**Una breve lista de los más utilizados:**

^ Indica el principio de una cadena

$ Indica el final de una cadena

() Un agrupamiento de parte de una expresión

[] Un conjunto de caracteres de la expresión

{} Indica un número o intervalo de longitud de la expresión

. Cualquier carácter salvo el salto de línea

? 0-1 ocurrencias de la expresión

+ 1-n ocurrencias de la expresión

\* 0-n ocurrencias de la expresión

\ Para escribir un carácter especial como los anteriores y que sea tratado como un literal

| Para indicar una disyunción lógica (para elegir entre dos valores: a|b se tiene que cumplir al menos uno de los dos)

Expresión regular para fecha

Imaginemos la fecha en formato dd/mm/yyyy. Son grupos de dos cifras separadas por barras. En una expresión regular \d representa una cifra. El día pueden ser una o dos cifras, es decir \d{1,2], el mes igual y el año vamos a obligar que sean cuatro cifras exactamente \d{4}

Si queremos comprobar que una cadena leída por teclado cumple ese patrón, podemos usar la clase Pattern. A la clase Pattern le decimos el patrón que queremos que cumpla nuestra cadena y nos dice si la cumple o no.

El siguiente ejemplo comprueba si la cadena cumple con la expresión regular. Ten en cuenta que cuando en java metemos un caracter \ dentro de una cadena delimitada por "", debemos "escapar" esta \ con otra \, por ello todas nuestras \ en la expresión regular, se convierten en \\ en nuestro código java.

String regexp = "\\d{1,2}/\\d{1,2}/\\d{4}";

// Lo siguiente devuelve true

System.out.println(Pattern.matches(regexp, "11/12/2014"));

System.out.println(Pattern.matches(regexp, "1/12/2014"));

System.out.println(Pattern.matches(regexp, "11/2/2014"));

// Los siguientes devuelven false

System.out.println(Pattern.matches(regexp, "11/12/14")); // El año no tiene cuatro cifras

System.out.println(Pattern.matches(regexp, "11//2014")); // el mes no tiene una o dos cifras

System.out.println(Pattern.matches(regexp, "11/12/14perico")); // Sobra "perico"

**Supongamos que queremos que el mes se exprese como "ene", "feb", "mar", ... en vez de como un número.** Cuando hay varias posibles cadenas válidas, en la expresión regular se ponen entre paréntesis y separadas por |. Es decir, algo como esto (ene|feb|mar|abr|may|jun|jul|ago|sep|oct|nov|dic). Si además nos da igual mayúsculas o minúsculas, justo delante ponemos el flag de case insensitive (?i) (la 'i' es de ignore case)

El siguiente código muestra un ejemplo completo de esto.

String literalMonthRegexp = "\\d{1,2}/(?i)(ene|feb|mar|abr|may|jun|jul|ago|sep|oct|nov|dic)/\\d{4}";

// Lo siguiente devuelve true

System.out.println(Pattern.matches(literalMonthRegexp, "11/dic/2014"));

System.out.println(Pattern.matches(literalMonthRegexp, "1/nov/2014"));

System.out.println(Pattern.matches(literalMonthRegexp, "1/AGO/2014")); // Mes en mayúsculas

System.out.println(Pattern.matches(literalMonthRegexp, "21/Oct/2014")); // Primera letra del mes en mayúsculas.

// Los siguientes devuelven false

System.out.println(Pattern.matches(literalMonthRegexp, "11/abc/2014")); // abc no es un mes

System.out.println(Pattern.matches(literalMonthRegexp, "11//2014")); // falta el mes

System.out.println(Pattern.matches(literalMonthRegexp, "11/jul/2014perico")); // sobra perico

Expresión regular para DNI

En España existe el DNI (Documento Nacional de Identidad), últimamente también llamado NIF (Número de identificación fiscal), que lleva un número único y sirve para identificar a la persona. Este número son 8 cifras seguidas de una letra, que normalmente se escribe en mayúscula. Esta letra es una especie de checksum de las cifras anteriores, por lo que hay un algoritmo para validar que el número completo es correcto. Quedan excluidas las letras 'I', 'O' y 'U'. Las dos primeras por poder confundirse con uno y cero respectivamente. La tercera por algún extraño motivo.

Una expresión regular no va a realizar este checksum, pero sí nos puede ayudar a hacer una primera comprobación: 8 cifras y una letra mayúscula. La expresión regular puede ser así \d{8}[A-HJ-NP-TV-Z]

El siguiente código muestra un ejemplo completo de la expresión regular de DNI

String dniRegexp = "\\d{8}[A-HJ-NP-TV-Z]";

// Lo siguiente devuelve true

System.out.println(Pattern.matches(dniRegexp, "01234567C"));

// Lo siguiente devuelve faslse

System.out.println(Pattern.matches(dniRegexp, "01234567U")); // La U no es válida

System.out.println(Pattern.matches(dniRegexp, "0123567X")); // No tiene 8 cifras

**Expresión regular para email**

Antes de nada, no existe una expresión regular para email que sea 100% fiable, puesto que hay muchos formatos válidos de email y muy complejos. Aquí vamos a usar una expresión regular más o menos sencilla: [^@]+@[^@]+\.[a-zA-Z]{2,}. Significa lo siguiente, un email válido está compuesto de:

[^@]+ cualquier caracter que no sea @ una o más veces seguido de

@ una @ seguido de

[^@]+ cualquier caracter que no sea @ una o más veces seguido de

\. un punto seguido de

[a-zA-Z]{2,} dos o más letras minúsculas o mayúsculas

Un ejemplo en código java de esta expresión regular

String emailRegexp = "[^@]+@[^@]+\\.[a-zA-Z]{2,}";

// Lo siguiente devuelve true

System.out.println(Pattern.matches(emailRegexp, "a@b.com"));

System.out.println(Pattern.matches(emailRegexp, "+++@+++.com"));

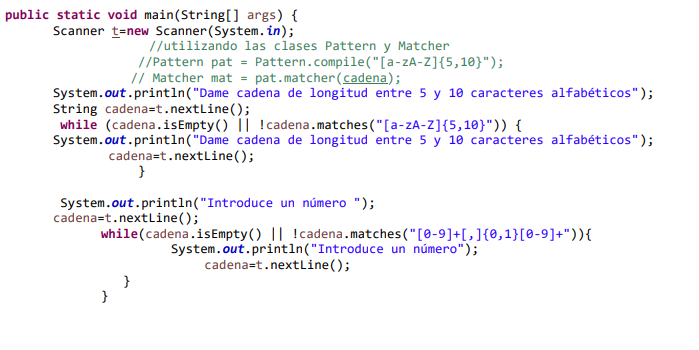
// Lo siguiente devuelve false

System.out.println(Pattern.matches(emailRegexp, "@b.com")); // Falta el nombre

System.out.println(Pattern.matches(emailRegexp, "a@b.c")); // El dominio final debe tener al menos dos letras.

**Ejemplo con el método matches de String**

Podemos comprobar si una cadena de caracteres cumple con un patrón usando el método matches de la clase String. Este método recibe como parámetro la expresión regular.



**Ejemplo con el método split de String**

Este método divide el String en cadenas según la expresión regular que recibe. La expresión regular no forma parte del array resultante.

Ejemplo 1:

String str = "blanco-rojo:amarillo.verde\_azul";

String [] cadenas = str.split("[-:.\_]");

for(int i = 0; i<cadenas.length; i++){

System.out.println(cadenas[i]);

}

Muestra por pantalla:

blanco

rojo

amarillo

verde

azul

Ejemplo 2:

String str = "esto es un ejemplo de como funciona split";

String [] cadenas = str.split("(e[s|m])|(pl)");

for(int i = 0; i<cadenas.length; i++){

System.out.println(cadenas[i]);

}

Salida:

to

un ej

o de como funciona s

it