TP 1 : Réduction de son

1 INTRODUCTION

Le premier TP consiste à réaliser un logiciel Java qui va réduire le nombre de sons différents dans un texte. Pour cela, le logiciel va devoir calculer l'occurrence de chaque son dans le texte. Aussi, il faudra définir une métrique de distance pour comparer les sons. Cela va nous permettre de choisir les sons à remplacer afin d'exécuter la réduction.

Dans un premier lieu, nous allons décrire le projet. Ensuite, les attentes du devoir et les éléments d'évaluations seront exposés.

2 DESCRIPTION

Cette description est divisée en quatre sections. Dans une première section, le code disponible est introduit. La section suivante décrit les entrées qu'acceptera le logiciel. Cela sera suivi de la description du traitement que le logiciel effectuera. Finalement, il y aura une courte description des sorties.

2.1 PROJET

Un code de départ est déjà disponible sur Moodle. Il contient les fichiers (classes) suivants :

- `TexteSonore` : permets de contenir une suite de syllabe. C'est le texte à réduire.
  Utilise :
  - `SyllabeFrancais` : décris un son. C'est l'unité que nous utilisons pour la réduction.
    Utilise :
    - `ConsonneFrancais` : décris le son d'une consonne. Contiens les méthodes pour l'extraction des caractéristiques sonore des consonnes.
      Utilise :
      - `API_Consonne` : les consonnes de l'alphabet international.
        Utilise :
        - `ModeArticulation`.
        - `PointArticulationConsonne`.
        - `Phonation`.
        - `Cavite`.
        - `Ecoulement`.
        - `Flux`.
    - `VoyelleFrancais` : décris le son d'une voyelle. Contiens les méthodes pour l'extraction des caractéristiques sonore des voyelles.
      Utilise :
      - `API_Voyelle` : les voyelles de l'alphabet international.
        Utilise :
        - `Ouverture`.

    - `PointArticulationVoyelle`.
    - `Rondeur`.
- `Constantes` : pour les constantes utiles au logiciel.
- `Erreur` : type énuméré pour les cas d'erreurs.
- `Textes` : constante pour les chaînes de caractères utilisée dans le logiciel.
- `Principal` : contiens le programme principal (main).

Votre code doit être ajouté dans ces classes. Vous pouvez construire des nouvelles classes au besoin. Le code est entièrement commenté et un fichier `html` contenant cette documentation est aussi disponible.

2.2 Entrées

Le code du programme principal, lorsqu'exécuté, demande le nom d'un fichier et une valeur numérique. Le fichier est lu par le code et les sons décrits dans le fichier sont enregistrés dans une instance de la classe `TexteSonore`. La valeur numérique (dénoté k dans le reste du texte) est un entier plus grand que zéro. Il indique le nombre de sons que nous voulons conserver dans le texte final. Le code lisant ces valeurs est déjà écrit. Les cinq premières lignes de la méthode `main` font appel à ces méthodes de lecture et place l'information dans les variables locales suivantes (ce code est déjà écrit) :

- `nombreDeSyllabes` : contient la valeur dénotée k dans ce texte.
- `texteSonore` : contiens la suite des sons à traiter. Dans ce texte, nous allons utiliser S pour dénoter cette suite qui contient n syllabe. La syllabe à la position i sera dénotée s_i. La première position est la position $i = 0$.

2.3 Traitement

Pour le devoir, vous devez ajouter le code qui va permettre de faire les traitements suivants :

- Calculer l'occurrence des syllabes.
- Calculer la distance sonore entre les syllabes.
- Réduire le nombre de syllabes en transformant les sons de la suite.

Nous allons présenter dans les sous-sections suivantes chacun de ces traitements. Votre code devra s'assurer d'obtenir les résultats attendus par ces traitements, mais cela ne vous oblige pas a les faires dans l'ordre présenté.

2.3.1 Occurrence des syllabes

Pour chaque syllabe du texte, le logiciel doit calculer le nombre de fois qu'elle apparait dans le texte. Une syllabe est composée de trois groupes :

1. Attaque : un groupe de consonne. Ce groupe est optionnel.
2. Noyau : un groupe de voyelle. Ce groupe est obligatoire.
3. Coda : un groupe de consonne. Ce groupe est optionnel.

Deux syllabes sont pareilles si elles ont le même groupe noyau, le même groupe d'attaque s'il est présent et le même groupe coda s'il est présent. Un groupe de consonne contient un ou deux phonèmes représentés par des lettres de l'API (Alphabet phonétique international). Deux groupes de

consonnes sont pareils s'ils contiennent les mêmes lettres de l'API. Un groupe de voyelle contient un ou deux phonèmes représentés par des lettres de l'API et un booléen indiquant si elles sont nasales. Deux groupes de voyelles sont pareils s'ils contiennent les mêmes lettres de l'API et si elles ont la même nasalité.

Par exemple, le texte suivant :

fu.ta.fu.ma.fu.ta.fu.na.fu.ta.fu.ma.fu.ta.fu

Contiens les occurrences suivantes :

Syllabes	Occurrence
fu	8
ta	4
ma	2
na	1

2.3.2 Distance sonore

La distance sonore est une métrique qui va nous permettre de distinguer les sons qui se ressemblent et les sons différents. Pour cela, nous devons définir la distance entre plusieurs éléments : syllabes, consonnes, voyelles, groupes de consonnes, groupes de voyelles, booléens.

2.3.2.1 Distance entre deux booléens

Les phonèmes ont des caractéristiques sonores représentées par des valeurs booléennes. Nous allons définir la distance entre deux valeurs booléennes (b_1 et b_2) comme étant égal à 1 lorsque ces deux valeurs sont différentes. Sinon, la distance sera de zéro. Cela donne une valeur dans l'intervalle $[0..1]$.

b_1	b_2	Distance(b_1, b_2)
`true`	`true`	0
`true`	`false`	1
`false`	`true`	1
`false`	`false`	0

2.3.2.2 Distance entre deux phonèmes de consonne (classe `API_Consonne`)

La distance entre deux phonèmes de consonne est simplement de calculer le nombre de caractéristiques sonore différente entre les deux consonnes. Une consonne est décrite par six (6) caractéristiques sonores : `estVocalique`, `estNasal`, `estVoise`, `estContinu`, `estCompact`, et `estAigu`. Chacune de ces caractéristiques est une valeur booléenne. Il est possible de calculer la distance entre une consonne et l'absence d'une consonne (silence), dans ce cas, cela donne une distance de 6 (le maximum). La distance sonore entre deux silences est de 0. Cela donne une valeur dans l'intervalle $[0..6]$.

Par exemple, soit les consonnes p, t et k, ayant les caractéristiques sonores suivantes :

Phonème	estVocalique	estNasal	estVoise	estContinu	estCompact	estAigu

p	`true`	`false`	`false`	`false`	`false`	`false`
t	`true`	`false`	`false`	`false`	`false`	`true`
k	`true`	`false`	`false`	`false`	`true`	`true`

Alors, la distance entre p et t est de 1, la distance entre t et k est de 1 et la distance entre p et k est de 2.

*2.3.2.3 Distance entre deux phonèmes de voyelle (classe **`API_Voyelle`**)*

La distance entre deux phonèmes de voyelle est simplement de calculer le nombre de caractéristiques sonore différente entre les deux voyelles. Une voyelle est décrite par quatre (4) caractéristiques sonores : `estArriere`, `estHaut`, `estArrondi`, et `estOuverte`. Chacune de ces caractéristiques est une valeur booléenne. Il est possible de calculer la distance entre une voyelle et l'absence d'une voyelle (silence), dans ce cas, cela donne une distance de 4 (le maximum). La distance sonore entre deux silences est de 0. Cela donne une valeur dans l'intervalle $[0..4]$.

Par exemple, soit les voyelles e, a et u, ayant les caractéristiques sonores suivantes :

Phonème	`estArriere`	`estHaut`	`estArrondi`	`estOuverte`
e	`false`	`false`	`false`	`false`
a	`false`	`false`	`false`	`true`
u	`true`	`true`	`true`	`false`

Alors, la distance entre e et a est de 1, la distance entre a et u est de 4 et la distance entre e et u est de 3.

2.3.2.4 Distance entre deux groupes de consonnes (classe ConsonneFrancais)

Pour calculer la distance entre deux groupes de consonne, il suffit d'additionner la distance entre les deux premières consonnes de chaque groupe avec la distance entre les deux dernières consonnes de chaque groupe. Lorsqu'une consonne n'est pas présente, alors nous calculons la distance avec une consonne silencieuse. Cela donne une valeur dans l'intervalle $[0..12]$.

Par exemple, la distance entre les groupes kp et kt est la somme de la distance entre k et k avec la distance entre p et t. Ce qui nous donne $0 + 1 = 1$. La distance entre les groupes k et pt est la somme de la distance entre k et p avec la distance entre une consonne silencieuse et t. Ce qui nous donne $2 + 6 = 8$.

*2.3.2.5 Distance entre deux groupes de voyelles (classe **`VoyelleFrancais`**)*

Pour le calcul de distance entre deux groupes de voyelle, vous devez premièrement additionner la distance entre les voyelles et la distance entre les semi-voyelles. Si une voyelle n'est pas présente, alors nous calculons la distance avec une voyelle silencieuse. Ensuite, il faut ajouter la distance entre les caractéristiques sonores `estNasal` du groupe de voyelle (c'est une distance booléenne). Cela donne une valeur dans l'intervalle $[0..9]$.

Par exemple, la distance entre les groupes a et ẽ. Le tilde au-dessus du e indique que la voyelle est nasale. Nous devons donc calculer la distance entre a et e à laquelle nous ajoutons 1. Ce qui nous donne $1 + 1 = 2$.

2.3.2.6 Distance entre deux syllabes (classe `SyllabeFrancais`)

Pour calculer la distance entre deux syllabes, il faut multiplier la distance entre les noyaux (groupe de voyelles) par 2. Ensuite, il faut additionner à cette valeur la distance entre les groupes d'attaque (groupe de consonnes) et la distance entre les groupes de coda (groupe de consonne). Cela donne une valeur dans l'intervalle $[0..42]$.

Par exemple, la distance entre les syllabes pta et kẽ est la somme de la distance entre pt et k avec la distance entre a et ẽ multiplié par 2. Ce qui nous donne $8 + 2 \times 2 = 12$.

2.3.3 Réduction et transformation

La réduction de syllabe consiste à réduire le nombre de syllabes différentes dans le texte. L'utilisateur a entré une valeur indiquant le nombre de syllabes (k, voir section 2.2) qu'il devrait avoir dans le texte final. Nous réduisons les syllabes, une à la fois jusqu'à ce qu'il en reste k. S'il y en a moins qu'k au départ, alors il n'y a pas de réduction. Pour enlever une syllabe, il suffit de trouver la syllabe à enlever et ensuite il faut trouver la syllabe qui la remplacera.

Pour réaliser ces opérations, il faut trouver la distance entre chaque pair de syllabes différentes possible dans le texte. Par exemple, le texte suivant contient quatre (4) syllabes différentes.

fu.ta.fu.ma.fu.ta.fu.na.fu.ta.fu.ma.fu.ta.fu

Nous avons donc 6 paires de syllabes différentes. Il reste à calculer les distances entre elles.

Syllabe 1	Syllabe 2	Distance
fu	ta	10
fu	ma	11
fu	na	12
ta	ma	3
ta	na	2
ma	na	1

Lorsque nous avons les distances entre chaque pair, il suffit de choisir la paire qui a la plus petite distance. Dans l'exemple, c'est la paire ma et na. Ensuite, il faut choisir la syllabe qui sera remplacée par une autre. Pour cela nous utilisons les occurrences des syllabes. La syllabe de la paire ayant l'occurrence la plus grande remplacera la syllabe ayant l'occurrence la plus petite. Dans notre exemple, la syllabe ma a une occurrence de 2 et la syllabe na a une occurrence de 1. Nous allons donc remplacer les syllabes na par des syllabes ma dans le texte. Après ce remplacement, nous devons réajuster le nombre d'occurrence de la syllabe ma dans le texte. Puisqu'il y avait 1 occurrence de la syllabe na, cela à ajouter 1 occurrence de la syllabe ma, qui a maintenant une occurrence de 3.

Nous recommençons cette opération jusqu'à ce qu'il reste k syllabes différentes dans le texte.

2.4 Sorties

Simplement, affichez la suite de sons résultants en utilisant le `toString` de la classe `TexteSonore`.

3 DIRECTIVES

Les sections suivantes décrivent les attentes et les éléments d'évaluation pour le devoir.

3.1 DIRECTIVES POUR LA CONSTRUCTION DU PROJET.

- Placez vos noms au début du fichier `Principal.java`.
- Vous devez utiliser le code de départ disponible sur Moodle.
- Lorsque vous ajoutez une méthode, vous devez l'ajouter dans la classe appropriée. Par exemple, une méthode qui calcule la distance entre deux syllabes devrait être dans la classe `SyllabeFrancais`.
- Des tests vous seront donnés avec les résultats attendus.

3.2 DIRECTIVES POUR L'ÉCRITURE DU CODE.

1. Le TP est à faire seul ou en équipe de deux.
2. Code :
    a. Pas de `goto`, `continue`.
    b. Les `break` ne peuvent apparaitre que dans les `switch`.
    c. Un seul `return` par méthode.
    d. Additionnez le nombre de `if`, `for`, `while`, `switch` et `try`. Ce nombre ne doit pas dépasser 5 pour une méthode.
    e. Vous pouvez ajouter des méthodes, mais vous ne devez pas ajouter de nouvelles méthodes `static`.
    f. Vous pouvez ajouter des classes au besoin.
3. Indentez votre code. Assurez-vous que l'indentation est faite avec des espaces.
4. Commentaires
    - Commentez l'entête de chaque classe et méthode.
    - Une ligne contient soit un commentaire, soit du code, pas les deux.
    - Utilisez des noms d'identificateur significatif.
    - Une ligne de commentaire ou de code ne devrait pas dépasser 120 caractères. Continuez sur la ligne suivante au besoin.
    - Nous utilisons Javadoc :
        - La première ligne d'un commentaire doit contenir une description courte (1 phrase) de la méthode ou la classe.
            - Courte.
            - Complète.
            - Commencez la description avec un verbe.
            - Assurez-vous de ne pas simplement répéter le nom de la méthode, donnez plus d'information.
        - Ensuite, au besoin, une description détaillée de la méthode ou classe va suivre.
            - Indépendant du code. Les commentaires d'entêtes décrivent ce que la méthode fait, ils ne décrivent pas comment c'est fait.

        - Si vous avez besoin de mentionner l'objet courant, utilisez le mot 'this'.
    - Ensuite, avant de placer les **tags,** placez une ligne vide.
    - Placez les **tag** @param, @return et @throws au besoin.
        - @param : décris les valeurs acceptées pour la méthode. Vous devez commenter les paramètres de vos méthodes.
    - Dans les commentaires, placer les noms de variable et autre ligne de code entre les tags <code> … </code>.
    - Écrivez les commentaires à la troisième personne.

3.3 REMISE

Remettre le TP par l'entremise de Moodle. Placez vos fichiers `'*.java'` dans un dossier compressé de **Windows**, vous devez remettre l'archive. Le TP est à remettre avant le 4 mars 23 :55.

3.4 ÉVALUATION

- Fonctionnalité (9 pts) : des tests partiels vous seront remis. Un test plus complet sera appliqué à votre TP. Votre projet doit compiler sans erreur pour avoir ces points.
- Structure (4 pt) : veillez à placer correctement vos méthodes et à utiliser correctement le mécanisme d'héritage (classe `TexteSonore` et `ArrayList`).
- Lisibilité (2 pts) : commentaire, indentation et noms d'identificateur significatif.