EXPRESIONES REGULARES

Introducción

Una expresión regular define un patrón que puede o no cumplir una cadena de caracteres. Los patrones de búsqueda pueden servir para comprobar:

- que la fecha leída cumple el patrón dd/mm/aaaa
- que un NIF está formado por 8 cifras, un guión y una letra
- que una dirección de correo electrónico es una dirección válida.
- que una contraseña cumple unas determinadas condiciones.
- que una URL es válida.
- cuántas veces se repite dentro de la cadena una secuencia de caracteres
- etc.

La comprobación del patrón de búsqueda en un String se realiza de izquierda a derecha y se analiza si se cumpe el patrón, 0, 1 o más veces.

Por ejemplo, La expresión regular "101" la encontramos dentro del String "0**101**000**101**001" la encontramos 2 veces.

Símbolos utilizados en las expresiones regulares

Expresión	Descripción
.	Un punto indica cualquier carácter
^*expresión*	En este caso el String debe contener la expresión al principio.
***expresión*$**	El símbolo $ indica el final del String. En este caso el String debe contener la expresión al final.
[abc]	Los corchetes representan una definición de conjunto. En este caso el String debe contener las letras a ó b ó c.
[abc][12]	El String debe contener las letras a ó b ó c seguidas de 1 ó 2
[^abc]	El símbolo ^ **dentro de los corchetes** indica **negación**. En este caso el String debe contener cualquier carácter excepto a ó b ó c.
[a-z1-9]	Rango. Indica las letras minúsculas desde la a hasta la z (ambas incluidas) y los dígitos desde el 1 hasta el 9 (ambos incluidos)
AB	Concatenación. A seguida de B
A|B	OR. El carácter A o el B

Expresión	Descripción
\\d	Dígito. Equivale a [0-9]
\\D	No dígito. Equivale a [^0-9]
\\s	Espacio en blanco. Equivale a [\t\n\x0b\r\f]
\\S	No espacio en blanco. Equivale a [^\s]
\\w	Una letra mayúscula o minúscula, un dígito o el carácter _ Equivale a [a-zA-Z0-9_]
\\W	Es la negación de //w
\\b	Límite de una palabra.

Expresión	Descripción
{X}	Indica que lo que va justo antes de las llaves se repite X veces
{X,Y}	Indica que lo que va justo antes de las llaves se repite mínimo X veces y máximo Y veces. También podemos poner {X,} indicando que se repite un mínimo de X veces sin límite máximo.
*****	Indica 0 ó más veces. Equivale a {0,}
+	Indica 1 ó más veces. Equivale a {1,}
?	Indica 0 ó 1 veces. Equivale a {0,1}

Expresión	Descripción
patrón1|patrón2	El carácter | equivale a la expresión lógica OR. Comprueba si cumple el *patrón1* o el *patrón2*
patrón1 (?=patrón2)	Encuentra el patrón1 solo si está seguido del patrón2
patrón1 (?!patrón2)	Encuentra el patrón1 solo si **no** está seguido del patrón2

Clases implicadas en el uso de patrones

Para usar expresiones regulares en Java se usa el package **java.util.regex** que contiene las clases **Patter** y **Matcher** y la excepción **PatternSyntaxException**

- Clase **Pattern:** Un objeto de esta clase representa la expresión regular. Contiene el método **compile**(*String regex*) que recibe como parámetro la expresión regular y devuelve un objeto de la clase Pattern.
- Clase **Matcher:** Un objeto Pattern contiene el método **matcher**(*cadena*) que devuelve un objeto Matcher**.** Esta clase contienen los métodos:
  - **matches**(CharSequence input) que recibe como parámetro el *String* a validar y devuelve *true* si coincide con el patrón en su totalidad.

```
Pattern pat = Pattern.compile(".*dam.*");
Matcher mat = pat.matcher(cadena);
if (mat.matches()) {//Si cadena coincide exactamente con el patrón
      System.out.println("SI");
} else {
      System.out.println("NO");
}
```

  - **find**() indica si el *String* contienen el patrón que puede coincidir con una subcadena.

```
Pattern pat = Pattern.compile("dam");
Matcher mat = pat.matcher(cadena);
if (mat.find()) { //Si encuentra el patrón en alguna subcadena
      System.out.println("SI");
} else {
      System.out.println("NO");
}
```

Ejemplos

Comprobar si el String *cadena* **comienza** con la la subcadena *"dam".*

```
Pattern pat = Pattern.compile("^dam");
Matcher mat = pat.matcher(cadena);
if (mat.find()) {
    System.out.println("SI");
} else {
    System.out.println("NO");
}
```

Comprobar si el String *cadena* **comienza** por "abc".

```
Pattern pat = Pattern.compile("^abc.*");
Matcher mat = pat.matcher(cadena);
if (mat.matches()) {
   System.out.println("Válido");
} else {
   System.out.println("No Válido");
}
```

Comprobar si el String *cadena* **comienza** por "abc" o por "Abc".

```
Pattern pat = Pattern.compile("^[aA]bc.*");
Matcher mat = pat.matcher(cadena);
if (mat.matches()) {
   System.out.println("Válido");
} else {
   System.out.println("No Válido");
}
```

Comprobar si el String *cadena* **está formado** por un **mínimo de 5 letras mayúsculas** o **minúsculas y un máximo de 10**.

```
Pattern pat = Pattern.compile("[a-zA-Z]{5,10}");
Matcher mat = pat.matcher(cadena);
if (mat.matches()) {
    System.out.println("SI");
} else {
    System.out.println("NO");
}
```

Comprobar si el String *cadena* **no comienza** por un dígito.

```
Pattern pat = Pattern.compile("^[^\\d].*");
Matcher mat = pat.matcher(cadena);
if (mat.matches()) {
    System.out.println("SI");
} else {
    System.out.println("NO");
}
```

Comprobar si el String *cadena* **finaliza** en un dígito.

```
Pattern pat = Pattern.compile(".*[\\d]$");
Matcher mat = pat.matcher(cadena);
if (mat.matches()) {
    System.out.println("SI");
} else {
    System.out.println("NO");
}
```

Comprobar si el String *cadena* **solo** contiene los caracteres a o b

```
Pattern pat = Pattern.compile("(a|b)+");
Matcher mat = pat.matcher(cadena);
if (mat.matches()) {
    System.out.println("SI");
} else {
    System.out.println("NO");
}
```

Comprobar si el String *cadena* contiene un 1 y ese 1 no tiene a continuación un 2

```
Pattern pat = Pattern.compile(".*1(?!2");
Matcher mat = pat.matcher(cadena);
if (mat.matches()) {
    System.out.println("SI");
} else {
    System.out.println("NO");
}
```

Comprobar la introducción de un **NIF** válido

```
Scanner miScanner = new Scanner(System.in);

Pattern pat = Pattern.compile("[0-9]{7,8}[A-Z a-z]");

System.out.println("Introduce un Dni correcto: ");
String dniIntroducido = miScanner.nextLine();

Matcher mat = pat.matcher(dniIntroducido);

while(!mat.matches()){
   System.out.println("El Dni introducido es incorrecto, por favor "
                   + "introduzca un Dni válido.");
   System.out.print("Introduce un Dni correcto:");
   dniIntroducido = miScanner.nextLine();
   mat = pat.matcher(dniIntroducido);
}

System.out.println("Bien, el Dni " + dniIntroducido + " es un Dni
válido");
```

Comprobar la introducción de un **email** válido

```
Scanner sc = new Scanner(System.in);
String email;
System.out.print("Introduce email: ");
email = sc.nextLine();
Pattern pat = Pattern.compile("^[\\w-]+(\\.[\\w-]+)*@[A-Za-z0-
9]+(\\.[A-Za-z0-9]+)*(\\.[A-Za-z]{2,})$");
Matcher mat = pat.matcher(email);
if(mat.find()){
    System.out.println("Correo Válido");
}else{
    System.out.println("Correo No Válido");
}
```

Hemos usado la siguiente expresión regular para comprobar si un email es válido: `"^[\\w-]+(\\.[\\w-]+)*@[A-Za-z0-9]+(\\.[A-Za-z0-9]+)*(\\.[A-Za-z]{2,})$"`

La explicación de cada parte de la expresión regular es la siguiente:

[\\w-]+	**Inicio del email:** **El signo + indica que debe aparecer uno o más elementos que están indicados entre corchetes.** **\\w indica caracteres de la A a la Z tanto mayúsculas como minúsculas, dígitos del 0 al 9 y el carácter _** **Carácter – que puede existir, ya que en el patrón w no está incluido.** **En lugar de usar \\w podemos escribir el rango de caracteres con lo que esta expresión quedaría así:** **[A-Za-z0-9-_]+**
(\\.[\\w-]+)*	A continuación: **El *** indica que este grupo puede aparecer cero o más veces. El email puede contener de forma opcional un punto seguido de uno o más de los caracteres entre corchetes.
@	A continuación debe contener el **carácter @**
[A-Za-z0-9]+	**Después de la @** el email debe contener uno o más de los caracteres que aparecen entre los corchetes
(\\.[A-Za-z0-9]+)*	Seguido (opcional, 0 ó más veces) de un punto y 1 ó más de los caracteres entre corchetes
(\\.[A-Za-z]{2,})	Seguido de un punto y al menos 2 de los caracteres que aparecen entre corchetes (final del email)

Fuentes

http://puntocomnoesunlenguaje.blogspot.com/2013/07/ejemplos-expresiones-regulares-java-split.html

http://chuwiki.chuidiang.org/index.php?title=Expresiones_Regulares_en_Java