Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Introducción a la Programación y Computación 1 Sección C

Catedrático: Ing. Moisés Velásquez

Tutor académico: Carlos Campanero, William Corado

Práctica no. 1

Manual Técnico

Nombre: Diego Enrique Arriaga Meléndez

Canet: 202003892

Índice

Clase Principal……1-3

Clase Encriptar……4-12

Clase Desencriptar……13-21

Clase Ataque con Texto Plano……22-34

Clase Reportes……..35-36

Reporte del Proceso de Deseencriptar……..……36-40

Reporte del Proceso de Encriptar……41-43

Reporte del Proceso de Ataque con Texto Plano……43-48

Clase Principal

La clase principal del programa se llama *Practica1LabIPC1*.

```
public class PracticalLabIPC1 {

    /**
     * @param args the command line arguments
     */
    static encriptador opcion1=new encriptador();
    static desencriptador opcion2=new desencriptador();
    static Ataque_Texto opcion3= new Ataque_Texto();
    static reportes opcion4=new reportes();
    static Scanner sc=new Scanner(System.in);
    static String op;

    //Colores de texto
       static String verde="\033[32m";
       static String cyan="\033[36m";
       static String negro="\033[30m";
       static String rojo="\033[31m";
       static String amar="\033[33m";
        //----------------------
    public static void main(String[] args) {
        int cont=1;
        while(cont>0){
            System.out.println("================MENU====================");
            System.out.println("|"+verde+" 1."+negro+" Encriptar                              |");
            System.out.println("|"+verde+" 2."+negro+" Desencriptar                           |");
            System.out.println("|"+verde+" 3."+cyan+" Ataque"+negro+" con texto plano                 |");
            System.out.println("|"+verde+" 4."+cyan+" Generar"+negro+" Reportes                       |");
            System.out.println("|"+verde+" 5."+rojo+" Finalizar"+negro+" Proyecto                     |");
            System.out.println("=========================================");
            System.out.print("Ingrese el número de acción que quiere realizar: ");
            op=sc.nextLine();System.out.println();
```

Primero se importó una librería con funciones tipo escáner que se utilizarán posteriormente.

```
import java.util.Scanner;
```

Se mandan a llamar las clases que se encargan de realizar todos los procesos del programa. El nombre que se les puso hace referencia al número de opción que tienen en el menú de inicio.

```
static encriptador opcion1=new encriptador();
static desencriptador opcion2=new desencriptador();
static Ataque_Texto opcion3= new Ataque_Texto();
static reportes opcion4=new reportes();
```

Se crea una variable tipo *Scanner* para poder analizar la información que él usuario ingrese en la consola. La variable tipo *string* servirá para almacenar la información que escriba el usuario cuando sea necesario.

```
static Scanner sc=new Scanner(System.in);
static String op;
```

Se crearon variables tipo *string*, se utilizarán para que el texto que imprima el programa tenga color. Estas variables se crearon en todas las clases que impriman texto, por lo que ya se volverán a explicar.

```
//Colores de texto
   static String verde="\033[32m";
   static String cyan="\033[36m";
   static String negro="\033[30m";
   static String rojo="\033[31m";
   static String amar="\033[33m";
    //----------------------
```

Inicio del método principal. Primero se creo una variable tipo *int* la cual servirá como contador del ciclo que será la base del funcionamiento del menú de inicio.

Posteriormente comienza un ciclo *while*, el cual tendrá como condición la variable *int* explicada anteriormente. El valor de la variable *cont* se modificará hasta que el usuario indique que quiere salir del programa, provocando que la condición ya no se cumpla y el ciclo deje de repetirse. Se utilizará un ciclo igual para el menú de todos los métodos, por lo que no se volverá a explicar.

```
int cont=1;
while(cont>0){
```

Una vez iniciado el ciclo *while* el programa empezara a escribir en la consola un texto que se asemeje a un menú convencional mediante la función *System.out.println*, se le incluyeron las variables *string* anteriormente creadas para que el texto se imprima de diferentes colores.

```
System.out.println("================MENU==================");
System.out.println("|"+verde+" 1."+negro+" Encriptar                          |");
System.out.println("|"+verde+" 2."+negro+" Desencriptar                        |");
System.out.println("|"+verde+" 3."+cyan+" Ataque"+negro+" con texto plano              |");
System.out.println("|"+verde+" 4."+cyan+" Generar"+negro+" Reportes                    |");
```

Una vez se imprimió el menú se imprime la instrucción que le pide al usuario ingresar un número que haga referencia a la acción que quiera realizar.

Primero la variable *sc* se encargará de recibir lo escriba el usuario en la consola, posteriormente este valor se le dará a la variable *op* para que lo almacene. Se utiliza el *System.out.println* para poner un *Enter* en la consola.

```
System.out.print("Ingrese el número de acción que quiere realizar: ");
op=sc.nextLine();System.out.println();
```

Se establece un *switch* cuya condicional será la información que tenga la variable *op*. Si el usuario escribe del número 1-4 se mandará a llamar el menú del proceso correspondiente, según como se hayan ordenada en el menú de inicio. Si el usuario ingresa el número 5 el contador pasara a valer 0 por lo que el ciclo se parara y el programa dejara de funcionar. En caso de que el usuario ingrese cualquier otro valor, ya sea numérico o dígito, el programa le pedirá que ingrese un valor adecuado y se volverá a iniciar el ciclo *while* principal.

```
switch(op){
        case "1":
            opcion1.menu();
            break;
        case "2":
            opcion2.menu();
            break;
        case "3":
            opcion3.menu();
            break;
        case "4":
            opcion4.menu();
            break;
        case "5":
            cont--;
            break;
        default:
            System.out.println(rojo+"ERROR. Ingrese un numero de opción adecuado."+negro);
            break;
```

Fin del método principal.

Clase de Encriptar

Esta clase se encarga de almacenar un mensaje que ingrese el usuario y se encargue

```
import java.util.Scanner;
import java.io.*;

/**
 *
 * @author Diego
 */
public class encriptador {

//Colores de texto
      static String verde="\033[32m";
      static String cyan="\033[36m";
      static String negro="\033[30m";
      static String rojo="\033[31m";
      static String amar="\033[33m";
       //----------------------
   static Scanner sc=new Scanner(System.in);
   static String op;
   static String[] caracterS;
   static double mb;
   public static int caracterI[],M[][], A[][]=new int[3][3], B[][],colum,Ri[][], Rf[][];

   //Metodo principal de la clase
   public static void menu(){
       int cont=1;
       while(cont>0){
```

Primero se importó una librería con funciones tipo escáner que se utilizarán posteriormente. También se importó una librería que entre muchas de sus funciones tiene la capacidad de detectar algún error logístico en programa y evita que este se cierre por culpa de este error.

```
import java.util.Scanner;
import java.io.*;
```

Se creo un arreglo que se encargara de almacenar cada carácter del mensaje que ingresara el usuario. Una variable que se encargara de almacenar la cantidad de caracteres que se ingresaron. Una variable que encargara de almacenar la cantidad de columnas que debe tener la matriz. También se crearon varias matrices que almacenaran la información que ingrese el usuario y también para almacenar las operaciones.

```
static String[] caracterS;
static double mb;
public static int colum,caracterI[],M[][], A[][]=new int[3][3], B[][],Ri[][], Rf[][];
```

Se crea el menú de la misma manera de como se hizo con el menú de inicio.

```
int cont=1;
while(cont>0){
    System.out.println("=============="+verde+"Menu "+cyan+"Encriptar"+negro+"==============");
    System.out.println("|"+amar+" 1."+verde+" Ingreso "+negro+"Mensaje                          |");
    System.out.println("|"+amar+" 2."+verde+" Ingreso Matriz Clave "+negro+"A                   |");
    System.out.println("|"+amar+" 3."+verde+" Ingreso Matriz Clave "+negro+"B                   |");
    System.out.println("|"+amar+" 4."+negro+" Encriptar                                  |");
    System.out.println("|"+amar+" 5."+cyan+" Regresar"+negro+" al Menú Principal             |");
    System.out.println("=============================================");
    System.out.print("Ingrese el número de acción que quiere realizar: ");
    op=sc.nextLine();System.out.println();
```

Se creo un *switch* que funciona de la misma manera que el del método principal.

```
switch(op){
    case "1":
        matrizM();
        break;

    case "2":
        claveA();
        break;

    case "3":
        claveB();
        break;

    case "4":
        encriptar();
        break;

    case "5":
        cont--;
        break;
    default:
            System.out.println(rojo+"ERROR. Asegurese que el dígito ingresado sea 1, 2, 3 o 4"+negro);
            break;
```

Se creo un método encargado de almacenar el mensaje que ingrese el usuario en una matriz. Primero la variable *op* almacena la información detectada con la variable *sc.*

```
private static void matrizM(){
    try{
    System.out.print(verde+"Ingrese "+negro+"mensaje a encriptar (Sin tildes): ");
            op=sc.nextLine();System.out.println();
```

La variable *mb* almacena la cantidad de caracteres que tiene el mensaje ingresado por el usuario. Posteriormente con este valor se calculo la cantidad de columnas que tiene que tener la matriz del mensaje y se inicializo la matriz del mensaje, indicando que la cantidad de columnas que tiene que tener es la ya calculada.

```
mb=op.length();
colum=(int)Math.ceil(mb/3);
M=new int[3][colum];
```

Primero un arreglo se encarga de almacenar cada carácter del mensaje, el mensaje se dividió mediante la función *split* que no tiene separador, por lo que toma todos los signos del mensaje. Se inicializo un arreglo que se encargara de almacenar el mensaje en su forma númerica. Se crearon dos variables que utilizaran posteriormente como contadores.

```
caracterS=op.split("");
caracterI=new int[(int)mb];
int c=0,d=0;
```

Se creo un ciclo que se repite la misma cantidad de caracteres que tenga el mensaje original. Dentro del ciclo se encuentran distintos condicionales *if* que dependiendo del signo que tenga el arreglo *string* le dará un valor especifico al arreglo *int*.

```
while(c<mb){
    if (caracterS[c].equals("A") || caracterS[c].equals("a")) {caracterI[c]=0;}
    else if (caracterS[c].equals("B") || caracterS[c].equals("b")) {caracterI[c]=1;}
    else if (caracterS[c].equals("C") || caracterS[c].equals("c")) {caracterI[c]=2;}
    else if (caracterS[c].equals("D") || caracterS[c].equals("d")) {caracterI[c]=3;}
    else if (caracterS[c].equals("E") || caracterS[c].equals("e")) {caracterI[c]=4;}
    else if (caracterS[c].equals("F") || caracterS[c].equals("f")) {caracterI[c]=5;}
    else if (caracterS[c].equals("G") || caracterS[c].equals("g")) {caracterI[c]=6;}
    else if (caracterS[c].equals("H") || caracterS[c].equals("h")) {caracterI[c]=7;}
    else if (caracterS[c].equals("I") || caracterS[c].equals("i")) {caracterI[c]=8;}
    else if (caracterS[c].equals("J") || caracterS[c].equals("j")) {caracterI[c]=9;}
    else if (caracterS[c].equals("K") || caracterS[c].equals("k")) {caracterI[c]=10;}
    else if (caracterS[c].equals("L") || caracterS[c].equals("l")) {caracterI[c]=11;}
    else if (caracterS[c].equals("M") || caracterS[c].equals("m")) {caracterI[c]=12;}
    else if (caracterS[c].equals("N") || caracterS[c].equals("n")) {caracterI[c]=13;}
    else if (caracterS[c].equals("Ñ") || caracterS[c].equals("ñ")) {caracterI[c]=14;}
    else if (caracterS[c].equals("O") || caracterS[c].equals("o")) {caracterI[c]=15;}
    else if (caracterS[c].equals("P") || caracterS[c].equals("p")) {caracterI[c]=16;}
    else if (caracterS[c].equals("Q") || caracterS[c].equals("q")) {caracterI[c]=17;}
    else if (caracterS[c].equals("R") || caracterS[c].equals("r")) {caracterI[c]=18;}
    else if (caracterS[c].equals("S") || caracterS[c].equals("s")) {caracterI[c]=19;}
    else if (caracterS[c].equals("T") || caracterS[c].equals("t")) {caracterI[c]=20;}
    else if (caracterS[c].equals("U") || caracterS[c].equals("u")) {caracterI[c]=21;}
    else if (caracterS[c].equals("V") || caracterS[c].equals("v")) {caracterI[c]=22;}
    else if (caracterS[c].equals("W") || caracterS[c].equals("w")) {caracterI[c]=23;}
    else if (caracterS[c].equals("X") || caracterS[c].equals("x")) {caracterI[c]=24;}
    else if (caracterS[c].equals("Y") || caracterS[c].equals("y")) {caracterI[c]=25;}
    else if (caracterS[c].equals("Z") || caracterS[c].equals("z")) {caracterI[c]=26;}
    else{caracterI[c]=27;}
    c++;
}
```

A todos los espacios de la matriz *M* se les pondrá como valor predeterminado el número 27, que es el equivalente al espacio vacío o cualquier otro signo que no sea una letra.

```
for (int b = 0; b < colum; b++) {
    for (int a = 0; a < 3; a++) {
        M[a][b]=27;
    }
}
```

Ciclo que sirve para introducir en la matriz *M* los valores que tiene el arreglo tipo *int*. Las condicionales *if* sirven para terminar el ciclo en caso de que se ya se hayan introducido todos los números, en caso de que las condiciones del ciclo *for* no lo hayan determinado anteriormente, sirviendo para evitar que haya errores en el código. La matriz *M* se llena de columna en columna.

```
for (int b = 0; b < colum; b++) {
    for (int a = 0; a < 3; a++) {
        M[a][b]=caracterI[d];
        d++;
        if(d==mb){break;}
    }
    if(d==mb){break;}
}
```

Se imprime la matriz *M* en la consola.

```
System.out.println("El mensaje"+verde+" transformado"+negro+" es:");
for (int a = 0; a < 3; a++) {
    for (int b = 0; b < colum; b++) {
        System.out.print("|\t"+M[a][b]+"\t");
    }
    System.out.print("|\n");
}
System.out.println("\n");
```

Se imprime un mensaje en caso de que durante el proceso se haya producido cualquier tipo de error.

```
}catch(Exception e){
    System.out.println(rojo+e+negro);
}
```

Se inicia el método para ingresar valores a la matriz A. A todos los espacios de la matriz se le introduce el valor 27 por defecto.

```
private static void claveA(){
    for (int a = 0; a < 3; a++) {
        for (int b = 0; b < 3; b++) {
            A[a][b]=27;
        }
    }
```

Se le pide al usuario que ingrese la ruta del archivo *.txt* que tiene la matriz A escrita. El archivo al que haga referencia la ruta que ingrese el usuario se guardara en una variable tipo *File* llamada *archivo*.

```
System.out.println("Ingrese la ruta en la que se encuentra el archivo .txt");
System.out.println(verde+"Ejemplo: C:\\Users\\Diego\\Desktop\\Clave_A.txt"+negro);
File archivo=new File(sc.nextLine());
```

La variable *archivoG* se encarga de almacenar toda la información que tenga el archivo que ingreso el usuario. La variable *lectura* servirá para que el programa pueda leer y manipular lo que tenga escrito el archivo.

```
FileReader archivoG=new FileReader(archivo);
BufferedReader lectura=new BufferedReader(archivoG);
```

Se creo una variable y un arreglo tipo *string* que servirán para almacenar la información que se recolecte del texto mediante la variable *lectura*. Se creo una variable tipo *int* que servirá como contador para la matriz *A* posteriormente.

```
String codigo;
String[] mensaje;
int c=0;
```

El ciclo *while* se encarga de leer el documento línea por línea, se lee una línea por cada repetición, así hasta que el documento ya no tenga más información. La variable *código* almacena todo el texto de la línea que se lea durante la repetición. El texto que tenga la variable se guarda en el arreglo *mensaje*, separando el texto siempre que se encuentre una coma (“,”). Posteriormente el ciclo se empieza un ciclo *for* que servirá para almacenar la información que tenga el arreglo *mensaje* en la matriz *A*. la variable c servirá como contador de la fila que se esta llenando. La matriz se llena de fila en fila.

```
while ((codigo=lectura.readLine())!=null) {
    mensaje=codigo.split(",");
    for (int b = 0; b < 3; b++) {
        A[c][b]=Integer.parseInt(mensaje[b]);
    }
    c++;
}
```

Se imprime la matriz *A.*

```
System.out.println("Matriz "+verde+"A"+negro);
for (int a = 0; a < 3; a++) {
    for (int b = 0; b < 3; b++) {
        System.out.print("|\t"+A[a][b]+"\t");
    }
        System.out.print("|\n");
}
System.out.print("|\n");
```

Se imprime un mensaje en caso de que durante el proceso se haya producido cualquier tipo de error.

```
}catch(Exception e){
    System.out.println(rojo+"ERROR al ingresar el archivo"+negro);
}
```

Se inicia el método para crear la clave *B*, se inicializa la matriz *B* y a tus sus espacios se les pone el número 27 por defecto.

```
private static void claveB(){
    B=new int[3][colum];
        for (int a = 0; a < 3; a++) {
            for (int b = 0; b < colum; b++) {
                B[a][b]=27;
            }
        }
```

Se le pide al usuario que ingrese la ruta del archivo *.txt* que tiene la matriz B escrita y se le da unas indicaciones al usuario sobre el texto que tenga el documento. El archivo al que haga referencia la ruta que ingrese el usuario se guardara en una variable tipo *File* llamada *archivo*.

```
try{
System.out.println("Ingrese la ruta en la que se encuentra el archivo .txt");
System.out.println(verde+"Ejemplo: C:\\Users\\Diego\\Desktop\\Clave_B.txt"+negro);
System.out.println("Asegurese que la cantidad de numeros que tiene el archivo no sea menor de "+(int)mb+
        + "En caso de que no sea así es probable que exista problemas al momento de\n"
        + "encriptar el mensaje");
File archivo=new File(sc.nextLine());
```

La variable *archivoG* se encarga de almacenar toda la información que tenga el archivo que ingreso el usuario. La variable *lectura* servirá para que el programa pueda leer y manipular lo que tenga escrito el archivo.

```
FileReader archivoG=new FileReader(archivo);
BufferedReader lectura=new BufferedReader(archivoG);
```

Se creo una variable y un arreglo tipo *string* que servirán para almacenar la información que se recolecte del texto mediante la variable *lectura*, el arreglo tendrá como capacidad máxima la cantidad de columnas que tenga la matriz *M*. Se creo una variable tipo *int* que servirá como contador para la matriz *A* posteriormente.

```
String codigo;
String[] mensaje=new String[(int)mb];
int c=0;
```

El ciclo *while* se encarga de leer el documento línea por línea, se lee una línea por cada repetición, así hasta que el documento ya no tenga más información. La variable *código* almacena todo el texto de la línea que se lea durante la repetición. El texto que tenga la variable se guarda en el arreglo *mensaje*, separando el texto siempre que se encuentre una coma (“,”). Posteriormente el ciclo se empieza un ciclo *for* que servirá para almacenar la información que tenga el arreglo *mensaje* en la matriz *B*. la variable c servirá como contador de la fila que se está llenando. La matriz se llena de fila en fila.

```
while ((codigo=lectura.readLine())!=null) {
    mensaje=codigo.split(",");
    for (int b = 0; b < colum; b++) {
        B[c][b]=Integer.parseInt(mensaje[b]);
    }
    c++;
}
```

Se imprime la matriz *B.*

```
System.out.println("Matriz "+verde+"B"+negro);
for (int a = 0; a < 3; a++) {
    for (int b = 0; b < colum; b++) {
        System.out.print("|\t"+B[a][b]+"\t");
    }
        System.out.print("|\n");
}
```

Se imprime un mensaje en caso de que durante el proceso se haya producido cualquier tipo de error.

```
}catch(Exception e){
    System.out.println(rojo+"ERROR al ingresar el archivo"+negro);
}
```

Se inicia el método para encriptar el mensaje. Se inicializan dos matrices que servirán para realizar las operaciones con matrices y se crean tres variables que servirán como contadores para las diversas matrices al momento de realizar las operaciones.

```
private static void encriptar(){
    try {
    Ri=new int[3][colum];int Ab=0,Ma,Mb=0;
    Rf=new int[3][colum];
```

Se creo un ciclo dentro de un ciclo que servirá para realizar las operaciones que resultan de la multiplicación de la matriz *A* y la matriz *M*. La matriz *Ri* almacenara el resultado de todas las sumas que se deben hacer para completar la multiplicación. Los contadores sirven para establecer que valores de la matriz se deben incluir en la operación.

```
//A*M
for (int a = 0; a < 3; a++) {
    for (int b = 0; b < colum; b++) {
        Ma=0;
        Ri[a][b]=(A[a][Ab]*M[Ma][Mb])+(A[a][Ab+1]*M[Ma+1][Mb])+(A[a][Ab+2]*M[Ma+2][Mb]);
        Mb++;
    }
    Mb=0;
}
```

Se crea otro ciclo *for* dentro de otro ciclo *for*, estos ciclos servirán para realizar la suma de matrices, el cual es una operación mucho más simple.

```
//A*M + B
for (int a = 0; a < 3; a++) {
    for (int b = 0; b < colum; b++) {
        Rf[a][b]=Ri[a][b]+B[a][b];
    }
}
```

Se imprime el mensaje ya encriptado en línea recta.

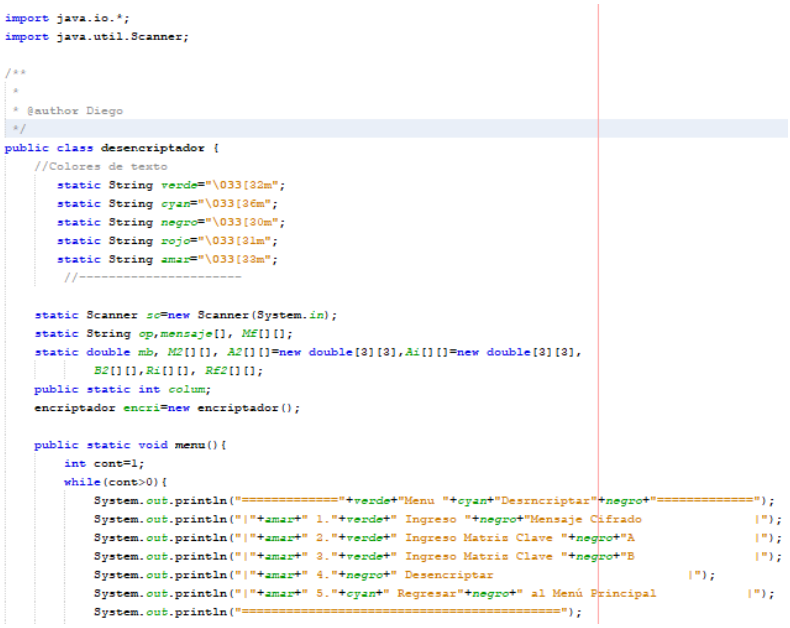
```
System.out.println("==============="+verde+"Menu "+cyan+"Encriptar"+negro+"===============");
System.out.println("| "+verde+"Mensaje Cifrado "+cyan+"es :"+negro+"                    |");
for (int b = 0; b < colum; b++) {
    for (int a = 0; a < 3; a++) {
        System.out.print(" "+Rf[a][b]+" ");
    }
}
```

Se imprime la matriz del mensaje ya encriptado.

```
System.out.println("\n"+"| "+verde+"Matriz del Mensaje Cifrado "+cyan+"es :"+negro+"          |");
for (int a = 0; a < 3; a++) {
    for (int b = 0; b < colum; b++) {
        System.out.print("|\t"+Rf[a][b]+"\t");
    }
    System.out.print("|\n");
}
System.out.println(negro+"=========================================="+"\n");
```

Clase de Desencriptar

Esta clase se encarga de almacenar un mensaje que ingrese el usuario y se encargue



```
import java.io.*;
import java.util.Scanner;

/**
 *
 * @author Diego
 */
public class desencriptador {
    //Colores de texto
       static String verde="\033[32m";
       static String cyan="\033[36m";
       static String negro="\033[30m";
       static String rojo="\033[31m";
       static String amar="\033[33m";
        //----------------------

    static Scanner sc=new Scanner(System.in);
    static String op,mensaje[], Mf[][];
    static double mb, M2[][], A2[][]=new double[3][3],Ai[][]=new double[3][3],
            B2[][],Ri[][], Rf2[][];
    public static int colum;
    encriptador encri=new encriptador();

    public static void menu(){
        int cont=1;
        while(cont>0){
            System.out.println("=============="+verde+"Menu "+cyan+"Desrncriptar"+negro+"==============");
            System.out.println("|"+amar+" 1."+verde+" Ingreso "+negro+"Mensaje Cifrado               |");
            System.out.println("|"+amar+" 2."+verde+" Ingreso Matris Clave "+negro+"A                 |");
            System.out.println("|"+amar+" 3."+verde+" Ingreso Matris Clave "+negro+"B                 |");
            System.out.println("|"+amar+" 4."+negro+" Desencriptar                              |");
            System.out.println("|"+amar+" 5."+cyan+" Regresar"+negro+" al Menú Principal           |");
            System.out.println("=============================================");
```

Primero se importó una librería con funciones tipo escáner que se utilizarán posteriormente. También se importó una librería que entre muchas de sus funciones tiene la capacidad de detectar algún error logístico en programa y evita que este se cierre por culpa de este error.

```
import java.util.Scanner;
import java.io.*;
```

Se creó la variable tipo *Scanner* para leer lo que ingrese el usuario en la consola y una variable tipo *string* que se encargue de manejar esta información. Se crearon un arreglo y una matriz tipo *string* que se encargaran de manejar el mensaje encriptado y desencriptado. Se crearon varias matrices tipo *double* que se utilizaran para realizar las operaciones requeridas. La variable *mb* se encarga de medir la cantidad de dígitos que tiene el mensaje y la variable *colum* se encarga de determinar la cantidad de columnas que debe tener la matriz del mensaje.

```
static Scanner sc=new Scanner(System.in);
static String op,mensaje[], Mf[][];
static double mb, M2[][], A2[][]=new double[3][3],Ai[][]=new double[3][3],
        B2[][],Ri[][], Rf2[][];
public static int colum;
```

Se crea el menú de la misma manera de como se hizo con el menú de inicio.

```
public static void menu(){
    int cont=1;
    while(cont>0){
        System.out.println("============"+verde+"Menu "+cyan+"Desrncriptar"+negro+"============");
        System.out.println("|"+amar+" 1."+verde+" Ingreso "+negro+"Mensaje Cifrado                |");
        System.out.println("|"+amar+" 2."+verde+" Ingreso Matriz Clave "+negro+"A                  |");
        System.out.println("|"+amar+" 3."+verde+" Ingreso Matriz Clave "+negro+"B                  |");
        System.out.println("|"+amar+" 4."+negro+" Desencriptar                              |");
        System.out.println("|"+amar+" 5."+cyan+" Regresar"+negro+" al Menú Principal             |");
        System.out.println("=========================================");
        System.out.print("Ingrese el número de acción que quiere realizar: ");
        op=sc.nextLine();System.out.println();
```

Se creo un *switch* que funciona de la misma manera que el del método principal.

```
switch(op){
    case "1":
        mensaje();
        break;
    case "2":
        claveA();
        break;
    case "3":
        claveB();
        break;
    case "4":
        desencriptar();
        break;
    case "5":
        cont--;
        break;
    default:
            System.out.println(rojo+"ERROR. Asegurese que el dígito ingresado sea 1, 2, 3 o 4"+negro
            break;
}
```

Se creo un método encargado de almacenar el mensaje que ingrese el usuario en una matriz. Primero la variable *op* almacena la información detectada con la variable *sc.* Posteriormente en el arreglo *mensaje* se almacenaron todos los números que ingreso el usuario, separando el mensaje cada vez que encuentre un espacio en blanco.

```
public static void mensaje(){
    try{
    System.out.println(verde+"Ingrese"+negro+" el mensaje cifrado. "+cyan+"Separe "+negro
            + "los números con un espacio en "+cyan+"blanco"+negro+":");
    op=sc.nextLine();
    mensaje=op.split(" ");
```

Se guarda en la variable *mb* cantidad de números que ingreso el usuario. Con este valor se calcula la cantidad de columnas necesarias que debe tener la matriz (la función *Math.ceil* se encarga de aproximar al entero superior el valor de la división. Ya con este valor se inicializo la matriz del mensaje encriptado.

```
mb=mensaje.length;
colum=(int)Math.ceil(mb/3);
M2=new double[3][colum];
```

Se creo una variable tipo *int* que servirá como contador. Se creo un ciclo *for* dentro de otro ciclo *for* para almacenar los valores que tengan el arreglo *mensaje* en la matriz *M2*, llenándola de columna en columna. Los condicionales *if* sirven para terminar los ciclos si ya se alcanzó el límite de dígitos que ingreso el usuario, en caso del que el ciclo *for* no lo haya detectado antes.

```
int c=0;
for (int b = 0; b < colum; b++) {
    for (int a = 0; a < 3; a++) {
        M2[a][b]=Double.parseDouble(mensaje[c]);
        c++;
        if (c==mb) {break;}
    }
        if (c==mb) {break;}
}
```

Se imprimirá la matriz *M2* en la consola.

```
for (int a = 0; a < 3; a++) {
    for (int b = 0; b < colum; b++) {
        System.out.print("|\t"+M2[a][b]+"\t");
    }
    System.out.print("|"+"\n");
}
```

Se imprime un mensaje en caso de que durante el proceso se haya producido cualquier tipo de error.

```
}catch(Exception e){
    System.out.println(rojo+"Asegurese de ingresar numeros enteros"+negro);
}
```

Se inicia el método para ingresar valores a la matriz A. A todos los espacios de la matriz se le introduce el valor 27 por defecto.

```
private static void claveA(){
    for (int a = 0; a < 3; a++) {
        for (int b = 0; b < 3; b++) {
            A[a][b]=27;
        }
    }
```

Se le pide al usuario que ingrese la ruta del archivo *.txt* que tiene la matriz A escrita. El archivo al que haga referencia la ruta que ingrese el usuario se guardara en una variable tipo *File* llamada *archivo*.

```
System.out.println("Ingrese la ruta en la que se encuentra el archivo .txt");
System.out.println(verde+"Ejemplo: C:\\Users\\Diego\\Desktop\\Clave_A.txt"+negro);
File archivo=new File(sc.nextLine());
```

La variable *archivoG* se encarga de almacenar toda la información que tenga el archivo que ingreso el usuario. La variable *lectura* servirá para que el programa pueda leer y manipular lo que tenga escrito el archivo.

```
FileReader archivoG=new FileReader(archivo);
BufferedReader lectura=new BufferedReader(archivoG);
```

Se creo una variable y un arreglo tipo *string* que se encargaran de manejar la información que tiene el archivo al que él usuario hace referencia.

```
String codigo;
String[] mensaje=new String[9];
```

El ciclo *while* se encarga de leer el documento línea por línea, se lee una línea por cada repetición, así hasta que el documento ya no tenga más información. La variable *código* almacena todo el texto de la línea que se lea durante la repetición. El texto que tenga la variable se guarda en el arreglo *mensaje*, separando el texto siempre que se encuentre una coma (“,”). Posteriormente el ciclo se empieza un ciclo *for* que servirá para almacenar la información que tenga el arreglo *mensaje* en la matriz *A*. la variable c servirá como contador de la fila que se esta llenando. La matriz se llena de fila en fila.

```
int c=0;
    while ((codigo=lectura.readLine())!=null) {
        mensaje=codigo.split(",");
        for (int b = 0; b < 3; b++) {
            A2[c][b]=Integer.parseInt(mensaje[b]);
        }
        c++;
    }
```

Se imprime la matriz *A* en la consola y se manda a llamar el método que calcula la inversa de la matriz *A*.

```
System.out.println("Matriz "+verde+"A"+negro);
for (int a = 0; a < 3; a++) {
    for (int b = 0; b < 3; b++) {
        System.out.print("|\t"+A2[a][b]+"\t");
    }
    System.out.print("|"+"\n");
}
System.out.print("\n");
inversa();
```

Se inicia el método *inversa()* y empieza calculando el número complementario del determinante de la matriz *A*.

```
public static void inversa(){
    try {
    double detA2=1/(((A2[0][0]*A2[1][1]*A2[2][2])+(A2[1][0]*A2[2][1]*A2[0][2])+(A2[2][0]*A2[0][1]*A2[1][2]))
            -((A2[0][2]*A2[1][1]*A2[2][0])+(A2[1][2]*A2[2][1]*A2[0][0])+(A2[2][2]*A2[0][1]*A2[1][0])));
```

Se asignaron valores a los espacios de la matriz inversa *Ai* mediante la formula de:

$$A^{-1} = \frac{1}{|A|}(Cofactor(A[x][y]))$$

```
Ai[0][0]=detA2*((A2[1][1]*A2[2][2])-(A2[1][2]*A2[2][1]));
Ai[0][1]=detA2*((A2[0][2]*A2[2][1])-(A2[0][1]*A2[2][2]));
Ai[0][2]=detA2*((A2[0][1]*A2[1][2])-(A2[0][2]*A2[1][1]));
Ai[1][0]=detA2*((A2[1][2]*A2[2][0])-(A2[1][0]*A2[2][2]));
Ai[1][1]=detA2*((A2[0][0]*A2[2][2])-(A2[0][2]*A2[2][0]));
Ai[1][2]=detA2*((A2[0][2]*A2[1][0])-(A2[0][0]*A2[1][2]));
Ai[2][0]=detA2*((A2[1][0]*A2[2][1])-(A2[1][1]*A2[2][0]));
Ai[2][1]=detA2*((A2[0][1]*A2[2][0])-(A2[0][0]*A2[2][1]));
Ai[2][2]=detA2*((A2[0][0]*A2[1][1])-(A2[0][1]*A2[1][0]));
```

Se imprime la matriz inversa *Ai.*

```
System.out.println("Matriz Inversa "+verde+"A"+negro);
    for (int a = 0; a < 3; a++) {
       for (int b = 0; b < 3; b++) {
        System.out.print("|\t"+Ai[a][b]+"\t");
       }
       System.out.print("|"+"\n");
   }
```

Se inicia el método para crear la matriz B, y se inicializa la matriz *B2* poniéndole el número 27 a todos los espacios de la matriz de manera predeterminada.

```
public static void claveB(){
    try{
    B2=new double[3][colum];
        for (int a = 0; a < 3; a++) {
            for (int b = 0; b < colum; b++) {
                B2[a][b]=27;
            }
        }
```

Se le pide al usuario que ingrese la ruta del archivo *.txt* que tiene la matriz B escrita y se le da unas indicaciones al usuario sobre el texto que tenga el documento. El archivo al que haga referencia la ruta que ingrese el usuario se guardara en una variable tipo *File* llamada *archiv*

```
System.out.println("Ingrese la ruta en la que se encuentra el archivo .txt");
System.out.println(verde+"Ejemplo: C:\\Users\\Diego\\Desktop\\Clave_B.txt"+negro);
System.out.println("Asegurese que la cantidad de numeros que tiene el archivo no sea menor de "+(int)mb+".\n"
        + "En caso de que no sea así es probable que exista problemas al momento de\n"
        + "encriptar el mensaje");

File archivo=new File(sc.nextLine());
```

La variable *archivoG* se encarga de almacenar toda la información que tenga el archivo que ingreso el usuario. La variable *lectura* servirá para que el programa pueda leer y manipular lo que tenga escrito el archivo.

```
FileReader archivoG=new FileReader(archivo);
BufferedReader lectura=new BufferedReader(archivoG);
```

Se creo una variable y un arreglo tipo *string* que servirán para almacenar la información que se recolecte del texto mediante la variable *lectura*, el arreglo tendrá como capacidad máxima la cantidad de columnas que tenga la matriz *M2*. Se creo una variable tipo *int* que servirá como contador para la matriz *B2* posteriormente.

```
String codigo;
String[] mensaje=new String[(int)mb];
```

El ciclo *while* se encarga de leer el documento línea por línea, se lee una línea por cada repetición, así hasta que el documento ya no tenga más información. La variable *código* almacena todo el texto de la línea que se lea durante la repetición. El texto que tenga la variable se guarda en el arreglo *mensaje*, separando el texto siempre que se encuentre una coma (“,”). Posteriormente el ciclo se empieza un ciclo *for* que servirá para almacenar la información que tenga el arreglo *mensaje* en la matriz *B*. la variable *c* servirá como contador de la fila que se está llenando. La matriz se llena de fila en fila.

```
int c=0;
    while ((codigo=lectura.readLine())!=null) {
        mensaje=codigo.split(",");
        for (int b = 0; b < colum; b++) {
            B2[c][b]=Integer.parseInt(mensaje[b]);
        }
        c++;
    }
```

Se imprime la matriz *B2.*

```
System.out.println("Matriz "+verde+"B"+negro);
for (int a = 0; a < 3; a++) {
    for (int b = 0; b < colum; b++) {
        System.out.print("|\t"+B2[a][b]+"\t");
    }
        System.out.print("|\n");
}
```

Se inicia el método que se encarga de desencriptar el mensaje. Primero se inicializa la matriz que servirá para realizar la primera operación. Se crean variables que servirán como contadores.

```
blic static void desencriptar(){
  try {
   Ri=new double[3][colum];
   int Ab=0,Ma,Mb=0;
```

Se realiza la resta de la matriz *M2* menos la matriz *B2* y se guardan en la matriz *Ri*.

```
//M-B
for (int a = 0; a < 3; a++) {
   for (int b = 0; b < colum; b++) {
       Ri[a][b]=M2[a][b]-B2[a][b];
   }
}
```

Se inicializa la matriz que servirá para realizar la segunda operación. Y mediante los ciclos *for* se realiza la multiplicación de la matrices *Ai* y *Ri*.

```
//Ai*(M-B)
 Rf2=new double[3][colum];
for (int a = 0; a < 3; a++) {
    for (int b = 0; b < colum; b++) {
        Ma=0;
        Rf2[a][b]=Math.round((Ai[a][Ab]*Ri[Ma][Mb])+(Ai[a][Ab+1]*Ri[Ma+1][Mb])+(Ai[a][Ab+2]*Ri[Ma+2][Mb]));
        Mb++;
    }
    Mb=0;
}
```

Se inicializa la matriz que tendrá almacenado el mensaje desencriptado y transformado en letras. Se crean ciclos *for* que dentro tendrán condicionales *if* que se encargan de cambiar los números por letras.

```
Mf=new String[3][colum];

for (int b = 0; b< colum; b++) {
    for (int a = 0; a < 3; a++) {
        if (Rf2[a][b]==0) {Mf[a][b]="A";}
        else if (Rf2[a][b]==1) {Mf[a][b]="B";}
        else if (Rf2[a][b]==2) {Mf[a][b]="C";}
        else if (Rf2[a][b]==3) {Mf[a][b]="D";}
        else if (Rf2[a][b]==4) {Mf[a][b]="E";}
        else if (Rf2[a][b]==5) {Mf[a][b]="F";}
        else if (Rf2[a][b]==6) {Mf[a][b]="G";}
        else if (Rf2[a][b]==7) {Mf[a][b]="H";}
        else if (Rf2[a][b]==8) {Mf[a][b]="I";}
        else if (Rf2[a][b]==9) {Mf[a][b]="J";}
        else if (Rf2[a][b]==10) {Mf[a][b]="K";}
        else if (Rf2[a][b]==11) {Mf[a][b]="L";}
        else if (Rf2[a][b]==12) {Mf[a][b]="M";}
        else if (Rf2[a][b]==13) {Mf[a][b]="N";}
        else if (Rf2[a][b]==14) {Mf[a][b]="Ñ";}
        else if (Rf2[a][b]==15) {Mf[a][b]="O";}
        else if (Rf2[a][b]==16) {Mf[a][b]="P";}
        else if (Rf2[a][b]==17) {Mf[a][b]="Q";}
        else if (Rf2[a][b]==18) {Mf[a][b]="R";}
        else if (Rf2[a][b]==19) {Mf[a][b]="S";}
        else if (Rf2[a][b]==20) {Mf[a][b]="T";}
        else if (Rf2[a][b]==21) {Mf[a][b]="U";}
```

Se imprime el mensaje desencriptado.

```
System.out.println("============="+verde+"Menu "+cyan+"Desencriptar"+negro+"=============");
System.out.println("| "+verde+"Mensaje Descifrado "+cyan+"es :"+negro+"                    |");
for (int b = 0; b < colum; b++) {
    for (int a = 0; a < 3; a++) {
        System.out.print(Mf[a][b]);
    }
}
System.out.print("\n");
```

Clase de Ataque de Texto Plano

Se creo la variable *filas* que se encargara de almacenar el número de filas que deben tener las matrices. Se crearon varias matrices que servirán para realizar las operaciones necesarias. Y varias variables y un arreglo tipo *String* que servirá para recibir la información de los archivos.

```
static Scanner sc=new Scanner(System.in);
static String op,doc1,doc2;
static int filas=0;
static double M1[][],M2[][],MT[][],MC1[][],MC2[][],Cam;
static int MF1[][]=new int[3][3],MF2[][]=new int[3][3];
static String mensaje[];
```

Se crea el menú de la misma manera de como se hizo con el menú de inicio.

```
static void menu() {
    int cont=1;
    while(cont>0){
        System.out.println("==========="+verde+"Ataque "+negro+"con texto "+cyan+"Plano"+negro+"==========");
        System.out.println("|"+amar+" 1."+verde+" Ingresar Matriz "+negro+"mensaje original      |");
        System.out.println("|"+amar+" 2."+verde+" Ingresar Matriz "+negro+"mensaje cifrado       |");
        System.out.println("|"+amar+" 3."+verde+" Obtener "+negro+"Clave                         |");
        System.out.println("|"+amar+" 4."+cyan+" Regresar"+negro+" al Menú Principal            |");
        System.out.println("=========================================");
        System.out.print("Ingrese el número de acción que quiere realizar: ");
        op=sc.nextLine();System.out.println();
```

Se creo un *switch* que funciona de la misma manera que el del método principal.

```
switch(op){
    case "1":
        matrizO();
        break;
    case "2":
        matrizC();
        break;
    case "3":
        clave();
        break;
    case "4":
        cont--;
        break;
    default:
            System.out.println(rojo+"ERROR. Asegurese que el dígito ingresado sea 1, 2, 3 o 4"+negro);
            break;
}
```

Se inicia el método que se encarga de recibir la matriz del mensaje sin encriptar. Se inicializa la variable que almacenará la cantidad de filas que tendrá la matriz. Se le pide al usuario que ingrese la ruta que hace referencia al archivo con la matriz del mensaje sin encriptar.

```
static void matrizO(){
    try {
        filas=0;
        System.out.println("Ingrese la ruta en la que se encuentra el archivo .txt");
        System.out.println(verde+"Ejemplo: C:\\Users\\Diego\\Desktop\\Mensaje_Prueba.txt"+negro);
```

Se utiliza una variable *string* que se encargara de almacenar la ruta ingresada por el usuario. Se creada una variable *File* para que el programa pueda obtener el texto que guarda el archivo ingresado por el usuario. La variable tipo *FileReader* se utiliza para que el programa pueda leer el texto y la variable *BufferedReader* se encarga de retener la información del texto. Se crea una variable tipo *string* que almacenara el texto del archivo.

```
doc1=sc.nextLine();
File archivo=new File(doc1);
FileReader archivoG=new FileReader(archivo);
BufferedReader lectura=new BufferedReader(archivoG);
String codigo;
```

Se inicia un ciclo *while* con el que se cuenta la cantidad de filas que tiene el texto del archivo.

```
while ((codigo=lectura.readLine())!=null) {
    mensaje=codigo.split(",");
    filas++;
}
```

Una vez se el programa posee la cantidad de filas que tiene el texto se inicializan las matrices que se utilizaran para realizar las operaciones pertinentes. También se crearon dos variables que servirán como contadores.

```
M1=new double[filas][3];M2=new double[filas][3];MT=new double[filas][6];
MC1=new double[filas][3];MC2=new double[filas][3];
int c=0,d=0;
```

Se vuelven a crear una variable *FileReader* y una variable *BufferedReader* que realizan la misma función anteriormente mencionada.

```
FileReader archivoG2=new FileReader(archivo);
BufferedReader lectura2=new BufferedReader(archivoG2);
```

Se crean un ciclo *while* que lee línea por línea el archivo referenciado por el usuario para almacenar la información de la línea en el arreglo *mensaje* mediante la variable *código*, que separa los dígitos siempre que encuentre una coma. Posteriormente el ciclo *for* se encarga de introducir los valores del arreglo en las matrices *M1* (Que servirá como matriz auxiliar) y *MT* (Que es la matriz en la que recaerán todas las ecuaciones principales del proceso para encontrar la clave *A*).

```
while ((codigo=lectura2.readLine())!=null && c<filas) {
    mensaje=codigo.split(",");
    for (int b = 0; b < 3; b++) {
        M1[c][b]=Integer.parseInt(mensaje[b]);
        MT[c][b]=Integer.parseInt(mensaje[b]);
    }
    c++;
}
```

Se imprime la matriz del mensaje sin encriptar.

```
System.out.println("=======Matriz del mensaje"+cyan+" original"+negro+"===============");
System.out.println("-----------------------------------------------");
for (int a = 0; a < filas; a++) {
    for (int b = 0; b < 3; b++) {
        System.out.print("|\t"+M1[a][b]+"\t");
    }
    System.out.println("|\n"+"-----------------------------------------------");
}
```

Se imprime la matriz semicompletada que servirá para calcular la clave *A*.

```
System.out.println("==================================Matriz"+cyan+" total"+negro+"=====================
System.out.println("-----------------------------------------------------------------------------------
for (int a = 0; a < filas; a++) {
    for (int b = 0; b < 3; b++) {
        System.out.print("|\t"+MT[a][b]+"\t");
    }
    System.out.print(":");
    for (int b = 3; b < 6; b++) {
        System.out.print("\t"+MT[a][b]+"\t|");
    }
    System.out.println("\n"+"-----------------------------------------------------------------------------------
}
```

Se le pide al usuario que ingrese la ruta que hace referencia al archivo que tiene la matriz del mensaje encriptado.

```
static void matrizC(){
    try {
        System.out.println("Ingrese la ruta en la que se encuentra el archivo .txt");
        System.out.println(verde+"Ejemplo: C:\\Users\\Diego\\Desktop\\Mensaje_Prueba_Cifrado.txt"+negro);
```

Se utiliza una variable *string* que se encargara de almacenar la ruta ingresada por el usuario. Se creada una variable *File* para que el programa pueda obtener el texto que guarda el archivo ingresado por el usuario. La variable tipo *FileReader* se utiliza para que el programa pueda leer el texto y la variable *BufferedReader* se encarga de retener la información del texto. Se crea una variable tipo *string* que almacenara el texto del archivo.

```
doc2=sc.nextLine();
File archivo=new File(doc2);
FileReader archivoG2=new FileReader(archivo);
BufferedReader lectura2=new BufferedReader(archivoG2);
String codigo;
```

Se crean un ciclo *while* que lee línea por línea el archivo referenciado por el usuario para almacenar la información de la línea en el arreglo *mensaje* mediante la variable *código*, que separa los dígitos siempre que encuentre una coma. Posteriormente el ciclo *for* se encarga de introducir los valores del arreglo en las matrices *M2* (Que servirá como matriz auxiliar) y *MT*.

```
while ((codigo=lectura2.readLine())!=null && c<filas) {
    mensaje=codigo.split(",");
    d=3;
    for (int b = 0; b < 3; b++) {
        M2[c][b]=Integer.parseInt(mensaje[b]);
        MT[c][d]=Integer.parseInt(mensaje[b]);
        d++;
    }
    c++;
}
```

Se imprime la matriz con el mensaje encriptado.

```
System.out.println("=======Matriz del mensaje"+cyan+" cifrado"+negro+"===============");
System.out.println("-------------------------------------------------");
for (int a = 0; a < filas; a++) {
    for (int b = 0; b < 3; b++) {
        System.out.print("|\t"+M2[a][b]+"\t");
    }
    System.out.println("|\n"+"-----------------------------------------------");
}
```

Se imprime la matriz completada que servirá para calcular la clave *A.*

```
System.out.println("================================Matriz"+cyan+" total"+negro+"=====================
System.out.println("-------------------------------------------------------------------------------------
for (int a = 0; a < filas; a++) {
    for (int b = 0; b < 3; b++) {
        System.out.print("|\t"+MT[a][b]+"\t");
    }
    System.out.print(":");
    for (int b = 3; b < 6; b++) {
        System.out.print("\t"+MT[a][b]+"\t|");
    }
    System.out.println("\n"+"-------------------------------------------------------------------------------------
}
```

Se crearon varias variables que servirán como contador y variaos arreglos que servirán para almacenar la información que se almacena en la matirz total. El número que posee cada variable sirve establecer a la columna que pertenecen, los 1 para la columna 1, los 2 para la columna 2 y los 3 para la columna 3, las letras a sirven para manejar las filas, las b las columnas y las c almacenaran el valor del inverso modular 27 del pivote.

```
static int d1[],d2[],d3[];
static int a1=0,a2=0,a3=0,b1=0,b2=0,b3=0,c1=0,c2=0,c3=0;
```

Se inicia el método que se encarga de encontrar la clave *A* se vuelven a poner valor a las variables en caso de que se tenga que reiniciar el proceso. Se inicializan los arreglos.

```
static void clave(){
    a1=0;a2=0;a3=0;b1=0;b2=0;b3=0;c1=0;c2=0;c3=0;
    d1=new int[filas];d2=new int[filas];d3=new int[filas];
```

Se inicia un ciclo *while* que se encarga de realizar el método Gauss-Jordan en la primera columna. El contador que guía al ciclo se cambiara cuando se haya realizado con éxito el método en la primera columna. La condición *if* que se crea al principio del ciclo sirve para que en el caso en que el proceso no se haya podido realizar con éxito el proceso en la columna 1 0 2, debido a que ningún número de esta tenga inverso modular 27, se detendrá el proceso, lo que posteriormente detendrá por completo el meto *clave()*, ya que tampoco se podrá realizar los procesos en las columnas 2 y 3, debido a que sus ciclos *while* tienen la misma condición *if*.

```
try {
    int con=0;
    //En primera columna
    while (con<1) {
        if (a1>=filas || a2>=filas) {break;}
```

Antes de empezar el método Gauss-Jordan se comprueba que el número que se encuentra en la columna actual posea inverso modular 27 mediante la función *moduloinverso* (Se explicara posteriormente) en caso de que no tenga inverso la función devolverá el valor de 0, y el contador *a* de la columna aumentara en *1* lo que significa que se pasara a la siguiente columna y se reiniciara el ciclo while en la nueva fila pivote.

```
if (moduloinverso((int)MT[a1][b1])==0) {
    a1=a1+1;
```

Se le asignara a la variable *c* el valor del inverso modular 27 del número pivote mediante el uso de la función *moduloinverso*. Posteriormente mediante el uso de un ciclo *for* se multiplica toda la fila del pivote por el valor de su inverso modular, y al valor de este producto se le encuentra su valor en modulo 27 mediante la función *modulo* (Que se explicara posteriormente).

```
c1=moduloinverso((int)MT[a1][b1]);
for (int j = 0; j < 6; j++) {
    MT[a1][j]=(double)modulo((int)(MT[a1][j]*c1));
}
```

Mediante un ciclo *for* el arreglo *d* guardara el valor de todos los números de la columna actual.

```
for (int i = 0; i < filas; i++) {
    d1[i]=(int)MT[i][0];
}
```

Se aplicará la fórmula $Fila\# = (Fila\# - (d * FilaPivote))(mod\ 27)$ a todas las filas que tengan mayor numeración que la fila pivote.

```
for (int i = a1+1; i < filas; i++) {
    for (int j = b1; j < 6; j++) {
        MT[i][j]=(double)modulo((int)(MT[i][j]-(d1[i]*MT[a1][j])));
    }
}
```

Se aplicará la fórmula $Fila\# = (Fila\# - (d * FilaPivote))(mod\ 27)$ a todas las filas que tengan menor numeración que la fila pivote. El contador aumentara en 1 concluyendo con el ciclo *while* que permiten reducir los valores de la columna actual y se vuelve a reiniciar la variable *con* para ser utilizado en ciclo while de la columna siguiente.

```
        for (int i = a1-1; i > -1; i--) {
            for (int j = b1; j < 6; j++) {
                MT[i][j]=(double)modulo((int)(MT[i][j]-(d1[i]*MT[a1][j])));
            }
        }
        con++;
    }
}
con=0;
```

El funcionamiento para aplicar el método Gauss-Jordan en las columnas 2 y 3 son prácticamente los mismos, exceptuando unas contadas excepciones que se explicaran.

El valor de *a2* es igual a *a1+1* debido a que para que el método Gauss-Jordan tenga sentido la fila a la que se le aplicara el proceso tiene que ser por lo menos 1 numeración mayor a la fila de la columna 1 que tiene como valor el número 1

```
b2=1;a2=a1+1;
```

Lo mismo aplica para la *a3* que corresponde a la columna 3 y tiene que sea igual a *a2+1*.

```
b3=2;a3=a2+1;
```

Tanto el ciclo *while* de la columna 2 y 3 poseen la condición de detenerse si se a alcanzado el límite de filas que tiene la matriz *MT*.

```
while (con<1) {
    if (a1>=filas || a2>=filas) {break;}
```

El siguiente *if* se llevará a cabo cuando se haya alcanzado el límite de filas en la columna 1 o 2, ya que esto significa que los valores que poseen no tienen inverso modular 27 a partir de cierto punto y puede ser necesario cambiar de lugar las filas de la matriz *MT*. Se explicara su funcionamiento mas a fondo.

```
if (a1>=filas || a2>=filas) {
    for (int i = 0; i < 3; i++) {
        Cam=M1[0][i];
        M1[0][i]=M1[1][i];
        M1[1][i]=Cam;

        Cam=M1[1][i];
        M1[1][i]=M1[filas-1][i];
        M1[filas-1][i]=Cam;

        Cam=M1[filas-1][i];
        M1[filas-1][i]=M1[(int)(filas/2)][i];
        M1[(int)(filas/2)][i]=Cam;
    }

    for (int i = 0; i < 3; i++) {
        Cam=M2[0][i];
        M2[0][i]=M2[1][i];
        M2[1][i]=Cam;

        Cam=M2[1][i];
        M2[1][i]=M2[filas-1][i];
        M2[filas-1][i]=Cam;

        Cam=M2[filas-1][i];
        M2[filas-1][i]=M2[(int)(filas/2)][i];
        M2[(int)(filas/2)][i]=Cam;
    }

    for (int i = 0; i < filas; i++) {
```

Primero se inicia un ciclo *for* que se encargará de cambiar las filas de la primera mitad de la matriz *MT* mediante el uso de la matriz auxiliar *M1*, debido que a después de aplicar Gauss-Jordan a una o dos columnas los valores que tendrá *MT* no serán los originales. Primero se almacenará el valor de la fila 1 en la variable *Cam*, posteriormente la fila 1 obtendrá los valores de la fila 2 y la fila 2 obtendrá los valores de la fila 1.

Posteriormente la variable Cam recibe el valor la fila 2, la fila 2 recibe el valor de la última fila y la última fila recibe el valor de la fila2.

Para terminar la variable Cam recibe el valor la última fila, última fila recibe el valor de la fila de la mitad y la fila de la mitad recibe el valor de la última fila.

```
for (int i = 0; i < 3; i++) {
    Cam=M1[0][i];
    M1[0][i]=M1[1][i];
    M1[1][i]=Cam;

    Cam=M1[1][i];
    M1[1][i]=M1[filas-1][i];
    M1[filas-1][i]=Cam;

    Cam=M1[filas-1][i];
    M1[filas-1][i]=M1[(int)(filas/2)][i];
    M1[(int)(filas/2)][i]=Cam;
}
```

Posteriormente se realiza exactamente el mismo proceso en la segunda mitad de la matriz *MT* utilizando la matriz auxiliar *M2*.

```
for (int i = 0; i < 3; i++) {
    Cam=M2[0][i];
    M2[0][i]=M2[1][i];
    M2[1][i]=Cam;

    Cam=M2[1][i];
    M2[1][i]=M2[filas-1][i];
    M2[filas-1][i]=Cam;

    Cam=M2[filas-1][i];
    M2[filas-1][i]=M2[(int)(filas/2)][i];
    M2[(int)(filas/2)][i]=Cam;
}
```

Se reasignan los valores de las matrices auxiliares en la matriz *MT*.

```
        for (int i = 0; i < filas; i++) {
            for (int j = 0; j < 3; j++) {
                MT[i][j]=M1[i][j];
            }
            int d=0;
            for (int j = 3; j < 6; j++) {
                MT[i][j]=M2[i][d];
                d++;
                }
        }
    }
```

En caso de que el método Gauss-Jordan se haya realizado con éxito el siguiente ciclo while se encarga de eliminar las filas en donde todos sus valores sean igual a 0 en la segunda mitad de la matriz *MT*, mediante el uso de varias variables que sirven como contador. El ciclo analiza la segunda mitad fila por fila, y antes de agregarla a la matriz de la clave *A* analiza los 3 valores de la fila y siempre que encuentre un 0 el *co1* aumentará en 1, si el contador llega a 3 se descartará la fila, se reiniciará *co1* y se pasará a la siguiente fila. Este proceso se repetirá hasta que se hayan analizado todas las filas o que la matriz de la clave *A* se haya llenado, para analizar esto último se encuentra el contador *co2*, que aumenta en 1 siempre que se complete una columna y cuando alcance el valor de 3 se concluirá el ciclo *for* y por consiguiente el ciclo *while*.

```
con=0;
int col=0,co2=0,a4=0,b4=0,b5=3;
while(con<1){
    if (a1>=filas || a2>=filas) {break;}
    for (int i = 0; i < filas; i++) {
        if (co2==3) {break; }
        for (int j = 3; j < 6; j++) {
            if (MT[i][j]==0) {col++;}
        }
            if (col==3) {col=0;}else{
                while(a4<3){
                    col=0;
                    MF1[a4][b4]=(int)MT[i][b5];
                    a4++;
                    b5++;
                }
                co2++;
                a4=0;b5=3;b4++;
            }
    }
    con++;
}
```

En caso de que no haya podido completarse el método Gauss-Jordan se le pedirá al usuario que ingrese el número que reinicie el método para encontrar la clave, pero esto lo hará con las filas de la matriz ya cambiadas

```
if (a1>=filas || a2>=filas) {
        System.out.println(verde+"Vuelva ingresar el número 3 en la consola"+negro);
```

Se crea la matriz transpuesta de la matriz con la clave *A*, esta matriz se creara posteriormente en el reporte que genera el usuario.

```
}else{
    for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
        MF2[i][j]=MF1[j][i];
    }
}
```

Se imprime la matriz de la clave *A.*

```
System.out.println("Matriz Clave "+verde+"A"+negro+"=====================================");
for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
        System.out.print("|\t"+MF1[i][j]+"\t");
    }
    System.out.print("|\n");
}
}
```

Se inicia la función *modulo* que sirve para calcular el modulo 27 del valor que reciba. Primero se calcula el residuo del valor que resulta de dividir el valor que se le ingresa a la función sobre el número 27. Si el valor del residuo es negativo se le sumara el número 27, en caso de que quede positivo se regresa el valor de a%27 sin realizar ningún calculo más.

```
static int modulo(int a){
    int mod=a%27;
    if (mod<0) {
        mod=mod+27;
    }
    return mod;
}
```

Se inicia la función *moduloinverso* que sirve para calcular el inverso modular 27 del número que sirve como pivote en el método Gauss-Jordan. Para calcular el inverso se utiliza el algoritmo *Euclidiano Extendido*.

```
static int moduloinverso(int n){;
                int mod=27,a=0,b=0,a2 = 1, a1 = 0, b2 = 0, b1 = 1,inv=0;
                int q = 0, r = 0;
                double div=mod;

                while(div>0){
                    q = (int)Math.floor(n/div);
                    r = (int)(n%div);
                    a = (int)(a2-q*a1);
                    b = (int)(b2 - q*b1);
                    n =(int) div;
                    div = r;
                    a2 = a1;
                    a1 = a;
                    b2 = b1;
                    b1 = b;
                }
                if(n>1){
                    return 0;
                }else{
                if (a2<0) {
                    inv=(int)(a2+mod);
                }else{
                    inv=(int)(a2);
                }
                }

    return inv;
}
```

Primero se crean varias variables que servirán para realizar los cálculos necesarios. La variable *mod* es igual a 27 ya que el programa se basa en 27 signos que se conviertan en números. La variable *div* recibe el valor de la variable *mod* ya que el algoritmo siempre empieza dividiendo el número al que se le quiere sacar el inverso por el número modular. La variable *n* es igual al número que el usuario quiere conseguir el inverso multiplicativo modular.

```
static int moduloinverso(int n){;
                int mod=27,a=0,b=0,a2 = 1, a1 = 0, b2 = 0, b1 = 1,inv=0;
                int q = 0, r = 0;
                double div=mod;
```

Primero se inicia un ciclo *while* que terminara hasta que el divisor se igual a cero, ya que en ese punto es cuando el algoritmo para. La primera ecuación que se realiza es para obtener del cociente entero de la división $\frac{n}{div}$ y almacenarlo en la variable *q* y la variable *r* recibe el residuo de la misma ecuación.

```
while(div>0){
    q = (int)Math.floor(n/div);
    r = (int)(n%div);
```

Las siguientes operaciones se realizan para obtener los valores que se deben meter en la siguiente vez que se repite el proceso.

```
a = (int)(a2-q*a1);
b = (int)(b2 - q*b1);
```

A continuación, se cambian los valores de las distintas variables, para posteriormente realizar nuevamente el algoritmo. La variable que almacena el valor que se regresara al usuario es la variable *a2*.

```
n =(int) div
div = r;
a2 = a1;
a1 = a;
b2 = b1;
b1 = b;
```

El primer *if* sirve para determinar si el valor que se ingreso a la función tiene o no inverso multiplicativo modular 27, en caso de que no regresara el valor de 0, en caso de que si tenga se realizara la siguiente condicional *if*, ya que todos los números tienen un inverso positivo y uno negativo, es necesario siempre devolver el positivo, y en caso de que el logaritmo devuelva el valor negativo se convertirá a su valor positivo y este último es el que se devolverá en caso de que el logaritmo devuelva la inversa positiva se regresara automáticamente.

```
            if(n>1){
                return 0;
            }else{
            if (a2<0) {
                inv=(int)(a2+mod);
            }else{
                inv=(int)(a2);
            }
            }

return inv;
```

Clase de Reportes

Se creó la clase que se encarga de generar los reportes de todos los procesos que puede realizar los programas. Se creo una clase diferente para cada reporte de cada proceso, por eso mismo primero se manda a llamar a las clases que generan estos reportes.

```
public class reportes {
static reporte_encriptacion RC=new reporte_encriptacion();
static reporte_desencriptado DRC=new reporte_desencriptado();
static reporte_Ataque AC=new reporte_Ataque();
```

Se crea el menú de la misma manera de como se hizo con el menú de inicio.

```
int cont=1;
while(cont>0){
    System.out.println("================"+verde+"Menu "+cyan+"Reporte"+negro+"===============");
    System.out.println("|"+amar+" 1."+verde+" Reporte "+negro+"del Encriptado                |");
    System.out.println("|"+amar+" 2."+verde+" Reporte "+negro+"del Desencriptado             |");
    System.out.println("|"+amar+" 3."+verde+" Reporte "+negro+"Ataque con texto claro        |");
    System.out.println("|"+amar+" 4."+cyan+" Regreso"+negro+" al Menú Principal             |");
    System.out.println("=============================================");
    System.out.print("Ingrese el número de acción que quiere realizar: ");
    op=sc.nextLine();System.out.println();
```

Se utilizara un switch de la misma manera que se creó en el método principal, pero en esta ocasión se mandan a llamar los métodos que generen el documento .html.

```
switch(op){
    case "1":
        RC.reporte();
        break;
    case "2":
        DRC.reporte();
        break;
    case "3":
        AC.reporte();
        break;
    case "4":
        cont--;
        break;
    default:
            System.out.println(rojo+"ERROR. Asegurese que el dígito ingresado sea 1, 2, 3 o 4"+negro)
            break;
}
```

En todas las clases que generan reportes en formato *HTML* se importaron las siguientes librerías java. La librería *Desktop* servirá para poder abrir de manera automática los archivos que se creen, la librería *io* se utiliza para detectar errores y evitar que el programa se cierre, las librerías *Calendar* y *GregorianCalendar* sirven para establecer la fecha y hora en que se generó el reporte.

```
import java.awt.Desktop;
import java.io.*;
import java.util.Calendar;
import java.util.GregorianCalendar;
```

Se inicia la clase para generar el reporte, todas las clases empiezan llamando a la clase que realiza el proceso que se quiere reportar. Se crea una variable tipo *Calendar* que se encarga de almacenar la fecha que tiene el equipo computacional en donde se genera el reporte. Y la variable *I* servirá para poder recibir la información de la clase del proceso que se va a reportar y poder manejarla de manera que nos convenga.

```
public class reporte_encriptacion {

    static encriptador info=new encriptador();
    static Calendar fecha=new GregorianCalendar();
    static String I;
```

Se inicia el método que se encarga de crear el archivo *.html* que poseerá el reporte del procedimiento. Se crea una variablem *Desktop* que servirá para abrir el archivo *html* que se cree. La variable *File* sirve para crear el archivo html y las variables *FileWriter, BufferedWriter* y *PrintWriter* se utilizan para poder escribir en el documento que se acaba de crear.

```
public static void reporte(){
    try{
    Desktop pc=Desktop.getDesktop();
    File f=new File("Reporte_encriptacion.html");
    FileWriter w=new FileWriter(f);
    BufferedWriter bw=new BufferedWriter(w);
    PrintWriter pw=new PrintWriter(bw);
```

Se empieza a escribir en el archivo *.html* mediante la función *write*. Primero se establece el tipo de lenguaje que maneja el documento. Con la etiqueta <*html*> se utiliza para indicar que todo lo que este por delante será texto tipo html. Con la etiqueta <*head*>Se inicia el espacio que servirá para el encabezado de la página web. Con la etiqueta <*title*> se establece para escribir el texto que tendrá la etiqueta de la pagina web en el buscador. <*meta charset=\"utf-8\"/*> sirve para establecer que pueda leer caracteres del idioma español.

```
pw.write("<!DOCTYPE html>\n");
pw.write("<html>\n");
    pw.write("<head>\n");
        pw.write("<title>"); pw.write("Reporte del Proceso de Encriptación"); pw.write("</title>\n");
        pw.write("<meta charset=\"utf-8\"/>\n");
    pw.write("</head>\n");
```

Comienza el cuerpo de la página web, con la etiqueta *h1* se le pone título a la página, mediante la variable *fecha* y la función *get* se obtiene el día, mes, año, hora, minutos y segundos en que se generó el documento.

```
pw.write("<body style =\"text-align: center;\">\n");
    pw.write("<h1 style=\"color:blue;\">Reporte del Proceso de Encriptación</h1>\n");
    pw.write("<h3>Documento Creado el "+fecha.get(Calendar.DAY_OF_MONTH)+
            "/"+((int)fecha.get(Calendar.MONTH)+1)+"/"+fecha.get(Calendar.YEAR)+" a las "
            +fecha.get(Calendar.HOUR)+":"+fecha.get(Calendar.MINUTE)+":"+fecha.get(Calendar.SECOND)+
```

A partir de este punto es cuando el código de las clases que generan los reportes son diferentes.

Se escribe en el documento el mensaje que escribió el usuario para encriptar mando a llamar el arreglo que tiene esta información mediante un ciclo *for*.

```
pw.write("\n<h4>(1) Ingreso del mensaje a Encriptar</h4>\n");
pw.write("<p>El mensaje ingresado por el usuario es: ");
    for (int i = 0; i < info.mb; i++) {I=info.caracterS[i];pw.write(I+"");}
pw.write("</p>\n");
```

Se imprime un listado que muestra el valor numérico que recibe cada letra o signo del mensaje que ingreso el usuario.

```
pw.write("<h4>(2) Conversión de letras a números</h4>\n");
pw.write("<p>");
for (int i = 0; i < info.mb; i++) {
    I=info.caracterS[i];
    pw.write("->"+I+" = ");
    I=info.caracterI[i]+"";
    pw.write(I+"<br>");
}
pw.write("</p>\n");
```

Se crea una tabla en html que mostrara la matriz que se formo con el mensaje que ingreso el usuario, utilizando los ciclos *for* para indicar cuando se debe crear una fila y una columna.

```
pw.write("<h4>(3) Pasar los numeros a la matriz M</h4>\n");
pw.write("<table  align=\"center\" border=\"black 2px solid\">\n");
for (int a = 0; a < 3; a++) {
    pw.write("<tr>\n");
    for (int b = 0; b < info.colum; b++) {
        pw.write("<td>");
            I=info.M[a][b]+"";
            pw.write(I);
        pw.write("</td>\n");
    }
    pw.write("</tr>\n");
}
```

Se realiza el mismo proceso, pero para crear una tabla que muestre la matriz de la clave *A* y *B* que se formaron gracias a los documentos *txt* que ingreso el usuario.

```
pw.write("<h4>(4) Ingresar valores a la matriz A</h4>\n");
pw.write("<h5>Valores ingresados por el usuario:</h5>\n");
pw.write("<table  align=\"center\" border=\"black 2px solid\">\n");
for (int a = 0; a < 3; a++) {
    pw.write("<tr>\n");
    for (int b = 0; b < 3; b++) {
        pw.write("<td>\n");
            I=info.A[a][b]+"";
            pw.write(I);
        pw.write("</td>\n");
    }
    pw.write("</tr>\n");
}
```

```
pw.write("<h4>(5) Ingresar valores a la matriz B</h4>\n");
pw.write("<h5>Valores ingresados por el usuario:</h5>\n");
pw.write("<table  align=\"center\" border=\"black 2px solid\">\n");
for (int a = 0; a < 3; a++) {
    pw.write("<tr>\n");
    for (int b = 0; b < info.colum; b++) {
        pw.write("<td>\n");
            I=info.B[a][b]+"";
            pw.write(I);
        pw.write("</td>\n");
    }
    pw.write("</tr>\n");
}
pw.write("</table>\n");
```

Se escribió en forma de lista las operaciones resultantes de la operación *A*M*.

```
pw.write("<h4>(6) Operación A*M</h4>\n");
int Ab=0,Ma,Mb=0;
for (int a = 0; a < 3; a++) {
    for (int b = 0; b < info.colum; b++) {
     Ma=0;
         pw.write("->Ri["+(a+1)+"]["+(b+1)+"] = ("+info.A[a][Ab]+"*"+info.M[Ma][Mb]+")"
             + " + ("+info.A[a][Ab+1]+"*"+info.M[Ma+1][Mb]+") + ("
             + info.A[a][Ab+2]+"*"+info.M[Ma+2][Mb]+") = "+info.Ri[a][b]+"</br>\n");
     Mb++; }
    Mb=0;}
```

Se escribió en forma de lista las operaciones resultantes de la operación *(A*M)+B*.

```
pw.write("<h4>(7) Operación (A*M)+ B</h4>\n");
for (int a = 0; a < 3; a++) {
    for (int b = 0; b < info.colum; b++) {
        pw.write("->C["+(a+1)+"]["+(b+1)+"] = "+info.Ri[a][b]+"+"+info.B[a][b]+" = "
                +info.Rf[a][b]+"</br>\n");
    }
}
```

Se escribe el mensaje ya encriptado en una sola línea.

```
pw.write("<h4>(8) Resultado del proceso de Encriptación</h4>\n");
pw.write("<p>Mensaje cifrado: ");
for (int b = 0; b < info.colum; b++) {
    for (int a = 0; a < 3; a++) {
            pw.write(info.Rf[a][b]+" ");
    }
}
pw.write("</p>\n");
```

Se crea una tabla en html que mostrara la matriz que se formó con el mensaje ya encriptado.

```
pw.write("<table  align=\"center\" border=\"black 2px solid\">\n");
for (int a = 0; a < 3; a++) {
    pw.write("<tr>\n");
    for (int b = 0; b < info.colum; b++) {
        pw.write("<td>\n");
            I=info.Rf[a][b]+"";
            pw.write(I);
        pw.write("</td>\n");
    }
    pw.write("</tr>\n");
}
pw.write("</table>\n");
```

Se cierra el espacio para escribir el código HTML.

```
    pw.write("</body>\n");
pw.write("</html>");
```

Se cierra el proceso de las variable *pw* y *bw* para indicar que ya no se modificara mas el texto y mediante la función *open* se abrirá de manera automática el documento que se acaba de crear.

```
pw.close();
bw.close();
pc.open(f);
```

```
public class reporte_desencriptado {
```

Se explicará únicamente lo más relevante que encuentre en el espacio entre *<body></body>* Porque todo lo que se encuentre antes o después es prácticamente igual a lo explicado anteriormente.

Se escribe en el documento el mensaje encriptado que escribió el usuario.

```
pw.write("<h4>(1) Ingreso del mensaje a Desencriptar</h4>\n");
pw.write("<p>El mensaje ingresado por el usuario es: ");
    for (int i = 0; i < info.mb; i++) {I=info.mensaje[i];pw.write(I+" ");}
pw.write("</p>\n");
```

Se crea una tabla en html que mostrara la matriz que se formó con el mensaje encriptado escrito por el usuario.

```
pw.write("<h4>(2) Pasar los numeros a la matriz C</h4>\n");
pw.write("<table  align=\"center\" border=\"black 2px solid\">\n");
for (int a = 0; a < 3; a++) {
    pw.write("<tr>\n");
    for (int b = 0; b < info.colum; b++) {
        pw.write("<td>\n");
            I=info.M2[a][b]+"";
            pw.write(I);
        pw.write("</td>\n");
    }
    pw.write("</tr>\n");
}
pw.write("</table>\n");
```

Se realiza el mismo proceso, pero para crear una tabla que muestre la matriz de la clave *A* que se formó gracias a los documentos *txt* que ingreso el usuario.

```
pw.write("<h4>(3) Ingresar valores a la matriz A</h4>\n");
pw.write("<h5>Valores ingresados por el usuario:</h5>\n");
pw.write("<table  align=\"center\" border=\"black 2px solid\">\n");
for (int a = 0; a < 3; a++) {
    pw.write("<tr>\n");
    for (int b = 0; b < 3; b++) {
        pw.write("<td>");
            I=info.A2[a][b]+"";
            pw.write(I);
        pw.write("</td>\n");
    }
    pw.write("</tr>\n");
}
```

Se crea una tabla que muestre la matriz inversa de la clave *A*.

```
pw.write("<h4>(4) Transformación de la matriz A a su inversa</h4>\n");
pw.write("<table  align=\"center\" border=\"black 2px solid\">\n");
for (int a = 0; a < 3; a++) {
    pw.write("<tr>\n");
    for (int b = 0; b < 3; b++) {
        pw.write("<td>");
            I=info.Ai[a][b]+"";
            pw.write(I);
        pw.write("</td>\n");
    }
    pw.write("</tr>\n");
}
pw.write("</table>\n");
```

Se realiza el mismo proceso, pero para crear una tabla que muestre la matriz de la clave *B* que se formó gracias a los documentos *txt* que ingreso el usuario.

```
pw.write("<h4>(5) Ingresar valores a la matriz B</h4>\n");
pw.write("<h5>Valores ingresados por el usuario:</h5>\n");
pw.write("<table  align=\"center\" border=\"black 2px solid\">\n");
for (int a = 0; a < 3; a++) {
    pw.write("<tr>\n");
    for (int b = 0; b < info.colum; b++) {
        pw.write("<td>");
            I=info.B2[a][b]+"";
            pw.write(I);
        pw.write("</td>\n");
    }
    pw.write("</tr>\n");
}
pw.write("</table>\n");
```

Se escribió en forma de lista las operaciones resultantes de la operación *C-B*.

```
pw.write("<h4>(6) Operación C-B</h4>\n");
for (int a = 0; a < 3; a++) {
    for (int b = 0; b < info.colum; b++) {
        pw.write("->(C-B)["+(a+1)+"]["+(b+1)+"] = "+info.M2[a][b]+"-"+info.B2[a][b]+" = "
                +info.Ri[a][b]+"</br>\n");
    }
}
```

Se escribió en forma de lista las operaciones resultantes de la operación *(A*M)-B*.

```
pw.write("<h4>(5) Operación Ai(C-B)</h4>\n");
int Ab=0,Ma,Mb=0;
for (int a = 0; a < 3; a++) {
    for (int b = 0; b < info.colum; b++) {
        Ma=0;
            pw.write("->M["+(a+1)+"]["+(b+1)+"] = ("+info.Ai[a][Ab]+"*"+info.Ri[Ma][Mb]+")"
                + " + ("+info.Ai[a][Ab+1]+"*"+info.Ri[Ma+1][Mb]+") + ("
             + info.Ai[a][Ab+2]+"*"+info.Ri[Ma+2][Mb]+") = "+info.Rf2[a][b]+"</br>\n");
    Mb++; }
Mb=0;}
```

Se muestra el mensaje ya desencriptado.

```
pw.write("<h4>(7) Resultado del proceso de la Desencriptación</h4>\n");
pw.write("<p>Mensaje cifrado: ");
for (int b = 0; b < info.colum; b++) {
    for (int a = 0; a < 3; a++) {
        pw.write(info.Mf[a][b]);
    }
}
pw.write("</p>\n");
```

```
public class reporte_Ataque {
```

Se explicará únicamente lo más relevante que encuentre en el espacio entre *<body></body>* Porque todo lo que se encuentre antes o después es prácticamente igual a lo explicado anteriormente.

Se escribe en el documento la ruta del txt con el mensaje sin encriptar que ingreso el usuario en la consola.

```
pw.write("<h4>(1) Ingreso de texto con el mensaje original convertido en Matriz</h4>\n");
pw.write("<p>Documento ingresado por el usuario: "+info.doc1+"\n");
```

Posteriormente se crea una tabla que muestra la matriz formada gracias a la información proporcionada por el usuario.

```
pw.write("<p>La matriz obtenida por el programa: "+"\n");
pw.write("<table align=\"center\" border=\"1 solid\">\n");
for (int a = 0; a < info.filas; a++) {
    pw.write("<tr>\n");
    for (int b = 0; b < 3; b++) {
        I=(int)(info.M1[a][b])+"";
        pw.write("<td>"+I+"</td>\n");
    }
    pw.write("</tr>\n");
}
pw.write("</table>\n");
```

Se escribe en el documento la ruta del txt con el mensaje encriptado que ingreso el usuario en la consola.

```
pw.write("<h4>(2) Ingreso de texto con el mensaje original cifrado convertido en Matriz</h4>\n")
pw.write("<p>Documento ingresado por el usuario: "+info.doc2+"\n");
```

Posteriormente se crea una tabla que muestra la matriz formada gracias a la información proporcionada por el usuario.

```
pw.write("<p>La matriz obtenida por el programa: "+"\n");
pw.write("<table align=\"center\" border=\"1 solid\">\n");
for (int a = 0; a < info.filas; a++) {
    pw.write("<tr>\n");
    for (int b = 0; b < 3; b++) {
        I=(int)(info.M2[a][b])+"";
        pw.write("<td>"+I+"</td>\n");
    }
    pw.write("</tr>\n");
}
pw.write("</table>\n");
```

Se crea una tabla que muestre la matriz que resulta de combinar las dos matrices que ingreso el ususario y que posteriormente servira para realizar todos los calculos.

```
pw.write("<h4>(3) Matriz resultante</h4>\n");
pw.write("<table align=\"center\" border=\"1 solid\">\n");
for (int a = 0; a < info.filas; a++) {
    pw.write("<tr>\n");
    for (int b = 0; b < 3; b++) {
        I=(int)(info.M1[a][b])+"";
        pw.write("<td>"+I+"</td>\n");
    }
    pw.write("<td>:</td>\n");
    for (int b = 0; b < 3; b++) {
        I=(int)(info.M2[a][b])+"";
        pw.write("<td>"+I+"</td>\n");
    }
    pw.write("</tr>\n");
}
pw.write("</table>\n");
```

Se muestra un listado de todas las operaciones que se hicieron cuando el método Gauss-Jordan se enfocaba en la primera columna.

```
pw.write("<h4>(4) Primera operación realizada en la matriz</h4>\n");
pw.write("<p>Fila #"+(info.a1+1)+" = (Fila #"+(info.a1+1)+"*"+info.c1+")modulo27</p></br>\n");
for (int i = 0; i < info.a1; i++) {
    pw.write("<p>Fila #"+(i+1)+" = (Fila #"+(i+1)+"- "+info.d1[i]+"*Fila #"+(info.a1+1)+")modulo
    }
for (int i = (info.a1+1); i < info.filas; i++) {
    pw.write("<p>Fila #"+(i+1)+" = (Fila #"+(i+1)+"- ("+info.d1[i]+"*Fila #"+(info.a1+1)+"))modu
    }pw.write("</br>\n");
```

Se muestra un listado de todas las operaciones que se hicieron cuando el método Gauss-Jordan se enfocaba en la segunda columna.

```
pw.write("<h4>(5) Segunda operación realizada en la matriz</h4>\n");
pw.write("<p>Fila #"+(info.a2+1)+" = (Fila #"+(info.a2+1)+"*"+info.c2+")modulo27</p></br>\n");
for (int i =0; i < info.a2; i++) {
    pw.write("<p>Fila #"+(i+1)+" = (Fila #"+(i+1)+"- "+info.d2[i]+"*Fila #"+(info.a2+1)+")modulo
    }
for (int i = (info.a2+1); i < info.filas; i++) {
    pw.write("<p>Fila #"+(i+1)+" = (Fila #"+(i+1)+"- ("+info.d2[i]+"*Fila #"+(info.a2+1)+"))modu
    }pw.write("</br>\n");
```

Se muestra un listado de todas las operaciones que se hicieron cuando el método Gauss-Jordan se enfocaba en la tercera columna.

```
pw.write("<h4>(6) Tercera operación realizada en la matriz</h4>\n");
pw.write("<p>Fila #"+(info.a3+1)+" = (Fila #"+(info.a3+1)+"*"+info.c3+")modulo27</p></br>\n");
for (int i = 0; i < info.a3; i++) {
    pw.write("<p>Fila #"+(i+1)+" = (Fila #"+(i+1)+"- "+info.d3[i]+"*Fila #"+(info.a3+1)+")modulo
    }
for (int i = (info.a3+1); i < info.filas; i++) {
    pw.write("<p>Fila #"+(i+1)+" = (Fila #"+(i+1)+"- ("+info.d3[i]+"*Fila #"+(info.a3+1)+"))modu
    }pw.write("</br>\n");
```

Se crea una tabla que muestra la matriz principal después de haber realizado el método Gauss-Jordan.

```
pw.write("<h4>(7) Matriz tras las operaciones</h4>\n");
pw.write("<table align=\"center\" border=\"1 solid\">\n");
    for (int i = 0; i < info.filas; i++) {
        pw.write("<tr>\n");
        for (int j = 0; j < 3; j++) {
            I=(int)(info.MT[i][j])+"";
            pw.write("<td>");
            pw.write(I);
            pw.write("</td>\n");
        }
        pw.write("<td>:</td>\n");
        for (int j = 3; j < 6; j++) {
            I=(int)(info.MT[i][j])+"";
            pw.write("<td>");
            pw.write(I);
            pw.write("</td>\n");
        }
        pw.write("</tr>\n");
    }
pw.write("</table>\n");
```

Se crea una tabla que muestra la versión reducida de la matriz principal.

```
pw.write("<h4>(8)Matriz Resultaten tras reducir el lado derecho de la matriz calculada</h4>\n");
 pw.write("<table align=\"center\" border=\"1 solid\">\n");
 for (int i = 0; i < 3; i++) {
     pw.write("<tr>\n");
     for (int j = 0; j < 3; j++) {
         pw.write("<td>");
         I=info.MF2[i][j]+"";
         pw.write(I);
         pw.write("</td>\n");
     }
     pw.write("</tr>\n");
 }
 pw.write("</table>\n");
```

Se crea una tabla que muestre la *clave A* calculada.

```
            pw.write("<h4>(9)Matriz Clave A</h4>\n");
            pw.write("<table align=\"center\" border=\"1 solid\">\n");
            for (int i = 0; i < 3; i++) {
                pw.write("<tr>\n");
                for (int j = 0; j < 3; j++) {
                    pw.write("<td>");
                    I=info.MF1[i][j]+"";
                    pw.write(I);
                    pw.write("</td>\n");
                }
                pw.write("</tr>\n");
            }
            pw.write("</table>\n");

        pw.write("</body>\n");
    pw.write("</html>");
```