MANUAL DE USUARIO

PORCIONES DE CÓDIGO

CLASE “ENCRIPTACION”, MÉTODO “MENSAJE”

EL FRAGMENTO DE CÓDIGO TIENE COMO FUNCIONALIDAD LEER DATOS QUE SON INGRESADOS MANUALMENTE A TRAVÉS DEL TECLADO, EN EL PARTICULAR CASO UNA CADENA DE CARACTERES QUE FORMARAN UN MENSAJE. EL ARREGLO DE TIPO CHAR LLAMADO “HOJA” CONVIERTE LA CADENA DE CARACTERES, ES DECIR EL MENSAJE QUE INGRESEMOS, Y CADA UNA DE LAS LETRAS DE DICHO MENSAJE OCUPARÁN UNA POSICIÓN EN EL ARREGLO UNIDIMENSIONAL O VECTOR. TAMBIÉN SE LE DESIGNA LA LONGITUD DEL VECTOR, ES DECIR, LA CANTIDAD DE LETRAS QUE TIENE A UNA VARIABLE LLAMADA “COLUMNAS”.

```
Scanner sc = new Scanner(System.in);
System.out.print("Ingrese el mensaje: ");
mensaje = sc.nextLine();//LEE LA CADENA INGRESADA EN EL SISTEMA
char[] hoja = mensaje.toCharArray();//CADENA A VECTOR
columnas = hoja.length;//LONGITUD DEL VECTOR
```

DEBIDO A QUE EL MENSAJE DEBÍA ESTAR CONTENIDO EN UNA MATRIZ DE 3XN, SU LONGITUD DEBERÍA SER MÚLTIPLO DE 3, ENTONCES SI SE DA EL CASO DE QUE LA CANTIDAD DE CARACTERES QUE TUVIERA EL MENSAJE NO CUMPLIERA ESA CONDICIÓN DEBERÍA HACERSE UN INCREMENTO HASTA QUE CUMPLA DICHA CONDICIÓN, POR LO QUE SE ASIGNÓ LA VARIABLE “COLUMNAS EN EL FRAGMENTO ANTERIOR. MIENTRAS QUE LA DIVISIÓN DE “COLUMNAS” ENTRE 3 NO FUERA EXACTA, SE AUMENTARÍA UNA UNIDAD A SU VALOR HASTA QUE SE CUMPLIERA LA CONDICIÓN.

```
while((columnas % 3) != 0) {
    columnas++;
}
```

PARA QUE EL VECTOR SE ALMACENARA EN LA MATRIZ, SE DIVIDIÓ EL VALOR DE “COLUMNAS” ENTRE 3 DE TAL MODO QUE SE CUMPLA QUE EL ARREGLO FUESE DE 3XN. POR ÚLTIMO, SE RECORRE LA MATRIZ COLUMNA A COLUMNA INSERTANDO CADA CARÁCTER DEL VECTOR DEL MENSAJE EN UNA POSICIÓN DE LA MATRIZ, CONVIRTIENDO CADA CARÁCTER EN UNA CADENA. CUANDO SE TERMINARA DE INSERTAR CADA UNO DE LOS ELEMENTOS DEL VECTOR DENTRO DE LA MATRIZ, PERO AÚN HAY ESPACIOS DISPONIBLES FUERON LLENADOS CON ESPACIOS POR MEDIO DE TRY CATCH.

```
sobre = new String[3][columnas/3];
codificado = new int[3][columnas/3];
//RECORRE EL ARREGLO
for(int j=0; j<columnas/3; j++) {
    for(int i=0; i<3; i++) {
        try {
            //VECTOR A MATRIZ
            if(hoja[letra]==32) {
                sobre[i][j] = " ";
            }else {
                sobre[i][j] = String.valueOf(hoja[letra]);
            }
        }catch(Exception e) {
            //RELLENA POSICIONES ADICIONADAS
            sobre[i][j] = " ";
        }
        letra++;
    }
}
```

SE HACE UN RECORRIDO POR EL ARREGLO DE TAL MODO QUE SE COMPARARA SI CONTENÍA UNA LETRA SE ESCRIBIERA SU RESPECTIVO NÚMERO EN OTRA MATRIZ DE TIPO ENTER, SEGÚN EL CÓDIGO DADO.

```
for(int i=0; i<3; i++) {
    for(int j=0; j<columnas/3; j++) {
        //CODIFICA LETRAS A NÚMEROS
        if(sobre[i][j].equals("A") || sobre[i][j].equals("a")) {
            codificado[i][j] = 0;
        }else if(sobre[i][j].equals("B") || sobre[i][j].equals("b")) {
...
        }else if(sobre[i][j].equals("Z") || sobre[i][j].equals("z")) {
            codificado[i][j] = 26;
        }else {
            codificado[i][j] = 27;
        }
    }
}
```

CLASE “ENCRIPTACION”, MÉTODO “CLAVEA” Y “CLAVEB”

CLAVEA Y CLAVEB REALIZAN LOS MISMOS PROCEDIMIENTOS.

EN ESTE SEGMENTO SE LE ENVIA EL PARÁMETRO QUE CORRESPONDE A LA RUTA O UBICACIÓN EN LA QUE SE ENCUENTRA EL ARCHIVO QUE LEERÁ EL PROGRAMA, CORRESPONDIENTE A MATRICES.

SEGUIDAMENTE SE ENTRA A UN CICLO EN EL QUE EL SCANNER LEERÁ LÍNEA POR LÍNEA HASTA LLEGAR A LA ÚLTIMA, ES IMPORTANTE QUE EN EL CICLO SE DEN SALTOS DE LÍNA PORQUE DE LO CONTRARIO LAS DIMENSIONES DE LAS MATRICES SERÁN ERRÓNEAS.

```
File archA = new File(ruta);
Scanner leerA = new Scanner(archA);
data="";
while(leerA.hasNextLine()) {
    data += leerA.nextLine()+"\n";
}
```

POR ÚLTIMO, FILAS SE DEFINE COMO LAS VECES QUE EL CICLO DIO UN SALTO DE LÍNEA, Y LAS COLUMNAS COMO LAS VECES EN QUE EN FILA 0 SE ENCONTRARON CADENAS DE TEXTO DIFERENTES DE COMA.

SE RECORRE EL ARREGLO PARA PODER ORDENAR LOS NÚMEROS LEIDOS EN EL ARCHIVO DE TEXTO CONVIRTIÉNDOLOS A VALORES ENTEROS.

```
filasA = data.split("\n");
columA=filasA[0].split(",");
ClaveA = new int[filasA.length][columA.length];
for(int i =0; i<filasA.length;i++) {
    columA=filasA[i].split(",");
    for(int j=0;j<columA.length;j++) {
        ClaveA[i][j]=Integer.parseInt(columA[j]);
    }
}
```

CLASE “ENCRIPTACION”, MÉTODO “ENCRIPTAR”

SE RECORRE EL ARREGLO, AUNQUE CON UNA VARIANTE, YA QUE EL PRODUCTO DE MATRICES MULTIPLICA FILAS POR COLUMNAS, SE CREÓ UN NUEVO CICLO, QUE HACE QUE SE CUMPLA ESA CONDICIÓN PARA

PRODUCTOS MATRICIALES, SUMÁNDOSE EL VALOR DEL NUEVO PRODUCTO ENTRE LA PRIMERA MATRIZ CLAVE CON LA MATRIZ DEL MENSAJE CODIFICADO CON EL ANTERIOR HASTA QUE SE COMPLETE EL CICLO.

```
for(int i=0; i<3; i++) {
    for(int j=0; j<(columnas/3); j++) {
        for(int k=0; k<3; k++) {
            oculto[i][j]+=ClaveA[i][k]*codificado[k][j];
        }
    }
}
```

UNA VEZ HECHO EL PRODUCTO MATRICIAL SE SUMÓ LA SEGUNDA MATRIZ CLAVE. UNA VEZ COMPLETO EL PROCESO EL MENSAJE ESTARÁ ENCRIPTADO.

```
for(int i=0; i<3; i++) {
    for(int j=0; j<(columnas/3); j++) {
        encriptado[i][j]=oculto[i][j]+ClaveB[i][j];
    }
}
```

CLASE “DESENCRIPTACION”, MÉTODO “DESENCRIPTAR”

SE CALCULO EL DETERMINANTE DE LA MATRIZ CLAVE A, QUE VIENE SIENDO UN ANÁLOGO DE EL PRODUCTO CRUZ O VECTORIAL.

```
x=ClaveA[0][0]*((ClaveA[1][1]*ClaveA[2][2])-(ClaveA[1][2]*ClaveA[2][1]));
y=ClaveA[0][1]*((ClaveA[1][0]*ClaveA[2][2])-(ClaveA[1][2]*ClaveA[2][0]));
z=ClaveA[0][2]*((ClaveA[1][0]*ClaveA[2][1])-(ClaveA[1][1]*ClaveA[2][0]));
determinante=x-y+z;
```

SE OBTUVO LA MATRIZ TRANSPUESTA DE LA MATRIZ CLAVE A PORQUE ES LA MATRIZ A LA QUE SE REQUIERE CALCULAR SU INVERSA, PARA POSTERIORMENTE USARLA EN LA DESENCRIPTACIÓN

```
for(int i=0; i<3; i++) {
    for(int j=0; j<3; j++) {
        transpuesta[i][j]=ClaveA[j][i];
    }
}
```

DE LA MATRIZ TRANSPUESTA SE CALCULARON LOS COFACTORES O DETERMINANTES DE LOS MENORES, ES DECIR OBVIANDO LA FILA Y COLUMNA DEL ELEMENTO AL QUE SE LE QUIERE CALCULAR SU COFACTOR.

```
m00=(transpuesta[1][1]*transpuesta[2][2])-(transpuesta[1][2]*transpuesta[2][1]);
m01=(transpuesta[1][0]*transpuesta[2][2])-(transpuesta[1][2]*transpuesta[2][0]);
m02=(transpuesta[1][0]*transpuesta[2][1])-(transpuesta[1][1]*transpuesta[2][0]);
m10=(transpuesta[0][1]*transpuesta[2][2])-(transpuesta[0][2]*transpuesta[2][1]);
m11=(transpuesta[0][0]*transpuesta[2][2])-(transpuesta[0][2]*transpuesta[2][0]);
m12=(transpuesta[0][0]*transpuesta[2][1])-(transpuesta[0][1]*transpuesta[2][0]);
m20=(transpuesta[0][1]*transpuesta[1][2])-(transpuesta[0][2]*transpuesta[1][1]);
m21=(transpuesta[0][0]*transpuesta[1][2])-(transpuesta[0][2]*transpuesta[1][0]);
m22=(transpuesta[0][0]*transpuesta[1][1])-(transpuesta[0][1]*transpuesta[1][0]);
```

SE FORMÓ LA MATRIZ DE COFACTORES O MATRIZ ADJUNTA, UTILIZANDO LOS COFACTORES CALCULADOS EN EL SEGMENTO DE CÓDIGO ANTERIOR.

```
adjunta[0][0]=(+m00);
adjunta[0][1]=(-m01);
adjunta[0][2]=(+m02);
adjunta[1][0]=(-m10);
adjunta[1][1]=(+m11);
adjunta[1][2]=(-m12);
adjunta[2][0]=(+m20);
adjunta[2][1]=(-m21);
adjunta[2][2]=(+m22);
```

UNA VEZ FORMADA LA MATRIZ DE COFACTORES SE CALCULÓ LA INVERSA DIVIDIENDO CADA ELEMENTO DE DICHA MATRIZ ENTRE EL DETERMINANTE DE LA MATRIZ.

```
for(int i=0; i<3; i++) {
    for(int j=0; j<3; j++) {
        inversa[i][j]=(adjunta[i][j]/determinante);
    }
}
```

RECORRE LA MATRIZ Y RESTA LA MATRIC CLAVE B DE LA ENCRIPTADA, Y LA INVERSA DE A MULTIPLICA ESA DIFERENCIA PARA ENCONTRAR LA MATRIZ M DEL MENSAJE. SEGUIDAMENTE COMPARA LOS VALORES QUE CONTIENEN LA MATRIZ DESENCRIPTADA Y SUSTITUYE EN ESA MISMA POSICIÓN PERO EN OTRA MATRIZ DE TIPO STRING CON LA LETRA CORRESPONDIENTE PARA ESE NÚMERO.

```
for(int i=0; i<3; i++) {
    for(int j=0; j<(columnas/3); j++) {
        for(int k=0; k<3; k++) {
            desencriptado[i][j]+=inversa[i][k]*(encriptado[k][j]-ClaveB[k][j]);
        }
        if(Math.round(desencriptado[i][j])==0) {
        }else if(Math.round(desencriptado[i][j])==26) {
            decodificado[i][j]="z";
        }else{
            decodificado[i][j]=" ";
        }
    }
}
```

...