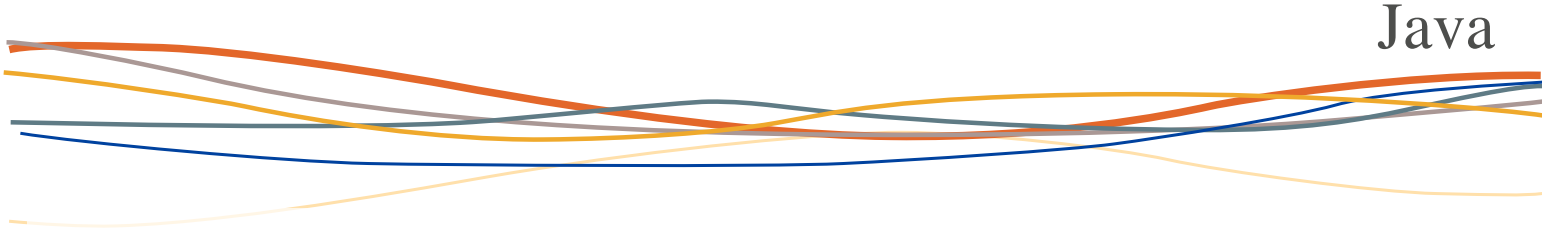
IUT Lyon 1

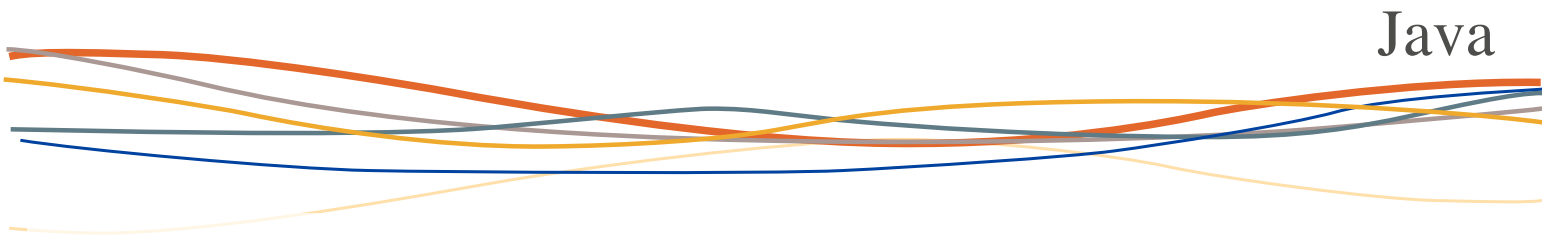
l'excellence technologique

Expressions régulières

LP IEM



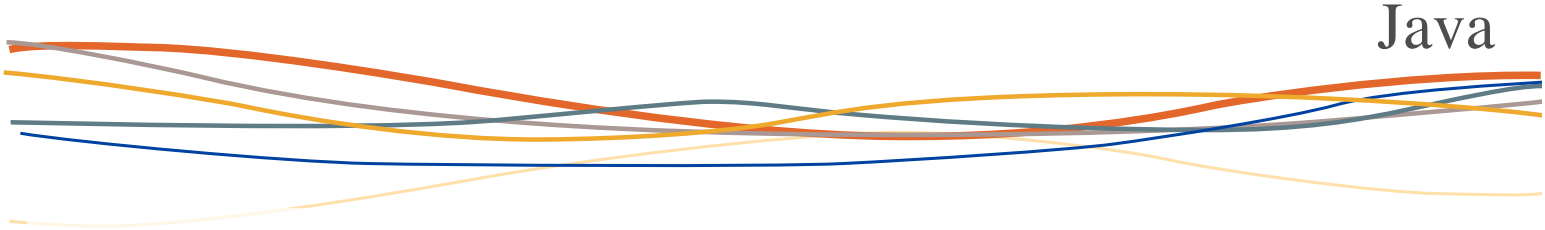
- Expressions régulières
  - Une expression régulière est une chaîne de caractère permettant de décrire un ensemble de chaîne de caractère
  - Utilisation
    - Validation de chaînes de caractères
      - email
      - mot de passe (nombre de caractères, …)
    - Recherche et extraction de données dans du texte
    - Traitement sur des chaînes
      - Substitution intelligente de sous-parties de chaînes



Expressions régulières

Aperçu de la syntaxe

^	Début de chaîne
$	Fin de chaîne
[abc]	Classe de caractère (a, b ou c)
[a-fA-F]	N'importe quel caractère pris dans a, b, …f ou A, B, …F
[^abc]	N'importe quel caractère sauf a,b et c
a|b	a ou b

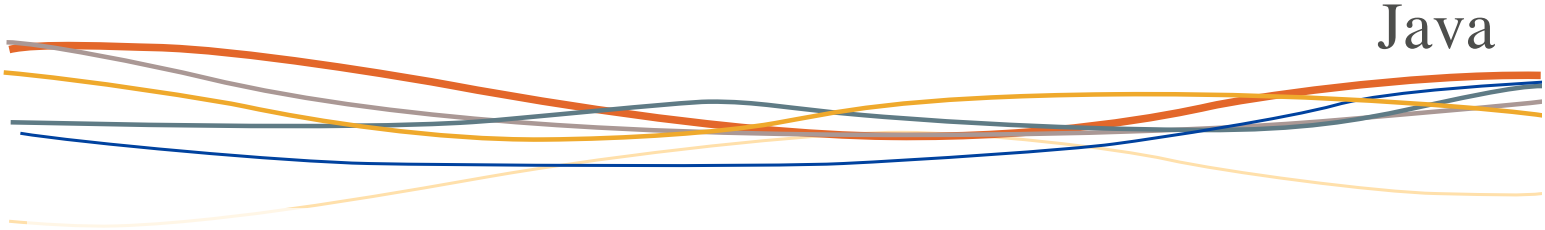


Expressions régulières

Aperçu de la syntaxe

Classes de caractères prédéfinies

\d	un digit [0-9]
\D	pas un digit [^0-9]
\w	Un caractère d'un mot [a-zA-Z_0-9]
\W	Tout sauf un caractère de mot [^\w]
\s	Un caractère d'espacement [\t\n\x0B\f\r]
\S	Tout sauf un caractère d'espacement [^\s]
.	N'importe quel caractère

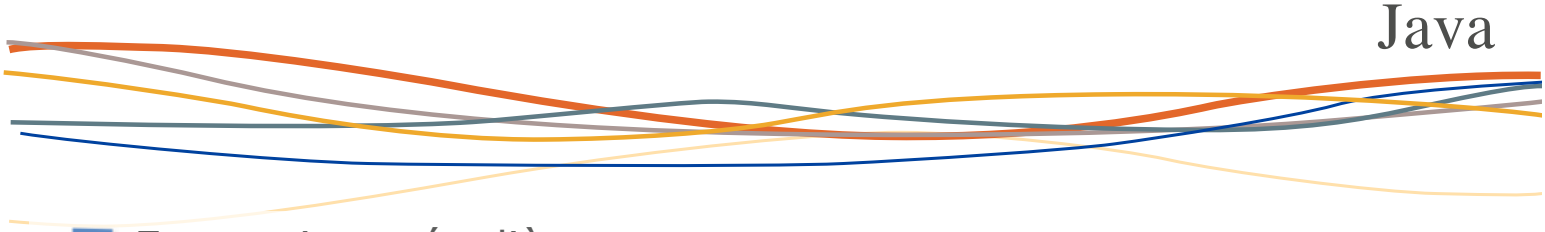


Expressions régulières

Aperçu de la syntaxe

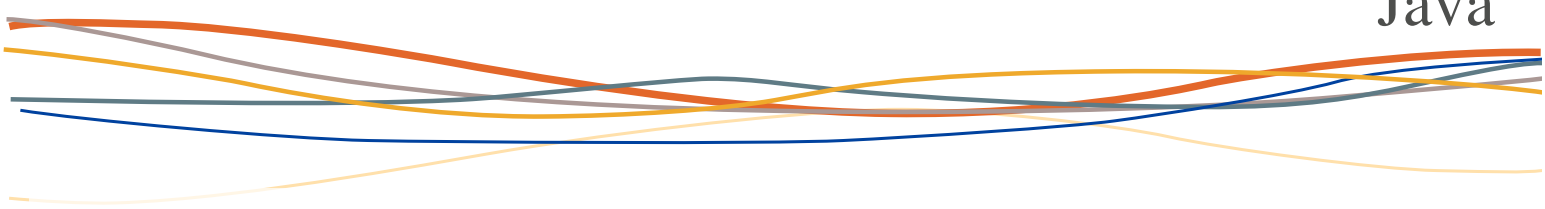
Multiplicité (quantificateurs)

Greedy	Reluctant	
a+	a+?	1 ou plusieurs a
a*	a*?	0 ou plusieurs a
a?	a??	0 ou 1 a
a{3}	a{3}?	Exactement 3 a
a{3,}	a{3,}?	Au moins 3 a
a{8,12}	a{8,12}?	8 à 12 a



- Expressions régulières
  - Aperçu de la syntaxe
    - Exemples
      - "^[A-Z][a-z]*\s.+[!]$"
        - "Hello world !" ➙ match
        - "hello world !" ➙ KO (doit commencer par une majuscule)
      - Validation de la saisie d'une case en bataille navale
        - Doit commencer par une lettre entre A et J (10 lignes) en minuscule ou en majuscule
        - Est suivi du numéro de colonne entre 1 et 10

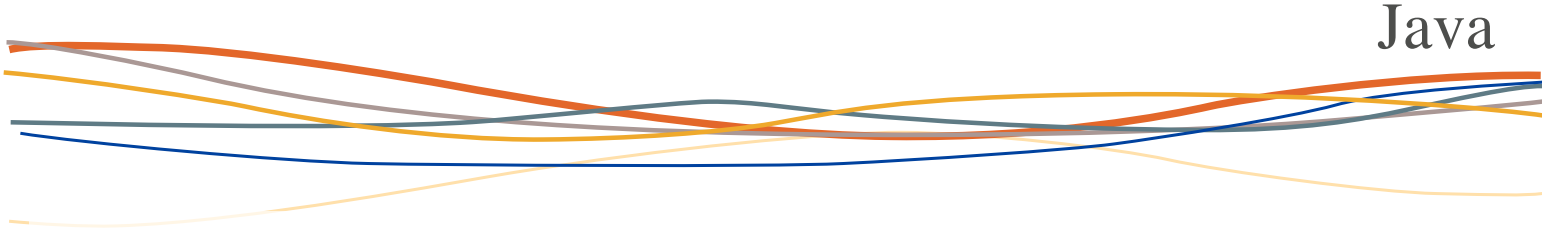
```
^[A-Ja-j]([1-9]|10)$
```



- Expressions régulières
  - Aperçu de la syntaxe
    - Validation coordonnée bataille navale

```
public static void main(String[] args) {
     System.out.println("B9".matches("^[A-Ja-j]([1-9]|10)$"));
     System.out.println("B11".matches("^[A-Ja-j]([1-9]|10)$"));
   }
```

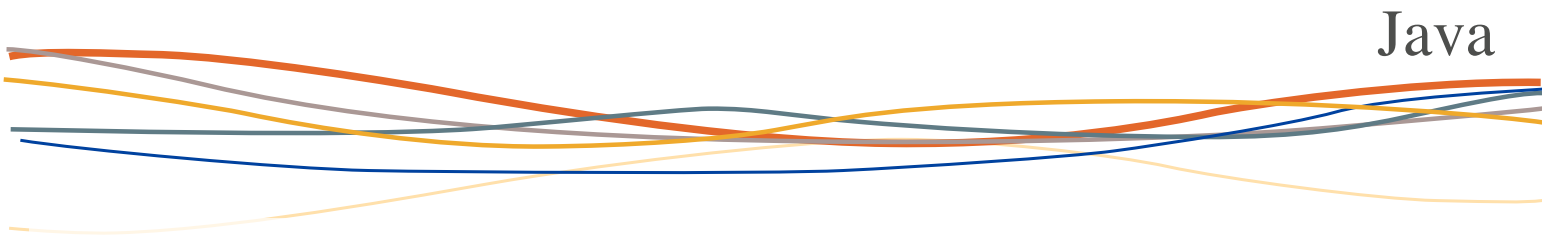
```
Résultat :
true
false
```



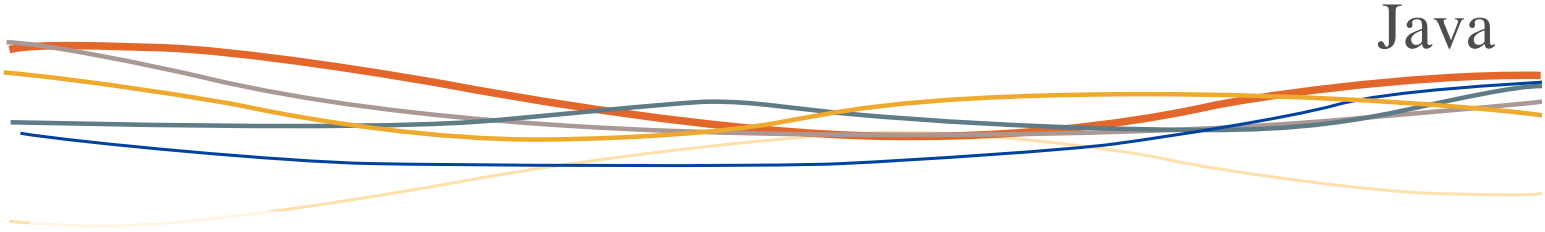
- Expressions régulières
  - Aperçu de la syntaxe
    - Groupes
      - Les parties de l'expression régulière entre parenthèse constituent des groupes qu'il est possible d'extraites ensuite

```
public static void main(String[] args) {
   Pattern pattern = Pattern.compile("<(.*)_(.*)>");
   Matcher matcher = pattern.matcher("<Toto_Tutu>");
   matcher.find();
   System.out.println(matcher.group(0));
   System.out.println(matcher.group(1));
   System.out.println(matcher.group(2));
}
```

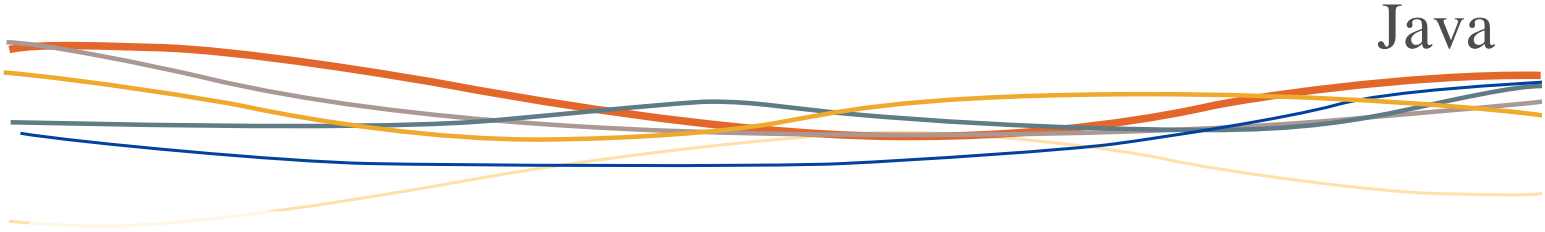
```
Résultat :
<Toto_Tutu>
Toto
Tutu
```



- Expressions régulières
  - Aperçu de la syntaxe
    - Greedy vs Reluctant
      - Les quantificateurs greedy mangent le maximum de caractères
      - Exemple
      - "<td>Contenu</td>"
        - L'expression régulière <.*> capturera la totalité de la chaîne
        - L'expression régulière <.*?> s'arrêtera à <td>



- Expressions régulières
  - Mise en oeuvre en Java
    - Nécessité d'échapper les \
      - remplacer par ex. \s par \\s
    - Utilisation des classes Pattern et Matcher
    - Certaines méthode de la classe String acceptent une expression régulière sous forme de chaîne



- Expressions régulières
  - Références
    - https://www.debuggex.com/#cheatsheet
    - http://regex101.com/#pcre
    - http://www.regular-expressions.info
    - http://docs.oracle.com/javase/tutorial/essential/regex/index.html
    - http://ocpsoft.org/opensource/guide-to-regular-expressions-in-java-part-1/