**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.

ArrayList<String> list = new ArrayList<>(Arrays.*asList*("fiorettinazareno@gmail.com", "manolomascapito@gmail.com","2617654332", "2617896543", "5678", "5523", "15/03/2005", "12/03/2006"));

1. **Validación de Datos**:s
   * 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.
2. package ProyectoLab.src;  
   import java.util.regex.Pattern;  
   public class RegEx {  
    public static void main(String args[]){  
    String ex = "^([a-zA-Z0-9.\_%-]+)@([a-zA-Z0-9.-]+)[.]([a-zA-Z]{2,6})$";  
    System.*out*.println(Pattern.*matches*(ex, "akjsfbkjsda@mail.com"));  
    }  
   }
   * Validar entrada de datos en un arraylist y aplicarlo.

package ProyectoLab.src;  
import java.util.ArrayList;  
import java.util.Scanner;  
import java.util.regex.Pattern;  
public class RegEx {  
 public static void main(String[] args){  
 ArrayList<String> list = new ArrayList<>();  
 Scanner sc = new Scanner(System.*in*);  
 String word="";  
 String ex = "^.\*[1-9$]";  
 word = sc.next();  
 while(list.size() < 6){  
 if(Pattern.*matches*(ex, word)){  
 System.*out*.println("Termina en numerico");  
 list.add(word);  
 } else {  
 System.*out*.println("No termina en numerico");  
 continue;  
 }   
 }  
 } //System.out.println(Pattern.matches(ex, "fiorettinazareno@gmail.com"));  
}

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 de Investigación sobre Expresiones Regulares y Validación de Datos**

**1. Introducción**

Las expresiones regulares son secuencias de caracteres que forman un patrón de búsqueda. Estos patrones son utilizados para realizar búsquedas y manipulaciones de cadenas de texto de manera eficiente. En el contexto de la programación, las expresiones regulares son herramientas poderosas para validar datos, lo que significa asegurarse de que los datos ingresados cumplan con ciertos criterios o formatos predefinidos.

**2. Comprensión de los Conceptos de Expresiones Regulares**

Las expresiones regulares consisten en caracteres literales y metacaracteres que definen un patrón de búsqueda. Algunos de los metacaracteres más comunes incluyen .\*, que coincide con cualquier caracter cero o más veces, y (?i), que hace que la expresión regular sea insensible a mayúsculas y minúsculas. Estos elementos permiten una flexibilidad significativa en la definición de patrones de búsqueda.

**3. Diseño y Desarrollo del Programa de Validación de Datos en Java**

En nuestro programa de validación de datos en Java, hemos utilizado expresiones regulares para garantizar que las cadenas ingresadas cumplan con ciertos criterios. Por ejemplo, si queremos validar que una cadena contenga la subcadena "naza" en cualquier parte, ignorando las diferencias de mayúsculas y minúsculas, utilizamos la expresión regular (?i).\*naza.\*.

Aquí está un fragmento del código Java:

java

import java.util.regex.\*;

public class ValidacionDatos {

public static void main(String[] args) {

String input = "hola naza";

String regex = "(?i).\*naza.\*";

Pattern pattern = Pattern.compile(regex);

Matcher matcher = pattern.matcher(input);

if (matcher.find()) {

System.out.println("La cadena cumple con el patrón.");

} else {

System.out.println("La cadena no cumple con el patrón.");

}

}

}

Este código demuestra cómo se puede implementar la validación de datos mediante expresiones regulares en Java.

**4. Ejemplos de Patrones de Expresiones Regulares Utilizados**

* Para validar que una cadena contenga la subcadena "abc" en cualquier parte, ignorando las diferencias de mayúsculas y minúsculas:

java

String regex = "(?i).\*abc.\*";

* Para validar que una cadena comience con "abc":

java

String regex = "^abc.\*";

Estos ejemplos ilustran la versatilidad de las expresiones regulares en la validación de datos.

**5. Claridad y Organización del Informe de Investigación**

Este informe se ha estructurado de manera clara para abordar los temas específicos solicitados. Se han proporcionado explicaciones detalladas de los conceptos de expresiones regulares, seguidas de ejemplos concretos de cómo se aplicaron en el programa de validación de datos. La información se presenta de manera ordenada, facilitando la comprensión del lector.

**Conclusión**

Las expresiones regulares son herramientas fundamentales en la validación de datos, permitiendo a los programadores establecer criterios específicos para garantizar la integridad y el formato de la información ingresada. Su comprensión y aplicación adecuadas son esenciales para el desarrollo de programas robustos y confiables.

1. **Ejercicios**:

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

package ProyectoLab.src;  
import java.util.regex.Pattern;  
public class RegEx {  
 public static void main(String[] args){  
 String ex = "^.\*[^0-9]$";  
 System.*out*.println(Pattern.*matches*(ex, "naza"));  
 }  
}

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

package ProyectoLab.src;  
import java.util.regex.Pattern;  
public class RegEx {  
 public static void main(String[] args){  
 String ex = "^[^1-9].\*$";  
 System.*out*.println(Pattern.*matches*(ex, "naza"));  
 }  
}

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

package ProyectoLab.src;  
import java.util.regex.Pattern;  
public class RegEx {  
 public static void main(String[] args){  
 String ex = "^.{5,10}$";  
 System.*out*.println(Pattern.*matches*(ex, "nazareno"));  
 }  
}

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

package ProyectoLab.src;  
import java.util.regex.Pattern;  
public class RegEx {  
 public static void main(String[] args){  
 String ex = "^[1-9]{7,8}$";  
 System.*out*.println(Pattern.*matches*(ex, "46328818"));  
 }  
}

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

package ProyectoLab.src;  
import java.util.ArrayList;  
import java.util.Scanner;  
import java.util.regex.Pattern;  
public class RegEx {  
 public static void main(String[] args){  
 ArrayList<String> list = new ArrayList<>();  
 Scanner sc = new Scanner(System.*in*);  
 String mail="";  
 String ex = "^([a-zA-Z0-9.\_%-]+)@([a-z]+)mail[.]([a-zA-Z]{1,5})$";  
 while(list.size() < 6){  
 System.*out*.println("Ingrese un mail");  
 mail = sc.next();  
 if(Pattern.*matches*(ex, mail)){  
 list.add(mail);  
 }  
 }  
 //System.out.println(Pattern.matches(ex, "fiorettinazareno@gmail.com"));  
 }  
}

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

package ProyectoLab.src;  
import java.util.ArrayList;  
import java.util.Scanner;  
import java.util.regex.Pattern;  
public class RegEx {  
 public static void main(String[] args){  
 ArrayList<String> list = new ArrayList<>();  
 Scanner sc = new Scanner(System.*in*);  
 String word="";  
 String ex = "^.\*abc.\*$";  
 word = sc.next();  
 if(Pattern.*matches*(ex, word)){  
 System.*out*.println("Contiene abc");  
 } else System.*out*.println("No contiene abc");  
 } //System.out.println(Pattern.matches(ex, "fiorettinazareno@gmail.com"));  
}