**Trabajo práctico nro. 12**

|  |  |  |
| --- | --- | --- |
| Logo Departamento 2.jpg | **Asignatura: Laboratorio I** | |
|  | |
| **Cursado:**Primer Trimestre | **Horassemanales**: |
|  | **Horas semestrales:**  *Cantidad estimada de horas semestrales/anuales.* |
| **Carrera**: *Tecnicatura Universitaria en Programación* | **Nivel (Año):** |
| **Ciclo Lectivo: 2023** |

1. **Tema:**

**Expresiones Regulares**

1. **Enunciados:**

**Empezaremos por unos ejercicios básicos de programas Java con estructura secuencial, es decir, en estos programas no hay instrucciones condicionales ni repetitivas. En la mayoría de ellos las operaciones a realizar son: lectura de datos por teclado, realizar alguna operación con esos datos y mostrar resultados por pantalla.**

**Recomendaciones: Leer la teoría, investigar, consultar si existen dudas y realizar el trabajo en clases. La entrega de este trabajo practico es individual.**

1. **Investigación de Expresiones Regulares**:
   * Investiga y comprende los conceptos básicos de las expresiones regulares, incluyendo caracteres especiales, cuantificadores y grupos.
   * Explora ejemplos de patrones comunes de expresiones regulares utilizados en la validación de datos.

Concepto básico de las expresiones regulares son:

1- Caracteres Literales: Los caracteres en una expresión regular que coinciden exactamente con ellos mismos se denominan "literales". Por ejemplo, el patrón a coincidirá con la letra "a" en una cadena.

2- Caracteres Especiales: Las expresiones regulares utilizan ciertos caracteres especiales para representar clases de caracteres o cuantificadores. Algunos de los caracteres especiales comunes son:

. (punto): Coincide con cualquier carácter excepto un salto de línea.

\*: Coincide con 0 o más repeticiones del carácter o grupo anterior.

+: Coincide con 1 o más repeticiones del carácter o grupo anterior.

?: Coincide con 0 o 1 repetición del carácter o grupo anterior.

| (barra vertical): Se utiliza para alternar entre varias opciones.

[] (corchetes): Define un conjunto de caracteres que pueden coincidir. Por ejemplo, [aeiou] coincidirá con cualquier vocal.

3- Cuantificadores: Los cuantificadores se utilizan para controlar la cantidad de repeticiones de un carácter o grupo. Algunos ejemplos comunes son:

{n}: Coincide exactamente con n repeticiones del carácter o grupo anterior.

{n,}: Coincide con al menos n repeticiones del carácter o grupo anterior.

{n, m}: Coincide con entre n y m repeticiones del carácter o grupo anterior.

4- Grupos: Los grupos se utilizan para agrupar elementos juntos y aplicar cuantificadores o operadores a ese grupo. Los grupos se crean utilizando paréntesis. Por ejemplo, (abc)+ coincidirá con una o más repeticiones de "abc".

Los patrones comunes de expresiones regulares utilizados en la validación de datos son:

Correo Electrónico:

Patrón: ^\w+([\.-]?\w+)\*@\w+([\.-]?\w+)\*(\.\w{2,3})+$

Descripción: Valida direcciones de correo electrónico con un formato típico.

URL:

Patrón: ^(http|https)://\S+

Descripción: Valida URLs que comienzan con "http://" o "https://".

Número Entero Positivo o Negativo:

Patrón: ^-?\d+

Descripción: Valida números enteros positivos o negativos.

Número Decimal Positivo o Negativo:

Patrón: ^-?\d+(\.\d+)?$

Descripción: Valida números decimales positivos o negativos.

Código Postal (EE. UU.):

Patrón: ^\d{5}(-\d{4})?$

Descripción: Valida códigos postales de EE. UU. estándar.

Contraseña Segura:

Patrón: ^(?=.\*[A-Z])(?=.\*[a-z])(?=.\*\d)(?=.\*[@$!%\*?&]).{8,}$

Descripción: Valida contraseñas que deben cumplir con ciertos criterios de seguridad.

Nombre de Usuario:

Patrón: ^[A-Za-z0-9\_-]{3,16}$

Descripción: Valida nombres de usuario alfanuméricos con guiones bajos y guiones, con una longitud de 3 a 16 caracteres.

1. **Creación de un Conjunto de Datos**:
   * Crea un conjunto de datos ficticio que contenga información variada, como direcciones de correo electrónico, números de teléfono, fechas, códigos postales, etc.

Nombre: Juan Pérez

Edad: 30

Correo Electrónico: juan.perez@example.com

Teléfono: (555) 123-4567

Fecha de Nacimiento: 1993-05-15

Dirección: 123 Calle Principal, Ciudad, Estado, 12345

Número de Seguro Social: 123-45-6789

Código Postal: 54321

URL del Sitio Web: http://www.ejemplo.com

Número de Tarjeta de Crédito: 1234-5678-9012-3456

1. **Validación de Datos**:
   * Diseña un programa en un lenguaje de programación (Java) que utilice expresiones regulares para validar y verificar los datos en tu conjunto de datos. Por ejemplo, puedes verificar si una dirección de correo electrónico es válida o si un número de teléfono sigue un formato específico.
   * Validar entrada de datos en un arraylist y aplicarlo.

import java.util.ArrayList;

import java.util.regex.Pattern;

import java.util.regex.Matcher;

public class DataValidation {

public static void main(String[] args) {

ArrayList<String> data = new ArrayList<String>();

// Agrega datos al ArrayList

data.add("juan.perez@example.com");

data.add("(555) 123-4567");

data.add("1993-05-15");

data.add("54321");

data.add("http://www.ejemplo.com");

data.add("1234-5678-9012-3456");

// Expresiones regulares para validación

String emailPattern = "^[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\\.[A-Za-z]{2,}$";

String phonePattern = "^\\(\\d{3}\\) \\d{3}-\\d{4}$";

Pattern emailRegex = Pattern.compile(emailPattern);

Pattern phoneRegex = Pattern.compile(phonePattern);

for (String item : data) {

if (emailRegex.matcher(item).matches()) {

System.out.println("Correo electrónico válido: " + item);

} else if (phoneRegex.matcher(item).matches()) {

System.out.println("Número de teléfono válido: " + item);

} else {

System.out.println("Dato no válido: " + item);

}

}

}

}

1. **Informe de Investigación**:
   * Prepara un informe que desarrolle el tema sobre las expresiones regulares y cómo se aplicaron en la validación de datos.
   * Incluye ejemplos de patrones de expresiones regulares utilizados en tu programa y cómo ayudaron en la validación de los datos.
   * Comprensión de los conceptos de expresiones regulares.
   * Diseño y desarrollo del programa de validación de datos.
   * Claridad y organización del informe de investigación.

# Informe sobre Expresiones Regulares en la Validación de Datos

Introducción

Las expresiones regulares, comúnmente conocidas como "regex" o "regexp", son herramientas poderosas para trabajar con cadenas de texto. Se utilizan para definir patrones de caracteres que permiten buscar, validar y manipular datos de manera más eficiente y precisa. En este informe, exploraremos cómo funcionan las expresiones regulares y cómo se aplicaron en un programa de validación de datos en el lenguaje de programación Java.

Comprensión de los Conceptos de Expresiones Regulares

Caracteres Literales

Los caracteres literales en expresiones regulares coinciden exactamente con ellos mismos. Por ejemplo, la expresión regular `abc` coincidirá solo con la cadena "abc".

Caracteres Especiales

Las expresiones regulares utilizan caracteres especiales para representar clases de caracteres o cuantificadores. Algunos de los caracteres especiales comunes son `.`, `\*`, `+`, `?`, `|`, `[...]`, entre otros.

Cuantificadores

Los cuantificadores se utilizan para controlar la cantidad de repeticiones de un carácter o grupo. Algunos ejemplos son `{n}`, `{n,}`, `{n, m}`.

Grupos

Los grupos se utilizan para agrupar elementos juntos y aplicar cuantificadores o operadores a ese grupo. Se crean utilizando paréntesis.

Diseño y Desarrollo del Programa de Validación de Datos

Para demostrar cómo funcionan las expresiones regulares en la validación de datos, se desarrolló un programa en Java. El programa toma una serie de datos variados, incluyendo direcciones de correo electrónico, números de teléfono, fechas, códigos postales y otros tipos de datos, y verifica su validez utilizando expresiones regulares.

Ejemplos de Patrones de Expresiones Regulares Utilizados

En el programa de validación de datos, se utilizaron los siguientes patrones de expresiones regulares:

1. Correo Electrónico:

- Patrón: `^\w+([\.-]?\w+)\*@\w+([\.-]?\w+)\*(\.\w{2,3})+$`

- Descripción: Valida direcciones de correo electrónico con un formato típico.

2. Número de Teléfono:

- Patrón: `^\(\d{3}\) \d{3}-\d{4}$`

- Descripción: Valida números de teléfono con el formato "(###) ###-####".

Proceso de Validación

El programa recorre la lista de datos y aplica las expresiones regulares definidas para verificar su validez. Si un dato coincide con una expresión regular, se considera válido; de lo contrario, se marca como no válido.

Resultados y Conclusiones

El programa de validación de datos basado en expresiones regulares es una herramienta eficaz para verificar la validez de datos variados. Las expresiones regulares permiten definir patrones precisos que ayudan a garantizar que los datos cumplan con los requisitos especificados.

Este enfoque de validación es ampliamente utilizado en aplicaciones web y sistemas de gestión de datos para garantizar la integridad y calidad de los datos. La flexibilidad de las expresiones regulares permite adaptar las validaciones a requisitos específicos, como direcciones de correo electrónico, números de teléfono, fechas y otros formatos.

En resumen, las expresiones regulares son una herramienta esencial en la validación de datos y se aplican con éxito para garantizar que los datos cumplan con los criterios específicos de formato y contenido.

1. **Ejercicios**:

* Comprobar si el String *cadena* no acaba con un dígito.

public class Main {

public static void main(String[] args) {

String cadena = "Hola123";

boolean noTerminaConDigito = !cadena.matches(".\*\\d$");

if (noTerminaConDigito) {

System.out.println("La cadena no acaba con un dígito.");

} else {

System.out.println("La cadena acaba con un dígito.");

}

}

}

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

public class Main {

public static void main(String[] args) {

String cadena = "123Hola";

boolean noEmpiezaConDigito = !cadena.matches("^\\d.\*");

if (noEmpiezaConDigito) {

System.out.println("La cadena no empieza con un dígito.");

} else {

System.out.println("La cadena empieza con un dígito.");

}

}

}

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

public class Main {

public static void main(String[] args) {

String cadena = "AbCdEfGhI"; // Cambia esta cadena según tus necesidades

boolean cumpleRequisitos = cadena.matches("^[A-Za-z]{5,10}$");

if (cumpleRequisitos) {

System.out.println("La cadena cumple con los requisitos.");

} else {

System.out.println("La cadena no cumple con los requisitos.");

}

}

}

* Validar el ingreso de un DNI (xx-xxx-xxx).

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Ingresa tu número de DNI (en formato xx-xxx-xxx): ");

String dni = scanner.nextLine();

scanner.close();

boolean esDniValido = dni.matches("^\\d{2}-\\d{3}-\\d{3}$");

if (esDniValido) {

System.out.println("El DNI es válido.");

} else {

System.out.println("El DNI no es válido o no cumple con el formato xx-xxx-xxx.");

}

}

}

* Guardar en un arraylist una serie de mails, pero antes de ser guardados deben cumplir con las condiciones de las expresiones regulares.

import java.util.ArrayList;

import java.util.Scanner;

import java.util.regex.Pattern;

import java.util.regex.Matcher;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

ArrayList<String> correosValidos = new ArrayList<String>();

String emailPattern = "^[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\\.[A-Za-z]{2,}$";

Pattern emailRegex = Pattern.compile(emailPattern);

System.out.println("Ingresa una serie de correos electrónicos. Presiona Enter después de cada uno. Ingresa 'fin' para finalizar:");

while (true) {

String input = scanner.nextLine();

if (input.equalsIgnoreCase("fin")) {

break;

}

Matcher matcher = emailRegex.matcher(input);

if (matcher.matches()) {

correosValidos.add(input);

System.out.println("Correo válido. Agregado a la lista.");

} else {

System.out.println("Correo no válido. No agregado a la lista.");

}

}

scanner.close();

System.out.println("Correos electrónicos válidos:");

for (String correo : correosValidos) {

System.out.println(correo);

}

}

}

* Comprobar si el String *cadena* contiene “abc”.

public class Main {

public static void main(String[] args) {

String cadena = "Esta es una cadena que contiene abc en algún lugar.";

if (cadena.contains("abc")) {

System.out.println("La cadena contiene 'abc'.");

} else {

System.out.println("La cadena no contiene 'abc'.");

}

}

}