Haute École de Bruxelles-Brabant
École Supérieure d'Informatique
Bachelor en Informatique

2018 – 2019
DEV1

DEV1 – Laboratoires Java I

TD 11 – 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'.

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, 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 à un seul joueur de ce jeu.

Créer un package `g12345.dev1.lemotlepluslong` et une classe `LeMotLePlusLong`. Créer 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 hasard(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 par exemple 6.

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

Exercice 2 Voyelle

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

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

Exercice 3 Consonne

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

Exercice 4 Afficher les lettres

Version du document
5 novembre 2018

Écrivez la méthode `void afficherLettres(char[] lettres)` qui affiche les lettres disponibles.

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 appelle 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 utilisé un tableau de travail, par exemple un tableau de booléens qui indiquera si une lettre à déjà été utilisé.

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.

Complètez 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 complet, ou plus exactement la liste de tous les mots acceptés par le jeu. Cette librairie vous est fournie dans un fichier `.jar` que vous devez ajouter à votre projet.

Mais qu'est-ce qu'un jar ?

Un JAR (Java Archive) est un fichier Zip utilisé pour distribuer un ensemble de classes Java. Ce format est utilisé pour stocker les définitions des classes, ainsi que des métadonnées, constituant l'ensemble d'un programme. [Wikipedia]

Comment ajouter un jar donné ?

Sous Netbeans, pour pouvoir utiliser les classes d'un .jar donné, il suffit :

1. de copier ce .jar dans un sous-dossier de votre projet (par exemple dans le dossier lib, que vous créez pour l'occasion).
2. d'ajouter ce .jar aux librairies de votre projet.

Une fois le jar ajouté à votre projet, comment l'utiliser ? Le jar 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 58è mot de ce dictionnaire.

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 possible on en choisit ici un seul.

Testez votre méthode avec plusieurs tests JUnit.

Intégrez cela dans a méthode principale.

Exercice 10 **Les meilleurs mots**

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

Testez votre méthode avec plusieurs tests JUnit.

Intégrez cela dans la méthode principale.

Exercice 11 **Bonus I**

Développer 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 probablement disponibles quelque part sur internet.

Exercice 12 **Bonus II**

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' ou un 's' qu'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