Département d'Informatique
IUT de Laval
Université du Mans

R2.02 – Intro aux IHM – JavaFX

Projet

Pierre-Jean PETITPREZ

Ressources :
- Documentation JavaFX : https://openjfx.io/javadoc/19/index.html
- SceneBuilder : https://gluonhq.com/products/scene-builder/

Programmation d'un jeu de Morpion en JavaFX

Durée : 4 séances de 2h soit 8h

1. Contexte

Lors de ce TP noté, vous allez concevoir et développer une application JavaFX. L'objectif est de vous faire manipuler les composants, layouts, etc. de JavaFX, afin de produire une application fonctionnelle, et ainsi vous montrer que les connaissances acquises sont applicables.
Pour ce faire, vous utiliserez les différentes ressources utilisées jusqu'ici (cours, TD, documentation Java, etc.).

Seront évalués :
1. Le nombre de fonctionnalités correctement implémentées
2. La qualité globale du code (organisation des fichiers, noms des variables et méthodes, commentaires au sein du code, indentation)
3. Le respect des consignes

Le thème de votre application sera un jeu très connu : le jeu de **Morpion**, ou **Tic-Tac-Toe.**
Il s'agit d'un jeu se jouant à 2 joueurs, chacun son tour, et dont le but est de créer un alignement de symboles : O pour un joueur, et X pour l'autre joueur.
Le jeu de base se joue sur une grille de 3x3 cases, mais de nombreuses variantes existent avec des grilles plus grandes, plus de joueurs, etc.

Il existe de nombreux exemples de morpion sur Internet. Je vous demande de **ne pas copier les exemples que vous pourriez trouver sur Internet** ; le but est justement de vous pousser à appliquer vous-même ce que vous avez retenu des TD. Le plagiat se repère facilement et **sera sanctionné sévèrement.**
Votre assiduité pendant les séances de TP sera prise en compte : si je remarque que vous ne travaillez pas sur votre projet, des sanctions seront appliquées à la note.

2. Instructions pour le rendu

Le code source de votre application et un rapport seront à rendre sur la plateforme UMTice.

Archive

Votre archive devra être de la forme suivante :
- **Nom_Prenom_tpjavafx.zip** (Exemple : DUPONT_Jean_tpjavafx.zip).
Vérifiez le format de l'archive (**merci de n'utiliser que le format ZIP**). Si ce format n'est pas respecté, une pénalité sera appliquée à la note finale (-1).
- A la racine de l'archive : le rapport, le dossier contenant votre code et le .jar de votre application si existant (voir section 3).

Rapport

Le rapport devra impérativement contenir (dans l'ordre suivant) :
- Une **explication succincte de la structure de votre application** : quels choix de séparation de code source vous avez fait, quels fichiers de ressources sont utilisés, etc.
- La liste de **ce que vous avez fait ou non**. Soyez honnête ! Si vous avez commencé à faire quelque-chose et que ça ne fonctionne pas, précisez-le aussi. Essayez d'expliquer les raisons pour lesquelles cela ne fonctionne pas.
- Des **captures d'écran** de toutes vos vues : cela me permettra de voir le résultat que vous obtenez de votre côté dans l'éventualité où je n'arriverais pas à exécuter votre application.
- Eventuellement tous détails nécessaires à la bonne compréhension de votre application : un manuel d'utilisation si votre application est complexe, etc.

Vous êtes bien sûr libre d'ajouter des choses (du contenu, des images, des dessins, etc.), mais il faut au minimum cette base.
Le rapport comptera dans la note finale.

3. Travail à faire

Les points marqués **[OBL]** sont les choses à faire **obligatoirement**. Si tous ces points sont faits correctement, cela ne suffira pas pour obtenir 20/20. Les points marqués **[OPT]** sont des améliorations **optionnelles** ; optionnelles dans le sens **« il ne faut pas toutes les faire »**, mais **il faudra quand même en faire quelques-unes** pour viser la note maximale. Vous êtes libres de proposer d'autres améliorations optionnelles. Si vous le faites, n'oubliez surtout pas de préciser dans le rapport ce que vous avez fait en plus. Si vous ne savez pas si votre idée permettrait d'apporter des points bonus, n'hésitez pas à me demander.

Notes :
- La seule contrainte technique est d'utiliser JavaFX. Vous pouvez utiliser FXML comme vous pouvez programmer en approche procédurale, ou mixer les deux. N'hésitez pas à justifier vos choix dans votre rapport.
- Vous pouvez coder les fonctionnalités dans l'ordre que vous voulez. Par exemple, si la fonctionnalité X vous pose problème, passez à la fonctionnalité Y et revenez à la X plus tard. Ceci vous permettra de perdre le moins de temps possible.
- Essayez d'appliquer un mimium de style à votre application afin de la rendre sympa et la différencier de celle de votre voisin
- Faites bien attention à la propreté de votre code.
- Pensez à commenter votre code pour faciliter la relecture.

Liste des fonctionnalités :
- **[OBL]** Un titre sur chaque fenêtre
- **[OBL]** Plusieurs fenêtres (modales et non modales)
- **[OBL]** Une barre de menu permettant **a minima** de réiniatiliser le jeu, de voir les règles du jeu et de quitter le jeu
- **[OBL]** Des éléments permettant de nommer les deux joueurs
- **[OBL]** Une interface permettant de choisir qui joue en premier : « Joueur 1 », « Joueur 2 », ou choix aléatoire
- **[OBL]** Dans la partie principale de la fenêtre principale : la zone de jeu avec 9 cases (3x3). Le type d'élément utilisé pour les cases est libre (boutons, canvas, ou autre : à vous de voir)
- **[OBL]** A chaque clic dans une case, celle-ci doit afficher le symbole du joueur dont c'est le tour : un O ou un X. Le changement de joueur doit s'effectuer automatiquement après chaque clic
- **[OBL]** A chaque tour de jeu, le joueur dont c'est le tour doit le savoir explicitement par une indication à l'écran (sous la forme de votre choix).
- **[OBL]** La fin de jeu doit se faire automatiquement : lorsqu'un joueur a aligné trois symboles, le jeu doit s'arrêter et indiquer (sous la forme de votre choix) le nom du joueur vainqueur. En cas de match nul (c'est-à-dire si toutes les cases ont été remplies sans avoir trois symboles identiques alignés), l'application doit le mentionner.
- **[OBL]** A la fin du jeu, une interface doit permettre de rejouer ou de quitter l'application. En choisissant « Rejouer », la grille de jeu doit se réinitialiser. Autre option à prendre en compte : garder les mêmes noms de joueurs ou changer.
- **[OBL]** A la fin de chaque partie, établir un tableau de scores montrant le nombre de victoires de chacun des joueurs.

Fonctionnalités optionnelles :

- **[OPT]** A la fin de la partie, montrer l'alignement gagnant par l'effet de votre choix (surbrillance des cases, ligne qui « barre » l'alignement, … : à vous de voir)
- **[OPT]** A la place des symboles textuels X et O, utiliser des images
- **[OPT]** Permettre aux joueurs de personnaliser leur symbole (textuel avec éditeur de police et de couleur, image, autre...)
- **[OPT]** Implémenter un tableau de scores permanent accessible depuis le menu (indice : regardez du côté de la classe Java ***Properties***)
- **[OPT]** Implémenter un mode « Un joueur », permettant de jouer seul face à l'ordinateur :
    - Niveau 1 : l'ordinateur joue aléatoirement dans une case libre
    - Niveau 2 : l'ordinateur choisit le meilleur coup pour tenter de gagner
- **[OPT]** Implémenter la possibilité de choisir la taille de la grille, pour jouer à des variantes à 16 cases, 25 cases, ou plus.
- **[OPT]** Générer un fichier JAR de votre application et l'ajouter à votre archive. Ce format permet de publier des applications Java (par exemple à des utilisateurs finaux).