EJERCICIO 1

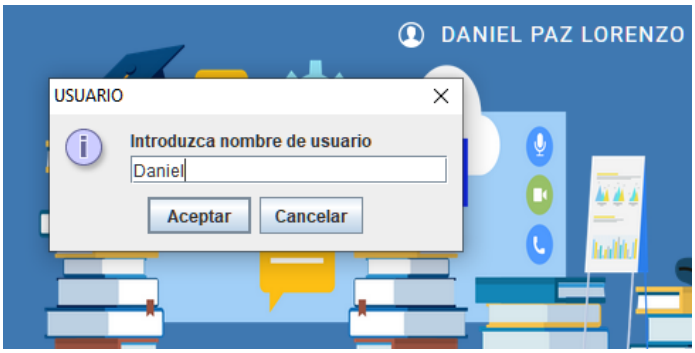
En esta tarea se nos pide que cifremos un texto dado con algoritmo "AES/ECB/PKCS5Padding" y con una clave de número aleatorio con semilla la cadena (usuario + password) y de longitud 128 bits.

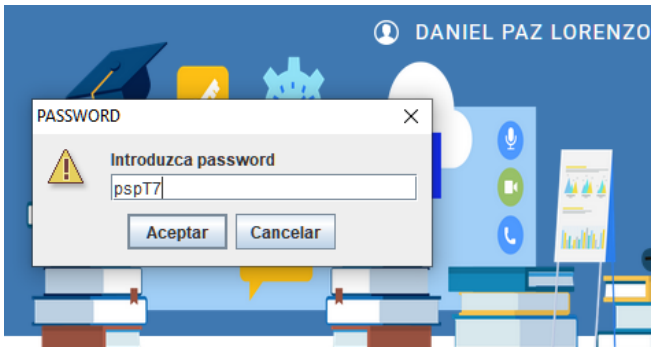
Para realizar esta tarea he decidido realizar una aplicación con interfaz gráfica y así poder interactuar con el usuario de una manera sencilla y visual. A grandes rasgos, la aplicación solicitará al usuario un nombre y contraseña, luego mostrará un formulario con distintas opciones: introducir texto, guardar fichero encriptado, mostrar fichero encriptado y mostrar fichero desencriptado.

La clave va a ser privada y la misma tanto para encriptar como para desencriptar, es decir vamos a realizar una encriptación simétrica en este caso con algoritmo AES como se nos pide en la tarea.

A continuación, voy a mostrar el funcionamiento de mi aplicación poniendo como ejemplo una ejecución tipo y explicando cada uno de los pasos:

1. Se nos va a solicitar usuario y contraseña mediante 2 cuadros de diálogo “JOptionPane”, una vez introducidas serán recogidas en ambas variables guardadas para tal fin:



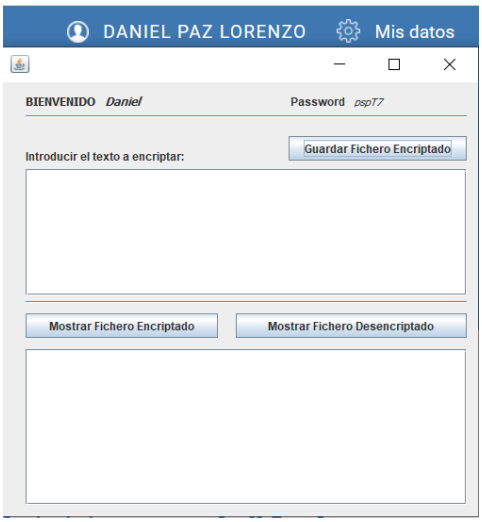



```
12  public class InicioAplicacion {
13
14      /**
15       * @param args the command line arguments
16       */
17      public static void main(String[] args) throws NoSuchAlgorithmException {
18          //Solicita usuario y contraseña y los recoge a través de JOptionPane
19          String usuario = JOptionPane.showInputDialog(null, "Introduzca nombre de usuario", "USUARIO", JOptionPane.INFORMATION_MESSAGE);
20          String password = JOptionPane.showInputDialog(null, "Introduzca password", "PASSWORD", JOptionPane.WARNING_MESSAGE);
21          //Crea instancia de la clase FormularioAplicacion
22          FormularioAplicacion formApli = new FormularioAplicacion(usuario, password);
23          formApli.setVisible(true);
24      }
25
26  }
```

```
Output - Tarea_7_PSP (run)
run:
Formato: RAW
Clave
r{$3_#□□□v□C□-□□
```

Este es el código de la clase que inicia la aplicación y en ejecución podemos ver que la salida nos muestra ya la clave encriptada

2. Una vez recogidos los datos y generada ya la clave se muestra el formulario principal de la aplicación con todas las opciones disponibles:



Pantalla de opciones de la aplicación

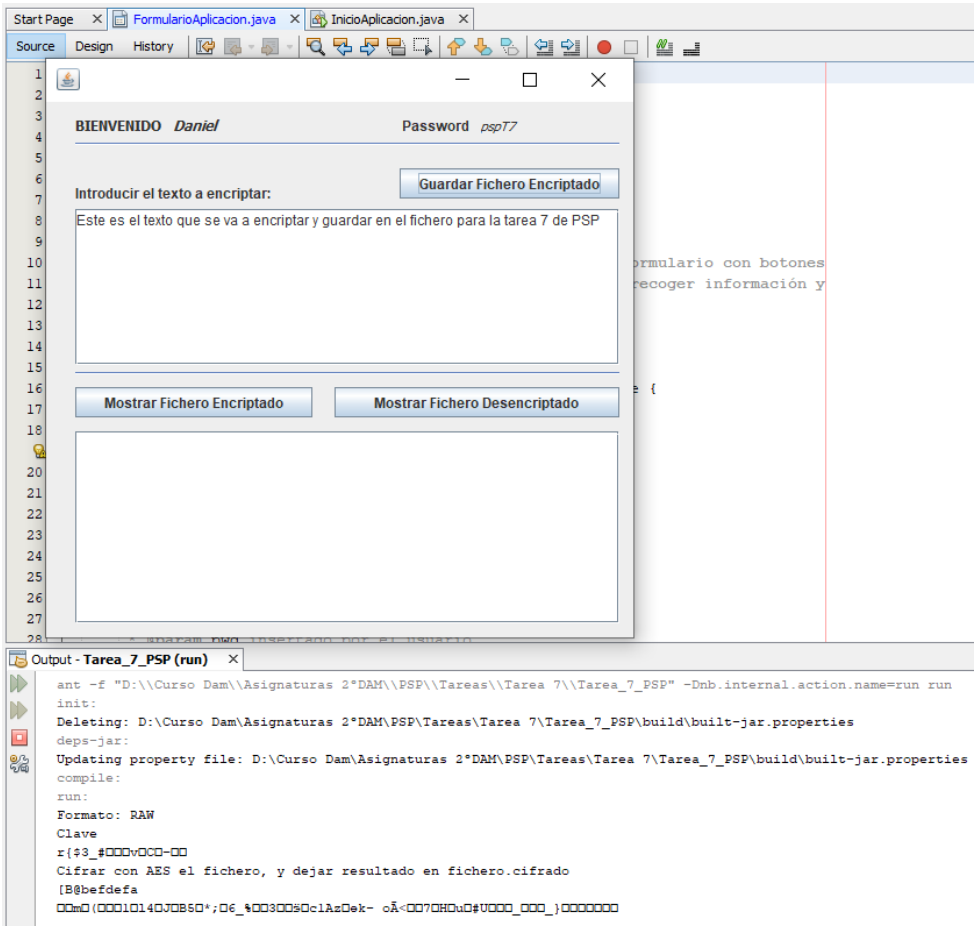
```java
/**
 * Constructor de la clase FormularioAplicacion
 *
 * @param usuario insertado por el usuario
 * @param pwd insertado por el usuario
 * @throws NoSuchAlgorithmException
 */
public FormularioAplicacion(String usuario, String pwd) throws NoSuchAlgorithmException {
    initComponents();
    //Inicializar variables de clase
    etiquetaUsuario.setText(usuario);
    etiquetaPwd.setText(pwd);
    this.usuario = usuario;
    this.password = pwd;
    this.fichero = new File("fichero");
    //Crear e inicializar clave
    this.clave = generarClaveSercreta(usuario, password);
}
```

Código del constructor que genera la pantalla de opciones cuando instanciamos la clase

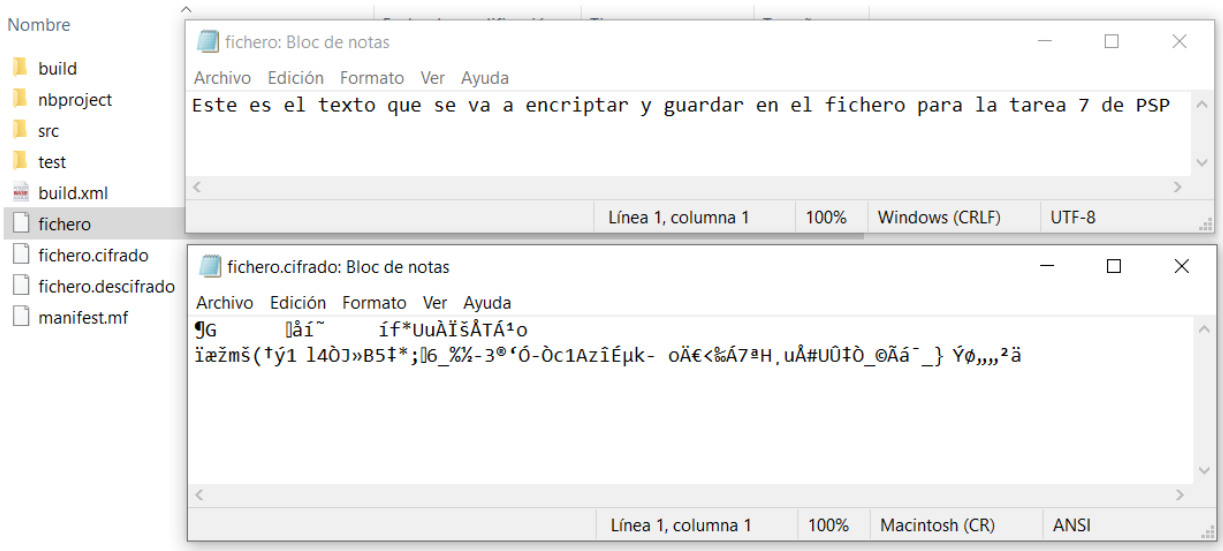
```java
/**
 * Método para crear una clave de 128 bits a partir de un número aleatorio
 * con semilla la suma de las cadenas usuario y password
 *
 * @param usuario
 * @param pwd
 * @return
 * @throws NoSuchAlgorithmException
 */
public SecretKey generarClaveSercreta(String usuario, String pwd) throws NoSuchAlgorithmException {
    //Suma de las cadenas
    String cadena = usuario + pwd;
    //Convertir cadena a array de bytes
    byte[] b = cadena.getBytes();

    //Declarar objeto SecureRandom con semilla el array de bytes
    SecureRandom sr = new SecureRandom();
    sr.setSeed(b);

    //Declaración de objeto KeyGenerator con algoritmo AES e inicializado a 128 bits con número de seguridad aleatorio
    KeyGenerator generador = KeyGenerator.getInstance("AES");
    generador.init(128, sr);
    //Crear objeto SecretKey a través de método de KeyGenerator
    SecretKey claveAES = generador.generateKey();
    System.out.println("Formato: " + claveAES.getFormat());
    System.out.println("Clave");
    mostrarBytes(claveAES.getEncoded());
    return claveAES;
}
```

Código para generar la clave de 128 bits, encriptada con el algoritmo AES y configurada a partir de un número aleatorio con semilla la cadena del nombre de usuario + password

3. Escribimos en el recuadro el texto que queremos encriptar y cuando ya lo tengamos pulsamos el botón de "guardar fichero encriptado":



Una vez se guarda el fichero encriptado muestra la clave y el texto, ya cifrados y un mensaje de verificación



Si abrimos la carpeta del proyecto podemos ver que se nos han creado los 2 ficheros y si abrimos cada uno de ellos veremos su contenido

Cuando nosotros pulsamos el botón se genera un evento y le diremos que encripte el texto, con el algoritmo "`Rijndael/ECB/PKCS5Padding`" y posteriormente lo guarde en un fichero (fichero.cifrado). Para ello he creado 2 métodos:

```
private void botonGuardarFicherosActionPerformed(java.awt.event.ActionEvent evt) {

    guardarFichero();
    encriptarFichero();
}
```

Recoge evento y llama a los métodos

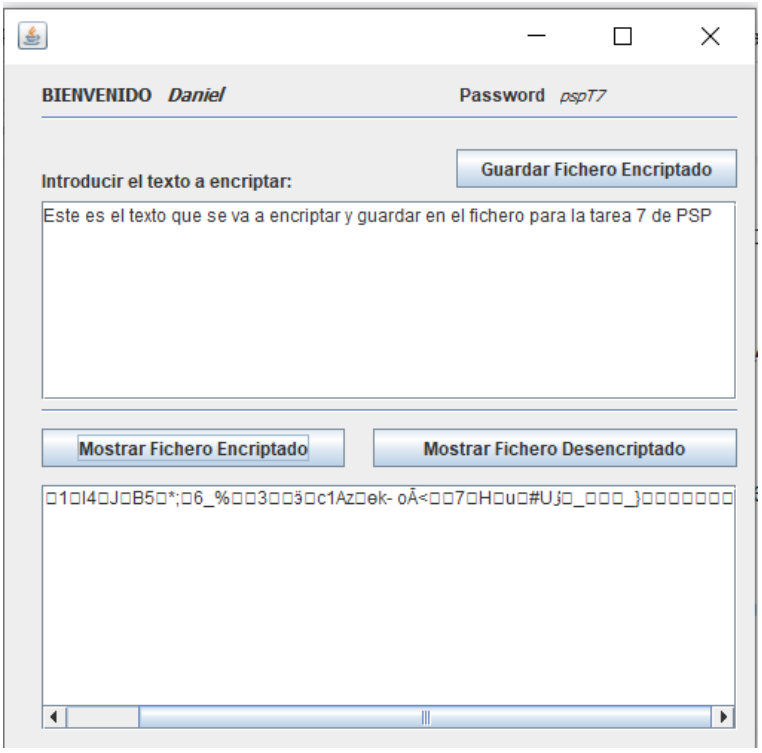
```
/**
 * Método para guardar en un fichero el texto recogido del usuario
 */
public void guardarFichero() {

    try {
        //Guardamos texto introducido en fichero
        byte[] bufferIn; //array de bytes
        bufferIn = cuadroIntroText.getText().getBytes();//Recoger texto del textBox
        FileOutputStream fs = new FileOutputStream(fichero); //flujo de salida
        fs.write(bufferIn); //Graba el texto introducido en fichero
    } catch (FileNotFoundException ex) {
        Logger.getLogger(FormularioAplicacion.class.getName()).log(Level.SEVERE, null, ex);
    } catch (IOException ex) {
        Logger.getLogger(FormularioAplicacion.class.getName()).log(Level.SEVERE, null, ex);
    }

}
```

Recoge el texto introducido y lo guarda en un fichero llamado "fichero" dentro del proyecto

```
/**
 * Método que lee "fichero", lo encripta con algoritmo AES y lo guarda en
 * "fichero.cifrado"
 */
public void encriptarFichero() {

    try {
        int bytesLeidos;
        String cadena = "";

        //Creación de objeto Cipher para cifrar, utilizando el algoritmo AES
        Cipher cifrador = Cipher.getInstance("AES/ECB/PKCS5Padding");
        //Inicialización del cifrador en modo CIFRADO o ENCRIPTACIÓN
        cifrador.init(Cipher.ENCRYPT_MODE, clave);
        System.out.println("\nCifrar con AES el " + fichero
                + ", y dejar resultado en " + fichero + ".cifrado");

        //declaración  de objetos
        byte[] bufferOut = new byte[1000]; //array de bytes
        byte[] bufferCifrado;
        FileInputStream fe = new FileInputStream(fichero); //flujo de entrada
        FileOutputStream fsc = new FileOutputStream(fichero + ".cifrado"); //flujo de salida

        //lee el fichero de 1k en 1k y pasa los fragmentos leidos al cifrador
        bytesLeidos = fe.read(bufferOut, 0, 1000);
        while (bytesLeidos != -1) {//mientras no se llegue al final del fichero
            //pasa texto claro al cifrador y lo cifra, asignándolo a bufferCifrado
            bufferCifrado = cifrador.update(bufferOut, 0, bytesLeidos);
            fsc.write(bufferCifrado); //Graba el texto cifrado en fichero
            bytesLeidos = fe.read(bufferOut, 0, 1000);
            System.out.println(bufferCifrado);
            cadena = new String(bufferCifrado, "UTF-8");
            System.out.print(cadena);
        }
        bufferCifrado = cifrador.doFinal(); //Completa el cifrado
        cadena = new String(bufferCifrado, "UTF-8");
        System.out.print(cadena);
        fsc.write(bufferCifrado); //Graba el final del texto cifrado, si lo hay
        //Cierra flujos
        fe.close();
        fsc.close();
    } catch (NoSuchAlgorithmException ex) {
        Logger.getLogger(FormularioAplicacion.class.getName()).log(Level.SEVERE, null, ex);
    } catch (NoSuchPaddingException ex) {
        Logger.getLogger(FormularioAplicacion.class.getName()).log(Level.SEVERE, null, ex);
```

Lee el fichero anterior, lo encripta y lo guarda en un fichero llamado "fichero.cifrado" dentro del proyecto

4. A continuación, pulsamos en "mostrar fichero encriptado" lo que hará que se muestre en el área de texto el fichero cifrado anteriormente:

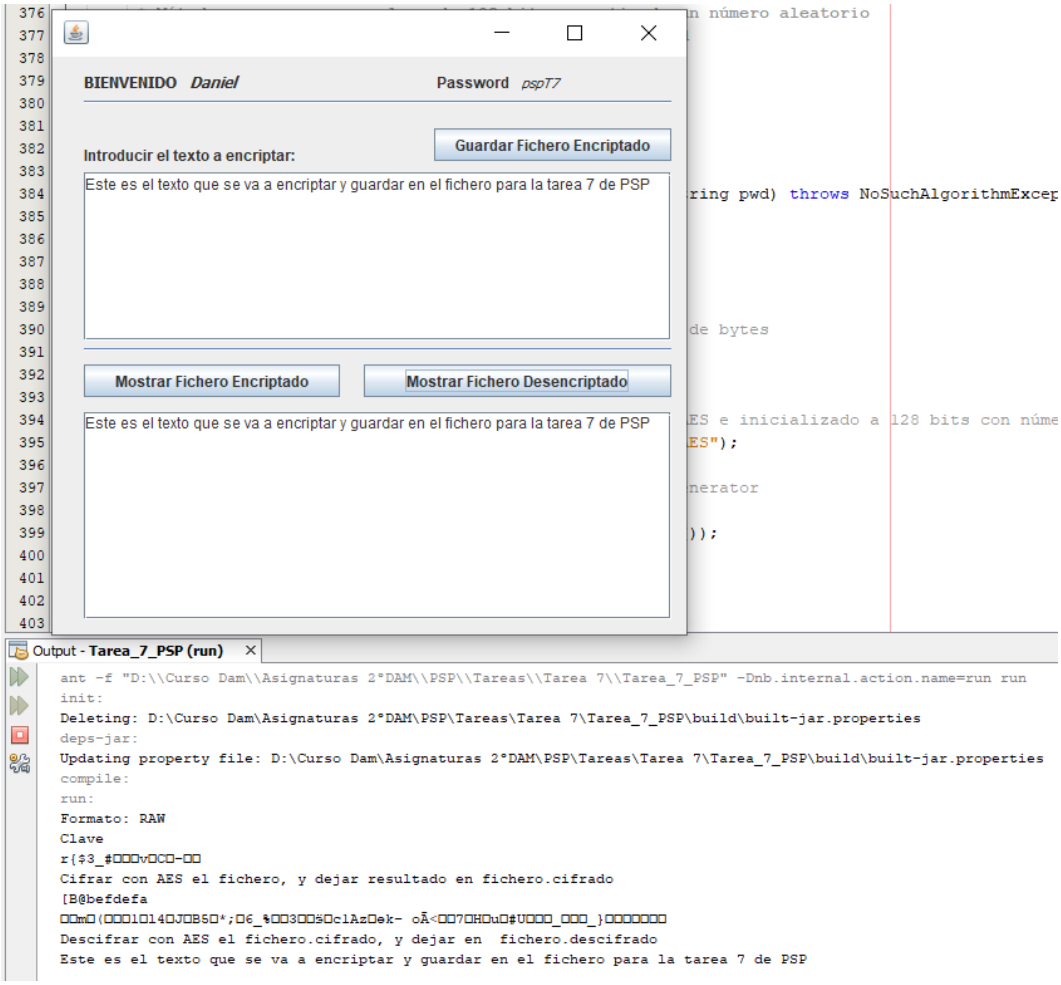


En el cuadro inferior vemos como quedaría el texto cifrado con el algoritmo AES

El código que responde al evento generado por el botón y que lee el fichero para mostrarlo al usuario sería el siguiente:

```
/**
 * Método que resuelve el evento generado al pulsar el botón de mostrar el
 * fichero encriptado mostrandolo en el area de texto
 *
 * @param evt
 */
private void botonMostrarEncriptadoActionPerformed(java.awt.event.ActionEvent evt) {

    File ficheroCifrado = new File(fichero + ".cifrado");
    try {
        //Flujo de entrada de fichero
        FileReader fis = new FileReader(ficheroCifrado);
        BufferedReader bis = new BufferedReader(fis);
        String linea;
        //Recorre el fichero y lo muestra en el area de texto
        while ((linea = bis.readLine()) != null) {
            areaTexto.setText(linea);
        }
    } catch (FileNotFoundException ex) {
        Logger.getLogger(FormularioAplicacion.class.getName()).log(Level.SEVERE, null, ex);
    } catch (IOException ex) {
        Logger.getLogger(FormularioAplicacion.class.getName()).log(Level.SEVERE, null, ex);
    }

}
```

5. La última de las opciones sería desencriptar el fichero cifrado, guardarlo en “fichero.descifrado” dentro del proyecto y a mostrar, en el área de texto de la aplicación, el resultado. Todo ello lo haremos pulsando el botón de “Mostrar fichero desencriptado”:



Observamos que el texto mostrado coincide con el inicial por lo que se ha desencriptado correctamente, a la vez muestra un mensaje de verificación en la salida por consola

En el código que recoge el evento del botón, recogemos en un objeto File el fichero ya descifrado que nos devolverá el método “descifrarFichero()” que hemos creado para ello. Una vez lo tenemos recogido solo queda leerlo e ir mostrando al usuario el resultado por pantalla:

```
/**
 * Método que recoge el evento de botón para mostrar el fichero
 * desencriptado
 *
 * @param evt
 */
private void botonMostrarFicDesencripActionPerformed(java.awt.event.ActionEvent evt) {

    File ficheroDesCifrado = descifrarFichero();
    try {
        //Lee fichero y lo muestra en el area de texto
        FileReader fis = new FileReader(ficheroDesCifrado);
        BufferedReader bis = new BufferedReader(fis);
        String linea;
        while ((linea = bis.readLine()) != null) {
            areaTexto.setText(linea);
        }

    } catch (FileNotFoundException ex) {
        Logger.getLogger(FormularioAplicacion.class.getName()).log(Level.SEVERE, null, ex);
    } catch (IOException ex) {
        Logger.getLogger(FormularioAplicacion.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Este es el método creado para descifrar el fichero en el que le indicamos que fichero queremos desencriptar, con que algoritmo y cuál es la clave para hacerlo:

```
/**
 * Método para descifrar el fichero cifrado
 *
 * @return
 */
public File descifrarFichero() {
    //Creamos los ficheros de cifrado y descifrado
    File ficheroCifrado = new File(fichero + ".cifrado");
    File ficheroDescifrado = new File(fichero + ".descifrado");
    try {
        int bytesLeidos;
        String cadena = "";
        //Crear objeto Cipher con algoritmo AES
        Cipher cifrador = Cipher.getInstance("AES/ECB/PKCS5Padding");
        //Poner cifrador en modo DESCIFRADO o DESENCRIPTACIÓN
        cifrador.init(Cipher.DECRYPT_MODE, clave);
        System.out.println("\nDescifrar con AES el " + ficheroCifrado
                + ", y dejar en  " + ficheroDescifrado);

        //declaración  de objetos
        FileInputStream fe = new FileInputStream(ficheroCifrado);
        FileOutputStream fs = new FileOutputStream(ficheroDescifrado);
        byte[] bufferClaro;
        byte[] buffer = new byte[1000]; //array de bytes

        //lee el fichero de 1k en 1k y pasa los fragmentos leidos al cifrador
        bytesLeidos = fe.read(buffer, 0, 1000);
        while (bytesLeidos != -1) {//mientras no se llegue al final del fichero
            //pasa texto cifrado al cifrador y lo descifra, asignándolo a bufferClaro
            bufferClaro = cifrador.update(buffer, 0, bytesLeidos);
            fs.write(bufferClaro); //Graba el texto claro en fichero
            bytesLeidos = fe.read(buffer, 0, 1000);
            cadena = new String(bufferClaro, "UTF-8");
            System.out.print(cadena);
        }
        bufferClaro = cifrador.doFinal(); //Completa el descifrado
        cadena = new String(bufferClaro, "UTF-8");
        System.out.println(cadena);
        fs.write(bufferClaro); //Graba el final del texto claro, si lo hay
        //Cerrar flujos
        fe.close();
        fs.close();
    } catch (NoSuchAlgorithmException ex) {
        Logger.getLogger(FormularioAplicacion.class.getName()).log(Level.SEVERE, null, ex);
    } catch (NoSuchPaddingException ex) {
        Logger.getLogger(FormularioAplicacion.class.getName()).log(Level.SEVERE, null, ex);
    } catch (InvalidKeyException ex) {
```