Université Paris Diderot INTERFACES GRAPHIQUES 17 mai 2019
8h30-10h30

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`
`https://openjfx.io/javadoc/12/index.html`
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 moteur de recherche (google et autre), de mail ou d'autres moyens de communication sera traitée comme une fraude.
Les documents papier interdits.
Vous pouvez utiliser le code de vos propres programmes.
L'écran de votre voisin n'est pas une ressource autorisée.
Le sujet comporte 4 pages.

Consignes

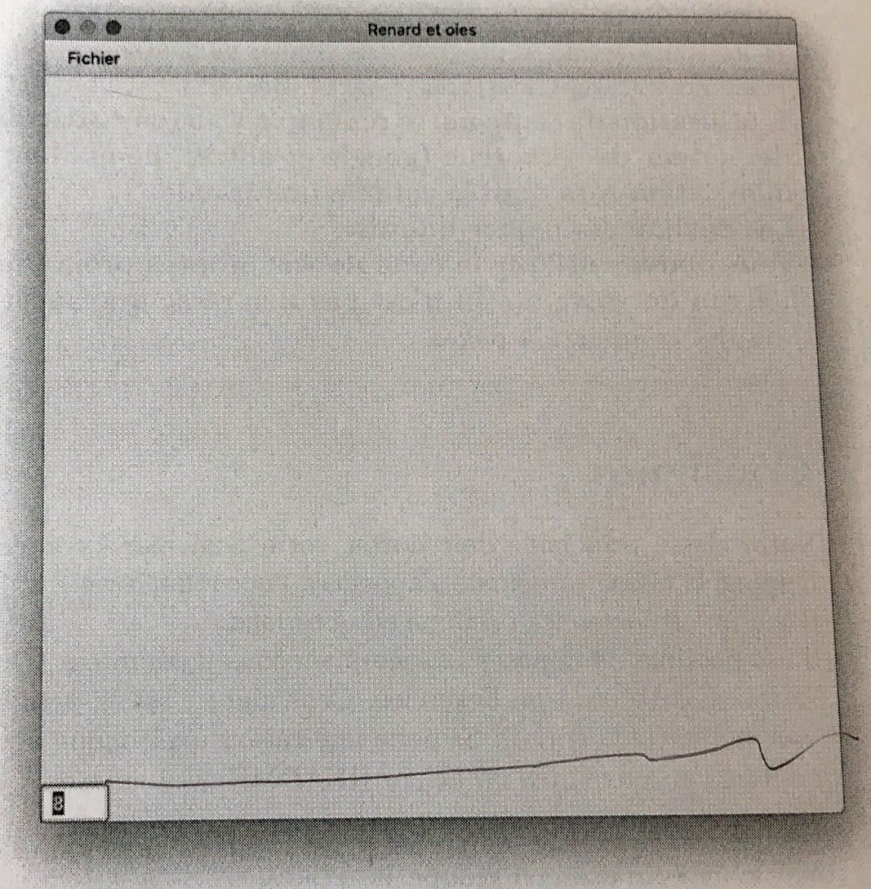
Votre classe principale doit porter votre nom, par exemple pour un dénommé Mathias Dupont la classe principale s'appellera `DupontMathias`.
Il y a un seul exercice découpé en questions.
Il est possible de déposer plusieurs versions numérotées (DupontMathiasExo1a, DupontMathiasExo1b etc.) de l'exercice. 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.

Vous devez faire un fichier archive (zip ou tar ou tar.gz) que vous déposerez sur moodle dans le dépôt d'examen groupe 1.
N'attendez pas la fin de l'examen pour déposez vos programmes, le dépôt sur moodle ferme à 10h30 pile.

Vos programmes seront examinés par un logiciel de détection de plagiat.
Dans toutes les questions, si vous avez à implémenter un interface avec une seule méthode vous devez utiliser une lambda expression. L'utilisation de classes anonymes (ou non anonymes) là où une simple lambda expression est suffisante sera pénalisée.

Exercice 1

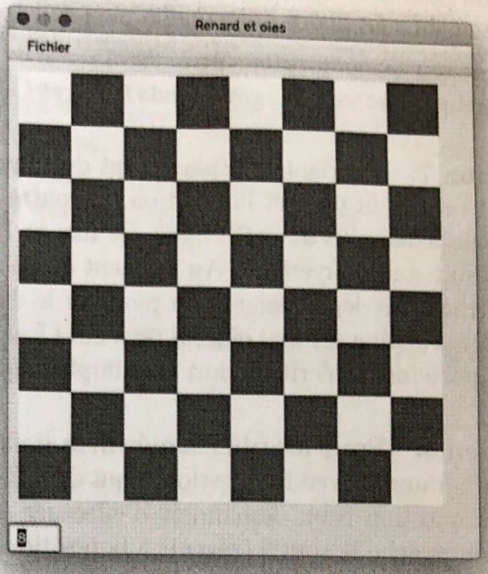
Question 1: Écrire une application qui comporte initialement un menu en haut avec deux MenuItems.
Le premier `MenuItem` : `quitter` permettra de quitter l'application. Le deuxième `MenuItem` : `jouer` sera utilisé dans les questions suivantes.
Au centre de la fenêtre vous mettrez un `Pane`, initialement vide.
En bas de la fenêtre il y une `TextField` qui contient initialement le nombre 8.
Vous devez construire cette interface en utilisant fxml.



Question 2: On dira que le `TextField` est **valide** s'il contient un nombre entier pair ≥ 8.
Quand utilisateur active le `MenuItem jouer` vous devez vérifier si le `TextField` contient un nombre valide. Si ce n'est pas le cas vous afficherez une `Alerte` avec le texte `nombre invalide` et vous remettrez 8 dans `TextField`.
Si `TextField` contient un entier valide vous sauvegarderez cet entier dans une variable d'instance. Ce nombre sera nécessaire pour la question suivante.

Question 3: Dans cette question on complète l'action `jouer`. Si on sélectionne le `MenuItem jouer` et si le TextField contient un entier valide vous devez créer un damier de taille $n \times n$ où n la valeur donnée dans le TextField.
Les carrés dans le damier doivent être soit blancs soit gris comme sur ce dessin :

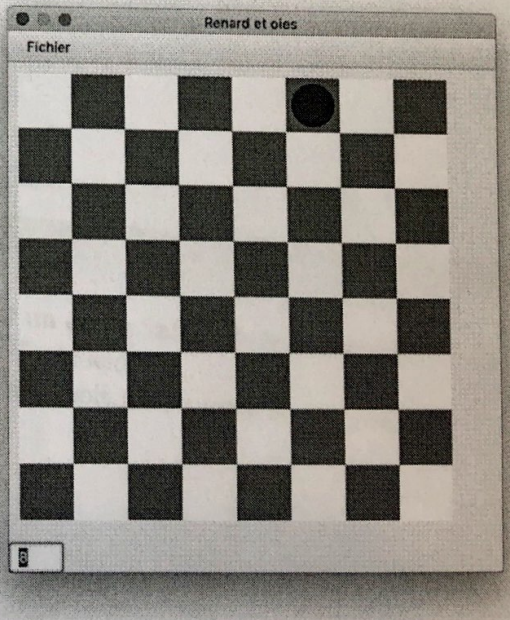
Indication 1. Pour ceux/celles qui ont le problème avec le calcule d'indices, supposant que le carrés sont stockés dans un tableau Rectangle[n][n], de telle sorte que le carré à gauche de la première ligne est Rectabngle[0][0] alors la couleur du carré Rectangle[i][j] dépend de la parité de la somme $(i + j)$.



Question 4: On complète la question précédente en plaçant un pion — un cercle rouge — sur de damier au centre d'un carré gris (on fait ce placement quand on dessine le damier). On peut choisir un carré gris quelconque sur la première ligne. Le cercle doit être légèrement plus petit que le carré.

Question 5: Donner au joueur la possibilité de déplacer le pion à l'aide de la souris, tout d'abord n'importe où sur le Pane.

Question 6: Dans cette question il faudra restreindre les déplacements du pion. L'idée est que le pion peut être déplacé uniquement d'un carré gris sur un autre carré gris.
On distinguera deux cas quand on termine le déplacement du pion :



1. quand on relâche la souris le centre du pion n'appartient à aucun carré gris. Dans ce cas le pion doit automatiquement revenir à sa position précédente (le pion peut être déplacé uniquement d'un carré gris sur un autre carré gris, il ne peut pas être abandonné ailleurs),
2. si le centre du pion se trouve dans un carré gris et l'utilisateur relâche la souris alors le centre

du pion doit être automatiquement positionné au centre de ce carré (on fait un ajustement nécessaire pour que le pion soit toujours au centre d'un carré gris).

Implémenter ce comportement.
Indication. `rectangle.getBoundsInParet().contains(a,b)`

Question 7: Pendant le déplacement du pion à l'aide de la souris on dessinera une ligne entre le centre du pion et la position du centre au début du mouvement.
Donc une extrémité de cette ligne est fixe et l'autre extrémité est toujours au centre du pion et suit son mouvement. Au moment où on relâche la souris la ligne doit se figer (l'idée est de tracer les déplacements du pion sur le damier).
(Bien sûr si le pion est mal placé il reviendra à sa position précédente et la ligne disparaîtra).
Le comportement décrit ici doit être implémenté à l'aide de binding.

Question 8: Pour les plus téméraires Refaire la question 6 (mais sauvegardez d'abord votre programme avec la question 6 qui déjà faite).
Dans la question 6 on demandait d'effectuer certains ajustement de la position du pion quand on relâche la souris (revenir à la position initiale, ou se fixer au centre d'un nouveau rectangle).
Au lieu de le faire d'un seul coup on demande ici de faire une animation, quand on relâche la souris le pion vient **progressivement** à la position indiquée dans la question 6 au lieu de le faire par un saut instantané.