**Trabajo práctico nro. 12**

|  |  |  |
| --- | --- | --- |
| Logo Departamento 2.jpg | **Asignatura: Laboratorio I** | |
|  | |
| **Cursado:**Primer Trimestre | **Horas** **semanales**: |
|  | **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.
2. **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.
3. **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.
4. **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.
5. **Ejercicios**:

* Comprobar si el String *cadena* no acaba con un dígito.
* Comprobar si el String *cadena* no empieza por un dígito.
* Comprobar si el String *cadena* está formado por un mínimo de 5 letras mayúsculas o minúsculas y un máximo de 10.
* Validar el ingreso de un DNI (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.
* Comprobar si el String *cadena* contiene “abc”.

**Introducción**

Las expresiones regulares, también conocidas como regex o regexp, son patrones de búsqueda utilizados para encontrar secuencias de caracteres en texto. Se utilizan en programación para validar, buscar y manipular datos textuales de manera eficiente. En este informe, exploraremos los conceptos básicos de las expresiones regulares y cómo se aplicaron en la validación de datos en el contexto de un programa Java.

**Conceptos Básicos de Expresiones Regulares**

Las expresiones regulares se componen de caracteres literales y caracteres especiales que representan reglas de búsqueda. Algunos conceptos clave incluyen:

Caracteres literales: Los caracteres que se buscan tal cual, como 'a' o '123'.

Metacaracteres: Caracteres especiales que tienen significados especiales, como '.', '^', '$', '?', '\*', '+', etc.

Cuantificadores: Especifican cuántas veces se espera que aparezca un elemento, como '{3}' para buscar exactamente tres repeticiones o '\*' para buscar cero o más repeticiones.

Grupos y alternativas: Se pueden agrupar elementos y utilizar el operador '|' para alternar entre diferentes patrones.

Diseño y Desarrollo del Programa de Validación de Datos

Para demostrar el uso de expresiones regulares en la validación de datos, hemos diseñado un programa Java que valida diferentes tipos de datos utilizando patrones regulares. Aquí hay un resumen de cómo se aplicaron las expresiones regulares en nuestro programa:

**Ejercicio 1: Comprobar si el String no acaba con un dígito**

Para verificar si una cadena no acaba con un dígito, utilizamos la expresión regular .\*[^0-9]$. Esto significa que buscamos cualquier cantidad de caracteres seguidos de un carácter que no sea un dígito al final de la cadena.

import java.util.regex.\*;

public class Ejercicio1 {

public static void main(String[] args) {

String cadena = "Ejemplo123";

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

if (noTerminaConDigito) {

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

} else {

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

}

}

}

**Ejercicio 2: Comprobar si el String no empieza por un dígito**

Para comprobar si una cadena no empieza con un dígito, utilizamos la expresión regular ^[^0-9].\*. Esto significa que buscamos un carácter que no sea un dígito al principio de la cadena, seguido de cualquier cantidad de caracteres.

import java.util.regex.\*;

public class Ejercicio2 {

public static void main(String[] args) {

String cadena = "123Ejemplo";

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

if (noEmpiezaConDigito) {

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

} else {

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

}

}

}

**Ejercicio 3: Comprobar si el String está formado por un mínimo de 5 letras y un máximo de 10**

Para validar si una cadena tiene entre 5 y 10 letras (mayúsculas o minúsculas), utilizamos la expresión regular ^[a-zA-Z]{5,10}$. Esto significa que buscamos una cadena que comience y termine con letras, y que tenga entre 5 y 10 letras en total.

**import java.util.regex.\*;**

**public class Ejercicio3 {**

**public static void main(String[] args) {**

**String cadena = "ABCDE";**

**boolean cumpleRequisitos = Pattern.matches("^[a-zA-Z]{5,10}$", cadena);**

**if (cumpleRequisitos) {**

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

**} else {**

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

**}**

**}**

**}**

**Ejercicio 4: Validar el ingreso de un DNI (xx-xxx-xxx)**

Para validar un número de DNI en el formato "xx-xxx-xxx," utilizamos la expresión regular \d{2}-\d{3}-\d{3}. Esto significa que buscamos dos dígitos, un guion, tres dígitos, otro guion y finalmente tres dígitos.

import java.util.regex.\*;

public class Ejercicio4 {

public static void main(String[] args) {

String dni = "12-345-678";

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

if (esDniValido) {

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

} else {

System.out.println("El DNI no es válido.");

}

}

}

**Ejercicio 5: Validación de Direcciones de Correo Electrónico**

En el programa, también validamos direcciones de correo electrónico. Utilizamos la expresión regular ^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$ para verificar si una dirección de correo electrónico es válida.

import java.util.regex.\*;

import java.util.ArrayList;

public class Ejercicio5 {

public static void main(String[] args) {

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

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

correos.add("correo1@example.com");

correos.add("correo2@invalid");

correos.add("correo3@example.org");

for (String correo : correos) {

if (Pattern.matches("^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$", correo)) {

correosValidos.add(correo);

}

}

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

for (String correoValido : correosValidos) {

System.out.println(correoValido);

}

}

}

**Resultados**

Nuestro programa aplicó con éxito las expresiones regulares a los datos de entrada y validó cada uno de los ejercicios propuestos. Los patrones regulares utilizados en cada ejercicio permitieron verificar datos de manera efectiva y precisa.

**Conclusiones**

Las expresiones regulares son una herramienta poderosa en programación para validar, buscar y manipular datos textuales. Con una comprensión adecuada de los conceptos básicos y un diseño cuidadoso, se pueden aplicar de manera efectiva para resolver problemas de validación de datos y búsqueda de patrones. El uso de expresiones regulares en Java proporciona una manera eficiente y flexible de trabajar con datos textuales.