TP 2 : Thaï vers API

1 INTRODUCTION

Pour le deuxième TP, vous allez construire une méthode qui traduit du thaï en Alphabet Phonétique International (API). La section 2 va décrire le programme à construire. La section 3 contient les directives pour le code. Finalement, il y a les modalités de remise (section 4).

2 DESCRIPTION

2.1 DESCRIPTION DE LA MÉTHODE

Vous allez construire une classe nommée `Thai`. Cette classe doit contenir la méthode suivante.

```
public static String versAPI( String texte )
```

Votre méthode va recevoir une chaîne de caractères (donc, en Unicode) représentant des caractères thaïs. Elle va retourner une chaîne de caractères (toujours en Unicode) représentant des caractères de l'API.

2.1.1 Traduction de la chaîne

La chaîne va contenir plusieurs syllabes, chaque syllabe comporte un ou plusieurs caractères. Il y a un espace entre chaque syllabe. Il n'y a pas de limite au nombre de syllabes qu'une chaîne peu contenir. Si la chaîne ne contient pas de syllabe, alors la méthode doit retourner une chaîne vide. Sinon, elle contient la traduction de chaque syllabe séparée par un espace.

2.1.2 Traduction d'une syllabe

Une syllabe se divise en trois composantes pour notre projet.

- Noyau : le groupe de voyelle. Pour notre projet, toutes les syllabes ont un noyau. Cependant, l'écriture thaïe peut avoir des syllabes où la voyelle est sous-entendue (n'est pas écrite).
- Attaque : le groupe de consonne initiale. Ce ne sont pas toutes les syllabes thaïes qui ont une consonne initiale. Dans notre projet, le groupe de consonne va toujours contenir une seule consonne, bien que l'écriture thaïe permet des groupes de deux consonnes en attaque. Notre logiciel ne traitera pas ces cas. Ils seront considérés comme des erreurs. Nous utilisons la variable c_i pour désigner la consonne initiale.
- Coda : le groupe de consonne finale. Ce groupe, composé d'une seule consonne, n'est pas toujours présent. Nous utilisons la variable c_f pour désigner la consonne finale.

Voici une table contenant les Unicode pour les consonnes thaïes. Les deux autres colonnes contiennent les Unicode pour les prononciations en fonction de la position de la consonne. Certaines prononciations se composent de plusieurs caractères. Les cases vides indiquent que la consonne ne peut pas occuper cette position.

Code thaï	Code API c_i	Code API c_f
U+0E01	U+006B	U+006B
U+0E02	U+006B U+02B0	U+006B
U+0E03	U+006B U+02B0	U+006B
U+0E04	U+006B U+02B0	U+006B
U+0E05	U+006B U+02B0	U+006B
U+0E06	U+006B U+02B0	U+006B
U+0E07	U+014B	U+014B
U+0E08	U+0074 U+0255	U+0074
U+0E09	U+0074 U+0255 U+02B0	
U+0E0A	U+0074 U+0255 U+02B0	U+0074
U+0E0B	U+0073	U+0074
U+0E0C	U+0074 U+0255 U+02B0	
U+0E0D	U+006A	U+006E
U+0E0E	U+0064	U+0074
U+0E0F	U+0074	U+0074
U+0E10	U+0074 U+02B0	U+0074
U+0E11	U+0074 U+02B0	U+0074
U+0E12	U+0074 U+02B0	U+0074
U+0E13	U+006E	U+006E
U+0E14	U+0064	U+0074
U+0E15	U+0074	U+0074
U+0E16	U+0074 U+02B0	U+0074
U+0E17	U+0074 U+02B0	U+0074
U+0E18	U+0074 U+02B0	U+0074
U+0E19	U+006E	U+006E
U+0E1A	U+0062	U+0070
U+0E1B	U+0070	U+0070
U+0E1C	U+0070 U+02B0	
U+0E1D	U+0066	
U+0E1E	U+0070 U+02B0	U+0070
U+0E1F	U+0066	U+0070
U+0E20	U+0070 U+02B0	U+0070
U+0E21	U+006D	U+006D
U+0E22	U+006A	U+006A
U+0E23	U+0072	U+006E
U+0E25	U+006C	U+006E
U+0E27	U+0077	U+0077
U+0E28	U+0073	U+0074
U+0E29	U+0073	U+0074
U+0E2A	U+0073	U+0074
U+0E2B	U+0068	
U+0E2C	U+006C	U+006E
U+0E2D	U+0294	
U+0E2E	U+0068	

Si plus de 1 patron sont valides, alors le plus précis des patrons l'emporte. Par exemple, pour U+0E01 U+0E32 U+0E22, il y a deux patrons : (c_i U+0E32 c_f) ou (c_i U+0E32 U+0E22), le deuxième est plus précis, et l'emporte. Les encadrés sur la table énumère les cas où un patron plus précis existe.

Voici une description de l'information contenue dans la table suivante.

- Colonne Code thaï : contiens la disposition (le patron) des consonnes et voyelles pour l'écriture d'une syllabe.
- Colonne Code API voyelle : les prononciations pour les voyelles.

Voyelle courte		Voyelle longue	
Code thaï	**Code API voyelle**	**Code thaï**	**Code API voyelle**
c_i U+0E30	U+0061 U+0294	c_i U+0E32	U+0061 U+02D0
c_i		c_i U+0E32 c_f	
c_i U+0E31 **c_f**			
c_i U+0E34	U+0069	c_i U+0E35	U+0069 U+02D0
c_i U+0E34 **c_f**		c_i U+0E35 c_f	
c_i U+0E36	U+026F	c_i U+0E37 U+0E2D	U+026F U+02D0
c_i U+0E36 **c_f**		c_i U+0E37 c_f	
c_i U+0E38	U+0075	c_i U+0E39	U+0075 U+02D0
c_i U+0E38 **c_f**		c_i U+0E39 c_f	
U+0E40 **c_i** U+0E30	U+0065	U+0E40 c_i	U+0065 U+02D0
U+0E40 **c_i** U+0E47 **c_f**		U+0E40 c_i c_f	
U+0E41 **c_i** U+0E30	U+025B U+0294	U+0E41 c_i	U+025B U+02D0
U+0E41 **c_i** U+0E47 **c_f**		U+0E41 c_i c_f	
U+0E42 **c_i** U+0E30	U+006F U+0294	U+0E42 c_i	U+006F U+02D0
c_i **c_f**		U+0E42 c_i c_f	
U+0E40 **c_i** U+0E32 U+0E30	U+0254 U+0294	c_i U+0E2D	U+0254 U+02D0
		c_i U+0E2D c_f	
c_i U+0E47 U+0E2D **c_f**		c_i c_f	
U+0E40 **c_i** U+0E2D U+0E30	U+0264 U+0294	U+0E40 c_i U+0E2D	U+0264 U+02D0
		U+0E40 c_i U+0E34 c_f	
		U+0E40 c_i U+0E2D c_f	
U+0E40 **c_i** U+0E35 U+0E22 U+0E30	U+0069 U+0061 U+0294	U+0E40 c_i U+0E35 U+0E22	U+0069 U+0061
		U+0E40 c_i U+0E35 U+0E22 c_f	
U+0E40 **c_i** U+0E37 U+0E2D U+0E30	U+026F U+0061 U+0294	U+0E40 c_i U+0E37 U+0E2D	U+026F U+0061
		U+0E40 c_i U+0E37 U+0E2D c_f	
c_i U+0E31 U+0E27 U+0E30	U+0075 U+0061 U+0294	c_i U+0E31 U+0E27	U+0075 U+0061
		c_i U+0E27 c_f	
c_i U+0E34 U+0E27	U+0069 U+0077		
U+0E40 **c_i** U+0E47 U+0E27	U+0065 U+0077	U+0E40 c_i U+0E27	U+0065 U+02D0 U+0077

$c_f \neq$ U+0E22
$c_f \neq$ U+0E27

$c_f \neq$ U+0E27

$c_f \neq$ U+0E22

$c_f \neq$ U+0E27

$c_f \neq$ U+0E23
$c_f \neq$ U+0E2D

$c_f \neq$ U+0E22

$c_f \neq$ U+0E22
$c_f \neq$ U+0E27

$c_f \neq$ U+0E2F

$c_f \neq$ U+0E22
$c_f \neq$ U+0E27
$c_f \neq$ U+0E2D

$c_f \neq$ U+0E27

$c_f \neq$ U+0E22

Si c_f = U+0E23

$c_f \neq$ U+0E27

$c_f \neq$ U+0E22

U+0E40 c_i U+0E32	U+0061 U+0077
c_i U+0E31 U+0E22 U+0E43 c_i U+0E44 c_i U+0E44 c_i U+0E22	U+0061 U+006A
c_i U+0E47 U+0E2D U+0E22	U+0254 U+006A
c_i U+0E38 U+0E22	U+0075 U+006A
U+0E33	U+0061 U+006D
U+0E24	U+0072 U+026F
U+0E26	U+006C U+026F

U+0E41 c_i U+0E27	U+025B U+02D0 U+0077
c_i U+0E32 U+0E27	U+0061 U+02D0 U+0077
U+0E40 c_i U+0E35 U+0E22 U+0E27	U+0069 U+0061 U+0077
c_i U+0E32 U+0E22	U+0061 U+02D0 U+006A
c_i U+0E2D U+0E22	U+0254 U+02D0 U+006A
U+0E42 c_i U+0E22	U+006F U+02D0 U+006A
U+0E40 c_i U+0E22	U+0264 U+02D0 U+006A
c_i U+0E27 U+0E22	U+0075 U+0061 U+006A
U+0E40 c_i U+0E37 U+0E2D U+0E22	U+026F U+0061 U+006A
U+0E24 U+0E45	U+0072 U+026F U+02D0
U+0E26 U+0E45	U+006C U+026F U+02D0

Lorsque vous avez trouvé les prononciations pour les consonnes et voyelles, il reste à construire la prononciation pour la syllabe. Vous placez premièrement la prononciation de la consonne initiale, elle sera suivie de la prononciation de la voyelle et finalement la prononciation de la consonne finale si présente.

Si votre code détecte un caractère non valide où une suite de caractère non valide, alors il lance l'exception `NoSuchElementException` de Java.

2.1.3 Exemples

1. Entrée : "\u0E24", sortie : "\u0072\u026F".
   La voyelle U+0E24 n'a pas de consonne (avant-dernière ligne de la table de gauche). Elle est simplement traduite.
2. Entrée : " \u0E0D\u0E34", sortie : " \u006A\u0069".
   Le premier caractère est une consonne, le deuxième est une voyelle. Cela correspond au patron de la deuxième entrée de la table de gauche. Dans ce cas, la voyelle devient \u0069.

3. Entrée : "\u0E40\u0E2B\u0E35\u0E22\u0E30", sortie : "\u0068\u0069\u0261\u0294".
   Le deuxième caractère est une consonne (\u0E2B), et sera traduit par \u0068. Le reste des caractères correspondent au patron de la 10e entrée de la table de gauche, ce qui nous donne la prononciation de la voyelle : \u0069\u0261\u0294.
4. Entrée : "\u0E1C\u0E27\u0E28", sortie : "\u0070\u02B0\u0075\u0061\u0074".
   Dans ce cas, nous avons le deuxième patron de la 12e entrée de la table de droite. Ce patron utilise une consonne initiale (\u0E1C dans l'exemple) et une consonne finale (\u0E28 dans l'exemple).

2.2 CODE

Votre code doit au minimum contenir une classe nommée `Thai`. Cette classe doit contenir la méthode suivante.

```
public static String versAPI( String texte )
```

C'est cette méthode qui sera appelée par le jeu d'essais (tests unitaires). Vous devez utiliser la classe `StringBuilder` de Java pour construire votre résultat. La méthode `toString` de la classe `StringBuilder` permet d'extraire le résultat.

Vous pouvez seulement importer des librairies `java.util`.

3 DIRECTIVES

Les sections suivantes décrivent les attentes pour le devoir.

3.1 DIRECTIVES POUR LA CONSTRUCTION DU PROJET.

- Placez vos noms au début du fichier `Thai.java`.
- Lorsque vous ajoutez une méthode, vous devez l'ajouter dans la classe appropriée.
- 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 6 pour une méthode.
   e. Construisez des classes au besoin.
3. Indentez votre code. Assurez-vous que l'indentation est faite avec des espaces.
4. Commentaires
   a. Commentez l'entête de chaque classe et méthode.
   b. Une ligne contient soit un commentaire, soit du code, pas les deux.

c. Utilisez des noms d'identificateur significatif.
d. Une ligne de commentaire ou de code ne devrait pas dépasser 120 caractères. Continuez sur la ligne suivante au besoin.
e. Nous utilisons Javadoc :
    1. La première ligne d'un commentaire doit contenir une description courte (1 phrase) de la méthode ou la classe.
    2. Courte.
    3. Complète.
    4. Commencez la description avec un verbe.
    5. Assurez-vous de ne pas simplement répéter le nom de la méthode, donnez plus d'information.
    ii. Ensuite, au besoin, une description détaillée de la méthode ou classe va suivre.
        1. Indépendant du code. Les commentaires d'entêtes décrivent ce que la méthode fait, ils ne décrivent pas comment c'est fait.
        2. Si vous avez besoin de mentionner l'objet courant, utilisez le mot 'this'.
    iii. Ensuite, avant de placer les **tags,** placez une ligne vide.
    iv. Placez les **tag** @param, @return et @throws au besoin.
        1. @param : décris les valeurs acceptées pour la méthode. Vous devez commenter les paramètres de vos méthodes.
    v. Dans les commentaires, placer les noms de variable et autre ligne de code entre les tags <code> … </code>.
    vi. Écrivez les commentaires à la troisième personne.
5. Vous pouvez seulement importer des librairies `java.util`.

4 REMISE

Remettre le TP par l'entremise de Moodle. **Placez vos fichiers `'*.java'` `seulement`, pas de sous-répertoire**. Votre archive doit être dans un dossier compressé de **Windows**, vous devez remettre l'archive. Le TP est à remettre avant le 21 juillet 23 :59.

4.1 ÉVALUATION

- Fonctionnalité (10 pts) : des tests partiels vous seront remis. Un test plus complet sera appliqué à votre TP.
- Structure (2 pt) : veillez à utiliser correctement le mécanisme de classe, interface et de méthode.
- Lisibilité (2 pts) : commentaire, indentation et noms d'identificateur significatif.
- Point bonus : Il est possible d'avoir jusqu'à un point de plus pour le TP.
    - Pour être éligible au bonus, votre projet doit
        - Avoir 10/10 à la fonctionnalité.
        - Aucun break, continue.
        - Un seul return par méthode.
    - Le calcul suivant est appliqué :

- nombre de mots ‘case’ et ‘catch’ + 2(nombre de mot ‘if’, ‘while’, ‘for’ + nombre d’opérateurs ternaire).
- L’étudiant qui a le plus petit score aura 1 point, l’étudiant qui a le plus grand score aura 0 point bonus. Les autres sont distribués dans l’intervalle]0..1[.