UNNOBA
UNIVERSIDAD NACIONAL
NOROESTE ▪ BUENOS AIRES

Tarea Expresiones Regulares

Asignatura: Ciencias de la Computación I

Figueroa Facundo

Introducción

Se propone actividad de lenguajes regulares sobre un lenguaje de programación en específico. En este caso, la actividad se basará sobre el lenguaje **Java**.

En Java, se utilizan expresiones regulares, también conocidas como lenguajes regulares, para buscar y manipular patrones de texto. Las expresiones regulares se basan en un conjunto de caracteres especiales y sintaxis para crear patrones que representan conjuntos de caracteres, palabras, números y otros elementos de texto.

La clase java.util.regex.Pattern es la principal clase utilizada para trabajar con expresiones regulares en Java. Esta clase se utiliza para compilar una expresión regular en un objeto Pattern, que se puede utilizar para buscar y manipular patrones en una cadena de texto.

Además, Java también proporciona la clase java.util.regex.Matcher, que se utiliza para aplicar un patrón compilado a una cadena de texto y encontrar las ocurrencias del patrón en la cadena. La clase Matcher proporciona métodos para buscar todas las ocurrencias del patrón en la cadena, reemplazar el patrón con otro texto y extraer partes de la cadena que coincidan con el patrón.

Java también proporciona un conjunto de caracteres especiales y sintaxis que se pueden utilizar para construir patrones de expresiones regulares, como caracteres especiales de coincidencia de patrones, clases de caracteres, cuantificadores, caracteres de escape y grupos de captura.

Las expresiones regulares son una herramienta muy útil y poderosa en Java para buscar y manipular patrones de texto. Si se utilizan correctamente, pueden ayudar a ahorrar tiempo y esfuerzo al procesar y manipular grandes cantidades de datos de texto.

Marco Teórico

Java es un lenguaje de programación y una plataforma informática que fue comercializada por primera vez en 1995 por Sun Microsystems.23

El lenguaje de programación Java fue desarrollado originalmente por James Gosling, de Sun Microsystems (constituida en 1983 y posteriormente adquirida el 27 de enero de 2010 por la compañía Oracle), y publicado en 1995 como un componente fundamental de la plataforma Java de Sun Microsystems. Su sintaxis deriva en gran medida de C y C++, pero tiene menos utilidades de bajo nivel que cualquiera de ellas. Las aplicaciones de Java son compiladas a bytecode (clase Java), que puede ejecutarse en cualquier máquina virtual Java (JVM) sin importar la arquitectura de la computadora subyacente.

La compañía Sun desarrolló la implementación de referencia original para los compiladores de Java, máquinas virtuales y librerías de clases en 1991, y las publicó por primera vez en 1995. A partir de mayo de 2007, en cumplimiento de las especificaciones del Proceso de la Comunidad Java, Sun volvió a licenciar la mayoría de sus tecnologías de Java bajo la Licencia Pública General de GNU. Otros han desarrollado también implementaciones alternas a estas tecnologías de Sun, tales como el Compilador de Java de GNU y el GNU Classpath. [1]

En base a expresiones regulares,todo comienza en 1943, cuando Warren S. McCulloch y Walter Pitts comenzaron a desarrollar modelos que describen cómo funciona el sistema nervioso humano. Su investigación se centró en tratar de comprender cómo el cerebro podría producir patrones complejos utilizando células simples que están unidas. En 1956, el matemático Stephen Kleen describió los modelos neuronales de McCulloch-Pitts con una notación de álgebra que él escribió "expresiones regulares". Influenciado por la noción de Kleen, en 1968, el matemático y pionero de Unix, Ken Thompson, implementó la idea de expresiones regulares dentro del editor de texto, 'ed'. Su objetivo era que los usuarios de ed

pudieran realizar coincidencias avanzadas de patrones en archivos de texto .ed , pronto evolucionó para tener la funcionalidad de búsqueda basada en expresiones regulares, aquí es cuando las expresiones regulares ingresaron al mundo de la informática. [2]

Desarrollo

- Creación de expresiones regulares para Direcciones de correo electrónico de los dominios Gmail y Hotmail

Para Gmail :

- Expresion Regular : **^[a-zA-Z0-9._%+-]+@gmail\\.com$**
- Descripción:

  ^ : indica el inicio de la cadena

  [a-zA-Z0-9._%+-]+ : indica que la dirección de correo electrónico debe comenzar con uno o más caracteres alfanuméricos, seguidos de uno o más caracteres de los siguientes: punto, guión bajo, porcentaje o signo más y menos.

  @gmail\\.com: indica que la dirección de correo electrónico debe tener el dominio "gmail.com".

  $: indica el final de la cadena.

Para Hotmail

- Expresión Regular : **^[a-zA-Z0-9._%+-]+@(hotmail|outlook)\\.com$**
- Descripción**:**

  ^ : indica el inicio de la cadena

  [a-zA-Z0-9._%+-]+ : indica que la dirección de correo electrónico debe comenzar con uno o más caracteres alfanuméricos, seguidos de uno o más caracteres de los siguientes: punto, guión bajo, porcentaje o signo más y menos.

  @(hotmail|outlook)\\.com : indica que la dirección de correo electrónico debe tener el dominio "hotmail.com" o "outlook.com".

  $: indica el final de la cadena.

- Creación de expresiones regulares para Direcciones postales que referencian números de puerta o intersección de calles

  Teniendo en cuenta las abreviaturas:

  carretera – Rd.

  calle: St.

  avenida – Ave.

  boulevard - Blvd.

  carril - Ln.

  Expresión regular :

^[0-9]+\\s+(\\p{L}+\\s+){1,2}(St\\.|Ave\\.|Rd\\.|Blvd\\.|Ln\\.)\\s*(\\p{L}+\\s+){1,3}(\\#\\s*[0-9]+)?$

  Descripción:

  ^: Representa el inicio de la cadena.

  [0-9]+: Representa uno o más dígitos del 0 al 9, que indican el número de calle de la dirección.

  \\s+: Representa uno o más espacios en blanco que separan el número de calle del nombre de calle.

  (\\p{L}+\\s+){1,2}: Representa el nombre de calle de la dirección. (\\p{L}+\\s+) indica una o más letras (incluso acentuadas) seguidas por uno o más espacios en blanco. {1,2} indica que el nombre de calle puede contener una o dos palabras

  (St\\.|Ave\\.|Rd\\.|Blvd\\.|Ln\\.): Representa el tipo de calle de la dirección. Esta parte permite que la dirección tenga uno de los tipos de calle válidos, como "St." o "Ave.".

  \\s*: Representa cero o más espacios en blanco que separan el tipo de calle del resto de la dirección.

  (\\p{L}+\\s+){1,3}: Representa cualquier información adicional de la dirección, como un nombre de edificio o una suite. Esta parte permite que la dirección contenga de una a tres palabras.

  (\\#\\s*[0-9]+)?: Representa un número de apartamento o suite opcional. \\# indica que el número de apartamento o suite está precedido por el símbolo #. \\s* representa cero o más espacios en blanco que separan el número de apartamento o suite del resto de la dirección. [0-9]+ : Representa uno o más dígitos que indican el número de apartamento o suite.

  $: Representa el final de la cadena.

Aplicación desarrollada en Java

- Evaluación de direcciones de correo electrónico**.**

```
/*RegEx*/
private static final String EMAIL_PATTERN
        = "^[a-zA-Z0-9._%+-]+@gmail\\.com$";
private static final Pattern patternGmail = Pattern.compile(EMAIL_PATTERN);

private static final String HOTMAIL_PATTERN
        = "^[a-zA-Z0-9._%+-]+@(hotmail|outlook)\\\\.com$";
private static final Pattern patterHotmail = Pattern.compile(HOTMAIL_PATTERN);
```

Fig.1. Implementación de atributos con la expresión regular.

```
/*Métod for evaluate email @gmail*/
public static boolean validateGmail(final String email) {
    Matcher matcher = patternGmail.matcher(email);
    return matcher.matches();
}

/*Métod for evaluate email @hotmail*/
public static boolean validateHotmail(final String email) {
    Matcher matcher = patterHotmail.matcher(email);
    return matcher.matches();
}
```

Fig.2. Implementación de métodos que evalúan si es válido o no lo escrito por el usuario en base a su expresión regular asociada.

```
run:
Ingrese email para direcciones gmail:
facu@gmail.com
Email Valido
Ingrese email - para direcciones hotmail :
facu@hotmai.com
Email Invalido
```

Fig.3. Resultado para direcciones válidas.

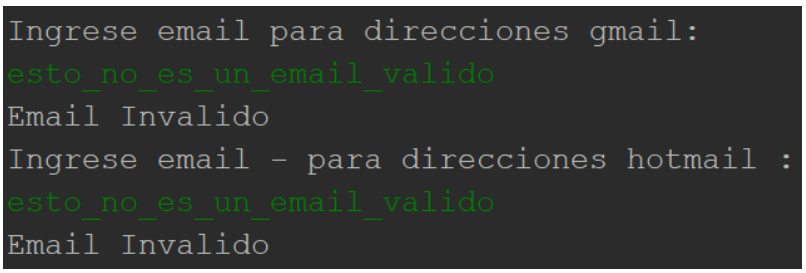

Fig.4. Resultado para direcciones inválidas.

- Evaluación de direcciones postales que referencian números de puerta o intersección de calles.

```
private static final String ADDRESS_PATTERN =
    "^[0-9]+\\s+(\\p{L}+\\s+){1,2}(St\\.|Ave\\.|Rd\\.|Blvd\\.|Ln\\.)\\s*(\\p{L}+\\s+){1,3}(\\#\\s*[0-9]+)?$";

private static final Pattern pattern = Pattern.compile(ADDRESS_PATTERN);

public static boolean validateAdress(final String address) {
    Matcher matcher = pattern.matcher(address);
    return matcher.matches();
}
```

Fig.5. Implementación de atributos con la expresión regular y método para validar la misma.

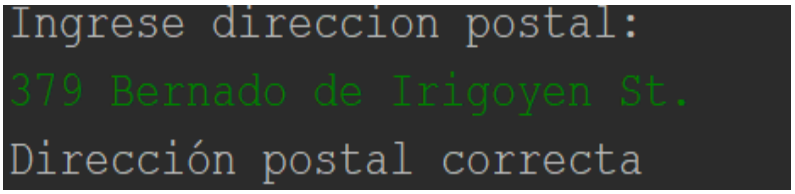

Fig.6. Resultado para direcciones válidas.

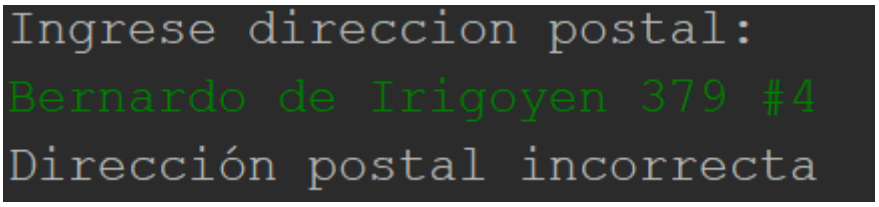

Fig.7. Resultado para direcciones inválidas.

Conclusión

El desarrollo de este trabajo me permitió el conocimiento de la aplicabilidad de expresiones regulares en el lenguaje de programación Java. De está manera, se logra aprender su uso en caso de ser necesario implementar una expresión regular para diferentes validaciones cotidianas.

Bibliografía

[1] https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)
[2] https://www.scantopdf.com/es/blog/the-history-of-regex/