Listado Expresiones Regulares.

1. **Describe las clases implicadas en java.util.regex para manejar las expresiones regulares.**
    - Matcher: Es un motor que interpreta el patrón y comprueba si coincide con la cadena introducida.
    - Pattern: Es una representación compilada de una expresión regular, es un patrón que se tiene que cumplir para poder cumplir la expresión regular.
    - PatternSyntaxException: Es una excepción de tipo Unchecked que indica que hay un error sintáctico en la expresión regular.
2. **Considera el literal "hola". Indica el índice de inicio y el índice de fin. Explica su significado.**
    - El índice de inicio es 0 y el final es 4, el índice se coloca entre las letras del literal, de ahí el índice 4.
3. **Explica qué son los metacaracteres. Enuméralos.**
    - Los metacaracteres son caracteres especiales en un determinado campo.
    - los metacaracteres soportados en Java son: <([{\^-=$!|]})?*+.>
4. **Indica cómo se fuerza a que un metacarácter sea tratado como un carácter ordinal.**
    - Para que un metacarácter sea tratado como un carácter ordinal se le tiene que agregar una barra de escape delante del metacarácter ("\") o bien encerrándolo entre \Q"(Metacarácteres)"\E, "\Q" para que empiece la cita y "\E" que la acabe.
5. **Indica los caracteres que representan:**
    - **\\ :** Representa una barra invertida, "\".
    - **\n :** Representa un salto de línea.
    - **\t :** Representa una tabulación.

6. **Explica el significado de los siguientes metacaracteres:**
    - **. :** Representa a cualquier carácter.
    - **^ :** Representa el inicio de la expresión regular.
    - **$:** Representa el final de la expresión regular.
    - **\ :** Representa la barra de escape.
    - **[…] :** Representa un grupo en una expresión regular.
    - **[^…] :** Representa una exclusión de los caracteres que haya dentro de ese grupo.
    - **[a-z] :** Representa un rango de la a minúscula a la z minúscula en un grupo.
    - **[…&&…] :** Representa una intersección que permite que sólo los caracteres que van después de los dos ampersand son permitidos en el rango que va antes.
    - **? :** Indica que el carácter o rango de caracteres solo puede aparecer 0 o 1 veces.
    - **+ :** Indica que el carácter o rango de caracteres pueden aparecer 1 o más veces.
    - *** :** Indica que el carácter o rango de caracteres pueden aparecer 0 o más veces.
7. **Analiza el carácter "-" en las siguientes expresiones. ¿En ambas funciona como metacarácter? Utiliza coincidencias positivas y negativas.**
    - **A-Z :** Aquí funciona como un carácter más junto con la A y la Z.
    - **[A-Z] :** Aquí funciona como un metacarácter que indica un rango de entre la A y la Z.
8. **Describe las siguientes clases de caracteres predefinidas. Busca su expresión regular equivalente mediante corchetes (clases)**
    - **\d :** Indica cualquier dígito, [0-9].
    - **\s :** Indica un espacio en blanco, [\t\n\x0B\f\r]
    - **\w :** Indica un carácter cualquiera, [a-zA-Z_0-9].

9. **Busca dos expresiones que representen lo opuesto de:**
    - **\d :** El opuesto es \D, [^0-9].
    - **\s :** El opuesto es \S, [^\s].
    - **\w :** El opuesto es \W, [^\w].
10. **Busca la coincidencia:**
    - **(hola){2} :** La coincidencia sería: “holahola” ya que la palabra “hola” se tiene que repetir dos veces.
11. **Explica la utilidad de los siguientes métodos de la clase java.util.regex.Pattern, indicando argumentos y valores devueltos:**
    - **compile():** Compila la expresión regular en un patrón.
    - **matcher():** Busca las coincidencias de la cadena con la expresión regular.
    - **matches():** Compila la expresión regular e intenta buscar coincidencias con él.
    - **split():** Trocea la cadena introducida para hacerla coincidir con el patrón.
12. **Explica la utilidad de los siguientes métodos de la clase java.util.regex.Matcher:**
    - **matches() :** Intenta coincidir la cadena entera con el patrón.
    - **lookingAt() :** Intenta coincidir el principio de la cadena con el patrón.
    - **find() :** Intenta encontrar la siguiente subsecuencia de la cadena que coincide con el patrón.
    - **group():** Devuelve la subsecuencia de la cadena de la coincidencia anterior.
    - **start() :** Devuelve el índice de inicio de la subsecuencia capturada por el grupo dado durante las coincidencias.
    - **end() :** Devuelve el índice del último carácter que ha coincidido con el patrón.

13. ¿Cuándo conviene utilizar un matches() u otro?

- Conviene usar matches() cuando queremos hacer algo cuando la cadena introducida coincide con el patrón dado.

14. Objetos que hay que crear antes de buscar coincidencias (matches) de la expresión regular. ¿Puede hacerse de otras formas? Investiga

- Los objetos que hay que crear antes de buscar coincidencias con matches() son el Pattern que es para compilar la expresión regular y convertirla en patrón (Pattern pattern = Pattern.compile("RegEx");), y el Matcher que será para buscar las coincidencias con el patrón que se compiló antes (Matcher matcher = pattern.matcher("cadena");).
- Se puede hacer directamente con el método matches() construyéndolo tal que así:
  String cadena = "foo";
  System.out.println(cadena.matches("^foo$")); //Muestra true si se cumple con el patrón, false de lo contrario.

15. Utilidad de los caracteres ^y $

- **^ :** Inicio de la expresión regular.
- **$:** Fin de la expresión regular.

16. Indica mediante coincidencias positivas y negativas las diferencias entre estas expresiones regulares:

- **t.*s :** Indica que tiene que haber una "t", seguido de un carácter cualquiera y que ese carácter cualquiera pueda aparecer 0 o más veces y una "s".
- **t.*?s :**

17. Indica las expresiones regulares correspondientes a las siguientes especificaciones:

- **Palabras que comiencen por la letra t :** \bt[A-z]+
- **Palabras que finalicen por la letra s :** [A-z]+s\b

18. Copia el siguiente código y comprueba su funcionamiento Entrega TestValidatorOnline y compruébalo con las expresiones regulares:

- **hola :** Coincide: hola, no coincide: Hola.
- **\sHola :** Coincide: Hola, no coincide: hola.
- **hola{3} :** Coincide: holaholahola, no coincide: holahola.
- **(hola){3} :** Coincide: holaholahola, no coincide: hola.
- **[hola]{3} :** Coincide: hhh, no coincide: huha.
- **.bat :** Coincide: cbat, no coincide: holacat.
- **\.bat :** Coincide: .bat, no coincide: bat.
- **\\.bat :** Coincide: \cbat, no coincide: cbat.
- **\\hola :** Coincide: \hola, no coincide: hola.
- **\whola :** Coincide: ehola, no coincide: hola.
- **\Whola :** Coincide: .hola, no coincide: huha.
- **([A-Z][a-zA-Z]*)\s\1 :** Coincide: Minion Minion, no coincide: Anais Minion.
- **\d\d :** Coincide: 22, no coincide: hi.
- **\b :** Coincide: hola, no coincide: .
- **\\b :** Coincide: \b, no coincide: b.

Envía el resultado de la ejecución con dos entradas para cada expresión: una que coincida y otra que no.