Module PCD - JavaFX

Préambule

L'objectif des 2 prochaines séances de TP est de se familiariser avec l'architecture MVW et l'API JavaFX en développant le jeu **Boggle**.
Pour mémoire, ce jeu propose un tableau de lettres. Le joueur sélectionne des lettres sur des cases contigües (8 voisins) pour former un mot. Si le mot formé est dans le dictionnaire, le joueur gagne autant de points que de lettres dans le mot ; sinon, il perd un point.

Vous devrez déposer votre travail sur Arche sous forme d'archive exécutable, contenant également les sources, avant le **dimanche 17 novembre 23h59**. Si besoin, la page suivante vous donne de l'aide pour apprendre à créer une archive.
`https://www.jetbrains.com/help/idea/creating-and-running-your-first-java-application.html`

L'architecture de cette application a été étudiée en CM. Il reste à programmer ce qui a été prévu. Cet énoncé vous propose un plan de travail pour résister à la tentation néfaste de tout écrire d'un seul coup et ainsi profiter des bienfaits d'un développement itératif. La bonne méthode de travail consiste à écrire et tester chaque composant l'un après l'autre.

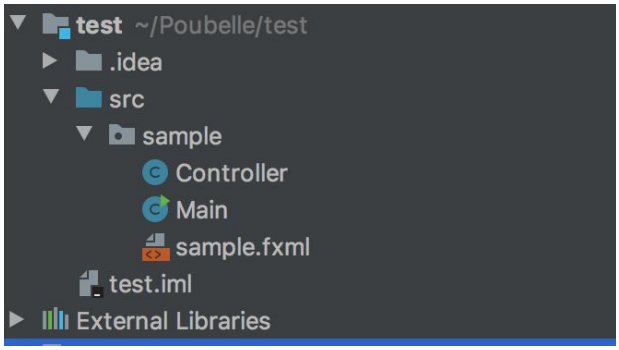
Environnement de développement

Avant toute chose, il faut mettre en place l'environnement de travail. Le développement de l'application se fait via **IntelliJ**.
Ouvrez **IntelliJ** : vous le trouvez dans le menu Programmation ou à défaut en exécutant la commande suivante dans un terminal :

```
/opt/intelliJ/idea/bin/idea.sh
```

Créez un projet JavaFX, portant le nom de votre choix. Il est possible qu'il soit nécessaire de définir le JDK utilisé. Utilisez **`/usr/local/logiciels/java-1.8.`**

La fenêtre de gauche ressemble à la capture ci-contre, en dehors du nom du projet qui vous est propre et n'a guère d'importance.



Un clic droit sur la classe **Main** vous permet de lancer l'exécution. Celle-ci doit se dérouler sans problème et afficher une fenêtre Hello World. Si ce n'est pas le cas, votre environnement à un problème qu'il faut résoudre ….

Modifiez le titre de la fenêtre à votre convenance.

Intégration des ressources fournies sur Arche

Sur Arche, vous trouverez les ressources suivantes (à utiliser en l'état) :

- les fichiers **Boggle.java** et **Dictionnaire.java** : la classe **Boggle** gère le modèle ; la classe **Dictionnaire** gère un dictionnaire que l'on peut construire à partir d'un fichier texte ;
- le dictionnaire de la langue française **dico.txt**

Vous allez adapter la souche créée par défaut par **IntelliJ** et y intégrer les ressources fournies. Dans cette souche, on trouve un répertoire **sample** contenant 3 fichiers : la classe **Controller**, la classe **Main** et un fichier **fxml**. L'adaptation nécessite de :

- renommer le répertoire **sample** pour qu'il porte un nom plus parlant (**boggle**) ; il s'agit du package principal de l'application ;
- supprimer les fichiers **Controller.java** et **sample.fxml**, car on n'utilise pas **fxml** pour développer cette application ;
- dans le répertoire (package) **boggle**, créer les sous-packages **model**, **view** et **ressources** ;
- placer les fichiers **Boggle.java** et **Dictionnaire.java** dans le sous-répertoire **model**.
- placer le fichier **dico.txt** dans le sous-répertoire **ressources** ;
- dans la fonction **start** de la classe **Main.java**, supprimez la ligne suivante (relative à **fxml**).

  Parent root = FXMLLoader.load(getClass().getResource("sample.fxml"));

Après tous ces changements, la compilation échoue … C'est normal, car on a perdu la déclaration de la variable **root** destinée à contenir le composant principal de l'interface graphique, racine de l'arborescence des composants …

Développement de l'application

N'oubliez pas que tout le développement doit être fidèle au diagramme de classes étudié en CM.

1. Pour que la fonction **start** de la classe **Main.java** soit opérationnelle, il faut créer le modèle (instance de **Boggle**) et la racine de l'arborescence de composants graphiques dans la variable **root**.

- Ajoutez l'instanciation du modèle.
- Ajoutez l'instanciation de la racine de l'arborescence graphique (**BorderPane)**.

Cette fois, la compilation n'échoue plus mais l'exécution ouvre une fenêtre vide. C'est normal, car on n'a rien mis dans le panneau principal.

2. Il faut remplir la racine avec ses différents constituants. Comme il est judicieux de construire l'interface graphique pas à pas, on commence par le composant qui affiche les lettres et on ignore les autres pour l'instant.

- Ecrivez la classe **view.VueLettres** ; contentez-vous d'écrire le corps du constructeur qui crée les boutons avec la lettre correspondante définie dans le modèle.
- Complétez la fonction **start** en instanciant cette vue pour la placer au centre du **BorderPane.**

Enfin, l'exécution ouvre une fenêtre avec le panneau des lettres.

3. On complète l'interface par le composant qui affiche les lettres choisies par le joueur au fur et à mesure.

- Ecrivez la classe **view.MotChoisi** ; contentez-vous d'écrire le corps du constructeur qui crée le label.
- Complétez la fonction **start** en instanciant cette vue pour la placer au sud du **BorderPane.**

Cette fois, la fenêtre affiche deux composants.

4. Il reste à rendre l'interface réactive.

- Dans le constructeur de **VueLettres**, attachez un écouteur à chaque bouton, pour prévenir le modèle qu'une nouvelle lettre a été ajoutée.
- Dans le constructeur de **MotChoisi,** enregistrez la vue auprès du modèle, pour qu'elle soit prévenue en cas de changement du modèle.
- Ecrivez la fonction **update** de **MotChoisi** pour rafraîchir le composant lorsque le modèle est modifié.

5. Puisque vous êtes arrivés là, c'est que vous avez compris le principal. Il vous reste à continuer de la même façon. Il faut ajouter le dernier composant. Et aussi apprendre à faire joli, en mettant du style, par le biais de la fonction **setStyle** applicable à tous les composants graphiques. Vous pouvez aussi ajouter un menu, ajouter la liste complète de tous les mots déjà choisis ou encore imaginer toute autre extension de l'application.

Quelques références utiles

Le cours sur Arche …

La documentation officielle de Java 11 SE :
https://docs.oracle.com/javase/11/javase-clienttechnologies.htm

La documentation officielle de l'API JavaFX :
https://docs.oracle.com/javase/11/javafx/api/toc.htm