

Universidad Central del Ecuador
Facultad de Ingeniería y Ciencias Aplicadas
Carrera de Ingeniería en Ciencias de la Computación



Criptografía y seguridad de la información

**Técnicas de cifrados utilizados en los algoritmos criptográficos
básicos**

Integrantes:

- Altamirano Ortiz Jonathan Danilo
- Flórez Rivera Yaniry Mabely
- Gualoto Tigrero Erika Paola
- Simbaña Pulupa Pablo Fernando

Semestre: 8 vo

Fecha: 04/Noviembre/2024

Quito-2024



Ejercicio 1:

Algoritmo que escriba todas las permutaciones posibles de una palabra de longitud n SIN espacios (Anagrama). La palabra se ingresa al iniciar el algoritmo. El algoritmo debe mostrar el número total de permutaciones y las 10 primeras ordenadas alfabéticamente.

➤ **Técnica:** Permutación.

Librerías importadas:

- **java.util.List:** Se utiliza para incluir la interfaz List, que forma parte del paquete java.util. Esta interfaz representa una colección ordenada de elementos que permite duplicados y proporciona métodos para manipular la lista, como agregar, eliminar y acceder a elementos.
- **java.util.ArrayList:** Se utiliza para almacenar las permutaciones generadas. ArrayList permite un tamaño dinámico y acceso rápido a los elementos.
- **java.util.Collections:** Esta librería proporciona métodos estáticos para operar sobre colecciones, como la ordenación de listas.
- **java.util.Scanner:** Se utiliza para la entrada de datos desde la consola, permitiendo al usuario ingresar la palabra para la que se desean generar las permutaciones.

Clase y Método Principal

- **public class Permutaciones:** Define la clase pública Permutaciones.
- **public static void main(String[] args):** Método principal donde comienza la ejecución del programa.
- **Scanner scanner = new Scanner(System.in);:** Crea un objeto Scanner para leer la entrada del usuario desde la consola.

- **String palabra = scanner.nextLine();**: Lee la palabra ingresada por el usuario.

```
public class Permutaciones {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Ingresa una palabra: ");
        String palabra = scanner.nextLine();
    }
}
```

Inicialización de la Lista de Permutaciones

- **List<String> permutaciones = new ArrayList<>();**: Crea una lista para almacenar las permutaciones generadas.
- **generarPermutaciones("", palabra, permutaciones);**: Llama al método generarPermutaciones para iniciar la generación de permutaciones con un prefijo vacío.

```
List<String> permutaciones = new ArrayList<>();
generarPermutaciones("", palabra, permutaciones);
```

Ordenar y Mostrar Resultados

- **Collections.sort(permutaciones);**: Ordena las permutaciones alfabéticamente.
- **System.out.println("Total de permutaciones: " + permutaciones.size());**: Muestra el número total de permutaciones generadas.

```
Collections.sort(permutaciones);
System.out.println("Total de permutaciones: " +
    permutaciones.size());
```

Mostrar las Primeras 10 Permutaciones

- **for (int i = 0; i < Math.min(10, permutaciones.size()); i++):** Itera sobre las primeras 10 permutaciones (o menos si hay menos de 10).

- **System.out.println(permutaciones.get(i));**: Imprime cada una de las permutaciones seleccionadas.

```
System.out.println("Las 10 primeras permutaciones:");
for (int i = 0; i < Math.min(10, permutaciones.size());
i++) {
    System.out.println(permutaciones.get(i));
}
```

Método Recursivo para Generar Permutaciones

- **private static void generarPermutaciones(...)**: Método recursivo que genera las permutaciones.
- **int n = palabra.length();**: Obtiene la longitud de la palabra actual.
- **if (n == 0)**: Comprueba si no quedan caracteres por permutar; si es así, agrega el prefijo a la lista de permutaciones.
- **for (int i = 0; i < n; i++)**: Itera sobre cada carácter de la palabra:
 - **generarPermutaciones(...)**: Llama recursivamente al método, construyendo un nuevo prefijo y eliminando el carácter actual de la palabra.

```
private static void generarPermutaciones(String prefijo,
String palabra, List<String> permutaciones) {
    int n = palabra.length();
    if (n == 0) {
        permutaciones.add(prefijo);
    } else {
        for (int i = 0; i < n; i++) {
            generarPermutaciones(prefijo + palabra.charAt(i),
palabra.substring(0, i) + palabra.substring(i + 1, n),
permutaciones);
        }
    }
}
```

Implementación:

Aquí el sistema da la opción al usuario de ingresar la palabra que se desea permutar.

Ingresa una palabra: anonimo

Al procesar la palabra el sistema, realiza el mayor número de permutaciones e imprime en pantalla las 10 primeras en orden alfabético.

```
Total de permutaciones: 5040
Las 10 primeras permutaciones:
aimnnoo
aimnnoo
aimnnoo
aimnnoo
aimnono
aimnono
aimnono
aimnono
aimnoon
aimnoon
```

Ejercicio 2:

Algoritmo que realice el cifrado de un mensaje por permutación de filas, teniendo como clave n filas. Tanto n como el texto del mensaje se ingresan al iniciar el algoritmo. El algoritmo debe controlar que el número de caracteres del mensaje (sin espacios), sea menor o igual que $n \times n$. Imprima la matriz de cifrado, el mensaje original y el mensaje cifrado. Si en la matriz de cifrado sobran espacios para almacenar los caracteres del mensaje original, estos deben llenarse con "*".

➤ **Técnica:** Permutación de filas.

- Utilizamos la cifrado por transposición, dada la matriz generada leemos fila x fila creamos los caracteres columna x columna para generar el mensaje cifrado, es decir se genera un tipo de reorganización de caracteres utilizando la transposición.

```
// Leer la matriz columna por columna para formar el mensaje cifrado
StringBuilder mensajeCifrado = new StringBuilder();
for (int j = 0; j < n; j++) {
    for (int i = 0; i < n; i++) {
        mensajeCifrado.append(matriz[i][j]);
    }
}
```

El StringBuilder() contendrá el mensaje reorganizado en el orden del cifrado en nuestro caso lee las columnas en lugar de filas.

- Para asegurar que el mensaje complete la matriz o llene los espacios en blanco, rellenamos con un carácter especial en este caso lo

completamos con "x". Verificamos que el mensaje no exceda el no exceda el valor de la matriz de la siguiente manera:

```
// Verificar que el mensaje quepa en la matriz de n x n
if (mensaje.length() > n * n) {
    System.out.println(x:"Error! El mensaje excede el valor de la matriz.");
    return;
}
```

- La matriz en la cual se almacenar los caracteres del mensaje utilizando la matriz[i][j] y vamos relleno la matriz

```
// Crear la matriz de n x n y llenarla con los caracteres del mensaje
char[][] matriz = new char[n][n];
int indice = 0;
for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
        matriz[i][j] = mensajeRelleno.charAt(indice++);
    }
}
```

- Utilizamos el while para controlar que el programa realice otro mensaje encriptado o finalice su ejecución

```
while (continuar.equalsIgnoreCase(anotherString:"si")) {
    // Solicitar el número de filas (n) para crear la matriz nxn
    System.out.print(s:"Ingrese el número de filas (n): ");
    int n = scanner.nextInt();
    scanner.nextLine(); // Limpiar el buffer del scanner
}
```

- Utilizamos equalsIgnoreCase para crear la sentencia que reconozca el comando "si" y siga en su ejecución en el caso de que sea "no" finalice su ejecución
- Al iniciar su ejecución es necesario especificar "n" (valor de la matriz) en el que se almacenara el mensaje a ser encriptado
- Limpiamos el buffer con scanner.nextLine();
- Para las impresiones utilizamos la matriz original que fue creada y almacenado en la matriz[i][j] y leemos la matriz cifrada almacena como StringBuilder en la variable **mensajeCifrado**

```
// Imprimir la matriz original
System.out.println(x:"Matriz original:");
for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
        System.out.print(matriz[i][j] + " ");
    }
    System.out.println();
}
```

Ejemplo:

```

Ingrese el mensaje a cifrar: Hola Mundo
Matriz original:
H o l a
M u n d
o * * *
* * * *

Mensaje cifrado: HMo*ou**ln**ad**

```

Se utiliza el cifrado por transposición generado por la técnica por filas.

Ejercicio 3:

Algoritmo que realice el cifrado de un mensaje por permutación de columnas, teniendo como clave n columnas. Tanto n como el texto del mensaje se ingresan al iniciar el algoritmo. El algoritmo debe controlar que el número de caracteres del mensaje (sin espacios), sea menor o igual que $n \times n$. Imprima la matriz de cifrado, el mensaje original y el mensaje cifrado. Si en la matriz de cifrado sobran espacios para almacenar los caracteres del mensaje original, estos deben llenarse con "*".

➤ **Técnica:** Permutación de columnas.

La permutación de columnas es una técnica de cifrado que organiza los caracteres de un mensaje en una matriz, llenándola columna por columna. Luego, el mensaje cifrado se obtiene leyendo los caracteres en orden de filas. El tamaño de la matriz es $n \times n$, donde n es la clave que define el número de columnas. Si el mensaje no llena completamente la matriz, los espacios restantes se completan con caracteres de relleno, como *, para mantener la estructura.

Explicación del Código

1. El programa solicita el mensaje y elimina espacios. Luego pide el número de columnas n , verificando que el mensaje no exceda $n \times n$. Si es demasiado largo, se pide al usuario que aumente n .

```
// Solicitar el mensaje
System.out.print("Ingrese el mensaje: ");
String mensaje = scanner.nextLine().replace(" ", ""); // Eliminar espacios del mensaje
int n = 0; // Variable para el número de columnas

// Bucle para solicitar el número de columnas hasta que sea suficiente
do {
    System.out.print("Ingrese el número de columnas: ");
    n = scanner.nextInt();
    scanner.nextLine(); // Limpiar el buffer

    // Comprobar si el mensaje cabe en la matriz de n x n
    if (mensaje.length() > n * n) {
        System.out.println("El mensaje es demasiado largo para la clave proporcionada. Por favor, ingrese un número mayor.");
    }
} while (mensaje.length() > n * n);
```

2. Se inicializa una matriz $n \times n$, rellenándola inicialmente con *.

```
// Crear la matriz de cifrado y llenarla con '*'
char[][] matriz = new char[n][n];
for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
        matriz[i][j] = '*';
    }
}
```

3. Se recorre la matriz columna por columna, colocando los caracteres del mensaje hasta que se complete. Si sobran posiciones, permanecen con *.

```
// Llenar la matriz con el mensaje, columna por columna
int index = 0;
for (int j = 0; j < n; j++) {
    for (int i = 0; i < n; i++) {
        if (index < mensaje.length()) {
            matriz[i][j] = mensaje.charAt(index++);
        }
    }
}
```

4. Para obtener el mensaje cifrado, se lee la matriz en orden de filas, concatenando los caracteres en el orden en que aparecen.

```
// Crear el mensaje cifrado leyendo la matriz en orden de filas
StringBuilder mensajeCifrado = new StringBuilder();
for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
        mensajeCifrado.append(matriz[i][j]);
    }
}
```


5. Finalmente, imprime la matriz de cifrado, el mensaje original y el mensaje cifrado.

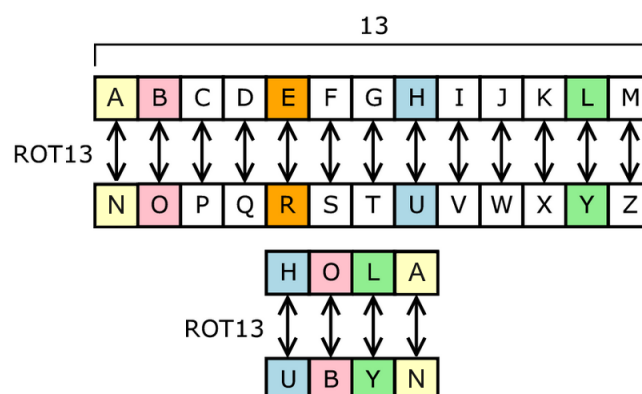
```
Problems @ Javadoc Declaration Console X
<terminated> ejercicio3 (1) [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (5 nov 2024, 2:41:36 a. m. - 2:41:46 a. m.) [pid: 4148]
Ingrese el número de columnas: 3
Matriz de cifrado:
D c d
i o *
s r *
Mensaje original: Discord
Mensaje cifrado: Dcdio*sr*
```

Ejercicio 4:

Algoritmo que realice el cifrado de una cadena de caracteres mediante un método de sustitución Mono alfabético de desplazamiento n caracteres a la derecha. Tanto la palabra como el valor de n se ingresan al iniciar el algoritmo. El algoritmo debe mostrar el alfabeto original, el alfabeto cifrado, la cadena de caracteres ingresada y su resultado.

- **Técnica:** Método de sustitución Mono alfabético de desplazamiento n caracteres a la derecha (Cifrado César).

El cifrado de desplazamiento, también conocido como Cifrado César, consiste en "desplazar" cada letra del mensaje una cantidad fija de posiciones n en el alfabeto. Por ejemplo, con un desplazamiento de 3, la letra "A" se convierte en "D", "B" en "E", y así sucesivamente. Esta técnica es simple y usa un alfabeto cifrado que es una versión desplazada del alfabeto original.



Explicación del Código

1. Se define el alfabeto original (abcdefghijklmnopqrstuvwxyz) y luego se crea el alfabeto cifrado desplazando n posiciones hacia la derecha.

```
// Solicitar al usuario que ingrese la cadena de caracteres y el valor de desplazamiento
System.out.print("Ingrese la cadena de caracteres: ");
String cadena = scanner.nextLine();
System.out.print("Ingrese el valor de desplazamiento: ");
int n = scanner.nextInt();

// Definir el alfabeto original
String alfabetoOriginal = "abcdefghijklmnopqrstuvwxyz";
```

2. Se utiliza un Map para asignar cada carácter del alfabeto original a su correspondiente carácter en el alfabeto cifrado, facilitando la sustitución.

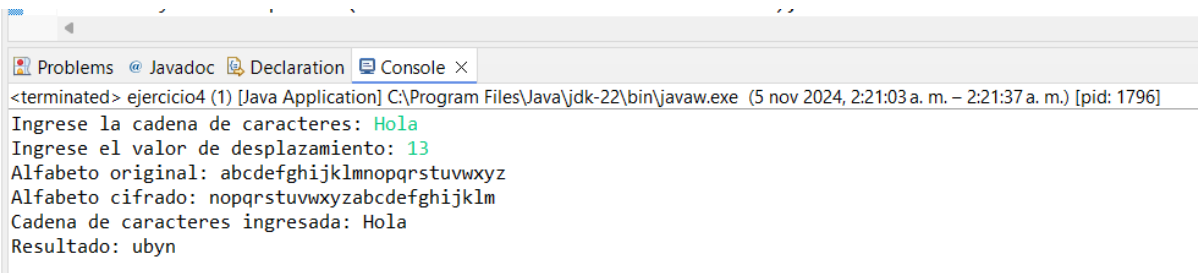
```
// Crear el alfabeto cifrado desplazando el alfabeto original n caracteres a la derecha
String alfabetoCifrado = alfabetoOriginal.substring(n) + alfabetoOriginal.substring(0, n);

// Crear un mapa de mapeo de cada carácter del alfabeto original a su correspondiente en el alfabeto cifrado
Map<Character, Character> mapeo = new HashMap<>();
for (int i = 0; i < alfabetoOriginal.length(); i++) {
    mapeo.put(alfabetoOriginal.charAt(i), alfabetoCifrado.charAt(i));
}
```

3. Para cada carácter en la cadena ingresada, se encuentra su equivalente en el alfabeto cifrado usando el mapeo. Si el carácter no está en el alfabeto (como un símbolo o número), se mantiene sin cambios.

```
// Cifrar la cadena de caracteres reemplazando cada carácter por su correspondiente en el alfabeto cifrado
StringBuilder resultado = new StringBuilder();
for (char caracter : cadena.toLowerCase().toCharArray()) {
    resultado.append(mapeo.getOrDefault(caracter, caracter));
}
```

4. Se imprimen el alfabeto original, el alfabeto cifrado, la cadena ingresada y el resultado cifrado.

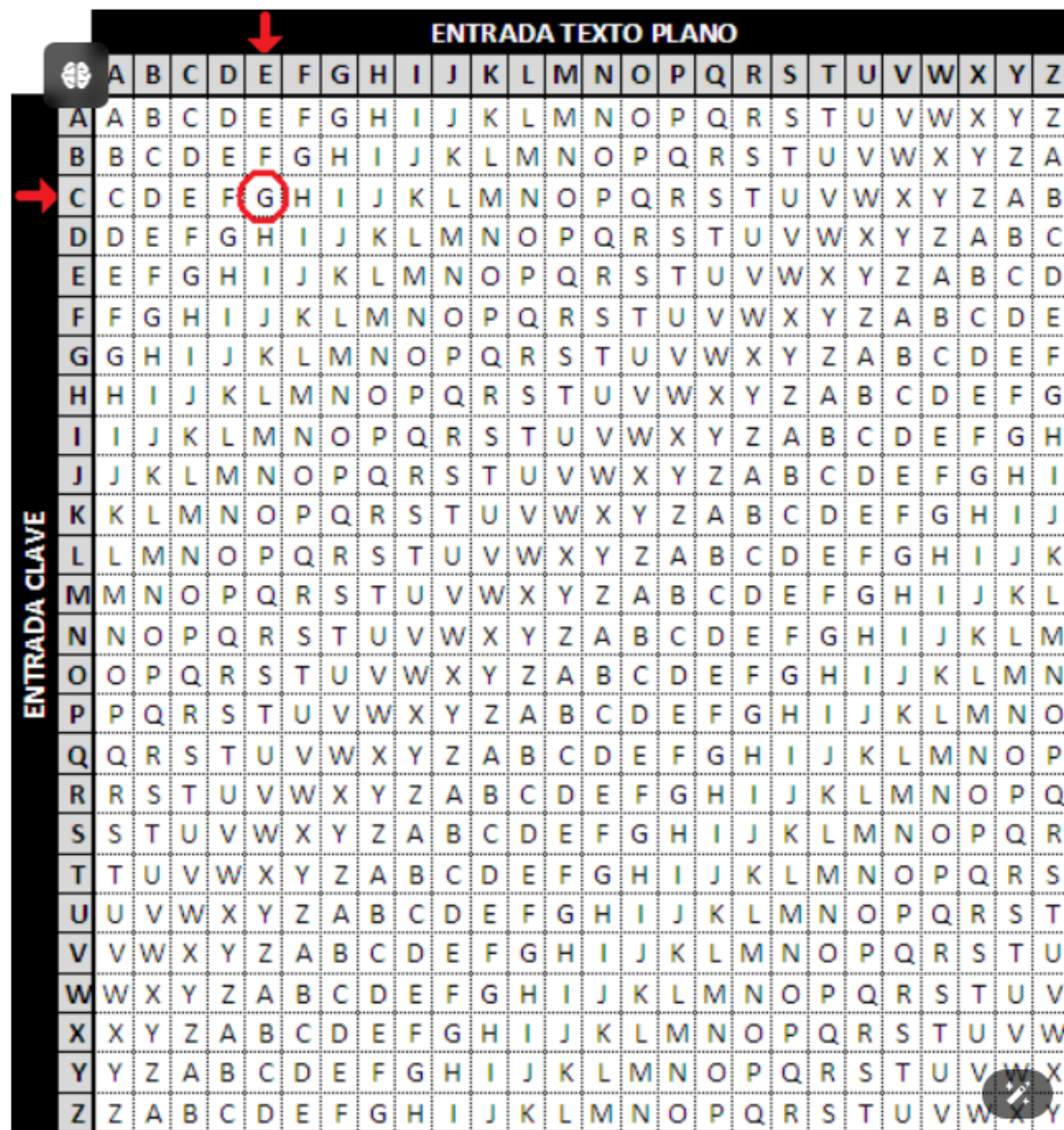


```
<terminated> ejercicio4 (1) [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (5 nov 2024, 2:21:03 a. m. - 2:21:37 a. m.) [pid: 1796]
Ingrese la cadena de caracteres: Hola
Ingrese el valor de desplazamiento: 13
Alfabeto original: abcdefghijklmnopqrstuvwxyz
Alfabeto cifrado: nopqrstuvwxyzabcdefghijklmnopqrstuvwxyz
Cadena de caracteres ingresada: Hola
Resultado: ubyn
```

Ejercicio 5:

Algoritmo que realice el cifrado de una cadena de caracteres mediante un método de sustitución Poli alfabético de Vigenère. La cadena se ingresa al iniciar el algoritmo. El algoritmo debe mostrar la cadena de caracteres ingresada, la clave de cifrado y la cadena de caracteres cifrada.

- **Técnica:** Método de sustitución Poli alfabético de Vigenère.



The image shows a Vigenère cipher table. The top row is labeled 'ENTRADA TEXTO PLANO' and contains the alphabet A-Z. The left column is labeled 'ENTRADA CLAVE' and also contains the alphabet A-Z. A red arrow points to the letter 'E' in the top row. Another red arrow points to the letter 'G' in the third row, second column. A red circle highlights the letter 'G' in the third row, second column. A small icon of a brain is in the top-left corner, and a small icon of a key is in the bottom-right corner.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

El cifrado de Vigenère es un método de sustitución polialfabético que utiliza una clave para cifrar el mensaje. La clave se repite hasta coincidir con la longitud del mensaje. Cada carácter del mensaje se desplaza un número de posiciones basado en el carácter correspondiente de la clave.

Así, el cifrado cambia en función de la posición y el carácter de la clave, haciendo que el cifrado sea más seguro que el cifrado César.

Explicación del Código

1. El alfabeto está definido como abcdefghijklmnopqrstuvwxyz. La clave ingresada se repite hasta que su longitud coincida con la del mensaje, lo que permite utilizar cada letra de la clave para cifrar una letra específica del mensaje.

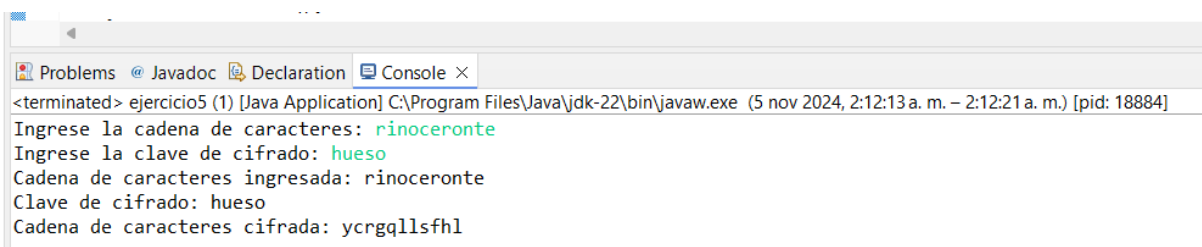
```
// Solicitar al usuario que ingrese la cadena de caracteres y la clave de cifrado
System.out.print("Ingrese la cadena de caracteres: ");
String cadena = scanner.nextLine();
System.out.print("Ingrese la clave de cifrado: ");
String clave = scanner.nextLine().toLowerCase();

// Definir el alfabeto
String alfabeto = "abcdefghijklmnopqrstuvwxyz";
```

2. Para cada carácter del mensaje, se obtiene su índice en el alfabeto y se suma al índice del carácter correspondiente de la clave. El resultado se ajusta al rango del alfabeto usando el módulo 26. Este índice cifrado se utiliza para obtener el carácter correspondiente en el alfabeto.

```
// Cifrar la cadena de caracteres utilizando la clave de cifrado
StringBuilder resultado = new StringBuilder();
for (int i = 0; i < cadena.length(); i++) {
    char caracter = Character.toLowerCase(cadena.charAt(i));
    char claveCaracter = clave.charAt(i % clave.length());
    if (alfabeto.indexOf(caracter) != -1) {
        int indice = (alfabeto.indexOf(caracter) + alfabeto.indexOf(claveCaracter)) % alfabeto.length();
        resultado.append(alfabeto.charAt(indice));
    } else {
        resultado.append(cadena.charAt(i)); // Dejar el carácter sin cambios si no está en el alfabeto
    }
}
```

3. Se imprime la cadena ingresada, la clave y el mensaje cifrado.



```
<terminated> ejercicio5 (1) [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (5 nov 2024, 2:12:13 a. m. - 2:12:21 a. m.) [pid: 18884]
Ingrese la cadena de caracteres: rinoceronte
Ingrese la clave de cifrado: hueso
Cadena de caracteres ingresada: rinoceronte
Clave de cifrado: hueso
Cadena de caracteres cifrada: ycrqllsfhl
```

Ejercicio 6:

Algoritmo que realice el cifrado de una cadena de caracteres utilizando la siguiente tabla de cifrado:

*	A	S	D	F	G
Q	a	b	c	d	e
W	f	g	h	i	j
E	k	l	m	n	o
R	p	q	r	s	t
T	u	v	x	y	z

➤ **Técnica:** Tabla de sustitución (matriz de equivalencia).

La técnica de cifrado por sustitución utiliza una matriz de equivalencias en la que cada letra del alfabeto se reemplaza por una combinación específica de caracteres. Este método es una forma básica de cifrado de sustitución, donde se mapea cada letra a un valor único en la matriz de cifrado. Si un carácter no está en la matriz, se sustituye por **.

Explicación del Código

Se crea una matriz de mapeo en un Map, que asocia cada letra del alfabeto con una combinación de dos letras, como QA, QS, etc.

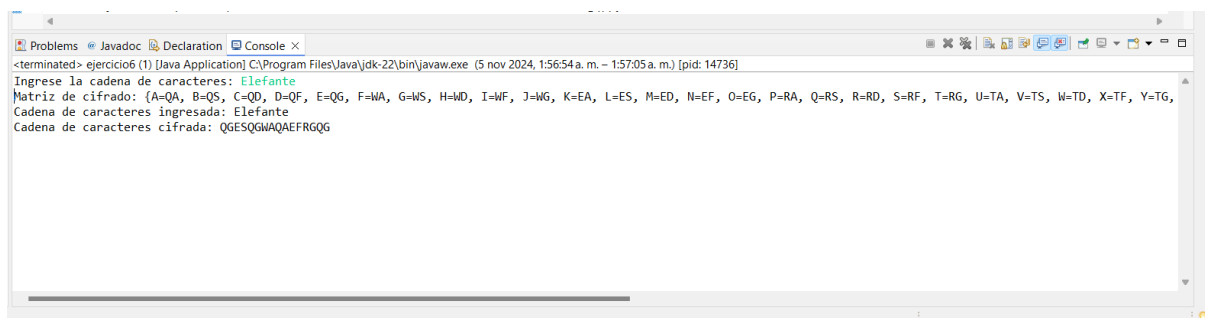
```
// Definir la matriz de cifrado
Map<Character, String> matrizCifrado = new HashMap<>();
String[] equivalencias = {
    "QA", "QS", "QD", "QF", "QG", // A-E
    "WA", "WS", "WD", "WF", "WG", // F-J
    "EA", "ES", "ED", "EF", "EG", // K-O
    "RA", "RS", "RD", "RF", "RG", // P-T
    "TA", "TS", "TD", "TF", "TG" // U-Y
};
```

Se recorre cada carácter de la cadena ingresada y se reemplaza con su equivalente en la matriz de cifrado. Si un carácter no tiene mapeo (como un símbolo o número), se sustituye por **.

```
// Asignar las equivalencias a las letras
for (int i = 0; i < equivalencias.length; i++) {
    matrizCifrado.put((char) ('A' + i), equivalencias[i]);
    matrizCifrado.put((char) ('a' + i), equivalencias[i]);
}

// Cifrar la cadena de caracteres
StringBuilder resultado = new StringBuilder();
for (char caracter : cadena.toCharArray()) {
    String cifrado = matrizCifrado.getOrDefault(caracter, "**");
    resultado.append(cifrado);
}
```

Finalmente, se imprime la matriz de cifrado, la cadena original y el mensaje cifrado, mostrando cómo cada letra del mensaje ha sido reemplazada.



Bibliografías:

1. H. Schildt, Java: The Complete Reference, 11th ed. McGraw-Hill Education, 2018.
2. J. Bloch, Effective Java, 3rd ed. Addison-Wesley, 2018.
3. Oracle, "Java Documentation," [Online]. Available: <https://docs.oracle.com/javase/8/docs/api/>. [Accessed: Nov. 4, 2024].
4. R. R. de Araujo, "Reticulados algébricos e aplicações a códigos e

criptografía," Doctoral dissertation, [sn], 2018.

5. "Ejemplos de uso del Cifrado César en un Escape Room educativo," eduescaperoom.com, [Online]. Available: <https://eduescaperoom.com/ejemplos-de-uso-del-cifrado-cesar-en-un-escape-room-educativo/#:~:text=El%20cifrado%20C%C3%A9sar%20consiste%20en,la%20Z%20por%20la%20A>. [Accessed: 04-Nov-2024].
6. M. García-Larragán, "Criptografía I," Blogspot, 2015. [Online]. Available: <https://mikelgarcialarragan.blogspot.com/2015/03/criptografia-i.html>. [Accessed: 04-Nov-2024].
7. P. J. Arellano Aguilar y M. Tapia Sánchez, "Criptografía en sistemas de información," *Revista Digital Universitaria*, vol. 7, no. 7, pp. 1-9, 2006. [Online]. Available: https://www.revista.unam.mx/vol.7/num7/art55/jul_art55.pdf. [Accessed: 04-Nov-2024].