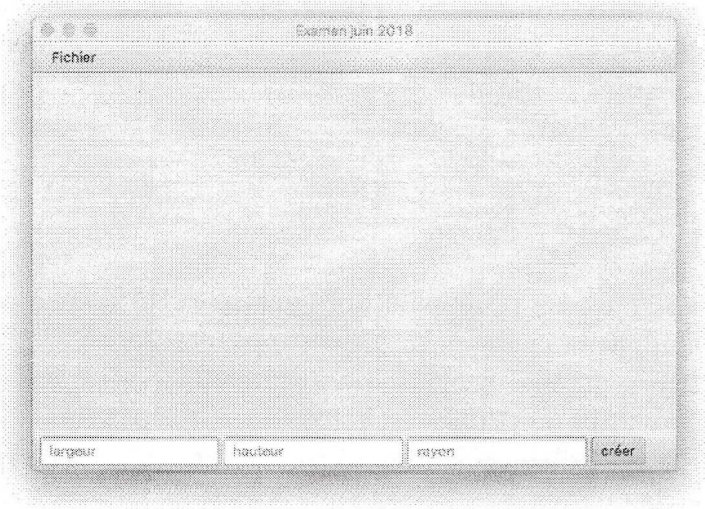
Question 1: Écrire une application qui comporte initialement :

(1) un menu en haut avec un seul item dont l'action est de quitter l'application. Il faut prévoir un accélérateur qui permet de quitter l'application juste par une combinaison de touches clavier (Q + plus une touche de modification qui dépend de système d'exploitation).

(2) un `Pane` vide au centre,

(3) un `ToolBar` en bas.

Dans le `ToolBar` il y a les éléments que vous pouvez voir sur cette image :
Les trois `TextFields` contiennent des PromptTexts : respectivement `largeur`, `hauteur` et `rayon`. Notez que le PrompText est visible uniquement quand le `TextField` ne possède pas de focus.



Question 2: Ajouter des `Tooltips` pour chaque `TextField`. Le Tooltip affichera le même texte que le PromptText de TextField correspondant.

Question 3: L'utilisateur écrit dans les trois TextFields des nombres doubles positifs et clique sur le bouton `créer`.
Il faut capter les exceptions qui peuvent survenir quand on transforme le contenu de TextField vers un nombre double. En cas d'exception on met un String vide dans le TextField correspondant.
Si après la transformation en double le nombre double est négatif alors on met un String vide dans le TextField correspondant.
Il n'y a aucune d'autre action à implémenter si l'exception ou un nombre négatif.
Si les trois nombres sont positifs mémorisez les dans trois variables pour l'utilisation postérieure et désactiver (disable) les trois TextFields et le bouton créer.

Question 4: Pour le `TextField largeur` (et uniquement pour ce premier `TextField`), s'il y a une exception (`NumberFormatException`) quand on transforme le contenu en nombre double vous lancerez un dialogue simple (`Alert` de type `Alert.AlertType.ERROR`) avec le message `rayon incorrect`. L'application est suspendue jusqu'à ce que l'utilisateur clique sur le bouton OK de la fenêtre d'`Alert`.

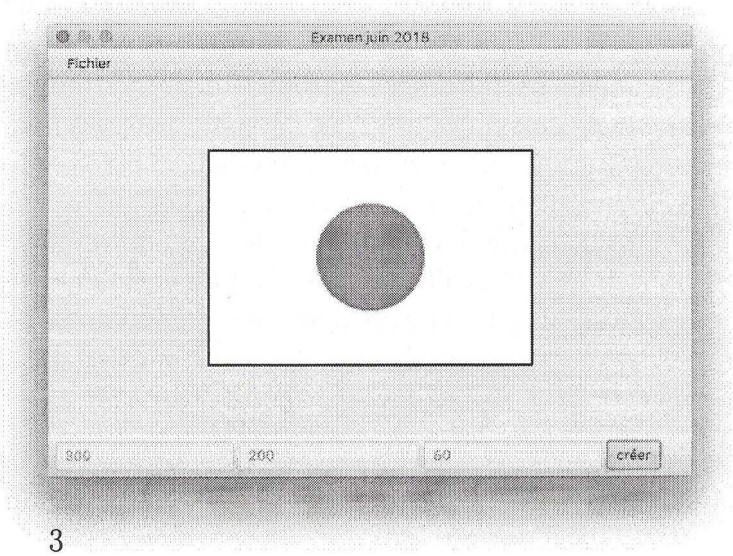
Question 5: Ajouter le comportement suivant. Quand l'utilisateur clique sur le bouton `créer` et les trois nombres dans les TextFields sont positifs alors on crée un `Rectangle` avec la largeur et hauteur données par les valeurs de deux premiers TextFields. Le rectangle est blanc avec le bord noir. Il doit être placé exactement au centre du `Pane` qui est au centre de la fenêtre de l'application. Quand on change la taille de la fenêtre il faut que le rectangle reste au centre. **Ce comportement doit être implémenté à l'aide de bindings.** Toute autre implémentation ne donne pas de points donc si vous ne savez pas faire avec bindings passez directement aux questions suivantes.
Indication pour ceux qui ont des problèmes insurmontables avec les maths élémentaires. Le rectangle est au centre du Pane si les conditions suivantes sont satisfaites à chaque instant :

$$rectangle.x = (pane.width - rectangle.width)/2$$
$$rectangle.y = (pane.height - rectangle.height)/2$$

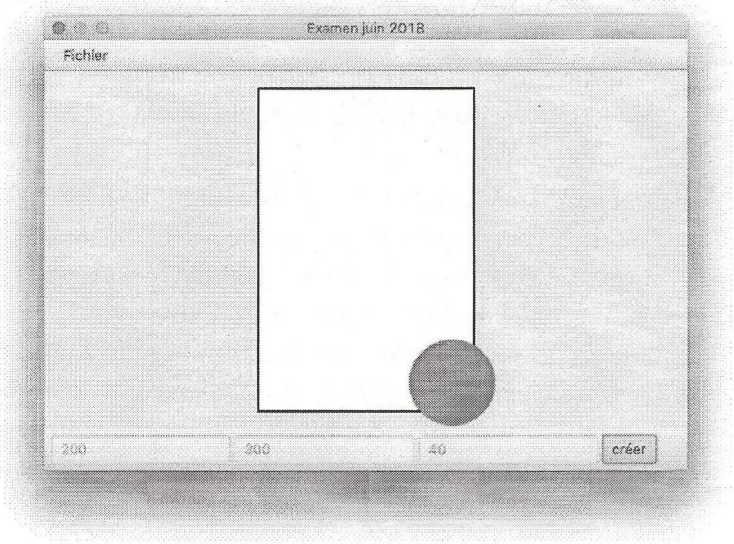
où, vous devinez (j'espère) que, `rectangle.a` désigne une propriété `a` du rectangle et `pane.b` une propriété `b` du Pane.

Question 6: Compléter l'action de la question précédente. Maintenant en plus d'un rectangle quand on clique sur le bouton `créer` on ajoute un cercle rouge dont le rayon est donné par la valeur du troisième `TextField`. Initialement le centre du cercle et au centre du Rectangle (mais ce n'est pas un binding, juste la position initiale du cercle).

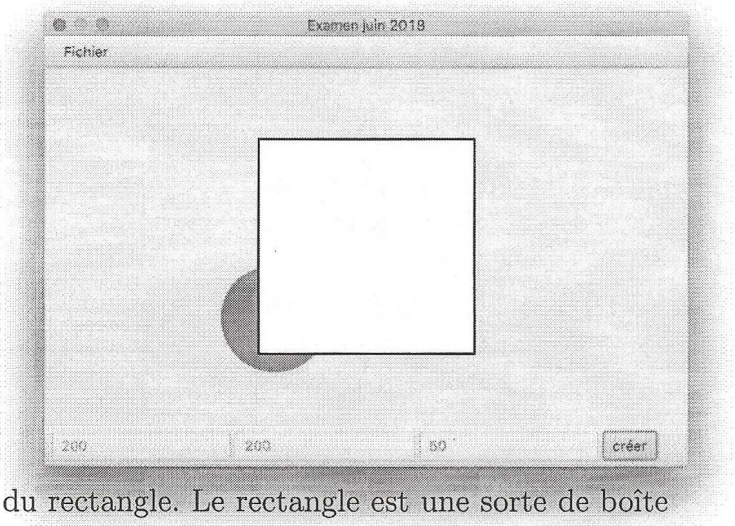
Question 7: Quand l'utilisateur clique sur le rectangle le centre du cercle fait un saut sur la position de la souris. Implémenter cette fonctionnalité. Notez que si on clique sur une position proche du bord du rectangle alors sur la nouvelle position le cercle peut déborder à l'extérieur du rectangle. Cela est utile pour la question suivante.



Question 8: Ajouter dans le menu un nouveau MenuItem `Inverser`. Quand l'utilisateur sélectionne ce MenuItem alors on inverse l'ordre du cercle et du rectangle. Initialement le cercle est au-dessus du rectangle. Si on inverse l'ordre le cercle se retrouve au-dessous du rectangle (ce qui peut le rendre invisible si le rectangle recouvre complètement le cercle). On sélectionnant `Inverser` encore une fois on revient sur l'ordre précédent : le cercle au-dessus du rectangle.



Question 9: Donner à l'utilisateur la possibilité de déplacer le cercle à l'aide de la souris (le mouvement `drag`).



Question 10: On supposera dans cet exercice que le rayon du cercle est tel que le cercle initialement ne déborde pas à l'extérieur du rectangle [1] (On suppose que l'utilisateur toujours respecte cette condition quand il crée le rectangle et le cercle, on ne demande pas que votre application vérifie que cette contrainte soir respecté).
Donc initialement le cercle est au milieu du rectangle. Le rectangle est une sorte de boîte qui contient le cercle. On demande la fonctionnalité suivante. Quand l'utilisateur déplace le cercle à l'aide de la souris il ne peut pas le sortir du rectangle, si le cercle déborde à l'extérieur on le ramène automatiquement à l'intérieur du rectangle comme si le bord du rectangle soit un mur infranchissable pour le cercle.
(Cette comportement est demandée uniquement pour le déplacement du cercle avec le mouvement drag de la question 9.
On ne demande pas que le cercle reste à l'intérieur du rectangle si le centre du cercle fait un saut quand on clique sur le rectangle comme dans la question 7. Le cercle peut déborder à l'extérieur par un saut mais pas par le mouvement drag.)

1. ce qui est vrai quand `cercle.rayon < min( rectangle.width, rectangle.height )/2`