Architecture du moteur de chaîne

1. Record Branch (package project.branch)

Variables du record :

- **word1**, **word2**: Champs représentant les deux mots liés par la branche.
- **score**: Champ représentant le score associé à la branche.

Méthodes :

- public **Branch(String word1, String word2, float score)**: Constructeur qui crée une nouvelle branche en vérifiant la validité des paramètres.
- public String **getOtherWord(String word)**: Méthode pour obtenir l'autre mot de la paire.
- @Override public String **toString()**: Méthode pour générer une représentation textuelle de la branche.

2. Classe Tree (package project.tree)

Variables de la classe :

- **branches**: Une liste d'instances de la classe Branch représentant les branches de l'arbre.

Méthodes :

- public **Tree()**: Constructeur qui initialise la liste de branches.
- public ArrayList<Branch> **getBranches()**: Méthode pour obtenir la liste des branches.
- public void **addBranch(Branch branch):** Méthode pour ajouter une branche à la liste.
- private void **cycleDetectorDFS(...)**: Méthode récursive utilisée pour détecter les cycles dans l'arbre.
- public Set<Set<Branch>> **detectAllCycles()**: Méthode pour détecter tous les cycles dans l'arbre.
- public void **removeWeakestBranchUntilNoCycle(DocumentHandler documentHandler)**: Méthode qui supprime la branche la plus faible jusqu'à ce qu'aucun cycle ne soit détecté.
- public boolean **isEqual(Tree tree)**: Méthode qui compare si deux arbres sont égaux en termes de branches.

- private Branch **getBranchWithLowerScoreBetweenWords(...)** : Méthode récursive qui permet d'obtenir la branche la plus faible entre le mot de départ et le mot d'arrivée.
- public float **getTreeScore(String word1, String word2) :** Mathéode qui renvoie le score le plus faible entre deux mots.
- @Override public String **toString()**: Méthode pour générer une représentation textuelle de l'arbre.

3. Classe DocumentHandler (package projects.documents)

Variables de la classe :

- **documentEntryPath**: Chemin d'accès au document d'entrée.
- **documentExitPath**: Chemin d'accès au document de sortie.
- **documentDeletedBranchesPath**: Chemin d'accès au document des branches supprimées.
- **WORDS_SECTION_HEADER**, **OFFSETS_SECTION_HEADER**, **DISTANCES_SECTION_HEADER**: Constantes représentant les en-têtes des sections du **document**.
- **WORD_OFFSET_PATTERN**, **WORD_DISTANCE_PATTERN**: Constantes représentant les modèles pour valider le format des sections.

Méthodes :

- public **DocumentHandler(String documentEntryPath)**: Constructeur qui initialise les chemins d'accès aux différents documents.
- private void **validateDocument(String document)**: Méthode privée pour valider le format du document.
- public void **addBranchesFromDocumentInTree(Tree tree)** throws IOException: Méthode qui ajoute des branches à un arbre à partir du document d'entrée.
- private String **writeAllBranchesInDocument(Tree tree)**: Méthode privée qui génère le contenu du document à partir de toutes les branches d'un arbre.
- private String **writeSingleBranchInDocument(Branch branch)**: Méthode privée qui génère le contenu du document pour une seule branche.
- public void **writeDocumentToFile(Tree tree, Branch branch)**: Méthode qui écrit le contenu d'un arbre ou d'une branche dans un fichier.

Format des documents utilisés

Afin de faciliter la manipulation des documents, la décision prise fut d'utiliser des documents .txt afin d'avoir le même type à chaque fois pour la suite du projet.