

DEV1 – JAVL – Laboratoires Java**TD 12 – Mise en pratique : Le mot le plus long**

Dans ce TD vous utiliserez les notions vues précédemment afin de réaliser le jeu « le mot le plus long ».



Les codes sources et les solutions de ce TD se trouvent à l'adresse :
<https://git.esi-bru.be/dev1/labo-java/tree/master/td12-mot-le-plus-long/>

Table des matières

1 Le mot le plus long

2

1 Le mot le plus long

Le mot le plus long est un jeu de lettres à deux joueurs qui se déroule en plusieurs manches. Le nombre de manches est choisi au départ, par exemple 5 manches. Une manche se déroule en 2 étapes.

Tout d'abord les joueurs, chacun à leur tour, tirent au hasard soit une voyelle, soit une consonne (le joueur choisit), jusqu'à obtenir 9 lettres. A chaque fois qu'une lettre est tirée (voyelle ou consonne), la lettre est dévoilée aux joueurs afin qu'ils puissent continuer en connaissance de cause.

Ensuite les joueurs cherchent le mot le plus long avec les lettres disponibles. Chaque joueur annonce la longueur du mot trouvé. Celui qui a trouvé le mot le plus long le forme avec les lettres disponibles. On vérifie alors que c'est bien un mot du dictionnaire. Le joueur gagne autant de points que le nombre de lettres de son mot.

Nous allons vous guider tout au long de ce TD afin de développer les différentes méthodes nécessaires pour une version simplifiée à un seul joueur de ce jeu.

Tout d'abord, créez une classe `LeMotLePlusLong` que vous placez dans le package `g12345.dev1.lemotlepluslong`. Créez aussi la méthode principale qui, dans un premier temps, affiche un message de bienvenue au 'mot le plus long'.

Exercice 1 Hasard

Écrivez la méthode `int tirerHasard(int min, int max)` qui retourne un nombre au hasard compris entre `min` et `max` reçus en paramètre.

Par exemple si la méthode reçoit 3 et 8, elle retourne 6.

Astuce : utilisez la méthode `random` de la classe `Math` qui retourne un nombre réel entre 0 (compris) et 1 (non-compris).

Exercice 2 Voyelle

Écrivez la méthode `char tirerVoyelle()` qui retourne une voyelle tirée au hasard.

Cette méthode fera un appel à la méthode `tirerHasard` de l'exercice précédent.

Exercice 3 Consonne

Écrivez la méthode `char tirerConsonne()` qui retourne une consonne tirée au hasard.

Exercice 4 Afficher les lettres

Écrivez la méthode `void afficherLettres(char[] lettres)` qui affiche le contenu du tableau `lettres`.

Par exemple si la méthode reçoit le tableau

```
{ 'E', 'E', 'I', 'O', 'C', 'D', 'M', 'R', 'Z' },
```

elle doit afficher :

```
E E I O C D M R Z
```

Cette méthode permettra d'afficher les lettres tirées par les joueurs.

Exercice 5 Demander les lettres

Écrivez la méthode `char[] demanderLettres()` qui demande à l'utilisateur s'il veut une voyelle ou une consonne et qui, en fonction de la réponse, tire une voyelle ou une consonne. La méthode affiche les lettres déjà tirées et répète la demande et le tirage 9 fois.

La méthode retourne un tableau de 9 caractères contenant les lettres tirées.

Intégrez cela dans la méthode principale, c'est-à-dire, ajoutez un appel à la méthode `demanderLettres` dans la méthode principale.

Exercice 6 Demander un mot

Écrivez la méthode `String demanderMot()` qui demande à l'utilisateur le mot le plus long qu'il a trouvé.

La méthode retourne le mot donné par l'utilisateur.

Intégrez cela dans la méthode principale.

Exercice 7 Vérifier les lettres

Écrivez la méthode `boolean vérifierLettres(char[] lettres, String mot)` qui vérifie que le mot proposé est possible avec les lettres disponibles.

Attention : vous ne pouvez pas modifier le tableau `lettres`.

Astuce : vous pouvez utiliser un tableau de travail, par exemple un tableau de booléens qui indiquera si une lettre a déjà été utilisée.

Cette méthode n'est pas triviale, elle doit être validée par des tests. Ajoutez des tests JUnit en prenant soin de tester les cas particuliers : utilisation de la même lettre mais disponible une seule fois, utilisation de la même lettre disponible plusieurs fois, utilisation d'une lettre non disponible, etc.

Complétez la méthode principale.

Exercice 8 Mot du dictionnaire

Écrivez la méthode `boolean dansDictionnaire(String mot)` qui vérifie que le mot proposé se trouve dans le dictionnaire.

Pour réaliser cet exercice, nous vous fournissons une librairie qui contient un dictionnaire, ou plus exactement la liste de tous les mots acceptés par le jeu. Cette librairie vous est fournie : devez l'ajouter à votre projet sous forme de dépendance.

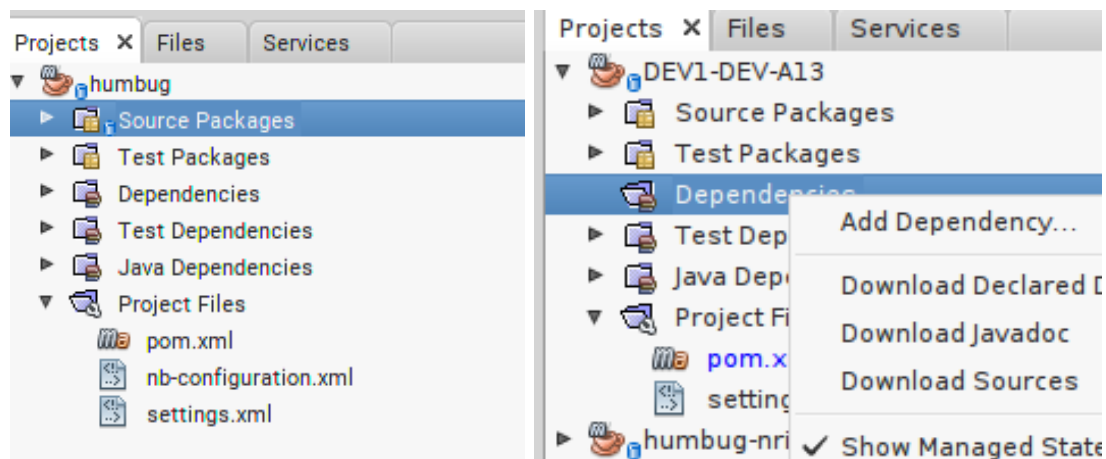


FIGURE 1 – Le fichier pom.xml (à gauche), et le menu pour ajouter une dépendance (à droite)

Qu'est-ce qu'une dépendance ?

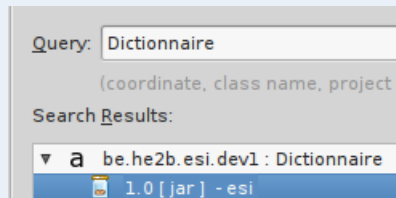
Un *dépendance* est du code dont votre projet *dépend*. Votre projet va *dépendre* du dictionnaire pour son fonctionnement. L'idée de dépendance est comprise par l'outil Maven via le fichier pom.xml.

Le fichier pom.xml ?

C'est un fichier de configuration de Maven. Vous le trouverez dans Netbeans sous *Project Files* (voir figure 1 page précédente, à gauche). Prenez quelques secondes pour essayer de comprendre le format de ce fichier : c'est un fichier XML ^a.

Ajouter la dépendance

1. Ajoutez d'abord le dépôt des librairies de l'ESI en ajoutant le code de la figure 2 dans la balise `project` (par exemple juste avant la ligne `</project>`). Vérifiez que Netbeans ne souligne rien en rouge, puis sauvez.
2. Ensuite, après un clic droit sur *Dependencies*, cliquez sur *Add dependency* (voir figure 1 page précédente, à droite). Une fenêtre s'ouvre. Dans *Query*, indiquez *Dictionnaire* et appuyez sur *Enter*. Rapidement, le résultat suivant doit s'afficher ; sélectionnez-le et confirmez :



C'est fait ! Re-lancez votre projet pour vérifier qu'aucune erreur n'est apparue.

^a. des balises qui s'ouvrent, possèdent (parfois) des attributs, ont du contenu, et se ferment. `<balise attribut="valeur">contenu</balise>` En outre, l'indentation permet de visualiser le fait que certaines balises sont "contenues" dans d'autres.

Une fois la dépendance ajoutée à votre projet, comment l'utiliser ? La librairie que nous vous fournissons contient une seule classe : la classe `Dictionnaire` qui se trouve dans le package `esi.dev1.util`. Cette classe possède une méthode `String[] mots()` qui retourne un tableau contenant tous les mots du dictionnaire. Pour l'utiliser il suffit donc d'appeler cette méthode et de récupérer une référence vers ce tableau de mots :

```
String[] dico = Dictionnaire.mots();
```

Vous avez maintenant un tableau contenant tous les mots du dictionnaire. Par exemple `System.out.println(dico[58]);` affichera le 59^e mot de ce dictionnaire.

```
<repositories>
  <repository>
    <id>esi</id>
    <name>esi</name>
    <url>https://maven.esi-bru.be/</url>
  </repository>
</repositories>
```

FIGURE 2 – Code à ajouter au fichier pom.xml

Exercice 9 Le meilleur mot

Écrivez la méthode `String meilleurMot(char[] lettres)` qui parcourt le dictionnaire à la recherche du mot le plus long faisable avec les lettres disponibles. Si plusieurs mots sont possibles on en choisit ici un seul.

Testez votre méthode avec plusieurs tests JUnit.

Intégrez cela dans la méthode principale.

Exercice 10 Bonus 1 - Les meilleurs mots

Écrivez la méthode `String[] meilleursMots(char[] lettres)` qui retourne la liste de tous les mots les plus longs du dictionnaire faisables avec les lettres disponibles.

Testez votre méthode avec plusieurs tests JUnit.

Intégrez cela dans la méthode principale.

Exercice 11 Bonus 2 - 2 joueurs

Développez le jeu à plusieurs manches, avec 2 joueurs, ainsi que la gestion du score et du gagnant. Pour cela vous pouvez vous référer aux règles officielles disponibles sur internet¹.

Exercice 12 Bonus 3 - Fréquence des lettres

Pour que le jeu soit agréable il faudrait que la fréquence des différentes lettres ne soit pas uniforme. En d'autres mots, il est préférable d'avoir plus souvent un 'e' ou un 'a' qu'un 'y', un 's' ou un 'z'. Modifiez vos méthodes `voyelle` et `consonne` afin d'ajuster la probabilité des différentes lettres.

Vous pouvez vous baser par exemple sur la fréquence des lettres en français : https://fr.wikipedia.org/wiki/Fr%C3%A9quence_d%27apparition_des_lettres_en_fran%C3%A7ais

1. Par exemple : <https://www.trictrac.net/jeu-de-societe/le-mot-le-plus-long-3/details>