INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Práctica 2 - Expresión Regular

Unidad de aprendizaje: Teoría computacional

Grupo: 2CM4

Alumno(a):
Nicolás Sayago Abigail

Profesor(a):
Sanchez García Luz María

8 de Marzo de 2018

Índice

1 Introducción **2**
1.1 Expresión regular 2
1.2 Expresión regular en programación 2

2 Planteamiento del problema **3**

3 Diseño de la solución **6**

4 Implementación de la solución **6**

5 Funcionamiento **8**

6 Conclusiones **10**

Referencias **10**

1. Introducción

En la siguiente práctica se implementara una expresión regular en el lenguaje de programación de JAVA. La expresión regular sera de un número de boleta que pertenece a un Sistema Administrativo y Escolar de Educación Básica (SAEEB), que está siendo desarrollado.

A lo largo del documento usaremos la palabra **Expresión Regular**, por lo cual tenemos que definir ¿Qué es una expresión regular?.

1.1. Expresión regular

Una **expresión regular** es una forma abreviada de representar cadenas de caracteres que se ajustan a un determinado patrón. Al conjunto de cadenas representado por la expresión *r* se lo llama *lenguaje generado por la expresión regular r* y se escribe *L(r)*. Una expresión regular se define sobre un alfabeto y es una cadena formada por caracteres de dicho alfabeto y por una serie de operadores también llamadas metacaracteres. Una expresión regular define un patrón de búsqueda para cadenas de caracteres. Los podemos utilizar para comprobar si una cadena contiene o coincide con el patrón. El contenido de caracteres puede coincidir con el patron 0, 1 o más veces.

1.2. Expresión regular en programación

Los usos de las expresiones regulares son muchos, por ejemplo, en algunos lenguajes de programación las expresiones regulares sirven para detectar patrones en distintas cadenas de texto. Para los programadores, esas expresiones regulares facilitan su labor en un solo paso. Un ejemplo muy sencillo es la comprobación vía expresión regular que los datos rellenados por los usuarios en un formulario son correctos o no. Alguien puede utilizar un breve formulario para recabar el nombre, los apellidos y el teléfono de contacto de sus clientes, y que algunos de ellos no completen bien este ultimo campo. La forma rápida de comprobarlo es a través de una expresión regular.

Algunos otros ejemplos de uso de expresiones regulares pueden ser:

→ Comprobar que la fecha leída cumple el patrón dd/mm/aaaa

→ Comprobar que un NIF esta formado por 8 cifras, un guión y una letra

→ Comprobar que una dirección de correo es válida

→ Comprobar que una contraseña cumple unas determinadas condiciones.

→ Comprobar que una URL es válida.

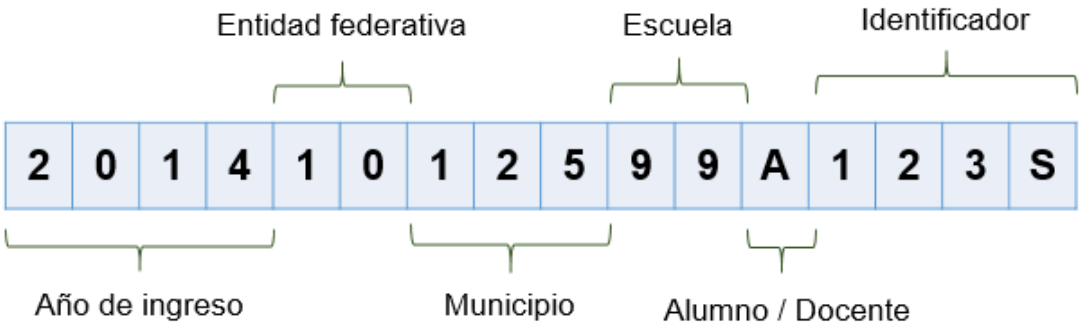
2. Planteamiento del problema

De forma general la expresión regular elegida define un número de identificación para un alumno que pertence a una escuela secundaria publica, debido a esto su numero de boleta estará dado por:

- Año de ingreso.
- Número de entidad federativa.
- Número de municipio en esa entidad federativa.
- Número de escuela en ese municipio.
- Puede ser Alumno o Docente, siendo identificados con A y D respectivamente.
- Número de usuario dado por 3 números y una letra.

Para esta práctica delimitamos el número de boleta para un alumno del Estado de México. Así que diseñamos la solución para implementar la expresión regular definida por lo siguiente:

- Niños que hayan ingresado en los años 2014-2017.
- Que la escuela pertenezca a la entidad federativa Estado de México.
- Que sea de uno de los 125 municipios de esa entidad federativa.
- Pertenezca a una de las escuelas en ese municipio (0-99)
- Puede ser Alumno o Docente, siendo identificados con A y D respectivamente.
- Número de usuario dado por 3 números y una letra. (Identificador).
- Tiene un total de 16 caracteres.



Algunos ejemplos son:

Cadenas validas son:

- 20141012599A123A
- 20141012599D144A
- 20151000198A126R
- 20171003474A112D

Cadenas no validas:

- 20181312599S123A12
- 20171013515D014S
- 20141012599A13S<
- 20131012610D123A

A continuación se ve la tabla que permite determinar el número de estado que tendrá asignado, así como también el número de municipios que se permite por entidad Federativa. Observamos unicamente la del Estado de México, aunque también cabe señalar que si se quiere implementar una expresión regular con cualquier otro estado, se podría de manera similar a la que vamos a hacer.

No.	**Entidad Federativa**	**Municipio**
1	Aguascalientes	1
2	Baja California	5
3	Baja California Sur	5
4	Campeche	11
5	Chiapas	122
6	Chihuahua	67
7	Coahuila	38
8	Colima	10
9	Durango	39
10	Estado de México	125
11	Guanajuato	46
12	Guerrero	81
13	Hidalgo	84
14	Jalisco	125
15	Michoacán	113
16	Morelos	33
17	Nayarit	20
18	Nuevo León	51
19	Oaxaca	570
20	Puebla	217
21	Querétaro	18
22	Quintana Roo	11
23	San Luis Potosí	58
24	Sinaloa	18
25	Sonora	72
26	Tabasco	17
27	Tamaulipas	43
28	Tlaxcala	60
29	Veracruz	212
30	Yucatán	106
31	Zacatecas	58
32	Ciudad de México	16

3. Diseño de la solución

A continuación mostraré el diseño que se tiene para implementar la solución, dados los requerimientos expresados antes. La expresión regular es:

$$20\,[4-7]10([0-1][0-9][0-9])([0-9][1-9])[AD][0-9][0-9][0-9][A-Z]$$

4. Implementación de la solución

Primero muestro una tabla con los métodos con los que ya cuenta el lenguaje de programación y que fueron utilizados en la realización de la implementación de la práctica.

Método	**Función**
Pattern	El método pattern devuelve la expresión regular que hemos compilado.
Pattern.matcher	Nos permite realizar operaciones sobre la secuencia de caracteres que queremos validar o la secuencia de caracteres en la que queremos buscar.
cadena.charAr(indice)	Devuelve el caracter que encuentra en una posición especifica.

De forma general el programa esta estructurado de tal forma que al dar click al boton **OK** haga algunas validaciones como son la longitud y el número de municipio, si pasa ese filtro, entonces manda al método **ValidarExpresion** la cadena a validar totalmente.

El método **ValidarExpresion** contiene una cadena que es prácticamente la expresión regular del número de alumno después se usa el método **Pattern.compile** que *compila* nuestra expresión regular. Posteriormente con el método **Matcher** se comprueban las cadenas contra el patrón indicado.

Cabe señalar que con el método **ValidarExpresion** era más que suficiente, sin embargo podemos evitar que se haga todo un desgaste de memoria haciendo ciertos ***filtros*** al recibir la cadena como la de la longitud

- IMPLEMENTACIÓN AL HACER CLIK AL BOTÓN

```
public void actionPerformed(ActionEvent e)
{
 JButton b = (JButton)e.getSource(); // Recibimos un boton
 String aux="";
 Boolean a; // Bandera
 int aux2, i;
 if(b == Enviar) // Si el boton es el de enviar
 {
  a = true;
  // Obtenemos la cadena a validar
  String cad = Cadena.getText();
  // Si la cadena es de longitud 16
  if(cad.length() == 16)
  {
   // Obtenemos el n mero de municipio
   for(i=6; i<9; i++)
    aux = aux + cad.charAt(i);
   aux2 = Integer.parseInt(aux);
   if (aux2 < 126) // Si el n mero de municipio esta en el rango
   {
    // Finalmente mandamos la cadena
    if(a = validarExpresion(cad))
     Valida.setText("CADENA VALIDA");
    else // Si no cumple con la E.R
     Valida.setText("CADENA NO VALIDA");
   }
   else // Si el municipio no esta en el rango
    Valida.setText("CADENA NO VALIDA");
  }
  else // Si la cadena no es de longitud 16
   Valida.setText("CADENA NO VALIDA");
 }
}
```

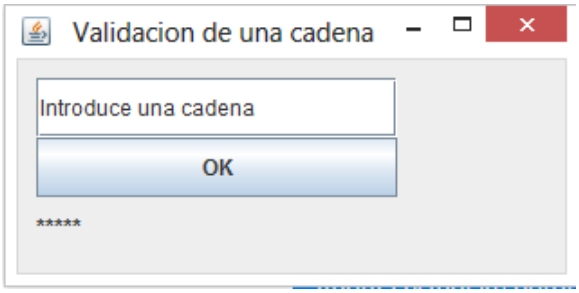
- IMPLEMENTACIÓN DEL MÉTODO VALIDAR EXPRESIÓN

```
private boolean validarExpresion(String CAD)
{   // Generamos la expresi n regular
 String regex = "([2]{1}[0]{1})(1[4-7])" +
     "1" + "0" +
     "([0-1][0-9][0-9])" +
     "([0-9][1-9])" +
     "[AD]{1}" +
     "[0-9][0-9][0-9][A-Z]{1}" ;
 // Se compila la expresi n regular
 Pattern patron = Pattern.compile(regex);
 // Si la cadena no coincide con la expresi n regular
 if(!patron.matcher(CAD).matches())
  return false;
 else
  return true;
}
```

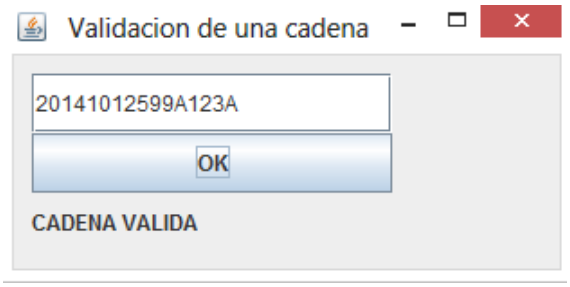
5. Funcionamiento

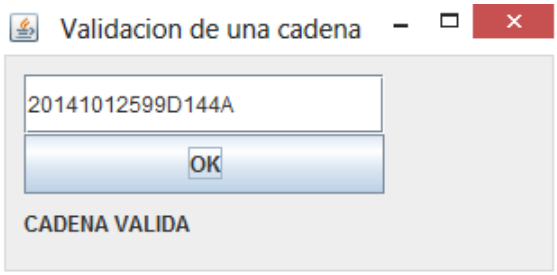
Primero que nada, mostramos la interfaz inicial. Observamos que el usuario tiene la oportunidad de ingresar una cadena. Al dar click en el botón OK, se mostrará **CADENA VALIDA** si la cadena que se ha ingresado cumple con el formato del número de boleta en caso contrario, se mostrará **CADENA NO VALIDA**.

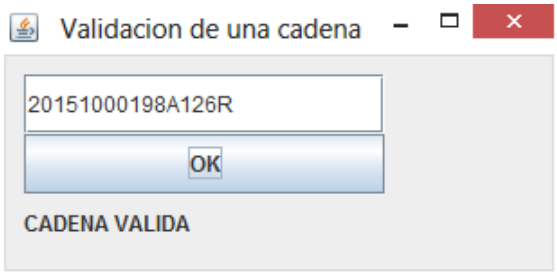
La pantalla inicial es la siguiente:

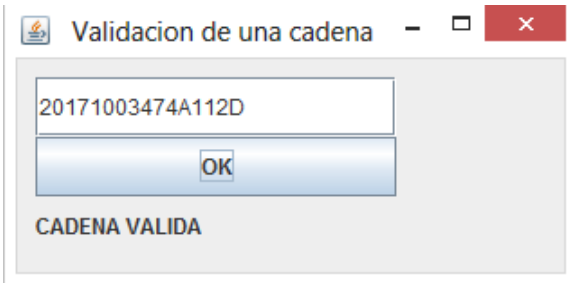


A continuación muestro ejemplos de **cadenas validas**:

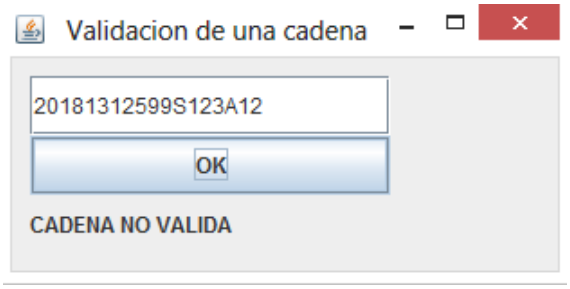


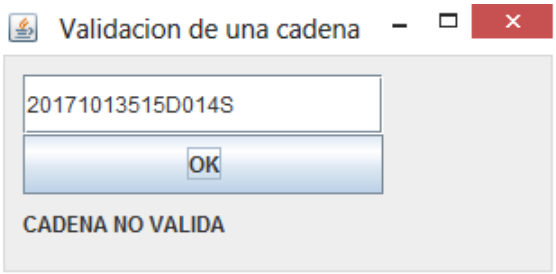


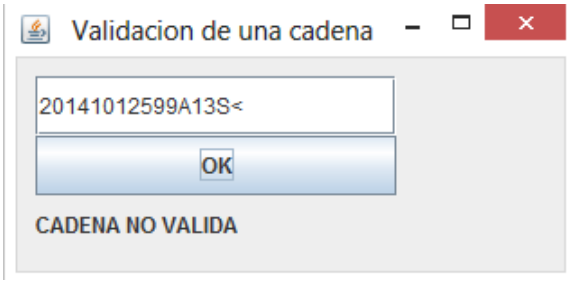


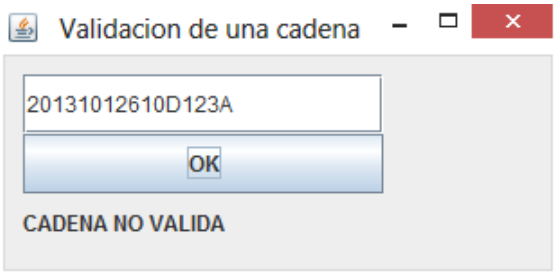


Ahora ejemplos de **cadenas no validas**:









6. Conclusiones

Al terminar está práctica pude observar la importancia de las expresiones regulares y la forma en la que son ocupadas para validar cadenas, especialmente porque las condiciones las especifique yo puesto que dichos número sde boleta serán ocupados en un sistema que será implementado en la clase de Análisis de Diseño Orientado a Objetos, así que usare lo desarrollado en está práctica para el sistema y las validaciones.

Referencias

[1] E. A. Martinez, *Alfabetos, símbolos y cadenas*. [Online]. Available: http://eafranco.com/docencia/teoriacomputacional/files/03/Clase$_1$5.*pdf*