

TP1 – Super Mots-Croisés Master 3000

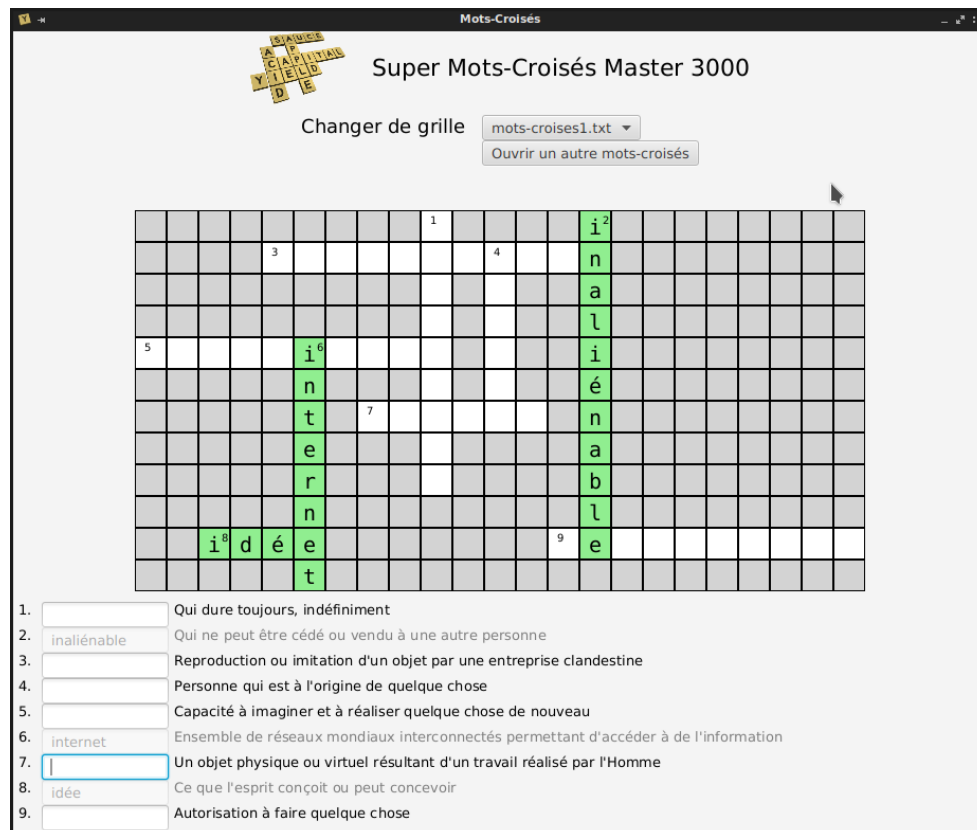
SIM – A22 – Développement de programmes dans un environnement graphique

Nicolas Hurtubise

➤ Ne commencez pas le TP avant d'avoir lu toutes les sections, en particulier **Organisation du code** et **Gradle** à la fin!

Contexte

Pour votre premier gros programme JavaFX, votre tâche sera de coder un jeu de **mots-croisés**.



Plus spécifiquement, vous devrez coder **deux versions** du programme : une en ligne de commande, et une autre en interface graphique avec JavaFX.

Votre tâche

Dans le `.zip` disponible sur Léa, je vous fournis des fichiers pour représenter des *grilles de mots-croisés*. Vous devrez créer deux interfaces pour permettre d'y jouer : une en ligne de commande, une avec JavaFX.

Je m'attends à ce que votre programme soit professionnel, au sens où il doit être **robuste**. Si on essaie d'utiliser une grille qui n'existe pas, si on essaie d'ouvrir un fichier qui ne contient pas de grille valide, ou si on essaie d'ouvrir un mots-croisés mal défini (ex.: avec des mots qui se croisent sur des lettres différentes), le programme devrait **nous aviser du problème sans que ça ne plante**.

Format des fichiers

Les fichiers de mots-croisés valides doivent respecter le format suivant :

```
# Possiblement des lignes au début qui commencent par des dièses
# Ces lignes constituent des commentaires
# et devraient être ignorées
Infos du mot 1
Infos du mot 2
Infos du mot 3
Infos du mot 4
```

Les infos sur les mots à deviner sont séparées sur la ligne par le caractère ':'

```
mot:numeroLigne:numeroColonne:H ou V:Indice à afficher pour deviner
```

Le mot lui-même est en premier, suivi des `numeroLigne` et `numeroColonne` dans la grille.

Notez : la grille a des numéros de ligne qui vont de **haut en bas**, des numéros de colonnes qui vont de **gauche à droite** et ces valeurs sont des nombres entiers **qui commencent à zéro** (comme pour les tableaux).

Notez également : **un fichier qui a l'air d'une grille pourrait très bien contenir quelque chose d'invalid**e. Le concept des mots-croisés est que les mots se croisent sur une même lettre : si on se retrouve avec des mots qui essaient de se croiser sur des lettres différentes, il s'agit d'un fichier erroné.

Je vous fournis des exemples de grilles invalides à tester avec votre programme pour valider que ça ne cause pas de bugs.

Version 1 : en ligne de commande

Vous devrez faire une première version du jeu dans laquelle la grille de mots et les descriptions sont affichées en ligne de commande.

La partie en ligne de commande se déroule de la façon suivante:

```
TANT QUE la grille n'est pas complétée :
  Afficher la grille
  Afficher les descriptions à entrer
  Demander quel mot on veut deviner
    On doit entrer un des numéros qui n'a pas encore été deviné,
      puis écrire notre réponse sur la ligne de commande
    On peut entrer `s` pour avoir la solution
    On peut entrer `q` pour quitter le jeu
```

La version en ligne de commande ne permet pas de choisir quel fichier on ouvre (par défaut, ça devrait être `mots-croises1.txt` à moins qu'on le change dans le code), et ne permet de jouer qu'une seule partie.

La façon d'afficher la grille dans la console est la suivante :

- On met un . pour identifier une case vide
- On met un ? pour identifier une case pas encore devinée, à l'exception de la première case d'un *mot* qui devrait plutôt contenir le numéro du mot entre 1 et 9
- Pour les cases appartenant à des mots déjà devinés, on affiche les lettres

Pour rendre la grille visuellement plus jolie, on devrait mettre un espace entre chaque case.

Voici de quoi devrait avoir l'air la grille de `mots-croises1.txt` au tout début :

```
. . . . . 1 . . . . 2 . . . . .
. . . . 3 ? ? ? ? ? 4 ? ? ? . . . . .
. . . . . ? . ? . ? . . . . .
. . . . . ? . ? . ? . . . . .
5 ? ? ? ? 6 ? ? ? ? . ? . ? . . . . .
. . . . . ? . . ? . ? . ? . . . . .
. . . . . ? . 7 ? ? ? ? ? . ? . . . . .
. . . . . ? . . ? . . . ? . . . . .
. . . . . ? . . ? . . . ? . . . . .
. . . . . ? . . . . . ? . . . . .
. . 8 ? ? ? . . . . . 9 ? ? ? ? ? ? ? ?
. . . . . ? . . . . . . . . . . .
```

La même grille après avoir deviné quelques mots :

```

. . . . . p . . . . i . . . . .
. . . . c o n t r e f a ç o n . . . . .
. . . . . r . u . . a . . . . .
. . . . . p . t . . l . . . . .
5 ? ? ? ? 6 ? ? ? é . e . . i . . . . .
. . . . . ? . . . t . u . . é . . . . .
. . . . . ? . 7 ? u ? r ? . n . . . . .
. . . . . ? . . . e . . . . a . . . . .
. . . . . ? . . . l . . . . b . . . . .
. . . . . ? . . . . . l . . . . .
. . 8 ? ? ? . . . . . 9 e ? ? ? ? ? ? ? ?
. . . . . ? . . . . . . . . . . . . .

```

La grille à la toute fin (ou si on affiche la solution) :

```

. . . . . p . . . . i . . . . .
. . . . c o n t r e f a ç o n . . . . .
. . . . . r . u . . a . . . . .
. . . . . p . t . . l . . . . .
c r é a t i v i t é . e . . i . . . . .
. . . . . n . . . t . u . . é . . . . .
. . . . . t . o e u v r e . n . . . . .
. . . . . e . . . e . . . . a . . . . .
. . . . . r . . . l . . . . b . . . . .
. . . . . n . . . . . l . . . . .
. . i d é e . . . . . p e r m i s s i o n
. . . . . t . . . . . . . . . . . . .

```

Voici un exemple du déroulement du jeu en ligne de commande :

```

. . . . . 1 . . . . 2 . . . . .
. . . . 3 ? ? ? ? ? 4 ? ? ? . . . . .
. . . . . ? . ? . ? . . . . .
. . . . . ? . ? . ? . . . . .
5 ? ? ? ? 6 ? ? ? ? . ? . ? . . . . .
. . . . . ? . . . ? . ? . ? . . . . .
. . . . . ? . 7 ? ? ? ? ? . ? . . . . .
. . . . . ? . . . ? . . . . ? . . . . .
. . . . . ? . . . ? . . . . ? . . . . .
. . . . . ? . . . . . . ? . . . . . .
. . 8 ? ? ? . . . . . . 9 ? ? ? ? ? ? ? ? ?
. . . . . ? . . . . . . . . . . . . .
1. Qui dure toujours, indéfiniment
2. Qui ne peut être cédé ou vendu à une autre personne
3. Reproduction ou imitation d'un objet par une entreprise clandestine
4. Personne qui est à l'origine de quelque chose
5. Capacité à imaginer et à réaliser quelque chose de nouveau
6. Ensemble de réseaux mondiaux interconnectés permettant d'accéder à de l'information
7. Un objet physique ou virtuel résultant d'un travail réalisé par l'Homme
8. Ce que l'esprit conçoit ou peut concevoir
9. Autorisation à faire quelque chose
Quel mot voulez-vous deviner?
(q pour quitter, s pour avoir la solution)
2
Tentative : inaliénable
Bonne réponse!
. . . . . 1 . . . . i . . . . .
. . . . 3 ? ? ? ? ? 4 ? ? n . . . . .
. . . . . ? . ? . a . . . . .
. . . . . ? . ? . l . . . . .
5 ? ? ? ? 6 ? ? ? ? . ? . i . . . . .
. . . . . ? . . . ? . ? . é . . . . .
. . . . . ? . 7 ? ? ? ? ? . n . . . . .
. . . . . ? . . . ? . . . a . . . . .
. . . . . ? . . . ? . . . b . . . . .
. . . . . ? . . . . . . l . . . . .
. . 8 ? ? ? . . . . . . 9 e ? ? ? ? ? ? ? ?
. . . . . ? . . . . . . . . . . . . .
1. Qui dure toujours, indéfiniment
3. Reproduction ou imitation d'un objet par une entreprise clandestine
4. Personne qui est à l'origine de quelque chose
5. Capacité à imaginer et à réaliser quelque chose de nouveau
6. Ensemble de réseaux mondiaux interconnectés permettant d'accéder à de l'information
7. Un objet physique ou virtuel résultant d'un travail réalisé par l'Homme
8. Ce que l'esprit conçoit ou peut concevoir
9. Autorisation à faire quelque chose
Quel mot voulez-vous deviner?
(q pour quitter, s pour avoir la solution)
... etc

```

Version 2 : JavaFX

Dans la version JavaFX, on devrait avoir plus d'options :

- À tout moment, on peut écrire des choses dans une boîte de réponse pour un des mots
- On peut soit jouer avec une grille fournie avec le programme, ou soit charger notre propre fichier de jeu
- À tout moment, on doit pouvoir **appuyer sur Escape pour fermer le jeu**.

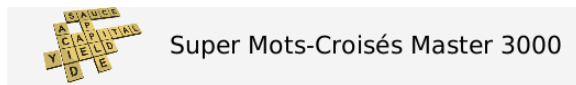
Utilisez `Platform.exit()`; pour quitter l'application JavaFX quand on appuie sur Escape.

Interface graphique

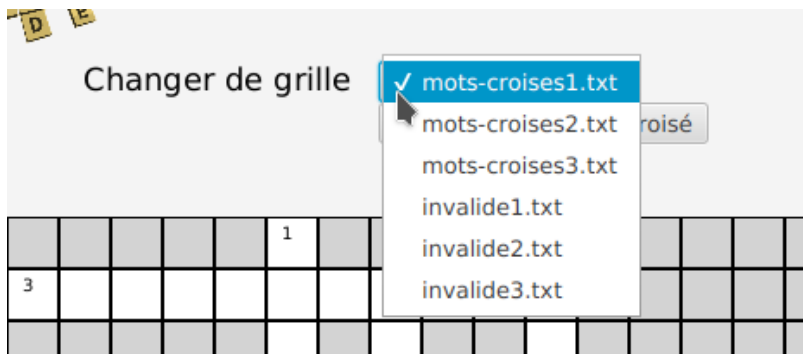
L'interface graphique doit être similaire à celle qui est présentée sur la première page de l'énoncé.

Il vous faut :

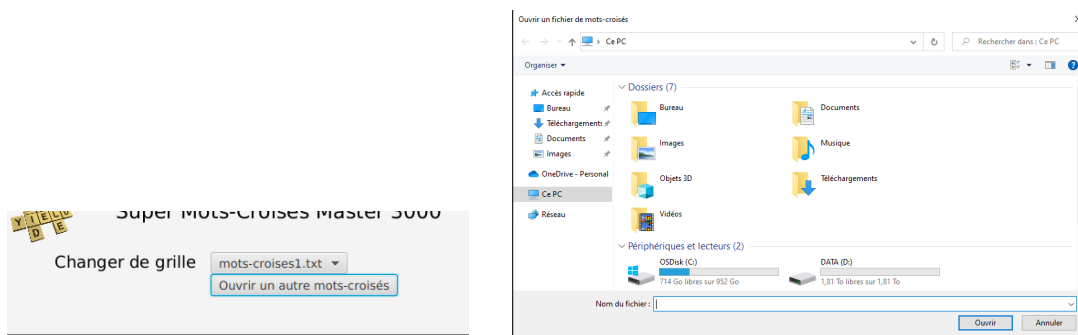
- Une barre de titre en haut, avec le logo (fichier `mots.png` fourni) et le titre








- Un petit menu pour changer de grille sur demande. On peut soit choisir une des grilles fournies dans le menu :



Ou soit cliquer sur le bouton Ouvrir un autre mots-croisés pour sélectionner un fichier sur notre disque.



- En dessous de tout ça, on devrait avoir **la grille de mots-croisés chargée**, avec :
 - En blanc, les cases à remplir 
 - En gris, les cases vides 
 - En vert, les cases qui ont déjà été devinées 
 - Un mini-texte dans chaque case qui débute un mot pour indiquer son numéro  
- Finalement, en bas, **la liste des mots à deviner** avec
 - Le numéro du mot, pour repérer où ça commence dans la grille
 - Un `TextField` pour permettre d'entrer une tentative
 - La description du mot à droite


1.	<input type="text"/>	Qui dure toujours, indéfiniment
2.	<input type="text" value="inaliénable"/>	Qui ne peut être cédé ou vendu à une autre personne
3.	<input type="text"/>	Reproduction ou imitation d'un objet par une entreprise clandestine
4.	<input type="text"/>	Personne qui est à l'origine de quelque chose
5.	<input type="text"/>	Capacité à imaginer et à réaliser quelque chose de nouveau
6.	<input type="text" value="Internet"/>	Ensemble de réseaux mondiaux interconnectés permettant d'accéder à de l'information

Pour essayer un mot, on entre le mot dans le `TextField` correspondant et on appuie sur **Enter**. Cela aura pour effet de déclencher l'événement défini dans le `.setOnAction()` du `TextField`.

Notez que les mots qui ont déjà été devinés ne devraient pas être *re-devinables*. Une fois le mot deviné, le `TextField` devrait être *désactivé*, trouvez comment faire ça.

- Quand la grille est complétée, on devrait afficher un petit message qui félicite la personne



- Dernière précision : assurez-vous de mettre une **icône**  et de **donner un titre** ("Mots-Croisés") à la fenêtre de votre programme

Affichage de la grille

Pour afficher la grille, je vous recommande fortement d'utiliser un **GridPane** contenant $N \times M$ **HBox**.

Voici une ébauche de code pour vous aider à avoir la même apparence que moi :

```
// Crée la cellule
var cellule = new HBox();
cellule.setPadding(new Insets(3, 8, 3, 8)); // Un peu de marge autour du contenu
cellule.setMaxSize(30, 30); // Force la taille de chaque cellule à 30x30 pixels
cellule.setMinSize(30, 30);

// Donne une bordure noire et une couleur d'arrière-plan à la cellule
cellule.setBorder(new Border(new BorderStroke(Color.BLACK,
    BorderStrokeStyle.SOLID, CornerRadii.EMPTY, BorderWidths.DEFAULT)));
Color color = Color.WHITE;
cellule.setBackground(new Background(new BackgroundFill(color, null, null)));

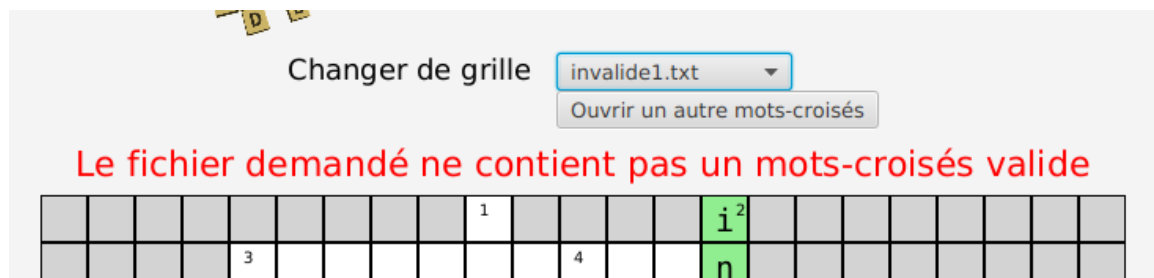
// Ajoute une lettre du mot qui va aller dans la case
Text lettre = new Text("i");
lettre.setFont(Font.font("monospace", 20));
cellule.getChildren().add(lettre);

// Le petit numéro à afficher si cette case est le début d'un mot
var numero = new Text("2");
numero.setFont(Font.font(10));
cellule.getChildren().add(numero);
```

Robustesse

Dans cette version, je m'attends à ce que votre programme soit robuste aux gens qui font n'importe quoi. Puisqu'on peut choisir le fichier qu'on veut pour l'ouvrir en mots-croisés, on pourrait très bien tenter d'ouvrir un fichier invalide.

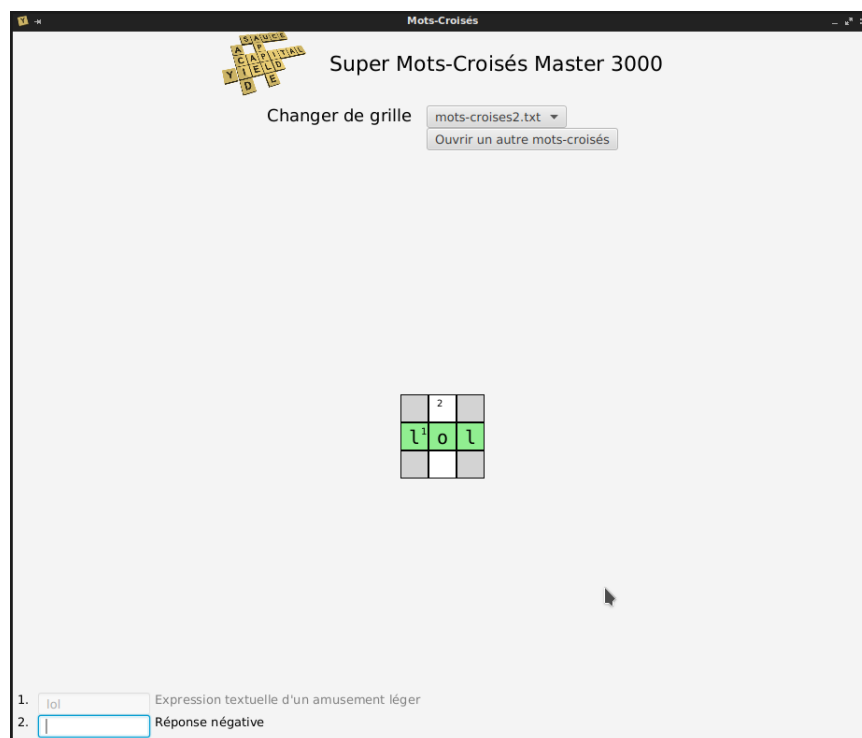
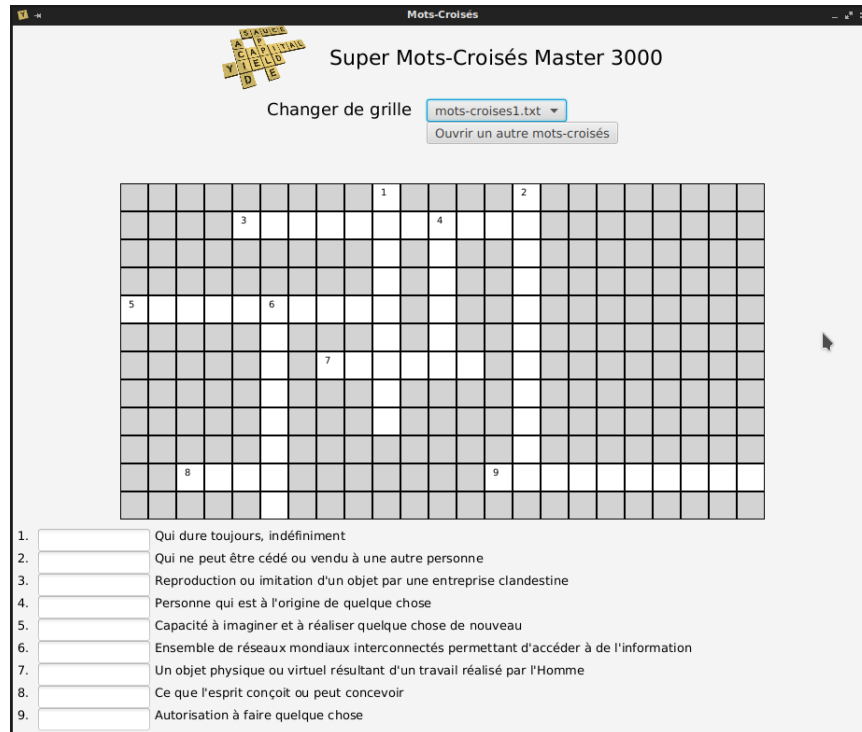
Dans un tel cas, on devrait afficher un message d'erreur :



Le code ne devrait **jamais** causer d'erreurs dans la console.

Exemples d'états de l'application

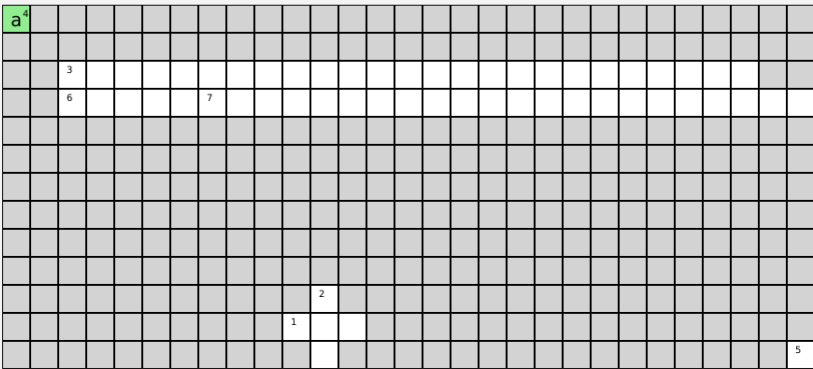
Voici quelques captures d'écran de mon programme à différents moments



Mots-Croisés

Super Mots-Croisés Master 3000

Changer de grille mots-croises3.txt
[Ouvrir un autre mots-croisés](#)



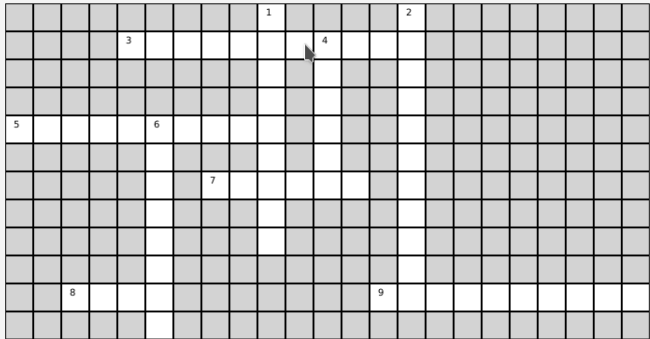
- Expression textuelle d'un amusement léger
- Réponse négative
- Mot le plus long de la langue française
- Première lettre de l'alphabet
- Dernière lettre de l'alphabet
- Mot encore plus long que le plus long de la langue française
- Diriger le pays

Mots-Croisés

Super Mots-Croisés Master 3000

Changer de grille mots-croises1.txt
[Ouvrir un autre mots-croisés](#)

Le fichier demandé ne contient pas un mots-croisés valide



- Qui dure toujours, indéfiniment
- Qui ne peut être cédé ou vendu à une autre personne
- Reproduction ou imitation d'un objet par une entreprise clandestine
- Personne qui est à l'origine de quelque chose
- Capacité à imaginer et à réaliser quelque chose de nouveau
- Ensemble de réseaux mondiaux interconnectés permettant d'accéder à de l'information
- Un objet physique ou virtuel résultant d'un travail réalisé par l'Homme
- Ce que l'esprit conçoit ou peut concevoir
- Autorisation à faire quelque chose

Organisation du code

Le découpage du code devrait avoir l'air de :

- Une classe `MainJavaFX`, qui contient...
 - La déclaration des composantes graphiques pour afficher la grille de mots et les définitions
 - La déclaration des événements
 - Une fonction pour **redessiner la grille** à mesure qu'on découvre des nouveaux mots
- Une classe `MainCmd`, qui contient...
 - La boucle de jeu (tant qu'il reste des mots à trouver, on affiche la grille, on demande quel mot on veut deviner, etc)
 - Une fonction pour **dessiner la grille** en ligne de commande
- Des classes pour la logique d'un `MotsCroises` en général
 - Une classe qui représente un mots-croisés
 - Des fonctions pour construire la grille à partir d'un fichier fourni
 - Des façons de valider si le fichier demandé est bon ou mauvais
 - D'autres classes pour vous aider si jamais vous en avez besoin

Vous **devez** réutiliser un maximum de code entre les deux versions du jeu.

Comme vous l'avez peut-être remarqué, un projet JavaFX contient rapidement beaucoup de code à cause des déclarations de composantes, des événements, des configurations, ...

Pour éviter d'avoir du code complètement incompréhensible, **on va deux interfaces (JavaFX et console) de la logique**.

Vous devriez avoir un **objet MotsCroises** qui gère toute la *logique* de la grille, sans se soucier de comment ça sera affiché par JavaFX ou par la ligne de commande.

La classe `MotsCroises` devrait gérer **TOUT** ce qui a rapport aux mots dans la grille, entre autres :

- Noter les descriptions et les positions de chaque mot
- Déterminer la taille totale de la grille
- Noter quelles cases contiennent quelles lettres
- Noter quels mots ont été devinés correctement et quels restent à deviner
- Valider si la grille qu'on essaie de charger est correcte ou si elle comporte des erreurs
- Déterminer si on a complété le mots-croisés ou non

Voici une esquisse de ce dont pourrait avoir l'air cette classe :

```
public class MotsCroises {
    // Attributs
    // ... à vous de choisir ...

    public /*...*/ getDescription(int numeroMot) {
        /* Donne la description du mot pour
           permettre aux gens de la deviner */
    }

    public /*...*/ etatCase(int ligne, int colonne) {
        /* Indique si la case contient est vide, si elle
           contient une lettre devinée, ou si elle contient
           une lettre à trouver */
    }

    public char getLettre(int ligne, int colonne) {
        /* Indique la lettre contenu à une certaine case */
    }

    public boolean estComplet() {
        /* Indique si oui ou non le mots-croisés est
           complété, donc si tous les mots ont été devinés */
    }

    // Autres méthodes
    // ...
}
```

Votre classe `MotsCroises` doit contenir **TOUTE** la logique relative à la grille de jeu et **RIEN** par rapport à l'affichage (console ou JavaFX). Vous pouvez vous faire d'autres classes pour vous aider au besoin.

Tout ce qui a rapport au fonctionnement de la grille de mots devrait se trouver dans cette classe. Si vous suivez cette façon de faire à la lettre, ça va **beaucoup** simplifier votre code.

Code en ligne de commande

Le code pour lancer le jeu en ligne de commande ne devrait pas être très compliqué et sert surtout à vous aider à tester votre jeu *avant* de vous lancer dans l'interface graphique.

```

public class MainCmd {
    public static void main(String[] args) {
        MotsCroises partie = new MotsCroises(...);

        [boucle pour afficher la grille et lire les commandes au clavier]
    }

    public static void dessinerGrille(MotsCroises partie) {
        // ...
    }

    public static void afficherDescriptions(MotsCroises partie) {
        // Affiche les descriptions des mots qui restent
        // à deviner
    }
}

```

Le code de cette classe devrait être tout petit (moins d'une centaine de lignes), puisqu'il devrait contenir **seulement la logique des interactions en ligne de commande**. À part des `Scanner` et des `System.out.println` et de la gestion de questions/réponses entrées au clavier, le code de cette classe ne fera pas grand chose.

Par exemple : ce n'est **pas** le rôle de `MainCmd` de se rappeler de quelles cases du mots-croisés ont déjà été devinées. Ce n'est **pas** non plus son rôle de lire un fichier pour déterminer si c'est un mots-croisés bien construit ou s'il est invalide.

Si je veux essayer de deviner un mot, je devrais demander à mon objet `MotsCroises` si ça a marché ou non, je ne devrais **pas** parcourir tous les mots dans le `MainCmd` et essayer de changer la grille depuis cette classe.

Code JavaFX

Pour garder le code simple, je vous recommande **fortement** de faire la chose suivante.

Plutôt que de définir :

Quand on devine un mot, on modifie les cases de ce mot

On devrait plutôt dire :

*Quand on devine un mot, on **reconstruit la grille au complet** à partir du nouvel état de l'objet `MotsCroises`*

Ça va vous permettre de coder une seule fois la fonction qui dessine une grille de mots-croisés à partir d'un objet `MotsCroises` en fonction de quelles cases sont ouvertes/vides/devinées, et ça devrait **beaucoup** vous faciliter la vie.

(Sérieusement, si vous ne faites pas ça, vous risquez de vous retrouver avec des cas bizarres qui fonctionnent mal et qui sont difficiles à tester!)

Gradle

➤ N'EXÉCUTEZ **PAS** VOTRE CODE SANS PASSER PAR GRADLE

Gradle est un outil très chouette, qui permet entre autres de définir *plusieurs façons d'exécuter le projet*.

Créez un package : `ca.qc.bdeb.inf203.tp1`

Et mettez-y **deux** classes différentes qui vont chacune avoir un `main()` :

- `MainCmd.java`, qui va démarrer votre projet en ligne de commande
 - Cette classe contient un `public static void main(String[] args)` comme vous êtes habitués de faire
- `MainJavaFX.java`, qui va contenir votre code pour démarrer le projet en JavaFX
 - Cette classe doit `extends Application` et définir votre méthode `start()` pour afficher la fenêtre

Pour expliquer à Gradle les deux façons de lancer votre programme, utilisez le fichier `build.gradle` suivant (disponible dans le `.zip` fourni sur Léa) :

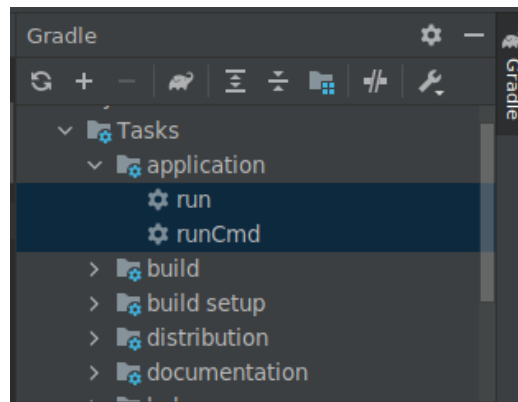
```
plugins {  
    id 'application'  
    id 'org.openjfx.javafxplugin' version '0.0.9'  
}  
  
repositories {  
    mavenCentral()  
}  
  
dependencies {  
}  
  
compileJava.options.encoding = 'UTF-8'  
  
javafx {  
    version = "16"  
    modules = ['javafx.controls']  
}  
  
mainClassName = 'ca.qc.bdeb.inf203.tp1.MainJavaFX'  
  
task(runCmd, dependsOn: 'classes', type: JavaExec) {  
    mainClass.set('ca.qc.bdeb.inf203.tp1.MainCmd')  
    classpath = sourceSets.main.runtimeClasspath  
    standardInput = System.in  
}  
  
configure(runCmd) {  
    group = 'application'  
    description = 'Lancer en ligne de commande'  
}
```

➤ N'EXÉCUTEZ **PAS** VOTRE CODE SANS PASSER PAR GRADLE

Passez toujours par le menu **Gradle** à droite dans IntelliJ

Avec ce fichier Gradle, vous aurez deux configurations :

- Tasks > Application > run pour lancer JavaFX
- Tasks > Application > runCmd pour lancer le jeu en ligne de commande



Remise

Vous devez remettre sur Léa votre projet IntelliJ (avec la configuration Gradle, le code, les ressources, etc.) **dans un fichier .zip**

La date de remise est spécifiée sur Léa.

Barème

- 60% : Fonctionnalités demandées implantées correctement
 - Respect des consignes
 - (30%) Interface JavaFX : définition de l'interface graphique et des événements
 - (10%) Interface en ligne de commande *qui ne plante pas même si je rentre n'importe quoi*
 - (20%) **Pas d'erreurs, même si je fais des choses invalides!**
Gérez les exceptions correctement
- 40% : Qualité du code
 - Code bien commenté
 - Respect du minusculeCamelCase pour les variables/méthodes, MajusculeCamelCase pour les noms de classes
 - Bon découpage en méthodes
 - Découpage demandé
Vues vs Modèle (classe `MotsCroises` bien séparée du reste)
 - Encapsulation : attributs `private`, avec getters/setters au besoin
 - **Pas de variables globales!**
 - **Pas de copier-coller de code!**

Note sur le plagiat

Le travail est à faire **individuellement**. Ne *partagez pas de code*, même pas “juste pour aider un ami”, ça constituerait un **plagiat**, et les deux personnes auraient la note de *zéro*.

Références

Les images fournies pour ce TP proviennent de openclipart.org