Université de Lille
IUT de Lille

**TD n°12**

R1.01 - Initiation au développement
BUT Informatique
Yann.Secq@univ-lille.fr
Fabien.Delecroix@univ-lille.fr
2022–2023

---

**Objectifs:** Comprendre les étapes de la décomposition en fonctions d'un problème:
- identifier les sous-problèmes à résoudre par des fonctions
- choisir comment représenter les informations
- implémenter et tester chacune de ces fonctions indépendamment

Comprendre l'importance de la manière de représenter les informations.
S'exercer à écrire un algorithme en utilisant des fonctions données.

---

# Un jeu avec des mots

Nous voulons implémenter un petit jeu dont le but est de construire le plus de mots liés à une catégorie en n'utilisant que les lettres d'un ensemble donné. Des exemples de catégories sont : noms d'animaux, pays, capitales, verbes, etc.

Par exemple, étant donnés les lettres *b e g i p r s t* et la catégorie *noms d'animaux*, on considère correctes les propositions *tigre, brebis, pie*. On notera donc qu'**on peut utiliser une même lettre plusieurs fois**.

Le jeu se déroulera comme suit :
- le programme affiche le nom de la catégorie et l'ensemble de lettres
- la joueuse ou le joueur saisit des mots (suivis par entrée), et indique la fin de sa réponse en saisissant le mot "fin"
- le programme teste combien de ces mots appartiennent à la catégorie et affiche le score obtenu.

En reprenant l'exemple ci-dessus, voici une trace d'exécution que produirait le programme (avec en gras les saisies).

```
Écrivez un maximum de noms d'animaux à partir des lettrees suivantes :
b e g i p r s t
pie
lion
brebis
gris
fin
Vous avez trouvé : pie brebis
Votre scoe est de 2/3
Mot(s) non-trouvé(s) : tigre
```

Ici le score obtenu est 2, car *lion* ne peut pas être écrit avec les lettres données, et *gris* n'est pas un nom d'animal.

Nous voulons un jeu jouable, où il est toujours possible de construire des mots de la catégorie avec les lettres données. Ainsi, les lettres ne seront pas choisies au hasard, mais de manière à pouvoir construire au moins trois mots de la catégorie. De plus, on voudrait afficher les lettres sans répétition et dans l'ordre alphabétique.

### Analyse du problème

**Q1.** Identifiez les différentes étapes du déroulement du jeu, depuis le lancement du programme, jusque l'affichage du score à la fin. Pour ce faire, analysez l'exemple d'exécution fourni.

**Q2.** Écrire une ébauche d'algorithme principal dans laquelle pour chacune des étapes de la question précédente on introduit une fonction. Pour l'instant on ne s'intéresse pas aux paramètres de ces fonctions ni à leur type de retour, mais uniquement à identifier un ensemble de fonctions qui devraient nous permettre de résoudre le problème de départ.

### Représenter les informations

On doit maintenant identifier les informations que le programme va manipuler, et trouver ou créer des types adaptés pour les représenter.

Dans une version aboutie, on externaliserait bien sûr les catégories et mots associés dans des fichiers, offrant ainsi la possibilité d'ajouter du contenu au jeu sans modifier le programme en lui-même. Ici, nous nous contenterons dans une première version d'avoir en dur dans le programme des constantes décrivant le nom d'une unique catégorie ainsi que les mots associés :

```
final String CATEGORIE = "NOMS D'ANIMAUX";
final String[] MOTS = new String[]{"brebis", "aigle", "perroquet", "lion",
               "mouche", "souris", "tigre", "chat", "loup", "koala", "kiwi", "pie"};
```

**Q3.** Dans la perspective d'avoir par la suite plusieurs catégories, comment les représenter ?

**Q4.** Comment représenter les mots proposés par la joueuse ?

Trouver une bonne manière de représenter l'ensemble de lettres est plus difficile. Un choix judicieux pourrait grandement faciliter la résolution du problème.

**Q5.** Quelles sont les sous-tâches dans lesquelles on devra manipuler l'ensemble de lettres ? Quels calculs seront effectués avec cet ensemble de lettres ?

Les deux possibilités qui viennent à l'esprit pour représenter un ensemble de lettres sont une chaîne de caractères et un tableau de caractères. Cependant, aucune de ces options ne permet de représenter *facilement* les lettres sans répétition, et de les afficher dans l'ordre alphabétique. Trouvez une autre option répondant à ces critères.

### Préciser les signatures des fonctions pour les sous-tâches

Après avoir décidé comment seront représentées les données, on peut préciser les signatures des différentes fonctions identifiées pour les sous-tâches. On va également décrire la *responsabilité* de chacune de ces fonctions, càd, les calculs qu'elle va effectuer. Voici les fonctions qui nous permettront de résoudre le problème:

1. `void ajouterAEnsemble(boolean[] ensembleLettres, String mot)` ajoute à `ensembleLettres` toutes les lettres de `mot`
2. `void initialiserEnsembleLettres(boolean[] ensembleLettres)` choisit aléatoirement trois mots de la catégorie et ajoute leurs lettres à `ensembleLettres`.
3. `void afficherEnsembleLettres(boolean[] ensembleLettres)` affiche l'ensemble des lettres dans l'ordre alphabétique
4. `boolean estDansCategorie(String mot, Categorie categorie)` teste si `mot` appartient à la catégorie
5. `boolean lettresSontDansEnsemble(boolean[] ensembleLettres, String mot)` teste si toutes les lettres `mot` appartiennent à l'ensemble des lettres

### Écrire l'algorithme principal

**Q6.** En supposant que les fonctions ci-dessus sont déjà écrites, donner l'algorithme principal `void algorithm()` qui implémente le jeu.

Notez qu'un exercice classique en DS est de vous présenter un problème et de vous demander de donner:
— les différentes informations et les types pour les représenter
— écrire l'algorithme principal comme dans la dernière question
— décrire en une phrase ce que fait chacune des fonctions utilisées dans l'algorithme principal, comme ci-dessus.

## Prolongement

**Q7.** Pour chacune des fonctions identifiées, implémenter la fonction et écrire une fonction de test adaptée. Les tests à écrire seront:
— des fonctions de test «classiques» (avec des assertions) `void testAjouterAEnsemble()`, `void testEstDansCategorie()` et `void testLettresSontDansEnsemble()`
— des tests à mettre dans `void algorithm()` pour les deux autres fonctions qui ne peuvent pas être testées par des assertions, l'une parce qu'elle fait des affichages, l'autre parce qu'elle fait intervenir du hasard et on ne peut donc pas prévoir son comportement.

**Q8.** Proposez un code permettant d'externaliser les données et de choisir une catégorie parmi celles disponibles.

**Q9.** Implémentez une version 2 joueurs dans laquelle le 1er joueur gagne 1 point par mot trouvé tandis que le second en gagne 2 par mot restant.

**Q10.** Implémentez une version à n joueurs où l'ordre dans lequel jouent les joueurs est aléatoire.