HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Professeur : Nastaran Fatemi
Assistants : Christopher Meier
Guillaume Hochet

# Labo 3 : Anagrammes

## Objectifs

L'objectif de ce laboratoire est de développer un programme permettant de trouver toutes les anagrammes d'une phrase en utilisant l'API de Collections Scala et les expressions `for`.

## Indications

Deux fichiers vous sont fournis. Le fichier `Anagrams.scala` qui contient une pré-implémentation du laboratoire et le fichier `linuxwords.txt` qui représente un dictionnaire.

- Ce laboratoire est à effectuer par **groupe de 2**, tout plagiat sera sanctionné par la note de `1`.
- Le laboratoire est à rendre pour le `26.04.2020` à `23h59` sur Cyberlearn, une archive nommée `SCALA_-labo3_NOM1_NOM2.zip (7z, rar, ...)` contenant les sources de votre projet devra y être déposé.
- Il n'est pas nécessaire de rendre un rapport, un code propre et correctement commenté suffit. Faites cependant attention à bien expliquer votre implémentation.
- Faites en sorte d'éviter la duplication de code.
- Préférez des implémentations récursives de fonctions.

## Descriptions

L'anagramme d'un mot est un réarrangement de ces lettres pour créer un nouveau mot avec un sens différent, par exemple, si on réarrange les lettres du mot *Elvis*, on peut obtenir le mot *lives*, qui est l'une de ses anagrammes. De la même manière, l'anagramme d'une phrase et un réarrangement de tous les caractères de la phrase afin de créer une nouvelle phrase avec un nouveau sens. par exemple, la phrase *I love you* est une anagramme de la phrase *You olive*. Ici, on considère aussi la permutation des mots de la phrase originale. Donc, dans cet exemple, *You I love* est aussi considéré comme une anagramme possible. Quand on génère les anagrammes, on ignore la casse et les caractères de ponctuation.

## Implémentation

Dans ce laboratoire, l'objectif final est d'implémenter la méthode `sentenceAnagrams`, qui prend une liste de mots représentant une phrase et qui retourne toutes les anagrammes de celle-ci. Afin de déterminer si un mot généré a un sens, un dictionnaire est fourni.

Dans le fichier `Anagrams.scala`, nous vous fournissons un *template* dans lequel vous allez compléter l'implémentation d'une série de méthodes qui seront utilisées pour réaliser `sentenceAnagrams`.

Afin de vérifier si deux mots sont des anagrammes, on génère une empreinte pour chaque mot du dictionnaire, en représentant chaque mot par une suite ordonnée de caractères. Pour simplifier l'accès à toutes les anagrammes d'un mot dans un dictionnaire, on génère un indexe représenté par une `Map`, afin de faire correspondre chaque empreinte à tous les mots du dictionnaire ayant cette empreinte.

Trouver les anagrammes d'une phrase est légèrement plus difficile. Pour cela, on va d'abord créer une empreinte de tous les caractères de tous les mots de la phrase. La phrase sera donc représentée par une séquence ordonnée de tous les caractères de tous ses mots. A partir de cette séquence de caractères, on doit générer toutes les possibilités de combinaison de caractères qui forment des mots existants dans le dictionnaire.

Pour trouver les mots, on va commencer par trouver tous les sous-ensembles de caractères de l'empreinte de la phrase. Pour chaque sous-ensemble, on l'extrait de la phrase et on vérifie si on trouve des mots correspondants dans le dictionnaire. On continue à résoudre le problème de façon récursive avec les caractères restants. A la fin de la récursion, on combine tous les mots trouvés pour former une phrase.

Par exemple, si on applique cette idée à la phrase *You olive*, l'empreinte de cette phrase est `eiloouvy`. On commence par extraire un sous-ensemble de caractères, par exemple `i`. Il nous reste la suite *eloovy*. On vérifie `i` dans l'indexe et on voit que cela correspond au mot `I`. Alors on continue à résoudre le problème récursivement pour le reste des caractères de *eloovy* et on obtient la liste des solutions `List(List(love, you), List(you, love))`. On peut combiner `I` avec cette liste pour obtenir la phrase *I love you* et *I you love* qui sont toutes les deux des anagrammes valables.