CONTENIDO DE CADA PRÁCTICA EN PARTICULAR.

Práctica 1. Análisis de Expresiones Regulares

1. IDENTIFICACIÓN

Nombre de la práctica:	Análisis de Expresiones Regulares		
No. de práctica:	1	No. de sesiones:	2
No. de integrantes máximo por equipo:	1	**CANO GARCIA ESMERALDA YOSELIN**	

2. INTRODUCCIÓN

Una ***Gramática Formal*** definida sobre un alfabeto $\sum$ es una tupla de la forma:

$$\mathbf{G} = \{\sum_{\mathbf{T}}, \sum_{\mathbf{N}}, \mathbf{S}, \mathbf{P}\}$$

donde:

- $\sum_{\mathbf{T}}$ es un alfabeto de símbolos terminales
- $\sum_{\mathbf{N}}$ es un alfabeto de símbolos no terminales
- $\boldsymbol{S}$ es el símbolo inicial de la gramática
- $\mathbf{P}$ es un conjunto de producciones gramaticales

Además, se cumple:

$$\mathbf{S} \in \sum_{\mathbf{N}}$$
$$\sum_{\mathbf{T}} \cap \sum_{\mathbf{N}} = \emptyset$$
$$\sum = \sum_{\mathbf{T}} \cup \sum_{\mathbf{N}}$$

La gramática formal $\mathbf{G}$ permite generan un lenguaje $\mathbf{L} = \{\mathbf{x} \in \sum_{\mathbf{T}}^{*} | \mathbf{S} \overset{*}{\rightarrow} \mathbf{x}\}$
Por lo tanto, las palabras del lenguaje estarán formadas por cadenas de símbolos terminales generadas a partir del símbolo inicial de la gramática utilizando las producciones que la definen.

Una ***expresión regular*** es una notación normalizada para representar lenguajes regulares, es decir, lenguajes generados por gramáticas regulares (Tipo 3). Las expresiones regulares permiten describir con exactitud y sencillez cualquier lenguaje regular.

3. OBJETIVO GENERAL

Construir programas en un lenguaje de programación de alto nivel que realicen el análisis de cadenas de símbolos que forman palabras basado en la implementación de estructuras de datos y algoritmos de análisis, además del uso de expresiones regulares como segunda estrategia de implementación.

4. OBJETIVOS ESPECIFICOS

- ★ Identificar el proceso básico de análisis de cadenas de símbolos de un alfabeto.
- ★ Implementar estructuras de datos y algoritmos de análisis de cadenas de símbolos en un lenguaje de programación de alto nivel.
- ★ Implementar expresiones regulares para el análisis de cadenas de símbolos en un lenguaje de programación de alto nivel.

5. EQUIPO Y SOFTWARE A UTILIZAR

Equipo de cómputo con software para programación a elección de los alumnos.

6. PROCEDIMIENTO PARA EL DESARROLLO DE LA PRÁCTICA

1. Utilizar un lenguaje de programación para construir programas basados en estructuras de datos para analizar cadenas de símbolos con el propósito de identificar:
   - Número entero
   - Palabras en minúsculas
   - Palabras en mayúsculas
   - Identificadores (Nombres de variables)

2. Definir el alfabeto de símbolos y las expresiones regulares que permita identificar las siguientes tipos de palabras de un lenguaje regular:
   - Números enteros
   - Palabras en minúsculas
   - Palabras en mayúsculas
   - Identificador (Nombres de variables)
   - Símbolos

3. Utilizar un lenguaje de programación para construir programas basados en expresiones regulares para analizar cadenas de símbolos con el propósito de clasificar y contabilizar las palabras en las categorias:
   - Número entero
   - Palabras en minúsculas
   - Palabras en mayúsculas
   - Identifcador (Nombre de variable)
   - Símbolo

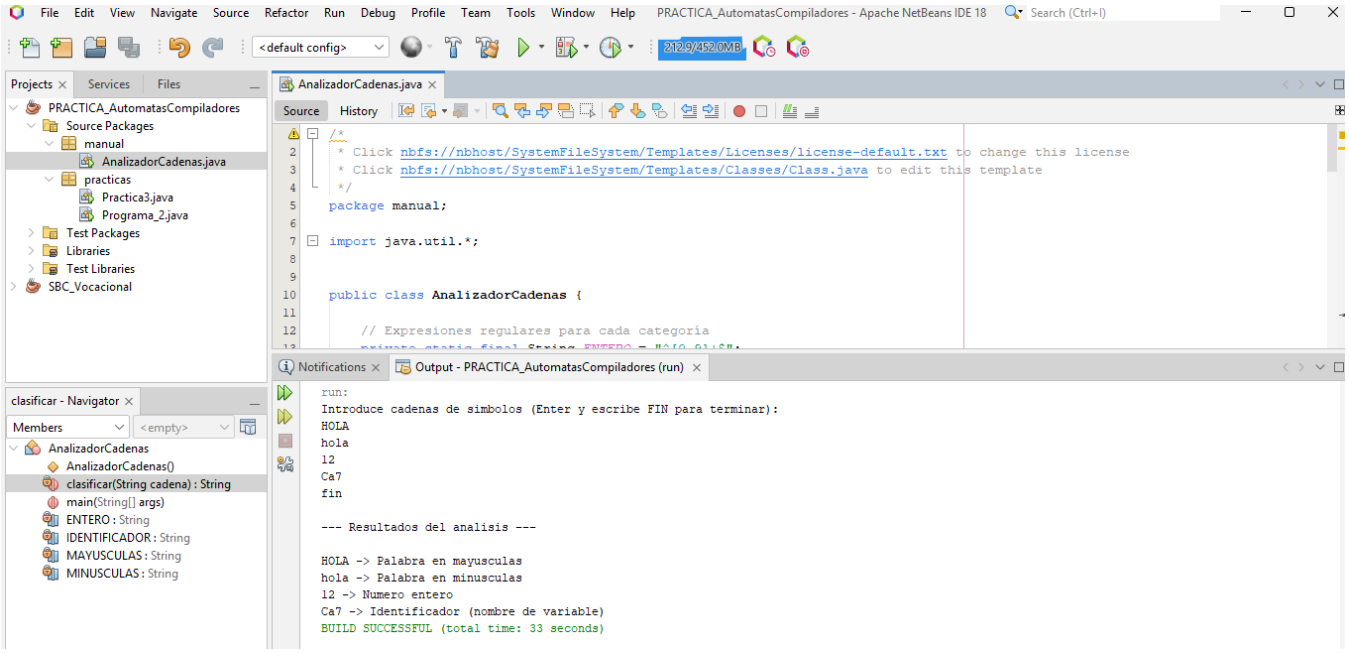
7. RESULTADOS

1. EJERCICIO 1

CODIGO:

```
package manual;

import java.util.*;


public class AnalizadorCadenas {

   // Expresiones regulares para cada categoría
   private static final String ENTERO = "^[0-9]+$";
   private static final String MINUSCULAS = "^[a-z]+$";
   private static final String MAYUSCULAS = "^[A-Z]+$";
   private static final String IDENTIFICADOR = "^[a-zA-Z_][a-zA-Z0-9_]*$";

   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      List<String> cadenas = new ArrayList<>();

      System.out.println("Introduce cadenas de simbolos (Enter y escribe FIN para terminar):");

      while (true) {
         String entrada = sc.nextLine().trim();
         if (entrada.equalsIgnoreCase("fin")) {
            break;
         }
         cadenas.add(entrada);
      }

      System.out.println("\n--- Resultados del analisis ---\n");
      for (String cadena : cadenas) {
         System.out.println(cadena + " -> " + clasificar(cadena));
      }
   }

   // Método para clasificar la cadena
   private static String clasificar(String cadena) {
      if (cadena.matches(ENTERO)) {
         return "Numero entero";
      } else if (cadena.matches(MINUSCULAS)) {
         return "Palabra en minusculas";
      } else if (cadena.matches(MAYUSCULAS)) {
         return "Palabra en mayusculas";
      } else if (cadena.matches(IDENTIFICADOR)) {
         return "Identificador (nombre de variable)";
      } else {
         return "No pertenece a ninguna categoría";
      }
   }
}
```

EVIDENCIA



EJERCICIO 2
Alfabeto Σ
Es un conjunto finito de símbolos o caracteres utilizados para construir cadenas.
Pueden Contener:

- Letras minúsculas: a–z
- Letras mayúsculas: A–Z
- Dígitos: 0–9
- Guion bajo _
- Símbolos especiales: + - * / =; () {} …

Expresiones Regulares

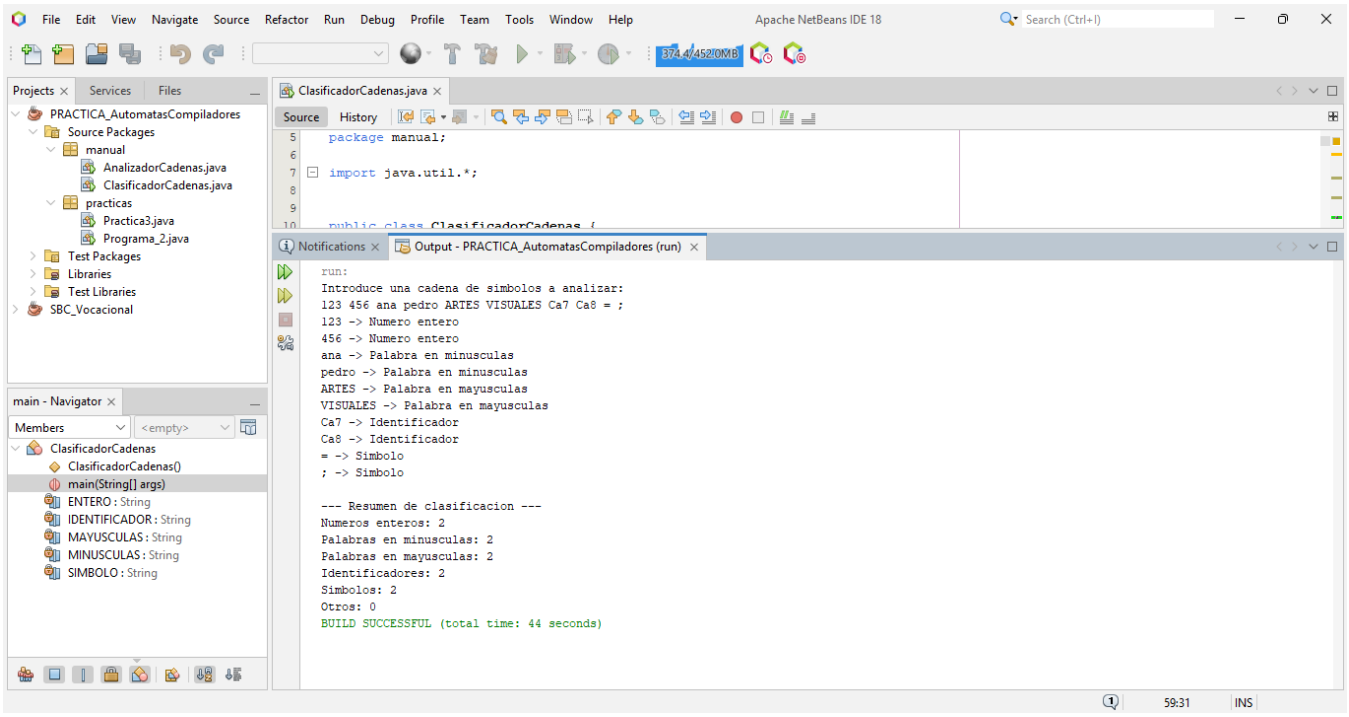
Números enteros: [0-9]+
Palabras en minúsculas: [a-z]+
Palabras en mayúsculas: [A-Z]+
Identificadores (nombres de variables): [a-zA-Z_][a-zA-Z0-9_]*
Símbolos: [+\-*/=;(){}]

EJERCICIO 3.
CODIGO

```
package manual;

import java.util.*;

public class ClasificadorCadenas {

    // Expresiones regulares
    private static final String ENTERO = "^[0-9]+$";
    private static final String MINUSCULAS = "^[a-z]+$";
    private static final String MAYUSCULAS = "^[A-Z]+$";
    private static final String IDENTIFICADOR = "^[a-zA-Z_][a-zA-Z0-9_]*$";
    private static final String SIMBOLO = "^[+\\-*/=;(){}]$";

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        // Contadores
        int enteros = 0, minusculas = 0, mayusculas = 0, identificadores = 0, simbolos = 0, otros = 0;

        System.out.println("Introduce una cadena de simbolos a analizar:");
        String entrada = sc.nextLine();

        // Dividimos la cadena en tokens separados por espacios
        String[] tokens = entrada.split("\\s+");

        for (String token : tokens) {
            if (token.matches(ENTERO)) {
                enteros++;
                System.out.println(token + " -> Numero entero");
            } else if (token.matches(MINUSCULAS)) {
                minusculas++;
                System.out.println(token + " -> Palabra en minusculas");
            } else if (token.matches(MAYUSCULAS)) {
                mayusculas++;
                System.out.println(token + " -> Palabra en mayusculas");
            } else if (token.matches(IDENTIFICADOR)) {
                identificadores++;
                System.out.println(token + " -> Identificador");
            } else if (token.matches(SIMBOLO)) {
                simbolos++;
                System.out.println(token + " -> Simbolo");
            } else {
                otros++;
                System.out.println(token + " -> No pertenece a ninguna categoria");
            }
        }
```

```
        // Mostrar resumen
        System.out.println("\n--- Resumen de clasificacion ---");
        System.out.println("Numeros enteros: " + enteros);
        System.out.println("Palabras en minusculas: " + minusculas);
        System.out.println("Palabras en mayusculas: " + mayusculas);
        System.out.println("Identificadores: " + identificadores);
        System.out.println("Simbolos: " + simbolos);
        System.out.println("Otros: " + otros);
    }
}
```

EVIDENCIA



8. CUESTIONARIO

1. ¿Cúal es utilidad de las expresiones regulares en la identificación de palabras de un lenguajes?

R. Las expresiones regulares permiten describir patrones de cadenas de manera compacta y precisa.
En la identificación de palabras de un lenguaje: Sirven para clasificar tokens (números, identificadores, símbolos, palabras reservadas), facilitan la validación de la estructura de una cadena sin tener que programar condiciones largas con if y bucles y son la base de analizadores léxicos en compiladores e intérpretes.

2. ¿Cuáles elementos del lenguaje de programación utilizaste para la implementación de las estructuras de datos?, ¿Cómo las utilizaste?

R. En Java utilicé principalmente: String[] (arreglo de cadenas) para dividir la entrada en tokens usando split(), contadores (int) para contabilizar cuántos elementos caen en cada categoría y clases de librería (Scanner) para capturar la entrada del usuario

3. ¿Cuál es el proceso que seguiste para analizar las cadenas de símbolos?

R. Definir el alfabeto y las expresiones regulares para cada categoría.
Leer la entrada del usuario (una cadena con palabras y símbolos).
Dividir la entrada en tokens usando espacios como separadores.
Comparar cada token contra las expresiones regulares.
Clasificar y contabilizar cada token según la categoría que cumpla.
Mostrar un reporte final con el conteo de cada tipo de palabra.

4. ¿Qué elementos del lenguaje de programación te permitieron implementar expresiones reguales?

En Java existen dos formas:
Método matches() de la clase String, compara un texto completo contra una regex.
Clases Pattern y Matcher (paquete java.util.regex) permiten reutilizar expresiones y buscar coincidencias parciales.
En el programa simple usamos String.matches() porque era suficiente.

5. ¿qué condiciones o recomendaciones se deben seguir al escribir las expresiones regulares?

- Definir claramente el alfabeto letras, dígitos y símbolos que se van a aceptar.
- Usar anclas `^` y `$` aseguran que la cadena completa cumpla la expresión regular
- Especificar cantidades con `+`, `*`, `{n,m}` según corresponda.
- Escapar caracteres especiales (`+`, `*`, `(`, `)`, `{`, `}`)
- Ser lo más específicos posible para evitar clasificaciones ambiguas.
- Probar con múltiples casos de entrada para validar que las regex funcionan correctamente.

9. BIBLIOGRAFÍA

Giró, J., Vázquez, J., Meloni, B., Constable, L. (2015). *Lenguajes formales y teoría de automátas*. Editorial Alfaomega. Argentina.

Ruiz Catalán, J. (2010). *Compiladores: teoría e implementación*. Editorial Alfaomega. México.

Brookshear, J. G. (1995). *Teoría de la Computación. Lenguajes formales, autómatas y complejidad*. Editorial Addison-Wesley Iberoamericana. Primera edición. U.S.A.