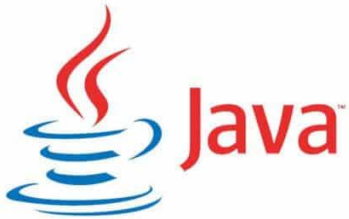

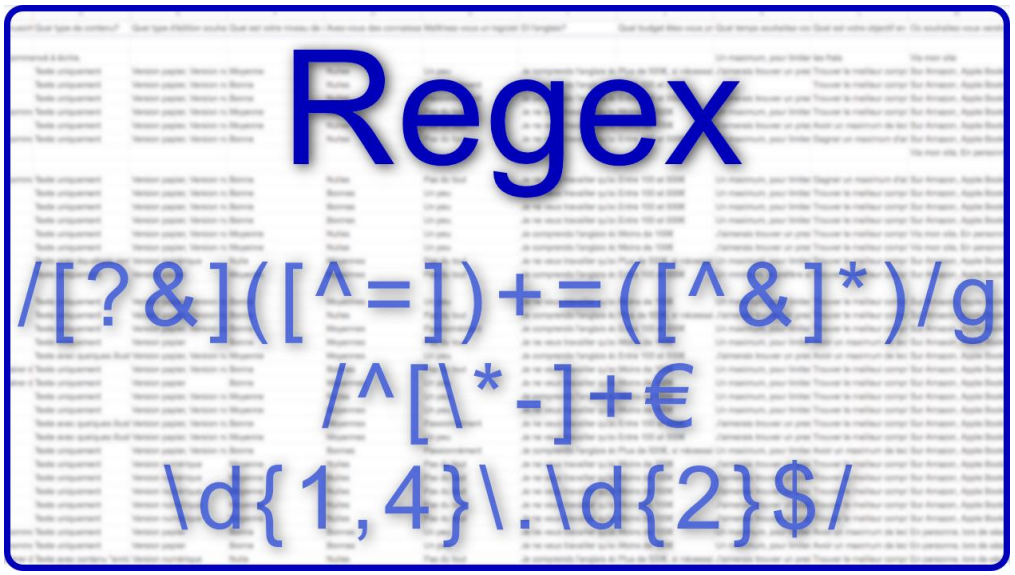

Les expression régulières (RegEx)

Introduction:

*Une expression **régulière ou rationnelle** est une chaine de caractères qui décrit un ensemble de chaines de caractères.*

*Par exemple l'expression régulière **[0-9][a-z]** décrit l'ensemble des chaines de caractères composées d'un chiffre et d'une lettre.*

Elles servent à réaliser :

- ***des filtres** : ne conserver que certaines lignes d'un fichier texte : par exemples que les lignes de la forme variable=valeur.*
- ***des contrôles** : vérifier qu'une donnée entrée par un utilisateur a bien le format d'une adresse ip par exemple.*
- ***des substitutions** : remplacer un motif par une chaine de caractères précise : par exemple, remplacer les majuscules par des minuscules.*
- ***des découpages** : récupérer une partie d'une chaine de caractères par exemple une date placée dans une chaine de caractères. Ou bien découper une ligne par rapport aux « ; » dans le cas d'un fichier .csv.*

Créer une expression régulière:

Pour créer une expression régulière en java, on respecte une certaine syntaxe qui est décrite dans la documentation officielle sur le site d'oracle: classe Pattern.

Créer une expression régulière:

Les caractères

X Le caractère X
**** Le caractère \
\t Le caractère tabulation
\n Le caractère nouvelle ligne
\r Le caractère retour chariot
\f Le caractère saut de page
. Un caractère quelconque

Exemple:
AB\tCD représente la chaine de caractère AB suivi d'une tabulation suivi de CD.

Les classes de caractères

[abc] Les caractères a, b ou c
[^abc] Les caractères qui ne sont pas a, b ou c
[a-z] Un caractère de a à z
[a-zA-Z] Un caractère de a à z minuscule ou majuscule
[0-9] Un caractère numérique

Exemple : [0-9][a-z] représente une chaine de caractères constituée d'un chiffre puis d'une lettre.

Créer une expression régulière:

Les caractères de répétition

X? X une fois ou zéro fois
X X zéro ou plusieurs fois*
X+ X une fois au moins (XX)*
X{n} X n fois
X{n,} X, au moins n fois
X{n,m} X entre n et m fois

Classes de caractères POSIX

\p{Lower} Une misnuscule : [a-z]
\p{Upper} Une majuscule :[A-Z]
\p{Digit} Un chiffre : [0-9]
\p{Punct} Ponctionation : !"#$%&'()+,-./:;<=>?@[\]^_`{|}~*
\p{Blank} Espace ou tabulation : [\t]
\p{XDigit} Un caractère hexadécimal: [0-9a-fA-F]
\p{Space} Un caractère blanc

Exemples:

Expression régulière correspondant à une adresse IP

Une adresse IP est de la forme suivante : 255.255.0.1 ou bien 127.0.0.1.

Son format est donc :

- *Un à trois chiffres,*
- *point,*
- *un à trois chiffres,*
- *point,*
- *un à trois chiffres,*
- *point,*
- *un à trois chiffres.*

[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}

Exemples:

Expression régulière correspondant à une date

Une date au format 28 décembre 2021 suit le motif suivant

Son format est donc :

- *un à deux chiffres,*
- *un espace,*
- *un nombre indéfini de lettres,*
- *un espace,*
- *puis quatre chiffres.*

[0-9]{1,2} [a-z]+ [0-9]{4}

Utilisation avec java:

La méthode matches de la classe String

`public boolean matches(String regex)`

La méthode matches de la classe String permet de savoir si une chaine de caractère respecte une expression régulière. Pour cela, on passe l'expression régulière en paramètre de la méthode et la méthode retourne true si l'expression régulière est vérifiée par la chaine de caractères.

La méthode replaceAll de la classe Stringpublic

`public String replaceAll(String regex, String replacement)`

La méthode replaceAll permet de remplacer toutes les occurrences d'une expression régulière par une expression de remplacement.

La méthode split de la classe String

`public String[] split(String regex)`

La méthode split permet de découper une chaine de caractères en fonction d'une expression régulière. Le résultat du découpage est retourné dans un tableau.

Applications:

Admettons que nous ayons récupéré le contenu du fichier dans une ArrayList et qu'on souhaite extraire une liste contenant uniquement les lignes qui ne sont pas des commentaires : (Une ligne est un commentaire si et seulement si le premier caractère non blanc est un #).

```
public static void main(String[] args) {
    ArrayList<String> listeInitiale = new ArrayList();
    listeInitiale.add("# 102.54.94.97 astier.ardeche.com # serveur source");
    listeInitiale.add(" # 38.78.63.10 x.ardeche.com # hôte client x");
    listeInitiale.add("");
    listeInitiale.add("127.0.0.1 localhost # machine courante");
    for(String s : listeInitiale){
        if(!s.matches("\\p{Space}*[#].*") && s.length()>0)
            System.out.println(s);
    }
}
```

Applications:

Réaliser des contrôles

Pour vérifier qu'un champ correspond bien à une adresse ip, on peut utiliser la méthode matches de String.

```
public static void main(String[] args) {
	String champ = "255.255.10.1";
	System.out.println(champ.matches("\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}"));
}
```

Réaliser des substitutions

Si on veut remplacer tout les espaces multiples d'un fichier par un seul espace, on peut utiliser la méthode rempaceAll avec une expression régulière.

```
public static void main(String[] args){
   String test = "1 2   3 4 5  6 7   8 ";
   System.out.println(test.replaceAll("\\p{Space}{2,}", " "));

}
```