TD n°7

Analyse lexicale — JFlex

Exercice 1 *On considère une spécification JFlex avec les règles lexicales suivantes :*

```
1 "abra"        {System.out.print("R1");}
2 "abracad"     {System.out.print("R2");}
3 [a-z]+        {System.out.print("R3");}
4 [ ]           {}
```

1. *Donner la séquence produite pour le flot d'entrée suivant :*

```
abra abracad abracadabra
```

2. *Donner la règle qu'il faudrait ajouter (et sa position dans la liste) pour que toute occurrence isolée du mot "evanesco" soit ignorée.*
   Par exemple, le flot d'entrée suivant devrait produire R1R1R3 :

```
abra evanesco abra evanesco evanescopus
```

3. *Est-ce que les deux spécifications JFlex suivantes sont interchangeables (dans le sens où les deux analyseurs associés, quand lancés sur un même flot d'entrée, produisent toujours la même sortie) ? Le cas échéant, donner un flot d'entrée qui les différencie.*

```
1 "abra"    {System.out.print("M1");}
2 "abracad" {System.out.print("M2");}
3 "ked"     {System.out.print("M3");}
4 "avra"    {System.out.print("M4");}
5 [ ]       {}
```

(a) Spécification 1

```
1 [ ]       {}
2 "avra"    {System.out.print("M4");}
3 "ked"     {System.out.print("M3");}
4 "abracad" {System.out.print("M2");}
5 "abra"    {System.out.print("M1");}
```

(b) Spécification 2

Exercice 2 *On considère l'alphabet initial* $\Sigma = \{0, \ldots, 9, +, \times\}$.

1. *Écrire une spécification JFlex qui permet d'afficher* `NOMBRE` *quand un nombre est lu,* `SYMBOLE` *quand c'est un symbole.*
2. *Ajouter une règle qui stoppe l'exécution en cas de caractère invalide (par exemple en utilisant l'instruction* `throw new Error();`*).*
3. *On étend* Σ *avec l'espace et les parenthèses. Ajouter une règle permettant de les ignorer.*
4. *On veut maintenant distinguer les chiffres (entiers compris entre* 0 *et* 9*) des nombres (entiers supérieurs ou égal à* 10*) : modifier la spécification initiale donnée à la question 1.*
5. *Par « sécurité », faire en sorte que les matricules d'agents secrets britanniques dangereux (un nombre commençant par deux* 0 *consécutifs) soient remplacés par* `CLASSIFIED`*.*
6. *Intuitivement, est-il possible de rejeter toute expression arithmétique mal parenthésée ?*

Exercice 3 *On s'intéresse aux ordinaux anglais abrégés, où le nombre (le cardinal) est écrit en chiffres :*

1st, 2nd, 3rd, 4th,. . . , 9th, 10th, 11th, 12th,. . . , 19th, 20th, 21st, 22nd, 23rd,. . .

Pour déterminer le suffixe, on considère le dernier chiffre du nombre : si c'est 1, on ajoute le suffixe "st" ; si c'est 2, le suffixe est "nd" ; si c'est 3, le suffixe est "rd" ; sinon le suffixe est "th". Il existe une exception : si l'avant-dernier chiffre du nombre est 1, le suffixe est toujours "th".

Écrire une spécification JFlex permettant d'ajouter le suffixe associé à chaque nombre rencontré (et ignorer le reste).

Exercice 4 *On souhaite mettre au pluriel des phrases très simples :*

<article> <nom> <verbe> <complément>
<pronom personnel> <verbe> <complément>

Dans cet exercice, le nom aura un pluriel obtenu par ajout d'un simple "s".
Le verbe sera un verbe du premier groupe, conjugué ici au présent de l'indicatif.
Le complément sera laissé tel quel.

Exemples :

1. *La chouette mange de l'herbe et des souris. → Les chouettes mangent de l'herbe et des souris.*
2. *Un renard marche dans le bois au clair de lune. → Des renards marchent dans le bois au clair de lune.*
3. *Je souhaite la paix. → Nous souhaitons la paix.*

Déterminer les états nécessaires (en plus de l'état `YYINITIAL`*) : l'idée générale est qu'un état indique la nature de ce que l'on vient de lire (article, nom (au singulier), etc). Dessiner l'automate. Écrire une spécification basée sur cet automate.*

Envisager le traitement des verbes pronominaux.

Exemples :

1. *Un passant se promène sur le quai brumeux. → Des passants se promènent sur le quai brumeux.*
2. *Tu te roules par terre. → Vous vous roulez par terre.*