**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.

1-Nombre: Jose Hernandez

2-Correo Electronico:jose@gmail.com

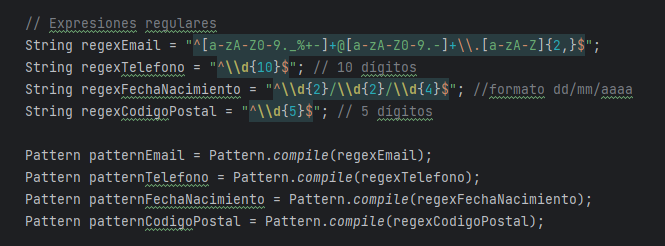
3-Telefono:2636717286

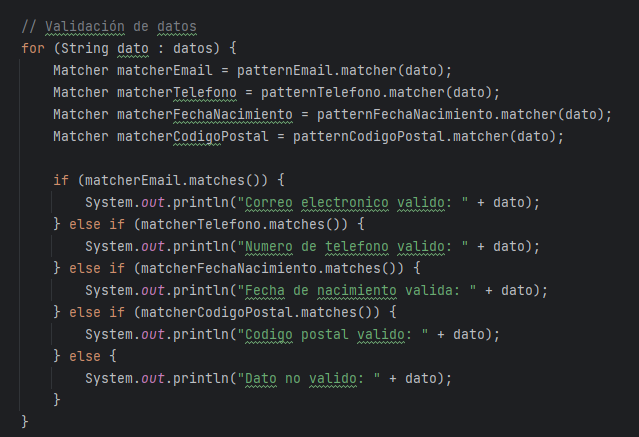
4-Fecha de Nacimiento:09/12/2000

5-Codigo Postal:5505

6-Direccion:Avenida San Jose 12

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.





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.

Expresiones Regulares

Una Expresión Regular es como una secuencia de caracteres que forman una secuencia o patrón que puede ser automatizada de alguna manera.

En un programa Java, una expresión regular se define mediante una cadena que obedece reglas especificas de coincidencia de patrones. Al ejecutar el código, la maquina Java compila esta cadena en un Pattern objeto y usa un Matcher objeto para encontrar coincidencias en el texto.

¿Que necesitamos para crear una Expresión Regular?

Para poder hacer uso de expresiones regulares en Java necesitamos importar el paquete [***regex***](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/regex/package-summary.html)

La clase regex nos aporta las siguientes clases:

* [Matcher](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/regex/Matcher.html): Esta clase nos permite hacer match sobre la secuencia de caracteres que nos define el Pattern.
* MatchResult: El resultado de la operación al hacer match
* Pattern: Es la representación de la expresión regular.
* PatternSyntaxException: Se lanza una [Unchecked Exception](https://refactorizando.com/crear-excepcion-personalizada-java/) para indicar un error de expresión en el patrón.

¿Como crear una expresión regular en Java?

Para crear una expresión regular haremos uso de los cuantificadores y metacaracteres.

Cuantificadores para una expresión regular:

Tenemos caracteres especiales que nos van a indicar el número de repeticiones de la expresión, la siguiente tabla muestra los caracteres

|  |  |
| --- | --- |
| **Cuantificador** | **Descripción** |
| n+ | Encuentra cualquier string con al menos un «n» |
| n\* | Encuentra cero o más ocurrencias de n |
| n? | Encuentra en el string la aparición de n cero o una vez |
| n{x} | Encuentra la secuencia de n tantas veces como indica x. |
| n{x,} | Encuentra una secuencia de X tantas veces como indica n |

Metacaracteres en una expresión regular:

|  |  |
| --- | --- |
| **Metacaracter** | **Descripción** |
| | | Símbolo para indicar OR. |
| . | Encuentra cualquier carácter |
| ^ | Sirve para hacer match al principio del string |
| $ | Hace match al final de un String |
| \d | Encuentra dígitos |
| \s | Busca un espacio |
| \b | Hace match al principio de una palabra. |
| \uxxxx | Encuentra el carácter Unicode especificado por el número hexadecimal xxxx |

Ejemplo de Expresión Regular para validar un Numero de Teléfono:

Expresión Regular: ^15\d{8}$

* ^ y $: Representan el inicio y el final de la cadena, asegurando que la expresión coincida con toda la cadena.
* 15: Coincide literalmente con “15”. Acá debería ir el prefijo común para números de teléfono del país (yo puse 15 como ejemplo).
* \d{8}:Coincide con exactamente 8 dígitos numéricos. Estos 8 dígitos representan el número de teléfono real.

Expresiones Regulares usadas en el programa:

1-Esta expresión regular se utiliza para verificar si una cadena cumple con el formato básico de una dirección de correo electrónico.

Expresión de Email: "^[a-zA-Z0-9.\_%+-]+ @[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$"

* ^ y $: Representan el inicio y el final de la cadena, asegurando que la expresión coincida con toda la cadena.
* [a-zA-Z0-9.\_%+-]+: Este grupo define el nombre de usuario de la dirección de correo electrónico.
  + [a-zA-Z0-9.\_%+-]: Coincide con cualquier carácter alfabético en minúsculas o mayúsculas, dígitos (0-9) y ciertos caracteres especiales: ".", "\_", "%", "+", y "-".
  + +: Indica que uno o más de estos caracteres pueden aparecer en el nombre de usuario.
* @: Coincide literalmente con el símbolo "@" que es un componente de una dirección de correo electrónico.
* [a-zA-Z0-9.-]+: Este grupo define el dominio de la dirección de correo electrónico.
  + [a-zA-Z0-9.-]: Coincide con cualquier carácter alfabético en minúsculas o mayúsculas, dígitos (0-9), y los caracteres "." y "-".
  + +: Indica que uno o más de estos caracteres pueden aparecer en el dominio. Esto asegura que el dominio debe tener al menos un carácter
* \\.: Coincide literalmente con el carácter ".". La barra invertida ( \ ) se utiliza para escapar el punto ya que el punto es un carácter especial en las expresiones regulares.
* [a-zA-Z]{2,}: Este grupo define la extensión de dominio (por ejemplo, ".com", ".org"):
  + [a-zA-Z]: Coincide con cualquier carácter alfabético en minúsculas o mayúsculas.
  + {2,}: Indica que debe haber al menos dos de estos caracteres para la extensión de dominio.

2- Esta expresión regular se utiliza para validar una cadena que debe contener exactamente 10 dígitos numéricos consecutivos. No permite ningún otro carácter, espacio o símbolo aparte de los dígitos.

Expresión Regular: "^\\d{10}$";

* ^ y $: Representan el inicio y el final de la cadena, asegurando que la expresión coincida con toda la cadena.
* \d{10}: Este grupo especifica una secuencia de dígitos numéricos que debe tener exactamente 10 dígitos:
  + \d: Coincide con cualquier dígito numérico del 0 al 9.
  + {10}: Indica que debe haber exactamente 10 dígitos en la secuencia.

3- Esta expresión regular se utiliza para validar una fecha en formato "dd/mm/aaaa". Verifica que la cadena cumpla con los siguientes criterios:

Dos dígitos para el día (01-31).

Un carácter de barra inclinada ("/").

Dos dígitos para el mes (01-12).

Otro carácter de barra inclinada ("/").

Cuatro dígitos para el año (por ejemplo, 2023).

Expresión Regular: "^\\d{2}/\\d{2}/\\d{4}$"

* ^ y $: Representan el inicio y el final de la cadena, asegurando que la expresión coincida con toda la cadena
* \d{2}: Este grupo se utiliza para validar el día y el mes de una fecha en formato "dd/mm/aaaa":
  + \d: Coincide con cualquier dígito numérico del 0 al 9.
  + {2}: Indica que debe haber exactamente 2 dígitos en la secuencia. Esto es para validar tanto el día como el mes, que consisten en dos dígitos cada uno.
* /: Coincide literalmente con el carácter de barra inclinada ("/"). Se utiliza para separar el día y el mes en la fecha.
* \d{4}: Este grupo se utiliza para validar el año en formato "dd/mm/aaaa":
  + \d: Coincide con cualquier dígito numérico del 0 al 9.
  + {4}: Indica que debe haber exactamente 4 dígitos en la secuencia. Esto es para validar el año, que consiste en cuatro dígitos.

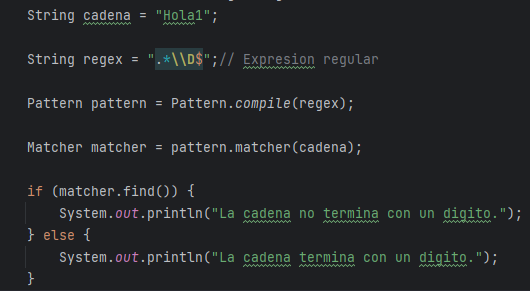
4- Esta expresión regular se utiliza para validar códigos postales que deben constar de exactamente cinco dígitos numéricos.

Expresión Regular: "^\\d{5}$";

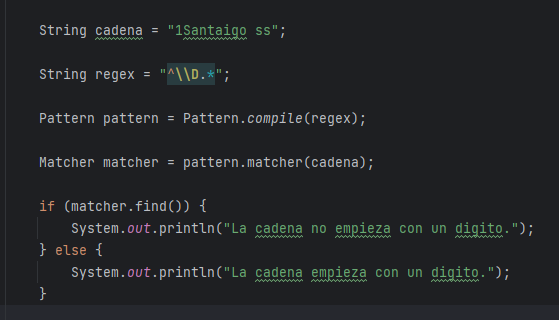
* ^ y $: Representan el inicio y el final de la cadena, asegurando que la expresión coincida con toda la cadena
* \d: Este patrón coincide con cualquier dígito numérico del 0 al 9.
* {5}: Indica que debe haber exactamente cinco dígitos en la secuencia.

1. **Ejercicios**:

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

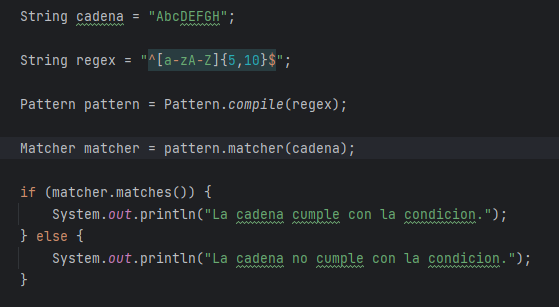


* Comprobar si el String *cadena* no empieza por un dígito.



* 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”.

