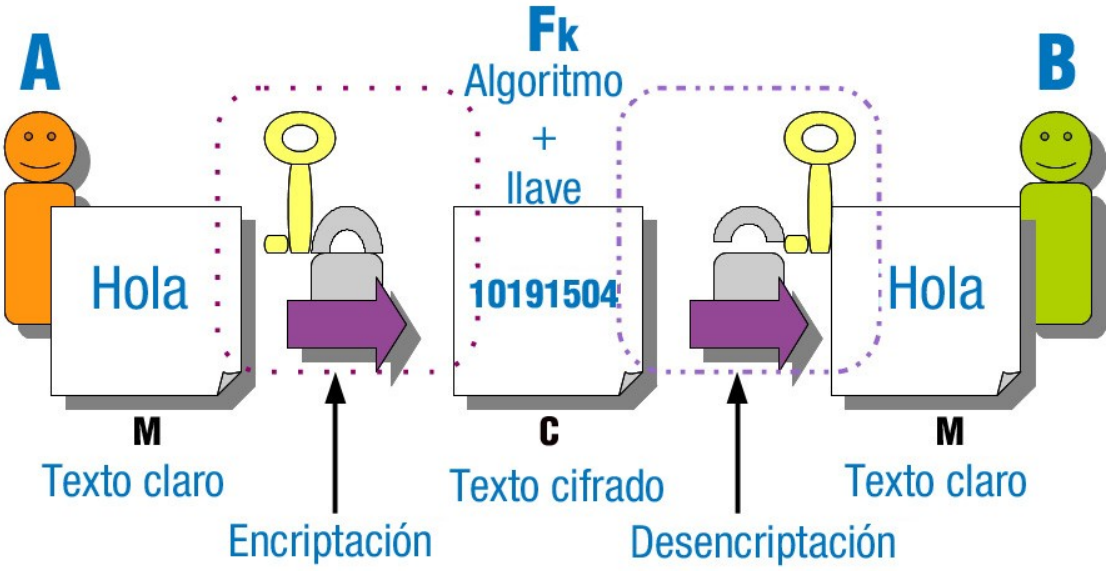
I.E.S Aguadulce

Titulación: Desarrollo de Aplicaciones Multiplataforma

Asignatura: Programación de servicios y procesos



Tarea número 6

– *Aplicaciones con comunicaciones seguras* –

Autor:	David Jiménez Riscardo
Teléfono:	618882196
E-Mail:	david.jimenez.riscardo@gmail.com

Dos Hermanas, 7 de Mayo del año 2022

Table of Contents

En esta tarea trabajaremos con algunos de los conceptos relacionados con las comunicaciones seguras que has visto en la unidad. Para ello tendrás que ir implementando cada uno de los programas que se proponen en los siguientes ejercicios así como rellenar la Ficha de la tarea donde explicarás el funcionamiento de cada componente y analizarás algunos escenarios específicos de prueba. La ficha la puedes descargar desde la sección Información de interés.

Para realizar estos programas no tienes que desarrollar ningún entorno gráfico. Serán aplicaciones que se ejecutarán en consola y que pueden recibir parámetros al ser ejecutadas.

1 Encriptación

Debes implementar un programa Java llamado Encriptador que reciba como argumentos desde la consola el nombre de un archivo y una contraseña de cifrado para a continuación generar un nuevo archivo que contendrá esa información encriptada. Para ello tendrás en cuenta lo siguiente:

- se utilizará el algoritmo Rijndael (también conocido como AES);
- se realizará encriptación de bloque (block cipher): modo ECB, con tipo de relleno: PKCS5Padding;
- la clave tendrá una longitud de 128 de bits y se generará a partir de una contraseña de encriptación solicitada al usuario una vez cargado el contenido del archivo de en memoria. Esa contraseña es la que se utilizará como semilla para la creación de la clave.

El nombre del nuevo archivo generado será encriptado.txt.

Recuerda que para crear una instancia de un objeto Cipher debes usar el método estático `Cipher.getInstance` donde indicarás, separados por el carácter barra ('/') el algoritmo de encriptación, el modo y el tipo de relleno (padding). En nuestro caso será algo tan simple como `Cipher.getInstance("Rijndael/ECB/PKCS5Padding")`.

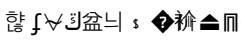
Ejemplo de ejecución

Dado un archivo de texto con el siguiente contenido:

```
Archivo de prueba para PSP06.
```

Si ejecutamos el programa y proporcionamos como entradas ese archivo y una determinada contraseña de encriptación, obtendremos como resultado un archivo encriptado.txt con un contenido ininteligible que podría ser algo así:

Posible contenido de encriptado.txt:



Tu programa debe ser robusto y no romperse con facilidad ante los posibles fallos que puedan darse. Aquí tienes una lista de los fallos que, como mínimo, debe controlar tu programa:

- No puede romperse porque no se proporcionen los parámetros necesarios.
- Debe controlarse que no exista un archivo con el nombre que se ha indicado (excepción `FileNotFoundException`).
- Debe controlarse que si se ha producido algún error de E/S (excepción `IOException`).
- Debe controlarse cualquier posible error al configurar el algoritmo criptográfico (excepciones `NoSuchAlgorithmException`, `NoSuchPaddingException`, `InvalidKeyException`).
- Debe controlarse cualquier posible error relacionado con el tamaño de bloque o el relleno (excepciones `IllegalBlockSizeException`, `BadPaddingException`).

En cualquier caso se finalizará ordenadamente y con un mensaje de error apropiado.

2 Desencriptación

Debes construir un nuevo programa en Java que permita desencriptar un archivo que haya sido encriptado con el programa anterior. Recibirá como argumento desde la consola el nombre del archivo para descifrar y una contraseña de descifrado. A continuación se procederá a la desencriptación del contenido del archivo. Si coincide con la usada para encriptar, el contenido del nuevo archivo debería coincidir exactamente con el del archivo original sin encriptar (cifrado simétrico).

El nombre del nuevo archivo generado será desencriptado.txt

Si intentas descifrar el archivo con una contraseña de encriptación diferente a la que se utilizó para cifrar, es más que probable salte una excepción de tipo `BadPaddingException` que tendrás que gestionar y finalizar el error con algún mensaje de error apropiado. Por otro lado, si intentas desencriptar un archivo que no esté encriptado es posible que te encuentres con una excepción de tipo `IllegalBlockSizeException`, que también tendrás que gestionar.

Al igual que en el caso anterior, tu programa debe ser robusto y no romperse con facilidad ante los posibles fallos que puedan darse.

En cualquier caso, ante cualquier problema que no pueda resolverse, el programa finalizará ordenadamente y con un mensaje de error apropiado. Los problemas y excepciones que debes controlar como mínimo serán los mismos que en el ejercicio anterior.

3 Documentación

Para documentar esta tarea deberás rellenar, al igual que en las anteriores, una ficha de la tarea. En este caso, para cada ejercicio tendrás que:

1. Proporcionar una breve explicación sobre cómo has construido el programa. Debe ser una explicación breve y concisa. No más de un par de párrafos.
2. Documentar una prueba del programa donde funcione correctamente con capturas de ejecución donde se pueda apreciar cómo se encripta (o desencripta) correctamente un archivo:
   - El contenido del archivo que se encriptará será el que se indica en el ejemplo de ejecución del ejercicio ("Archivo de prueba para PSP06.") y la contraseña de encriptación será "PSP06" (sin comillas).
   - En el caso de la prueba del desencriptador, se utilizará el archivo que se ha cifrado con el encriptador para observar que efectivamente se obtiene de nuevo el contenido original.
3. Documentar pruebas del programa, adjuntando capturas de ejecución, donde se ponga de manifiesto que tu programa es seguro y se están controlando los errores más habituales que podrían suceder. Habrá que documentar las siguientes pruebas:
   - Faltan parámetros (para ambos ejercicios).
   - Nombre de archivo no encontrado (para ambos ejercicios).
   - Intentar desencriptar un archivo no encriptado (solo para el ejercicio de desencriptación).
   - Intentar desencriptar un archivo encriptado utilizando una contraseña de encriptación diferente a la que se usó para encriptar (solo para el ejercicio de desencriptación).
4. Explicar, adjuntando capturas sobre vuestro propio código fuente cada uno de los posibles errores que se os ha pedido que controléis. Las explicaciones deben ser breves y concisas. No más de un par de líneas por cada posible error o excepción que se gestione. En algunos casos, pueden agruparse las explicaciones si te resulta más sencillo o son prácticamente las mismas.

La correcta entrega y cumplimentación de la ficha con toda esta documentación será imprescindible para poder superar esta tarea.

3.1 Explicación del Programa Encriptador

El desarrollo del programa lo he dividido en 3 partes:

1. Lectura de la entrada estándar de los datos necesarios: nombre del fichero a encriptar y contraseña.
2. Generación de la clave a partir de la contraseña.
3. Encriptado del contenido del fichero indicado por entrada estándar que guardaremos en un nuevo fichero llamado `"encriptado.txt"`.

En referencia a la lectura de la entrada estándar llevaré diferentes controles:

- Control para finalizar el programa. Si introducimos la cadena "FIN" el programa terminará.
- Control para comprobar si un fichero existe. Pediré repetidamente que el usuario introduzca un nombre en el caso que el archivo no exista.
- Control sobre entrada vacía. En el caso que el usuario presione la tecla ENTER le indicaré al usuario que es una entrada no válida.

Para la generación de la clave utilizo el método **`generarClave`**, al cual le paso la contraseña introducida por el usuario y su longitud devolviendo la clave generada. En caso de no generarse una clave porque se produzca alguna excepción devolvería el valor NULL. **Es reseñable el uso en este método de un algoritmo HASH para obtener un resumen de la contraseña, de esta forma no tendremos que tener un control estricto de la longitud de la contraseña introducida por el usuario; facilitando así la usabilidad.**

En el caso que la clave generada sea diferente a NULL seguiré con el programa lanzando el método **`encriptarFichero`**. Dicho método recibe el nombre del fichero a encriptar y la clave generada devolviendo TRUE en caso que la encriptación se efectúe correctamente y FALSE en caso contrario.

Durante el programa he utilizado un par de funciones helpers cuyo cometido paso a detallar:

`existFile`: comprueba si un fichero existe, devuelve true o false.

`deleteFile`: elimina un archivo en el caso que exista.

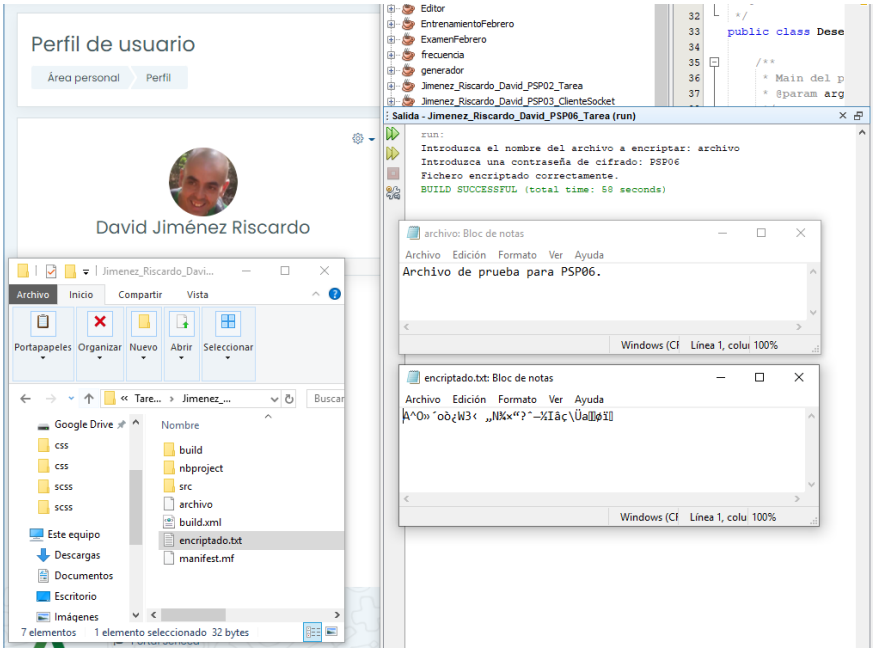
3.2 Explicación del Programa Desencriptador

Sigue el mismo diseño en tres fases que el programa Encriptador. Hace uso de las mismas funciones helpers y del método `generarClave`. Los controles para la entrada estándar serán los mismos.

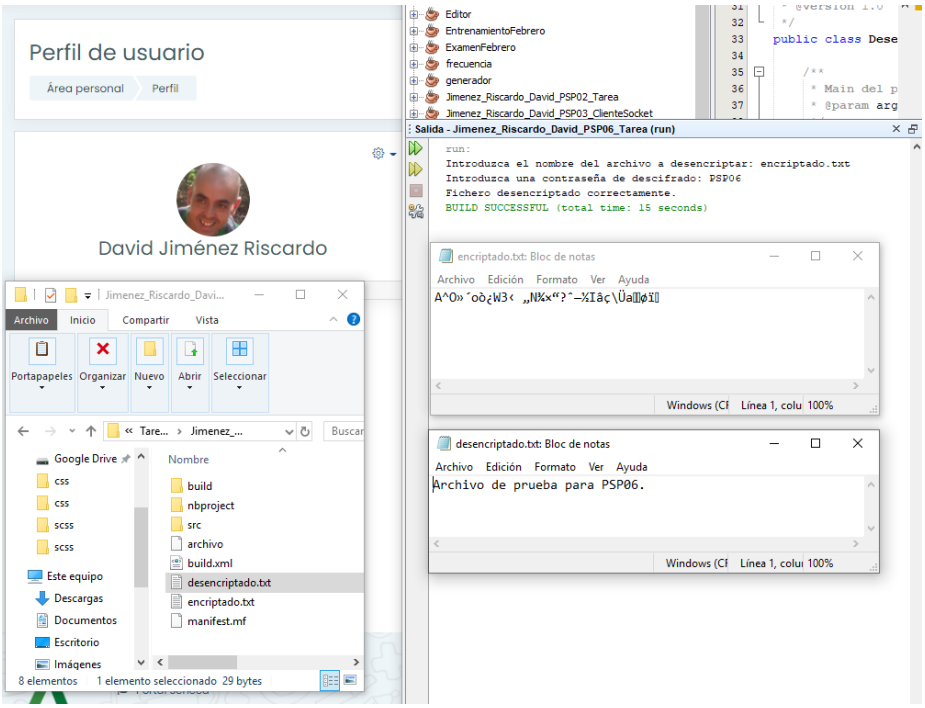
Para desencriptar el fichero dado utilizo el método **`desencriptarFichero`** cuyo desarrollo es muy parecido al del Encriptador salvo que hay control más específico de las excepciones.

Otra diferencia en referencia al programa Encriptador sería que en la entrada de datos agrego como condición adicional, que el fichero a desencriptar se llame `"encriptado.txt"`.

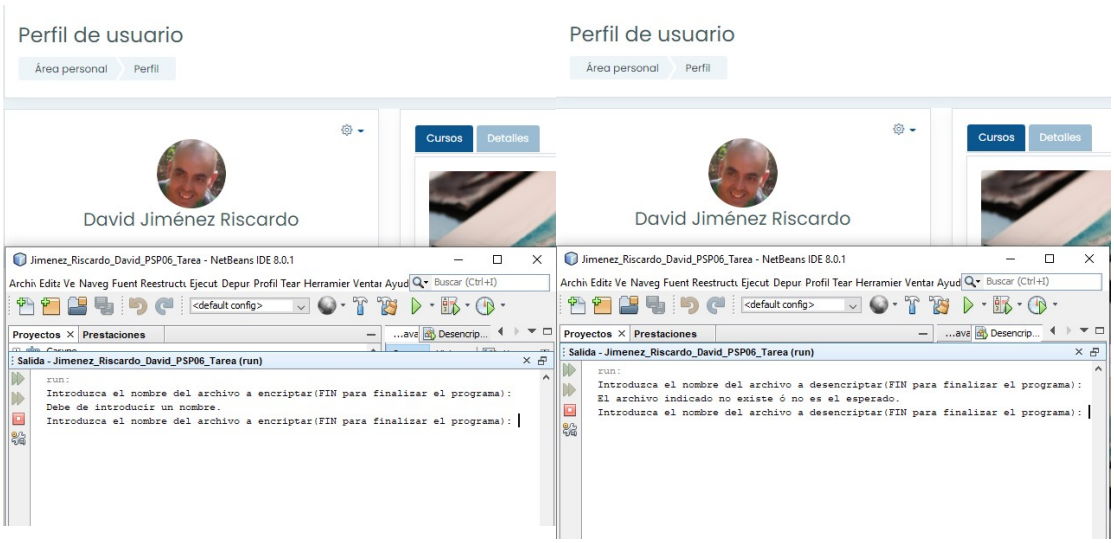
3.3 Prueba correcta de Encriptador



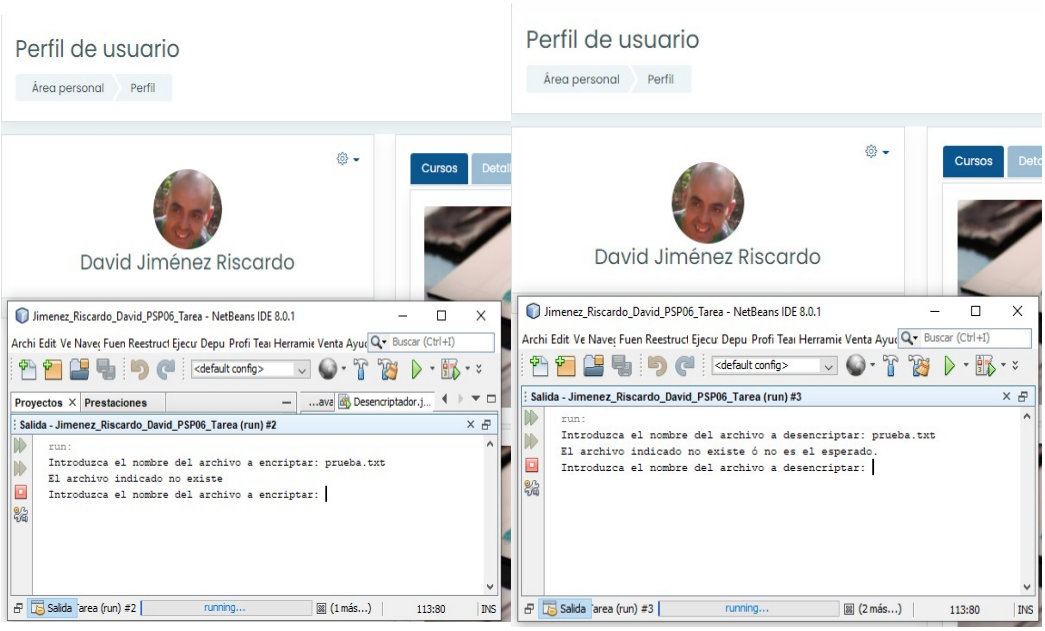
3.4 Prueba correcta de Desencriptador



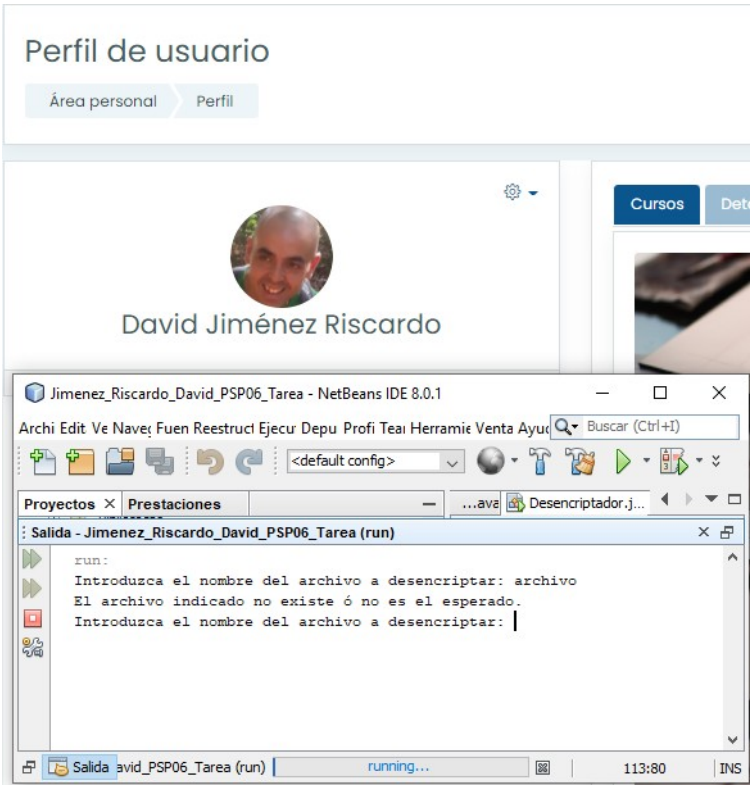
3.5 Control de errores más habituales



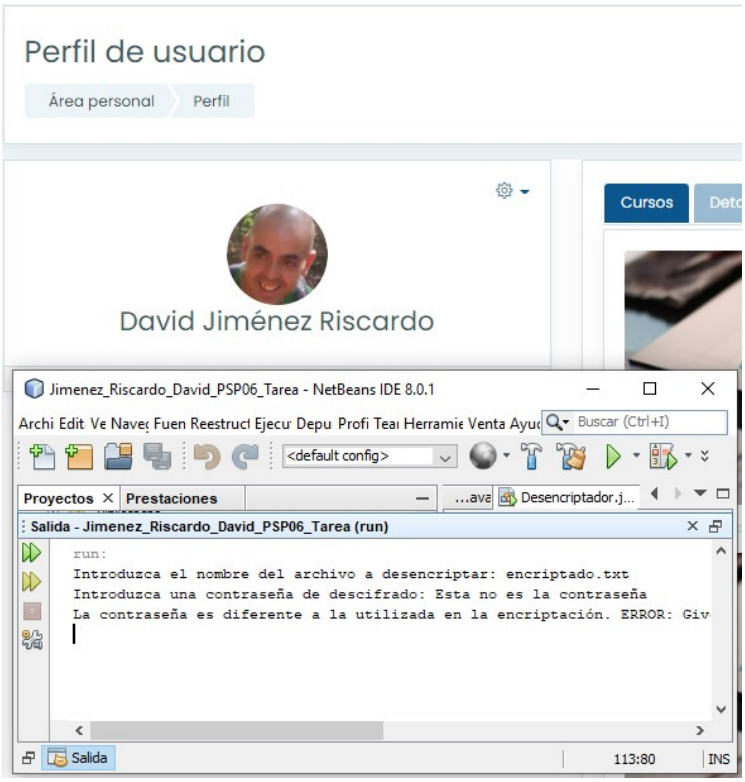
Faltan parámetros



Nombre de archivo no encontrado



Intentar desencriptar un archivo no encriptado



Intentar desencriptar un archivo encriptado utilizando una contraseña diferente

3.6 Explicación sobre las posibles Excepciones

Las excepciones más reseñables serían:

- **`UnsupportedEncodingException y NoSuchAlgorithmException`** que resultarían de un **problema existente en la configuración del algoritmo a la hora de generar la contraseña**. La gestión de esta excepción sería simplemente mostrar un mensaje del error. Al saltar dichas excepciones se devolvería al programa principal el valor NULL terminando la ejecución del programa correctamente. Esto es aplicable tanto al programa Encriptador como Desencriptador.

```
} catch (UnsupportedEncodingException | NoSuchAlgorithmException ex) {
    System.out.println("Error en la configuración del algoritmo. ERROR: " + ex.getMessage());
}
```

- **`NoSuchAlgorithmException, NoSuchPaddingException y InvalidKeyException`** se lanzarían al producirse un **posible error en la configuración del algoritmo utilizado en la encriptación / desencriptación**. De nuevo la gestión de esta excepción sería mostrar un mensaje del error, devolviéndose el valor FALSE al no haberse efectuado la encriptación / desencriptación esperada, terminando la ejecución del programa correctamente. Esto es aplicable tanto al programa Encriptador como Desencriptador.

```
} catch (NoSuchAlgorithmException | NoSuchPaddingException | InvalidKeyException ex) {
    System.out.println("Error en la configuración del algoritmo. ERROR: " + ex.getMessage());
}
```

- **`IllegalBlockSizeException y BadPaddingException`** (programa encriptador). Hacen referencia a un **error en el tamaño del bloque ó en el relleno** ocurriendo en la encriptación de la información. En tal caso muestro un mensaje de error y cierro el flujo asociado al fichero que estamos encriptando para posteriormente eliminarlo. Al no haberse podido efectuar la encriptación devolveríamos al programa principal el valor FALSE, terminando la ejecución del programa correctamente.

```
} catch (FileNotFoundException ex) {
    System.out.println("Archivo no encontrado. ERROR: " + ex.getMessage());
} catch (IOException ex) {
    System.out.println("Error de E/S. ERROR: " + ex.getMessage());
} catch (IllegalBlockSizeException | BadPaddingException ex){
    System.out.println("Error en el tamaño del bloque / relleno. ERROR: " + ex.getMessage());
    fe.close();
    //Debemos de cerrar el uso del fichero para poderlo eliminar
    fs.close();
    deleteFile("encriptado.txt");
}
```

- **`IllegalBlockSizeException y BadPaddingException`** (programa desencriptador). **El primero de ellos ocurriría al no estar encriptado el fichero y el segundo al comprobar que la contraseña con la que se encriptó el fichero es diferente a la introducida por el usuario**. En ambos casos muestro el mensaje de error oportuno y cierro el flujo con el archivo para poder eliminarlo posteriormente. Al no haberse podido efectuar la desencriptación devolveríamos al programa principal el valor FALSE, terminando la ejecución del programa correctamente.

```
} catch (FileNotFoundException ex) {
    System.out.println("Archivo no encontrado. ERROR: " + ex.getMessage());
} catch (IOException ex) {
    System.out.println("Error de E/S. ERROR: " + ex.getMessage());
} catch (IllegalBlockSizeException ex) {
    System.out.print("El archivo indicado no está encriptado. ERROR: "+ ex.getMessage());
    fe.close();
    //Debemos de cerrar el uso del fichero para poderlo eliminar
    fs.close();
    deleteFile("desencriptado.txt");
} catch (BadPaddingException ex) {
    System.out.print("La contraseña es diferente a la utilizada en la encriptación. ERROR: "+ ex.getMessage());
    fe.close();
    //Debemos de cerrar el uso del fichero para poderlo eliminar
    fs.close();
    deleteFile("desencriptado.txt");
}
```

4 Evaluación de la tarea

Cristerios de evaluación implicados

a) Se han identificado y aplicado principios y prácticas de programación segura.

b) Se han analizado las principales técnicas y prácticas criptográficas.

c) Se han definido e implantado políticas de seguridad para limitar y controlar el acceso de los usuarios a las aplicaciones desarrolladas.

d) Se han utilizado esquemas de seguridad basados en roles.

e) Se han empleado algoritmos criptográficos para proteger el acceso a la información almacenada.

f) Se han identificado métodos para asegurar la información transmitida.

g) Se han desarrollado aplicaciones que utilicen sockets seguros para la transmisión de información.

h) Se han depurado y documentado las aplicaciones desarrolladas.

Nuestros programas no pueden abortar porque se produzca algún error inesperado que haga que salte alguna excepción no controlada que no capturemos y finalmente sea capturada por la máquina virtual de Java haciendo que el programa termine de manera abrupta.

¡ESO NO PUEDE SUCEDER!

Debes ser especialmente cuidadoso con:

- El control de posibles **errores con los parámetros y valores de entrada**: no superar los límites del array `args`, controlar si los parámetros cumplen los criterios especificados (por ejemplo si deben ser números: excepciones del tipo `NumberFormatException`, `InputMismatchException`), etc.
- **Errores de E/S** en la gestión de archivos, sockets, flujos, etc.

- En general cualquier error que nosotros intuyamos que pueda producirse durante la ejecución de nuestro programa.

El hecho de que nuestros programas presenten este tipo de fragilidades y se "rompan" con esa facilidad eclipsa el buen hacer que hayamos podido desarrollar durante la implementación de nuestro código. Si durante la presentación de una aplicación, ésta falla nada más empezar vamos a producir una muy mala impresión a nuestros clientes, jefes, alumnos, profesores, asistentes, etc. echando por tierra todo el esfuerzo que hayamos dedicado a nuestro trabajo. Hay que cuidar este tipo de detalles en el acabado de nuestro producto.

Este tipo de fallos será severamente penalizado.

En la siguiente tabla se muestran los criterios de corrección aplicables a cada ejercicio, y la puntuación máxima asignada al mismo.

Criterios de corrección	*Puntuación*
• Se recibe como parámetro desde la línea de órdenes (command line) en consola el archivo que va a ser encriptado, así como la clave de encriptación. • Se llevan a cabo las comprobaciones apropiadas con los parámetros recibidos. • Se tienen en cuenta y se gestionan todos los posibles errores que se han indicado en el enunciado. • Se genera un objeto cifrador apropiadamente. • Se genera correctamente un archivo cifrado basándose en las condiciones indicadas en el enunciado. • Explicaciones y capturas de funcionamiento en la ficha de la tarea. • No contiene elementos extraños, redundantes o sin sentido. • No se "rompe" con facilidad.	Hasta 5 puntos
• Se recibe como parámetro desde la línea de órdenes (command line) en consola el archivo que va a ser desencriptado, así como la clave de encriptación. • Se llevan a cabo las comprobaciones apropiadas con los parámetros recibidos. • Se tienen en cuenta y se gestionan todos los posibles errores que se han indicado en el enunciado. • Se genera un objeto cifrador apropiadamente. • Se desencripta correctamente el archivo basándose en las	Hasta 5 puntos

condiciones indicadas en el enunciado. • Explicaciones y capturas de funcionamiento en la ficha de la tarea. • No contiene elementos extraños, redundantes o sin sentido. • No se "rompe" con facilidad.	