Manual Técnico

1. Importar clases y definir variables globales

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package practica1;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.time.LocalDateTime;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author dennis
 */
public class Practica1 {


    static int arrMsg[][];
    static int arrClaveA[][];
    static int arrClaveB[][];
    static int C[][];
    static int adjun[][];
    static float inv[][];
    static int multi[][];
```

Todas las clases que se logran ver en esta parte del código fueron las utilizadas para poder realizar opciones especificas en medida del objetivo que tenía la aplicación, en este caso se necesitaba leer archivos, generar archivos, controlar excepciones entre otras.
Y las variables globales que se ven, son las necesarias para el correcto funcionamiento del programa.

2. Final de variables globales e inicio de clase main

Se observan las ultimas variables globales necesarias para el código y para ciertas actividades específicas y se evidencia que en el main se trata de mantener una ética en relación a la limpieza de este por lo cual solo se dejan los menús y el switch que nos servirá para ejecutar ciertos bloques de código dependiendo de la entrada del usuario. El ciclo while en esta zona del código sirve para controlar la entrada del usuario.

```java
    static float inv[][];
    static int multi[][];
    static int resta[][];
    static int desencriptado[][];
    public static int columnas;
    public static int fil_M;
    public static int col_M;
    public static int fil_A;
    public static int col_A;
    public static int deter;
    public static String mensaje;

    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

//        Menu
        System.out.println("============ MENU ============");
        System.out.println("| 1. Encriptar               |");
        System.out.println("| 2. Desencriptar            |");
        System.out.println("| 3. Generar Reportes        |");
        System.out.println("| 4. Salir                   |");
        System.out.println("==============================");
        System.out.println("Ingrese una opcion: ");

        int opcion = Integer.parseInt(br.readLine());
        while(opcion < 1 || opcion > 4){
            System.out.println("Ingrese una opcion valida");
            opcion = Integer.parseInt(br.readLine());
        }

//        Opciones
          switch(opcion){
              case 1:
                  while(opcion != 5){
```

3. Visualización del contenido del switch.

```
//        Opciones
            switch(opcion){
                case 1:
                    while(opcion != 5){
                    System.out.println("======== MENU ENCRIPTAR ========");
                    System.out.println("| 1. Ingreso Mensaje          |");
                    System.out.println("| 2. Ingreso Matriz Clave A   |");
                    System.out.println("| 3. Ingreso Matriz Clave B   |");
                    System.out.println("| 4. Encriptar                |");
                    System.out.println("| 5. Regresar                 |");
                    System.out.println("===============================");

                    encriptar();
                    }
                    break;
                case 2:
                    desencriptar();
                    break;
                case 3:
                    reportes();
                    break;
                case 4:
                    System.exit(0);
                    break;
            }
    }

    public static void encriptar(){
        try {
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

            System.out.println("Ingrese la opcion");
            int opcionEnc = Integer.parseInt(br.readLine());
            while(opcionEnc < 1 || opcionEnc > 5){
```

Este switch sirve para determinar que funciones entraran a funcionar dependiendo la entrada del usuario y se observa un ciclo while dentro de la opción 1, esto sirve para que el usuario pueda continuar ingresando datos ya que esa opción en especial necesita almacenar ciertos datos en memoria y seria tedioso salir al menú principal e ingresando nuevamente al submenú.
Se observa el inicio de la función encriptar de la cual se ampliará en lo siguiente.

4. Función encriptar

En esta función como su nombre lo indica se busca encriptar por medio de matrices la entrada el usuario.

Se obtiene la entrada del usuario y se determina el número de columnas que ocupara en una matriz 3xn posteriormente se crean dos arreglos, uno de tipo carácter que almacenara los caracteres de la entrada del usuario y el otro de tipo entero que almacenara la posición en relación al alfabeto que iniciara en la posición 0 para A y en la posición 26 para Z otorgando el valor de 27 a caracteres desconocidos.

Se utiliza un ciclo for para poder recorrer las posiciones de los caracteres y a su vez almacenarlos en el vector de números, se definió unas variables que contiene en su valor el carácter “a” ya que este tiene un valor de 97 de forma hexadecimal partiendo de ese análisis para poder determinar las posiciones en relación al alfabeto que ocupara cada letra y se manejaron casos especiales como la “ñ”.

```
            System.out.println("Ingrese la opcion");
            int opcionEnc = Integer.parseInt(br.readLine());
            while(opcionEnc < 1 || opcionEnc > 5){
                System.out.println("Ingrese una opcion correcta");
                opcionEnc = Integer.parseInt(br.readLine());
            }

            switch(opcionEnc){
                case 1:
                    System.out.println("Escriba el mensaje");
                    mensaje = br.readLine();

//  Determina el numero de columnas que ocupara el mensaje
                    int filas = 3;
                    columnas = mensaje.length();

                    for(int i = 0;  i < columnas; i++){
                        if(columnas % 3 != 0){
                            columnas += 1;
                            mensaje += " ";
                        }
                    }
                    columnas = columnas/3;

                    mensaje = mensaje.toLowerCase();
                    char posiciones[] = new char[mensaje.length()];
                    int number [] = new int[mensaje.length()];

//  SE CONVIERTE EL MENSAJE EN UN ARREGLO CHAR Y SE DETERMINA QUE LETRA DEL ALFABETO ES
                    for (int i = 0; i < mensaje.length(); i++) {
                        posiciones[i] = mensaje.charAt(i);
                        char primerC = 'a';
                        number[i] = posiciones[i] - primerC;
//  SE CONVIERTE EL MENSAJE EN UN ARREGLO CHAR Y SE DETERMINA QUE LETRA DEL ALFABETO ES
                    for (int i = 0; i < mensaje.length(); i++) {
                        posiciones[i] = mensaje.charAt(i);
                        char primerC = 'a';
                        number[i] = posiciones[i] - primerC;
                        if(posiciones[i] == 'ñ'){
                            number[i] = 14;
                        }
                        if(number[i] >= 14){
                            number[i] +=1;
                        }
                        if(posiciones[i] < 'a' || posiciones[i] > 'z'){
                            number[i] = 27;
                        }
                    }
                    System.out.println();
//  SE CREA UN ARREGLO QUE CONTENDRA LAS POSICIONES DEL MENSAJE Y SE RECORRE
                    int index = 0;
                    arrMsg = new int [filas][columnas];
                    for (int i = 0; i < columnas; i++) {
                        for (int j = 0; j < 3; j++) {
                            arrMsg[j][i] = number[index];
                            index++;
                        }
                    }

//  Determina el tamano de filas y columnas del arreglo
                    fil_M = arrMsg.length;
                    col_M = arrMsg[0].length;
                    break;
                case 2:
                    System.out.println("Ingrese la direccion del archivo: ");
                    String pathFile = br.readLine();
```

5. Encriptar Pt2.

```
//                  Metodos para leer archivos, cada uno necesita del otro
                        FileReader fr = new FileReader(pathFile);
                        BufferedReader Bfread = new BufferedReader(fr);
                        Scanner scan = new Scanner(Bfread);

//                  Variables para recorrer y almacenar archivos en arreglo
                        int fila = 0;
                        String lineaEntrada = "";
                        arrClaveA = new int [3][3];

//                  Ciclo que observa que existan datos, los almacena en un arreglo
//                  Temporal y los almacena de manera fija en un arreglo 2D
                        try{
                            while(scan.hasNextLine()){
                                lineaEntrada = scan.nextLine();
                                String []arrEntrada = lineaEntrada.split(",");

                                for (int i = 0; i < 3; i++) {
                                    arrClaveA[fila][i] = Integer.parseInt(arrEntrada[i]);
                                }
                                fila++;
                            }
                        }catch(Exception e){
                            System.out.println(e);
                        }
//  Determina el numero de filas y columnas de matriz A
                        fil_A = arrClaveA.length;
                        col_A = arrClaveA[0].length;
                        break;
                    case 3:
                        System.out.println("Ingrese la direccion del archivo: ");
                        String pathFile1 = br.readLine();
```

El código de encriptar también funciona en base a un switch debido a que tendrá múltiples entradas del usuario, luego de convertir el texto a un vector y por medio de ciclos for convertirlo a un arreglo 2D, se procede a solicitar las ubicaciones de los archivos de texto que contendrán las matrices para encriptar el mensaje.

6. Encriptar Pt3

```
//  Metodos para leer archivos
                    FileReader fre = new FileReader(pathFile1);
                    BufferedReader Bfreader = new BufferedReader(fre);
                    Scanner scan1 = new Scanner(Bfreader);

//  variables para recorrer y almacenar archivos en un arreglo
                    int colB = 0;
                    String lineaEntrada1 = "";
                    colB = columnas;
                    lineaEntrada1 = "";
                    int f = 0;
                    arrClaveB = new int[3][colB];
                    try{
                        while(scan1.hasNextLine()){
                            lineaEntrada1 = scan1.nextLine();
                            String []a = lineaEntrada1.split(",");
                            for (int i = 0; i < 3; i++) {
                                arrClaveB[i][f] = Integer.parseInt(a[i]);
                            }
                            f++;
                        }
                    }catch(Exception e){
                        System.out.println(e);
                    }
                    break;
                case 4:
                    System.out.println("El mensaje Encriptado es: ");

                    int sum = 0;
                    multi = new int[fil_A][col_M];

                    for (int i = 0; i < 3; i++) {       //filas
                        for (int j = 0; j < col_M; j++) {   //columnas
```

Una vez ingresada la ubicación de la matriz A y almacenada en su respectivo arreglo 2D se procede a realizar lo mismo con la matriz B a diferencia que el número de columnas de esta matriz estará dado por el número de columnas de la matriz M.
Y se inicia con el proceso de encriptación.

7. Encriptar Pt4

```
                        for (int i = 0; i < 3; i++) {        //filas
                            for (int j = 0; j < col_M; j++) {    //columnas
                                for (int k = 0; k < 3; k++) {
                                    sum = sum + arrClaveA[i][k] * arrMsg[k][j];
                                }
                                multi[i][j] = sum;
                                sum = 0;
                            }
                        }

                        C = new int[multi.length][ multi[0].length];

                        for (int i = 0; i < multi.length; i++) {
                            for (int j = 0; j < multi[0].length; j++) {
                                C[i][j] = multi[i][j] + arrClaveB[i][j];
                                System.out.print(C[i][j] + " ");
                            }
                        }
                        System.out.println("\n");
                        break;
                    case 5:
                        main(null);
                        break;
                }
            } catch (IOException ex) {
                Logger.getLogger(Practica1.class.getName()).log(Level.SEVERE, null, ex);
            }
        }

//     Funciones para desenciptar

        public static void desencriptar() throws IOException{
            BufferedReader read = new BufferedReader(new InputStreamReader(System.in));
```

Para encriptar la entrada del usuario se realizan operación con matrices, específicamente multiplicación y suma respetando los criterios que rigen estas operaciones dando como resultado la matriz del mensaje encriptado.

8. Desencriptar

```
            System.out.println("El mensaje descifrado es: ");

            resta = new int[C.length][C[0].length];
            for (int i = 0; i < C.length; i++) {
                for (int j = 0; j < C[0].length; j++) {
                    resta[i][j] = C[i][j] - arrClaveB[i][j];
                }
            }

            int n = 3;
            adjun = new int[n][n];
            inv = new float[n][n];

            deter = determinante(arrClaveA, n);
            determinante(arrClaveA, n);
            adjunta(arrClaveA, adjun, n);
            inversa(arrClaveA, inv, n);

            float sum = 1;
            desencriptado = new int [arrMsg.length][arrMsg[0].length];
            for (int i = 0; i < resta.length; i++) {       //filas
                for (int j = 0; j < resta[0].length; j++) {   //columnas
                    for (int k = 0; k < 3; k++) {
                        sum = sum + (inv[i][k] * resta[k][j]);
                    }
                    desencriptado[i][j] = (int)sum;
                    sum = 0;
                }
            }

            int index = 0;
            int numbers[] = new int[(desencriptado.length * desencriptado[0].length)];
```

Para el proceso de desencriptación s realiza la ardua labor de múltiples operaciones con matrices debido a que se necesita encontrar la matriz adjunta de la matriz A y la matriz inversa al igual que el determinante de dicha matriz todas estas operaciones están aclaradas por medio de comentarios de su cómo es su funcionamiento.

Siguiente página funciones utilizadas al desencriptar:

```
252
253             System.out.println("El mensaje descifrado es: ");
254
255             resta = new int[C.length][C[0].length];
256             for (int i = 0; i < C.length; i++) {
257                 for (int j = 0; j < C[0].length; j++) {
258                     resta[i][j] = C[i][j] - arrClaveB[i][j];
259                 }
260             }
261
262             int n = 3;
263             adjun = new int[n][n];
264             inv = new float[n][n];
265
266             deter = determinante(arrClaveA, n);
267             determinante(arrClaveA, n);
268             adjunta(arrClaveA, adjun, n);
269             inversa(arrClaveA, inv, n);
270
271             float sum = 1;
272             desencriptado = new int [arrMsg.length][arrMsg[0].length];
273             for (int i = 0; i < resta.length; i++) {        //filas
274                 for (int j = 0; j < resta[0].length; j++) {    //columnas
275                     for (int k = 0; k < 3; k++) {
276                         sum = sum + (inv[i][k] * resta[k][j]);
277                     }
278                     desencriptado[i][j] = (int)sum;
279                     sum = 0;
280                 }
281             }
282
283             int index = 0;
284             int numbers[] = new int[(desencriptado.length * desencriptado[0].length)];
```

```
284             int numbers[] = new int[(desencriptado.length * desencriptado[0].length)];
285             for (int i = 0; i < desencriptado[0].length; i++) {
286                 for (int j = 0; j < desencriptado.length; j++) {
287                     numbers[index] = desencriptado[j][i];
288                     index = (index + 1) % numbers.length;
289                 }
290             }
291
292             char pos[] = new char[numbers.length];
293             for (int i = 0; i < numbers.length; i++) {
294                 char pC = 'a';
295                 pos[i] = (char) ((char)numbers[i] + pC);
296
297                 if (numbers[i] == 27) {
298                     pos[i] = 32;
299                 }else if(numbers[i] == 14){
300                     pos[i] = 164;
301                 }else if(numbers[i] >= 14){
302                     pos[i] -= 1;
303                 }
304             }
305             for (int i = 0; i < pos.length; i++) {
306                 System.out.print(pos[i]);
307             }
308             System.out.println();
309             main(null);
310     }
311             static void obtenerCof(int A[][], int temporal[][], int q, int p, int n){
312             int i = 0, j = 0;
313
314     //Ciclo que recorre cada elemento de la matriz
315             for (int fila = 0; fila < n; fila++) {
316                 for (int col = 0; col < n; col++) {
317     //Se copia en una matriz temporal los datos que no estan dados por las fila y col
```

```
317     //Se copia en una matriz temporal los datos que no estan dados por las fila y col
318                     if(fila != p && col != q){
319                         temporal[i][j++] = A[fila][col];
320
321
322     //Si la fila esta llena, entonces el index fila incrementa su valor y col se vuelve cero
323                         if(j == n - 1){
324                             j = 0;
325                             i++;
326                         }
327
328                     }
329                 }
330             }
331         }
332
333         public static int determinante(int A[][], int N){
334             int d = 0;
335             if(N == 1){
336                 d = A[0][0];
337             }else if(N == 2){
338                 d = A[0][0]*A[1][1] - A[1][0]*A[0][1];
339             }else{
340                 int [][] matrizNueva = new int[N-1][];
341                 d = 0;
342
343                 for (int saltarCol = 0; saltarCol < N; saltarCol++) {
344         //SaltarCol es el numero de columna que quedara fuera del determinante mas pequeno dividiendo la
345         //matriz(De teoria de matrices se sabe que si es mayor a 2x2 se crean matrices 2x2)
346                     for (int k = 0; k < (N-1); k++) {
347                         matrizNueva[k] = new int [N-1];
348                     }
349                     for (int i = 1; i < N; i++) {
350                         int saltarFila = 0;
```

```
350                         int saltarFila = 0;
351                         for (int j = 0; j < N; j++) {
352                             if(j == saltarCol){
353                                 continue;
354                             }
355                             matrizNueva[i-1][saltarFila] = A[i][j];
356                             saltarFila++;
357                         }
358                     }
359         //Fin de dividir matriz
360
361         //Calcular el determinante
362                     int potenciaNegativaPositiva;
363                     if(saltarCol % 2 == 0){
364                         potenciaNegativaPositiva = 1;
365                     }else {
366                         potenciaNegativaPositiva = -1;
367                     }
368
369                     d += potenciaNegativaPositiva * A[0][saltarCol] * determinante(matrizNueva,N-1);
370                 }
371             }
372             return d;
373         }
374
375         static void adjunta(int A[][], int [][]adj, int N){
376             if(N == 1){
377                 adj[0][0] = 1;
378                 return;
379             }
380     //Matriz temp es usada para almacenar los cofactores de A[][]
381             int signo = 1;
382             int [][]temp = new int[N][N];
383
```

```
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < N; j++) {
                //Obtener cofactor de A[i][j]
                obtenerCof(A, temp, i, j, N);

                //Signo de la adjunta adj[j][i] positivo si la suma de la fila y columna tienen index par
                signo = ((i + j) % 2 == 0)? 1: -1;

                //Intercambiando filas y columnas para obtener la matriz transpuesta de la de cofactores
                adj[j][i] = (signo) * (determinante(temp, N-1));
            }
        }
    }

//Funcion para calcular y almacenar la inversa, devuelve falso si el determinante es cero
    static boolean inversa(int A[][], float [][]inversa, int N){
        //Encontrar el determinante de A[][]
        int det = determinante(A, N);
        if(det == 0){
            System.out.println("Matriz singular, no se puede encontrar su inversa");
            return false;
        }
        //Encontrar la adjunta
        int [][]adj = new int[N][N];
        adjunta(A, adj, N);

        //Encontrar la inversa se usa la forma "Inversa(A) = adjunta(A) / determinante(A)"
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < N; j++) {
                inversa[j][i] = adj[i][j]/(float)det;
            }
        }
        return true;
```

9. Generar Reportes

```
416             return true;
417         }
418
419     //    Generar reportes
420         public static void reportes() throws IOException{
421             System.out.println("CREAR ARCHIVO");
422             File file = new File("Reporte.html");
                FileWriter fw = null;
424
425             if(!file.exists()){
426                 file.createNewFile();
427             }
428
                PrintWriter pw = new PrintWriter(file);
430             pw.println("<style>"
431                     + "body{"
432                     + "background-color: skyblue; "
433                     + "font-family: Arial;"
434                     + "}h1{text-align: center}"
435                     + "</style>");
436             pw.println("<h1>===========REPORTES===========</h1>");
437             pw.println("<h2>Reporte Encriptar</h2>");
438             pw.println("<h3>Texto a Encriptar</h3>");
439             pw.println(mensaje);
440             pw.println("<h3>Matriz del Mensaje</h3>");
441             for (int i = 0; i < fil_M; i++) {
442                 for (int j = 0; j < col_M; j++) {
443                     pw.println(arrMsg[i][j]);
444                 }
445             }
446             pw.println("<h3>Matriz del Clave A</h3>");
447             for (int i = 0; i < 3; i++) {
448                 for (int j = 0; j < 3; j++) {
449                     pw.println(arrClaveA[i][j]);
```

Para poder generar reportes se utilizaron ciertos constructores que permitían crear y escribir sobre archivos y se utilizo la sintaxis de HTML para poder generarlos.