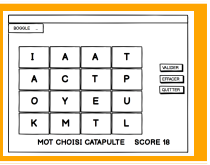
Interfaces Graphiques
TPs 2 et 3 - Boggle



Préambule

L'objectif de cette séance 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.

L'architecture de cette application a été étudiée en CM. Il reste à programmer ce qui a été prévu, en vous reportant aux explications du cours. 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.

Intégration des ressources fournies sur Arche

Vous disposez sur **gitlab** d'un dépôt que vous devez utiliser pour ce nouveau projet.

Créez un nouveau projet Java (et surtout pas JavaFX) dans IntelliJ, et intégrez les ressources suivantes (fournies sur Arche) :

- 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**

Développement de l'application

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

1. Écrivez le texte de l'interface **Observateur.** 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)**.

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.

- Écrivez 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.**

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.

- Écrivez la classe **view.VueInfos** ; 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.**

L’exécution ouvre une fenêtre avec le panneau des lettres et les informations relatives au jeu en cours.

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 **VueInfos,** enregistrez la vue auprès du modèle, pour qu’elle soit prévenue en cas de changement du modèle.
- Écrivez la fonction **re** de **VueInfos** 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 pour ajouter le dernier composant.

6. Il est temps de penser aux extensions pour apprendre à utiliser d’autres composants et/ou d’autres fonctionnalités.

- Mettre du style, par le biais de la fonction **setStyle** applicable à tous les composants graphiques.
- Ajouter un menu pour sauvegarder/reprendre une partie en cours, commencer une nouvelle partie en choisissant la taille du plateau de jeu.
- Ajouter sur l’interface la liste complète de tous les mots déjà choisis.
- Ajouter les lettres accentuées dans le modèle.
- Proposer à l’utilisateur de jouer avec un dictionnaire français ou anglais ou espagnol ou …

Vous pouvez imaginer toute autre extension de l’application, en respectant toujours le même principe de construction.