En este ejercicio se genera una cadena de texto que se deja almacenada en un fichero encriptado, en la raíz del proyecto creado, con el nombre “fichero.cifrado”.

Para encriptar el fichero, se utiliza el algoritmo Rijndael o AES, con las especificaciones de modo y relleno siguientes: Rijndael/ECB/PKCS5Padding.

La clave, se genera de la siguiente forma:

Obteniendo un hash de un password (un String) con el algoritmo "SHA-256".
Se copia con el método Arrays.copyOf los 192 bits a un array de bytes (192/8 bytes).
Se utiliza la clase SecretKeySpec para generar una clave a partir del array de bytes.
Para probar el funcionamiento, el mismo programa accede al fichero encriptado para desencriptarlo e imprimir su contenido.

```
/**
 *
 * @author JMFC
 */
public class Main {

    public static void main(String[] args) throws IOException {
        String textoACifrar = "Este es un texto de prueba para cifrar y luego descifrar usando el algoritmo AES.";
        String password = "josemfc86";
        //Iniciamos el FileWriter como null
        FileWriter fw = null;
        File fichero = new File( pathname: "fichero");
        //Declara e incializa objeto tipo clave secreta
        SecretKeySpec skeySpec = null;
```

Iniciamos el programa declarando las variables para crear la clave SecretKeySpec, la contraseña, el fichero y el contenido del fichero.

```
    //escribimos en el contenido
    fw = new FileWriter( file: fichero);
    BufferedWriter bw = new BufferedWriter( out: fw);
    bw.write( str: textoACifrar);
    bw.close();
```

Escribimos el ficho con el contenido del texto a cifrar.

```
try {

    MessageDigest sha256 = MessageDigest.getInstance( algorithm: "SHA-256"); //Realizamos el hash sha-256.
    sha256.update( input: password.getBytes( charsetName: "UTF-8"));
    byte[] resumen = sha256.digest(); //Obtiene el resumen.
    //miro los bytes que tiene hashed y tiene 32Bytes.
    // System.out.println(resumen.length);
```

Obtenemos el hash del password mediante el algoritmo “SHA-256”. compruebo que el hash es de 256bites o 32bytes usando un array de bytes e imprimiendo por pantalla la longitud del array de esos 32bytes.

```java
//Clave de encriptación/desencriptación
public static SecretKeySpec crearClave(byte[] resumen) {
    try {

        byte[] raw192 = Arrays.copyOf( original: resumen,  newLength: 24);
        SecretKeySpec skeySpec = new SecretKeySpec( key: raw192,  algorithm: "AES");
        System.out.print( s: "La Clave es: ");
        mostrarBytes( buffer: skeySpec.getEncoded()); //metodo mostrarBytes para leer la clave
        System.out.println();
        return skeySpec;
    } catch (Exception e) {
        return null;
    }
}
```

Creo un método para crear la clave de encriptación/desencriptación al cual le pasamos el array de bytes del resumen, en dicho método creo un nuevo array de bytes en el que introducimos los 24 primeros del “resumen” copiandolos con el método “Arrays.copyOf”, creamos la clave con la clase “SecretKeySpec” pasándole como parámetros dicho array y el algoritmo “AES”, por último mostramos la clave llamando al método “mostrarBytes” al cual se le pasa como parámetro la clave como un array de bytes.

```java
//Este método nos cifra el fichero con el texto llano y nos devuelve un nuevo fichero cifrado.
public static File cifrar(SecretKeySpec skeySpec, File fichero)
        throws NoSuchPaddingException, NoSuchAlgorithmException, InvalidKeyException, IllegalBlockSizeException,
        BadPaddingException, IOException {
    FileInputStream fe = null; //fichero de entrada
    FileOutputStream fs = null; //fichero de salida
    int bytesLeidos;
    //Se Crea el objeto Cipher para cifrar, utilizando el algoritmo AES.
    try {
        Cipher cifrador = Cipher.getInstance( transformation: "AES/ECB/PKCS5Padding");
        //Se inicializa el cifrador en modo CIFRADO o ENCRIPTACIÓN.
        cifrador.init( opmode: Cipher.ENCRYPT_MODE,  key: skeySpec);

        System.out.println("\n2. Ciframos con AES/ECB/PKCS5Padding el fichero: " + fichero);
        //Encriptamos los datos.
        byte[] bufferCifrado;
        byte[] leidos = new byte[1000];
        try {
            fe = new FileInputStream( file: fichero); //objeto fichero de entrada

            //control de posible excepcion
        } catch (FileNotFoundException ex) {
            ex.getMessage();
        }

        fs = new FileOutputStream(fichero + ".cifrado"); //fichero de salida
        //lee el fichero de 1 en 1 y pasa los fragmentos leidos al cifrador
        bytesLeidos = fe.read( b: leidos,  off: 0,  len: 1000);

        //mientras no se llegue al final del fichero
        while (bytesLeidos != -1) {
            //pasa texto claro al cifrador y lo cifra, asignándolo a bufferCifrado
            bufferCifrado = cifrador.update( input: leidos,  inputOffset: 0,  inputLen: bytesLeidos);

            //Graba el texto cifrado en fichero
            fs.write( b: bufferCifrado);
            bytesLeidos = fe.read( b: leidos,  off: 0,  len: 1000);
        }
        bufferCifrado = cifrador.doFinal(); //Completa el cifrado
        fs.write( b: bufferCifrado); //Graba el final del texto cifrado, si lo hay

        //Cierra ficheros
        fe.close();
        fs.close();

        return new File(fichero + ".cifrado");
        //control de posibles excepciones
    } catch (NoSuchAlgorithmException ex) {
        ex.getMessage();
        return null;
    } catch (NoSuchPaddingException ex) {
        ex.getMessage();
        return null;
    } catch (InvalidKeyException ex) {
        ex.getMessage();
        return null;
    } catch (FileNotFoundException ex) {
        ex.getMessage();
        return null;
    } catch (IOException ex) {
        ex.getMessage();
        return null;
    } catch (IllegalBlockSizeException ex) {
        ex.getMessage();
        return null;
    } catch (BadPaddingException ex) {
        ex.getMessage();
        return null;
    }
    //Se inicializa el cifrador en modo CIFRADO o ENCRIPTACIÓN
}
```

Para cifrar el fichero creamos un método estático al cual llamamos “cifrarFichero”, este recibe como parámetro la clave y el fichero que va a ser encriptado, creamos un objeto de la clase Cipher

que nos permite cifrar utilizando el algoritmo AES/ECB/PKCS5Padding, iniciamos el cifrado y le pasamos la clave como parámetro, ciframos el fichero leyendo fragmentos de 1000 bytes, con un bucle “while” pasamos texto llano del array de bytes al cifrador y luego lo grabamos en el fichero cifrado, al terminar de leer el array salimos del bucle “while” completamos el cifrado y lo grabamos en el fichero. Finalmente este método nos devuelve el fichero cifrado con la extensión “.cifrado” y lo deja en el directorio raíz del proyecto.

```java
//este método nos descifra el fichero encriptado que le pasamos por parámetro.
public static void descifrar(SecretKeySpec skeySpec, File ficheroEncriptado)
        throws IOException, NoSuchPaddingException, NoSuchAlgorithmException,
        InvalidKeyException, IllegalBlockSizeException, BadPaddingException {
    //Ahora a la inversa, leo los datos encriptados.
    try {
        //fichero de entrada
        FileInputStream fe = null;
        String descifrar = null;
        byte[] arrayDescif = null;
        fe = new FileInputStream( file:ficheroEncriptado);

        Cipher cifrador = Cipher.getInstance( transformation:"AES/ECB/PKCS5Padding");

        //Cifrador en modo DESCIFRADO o DESENCRIPTACIÓN
        cifrador.init( opmode:Cipher.DECRYPT_MODE,  key:skeySpec);
        System.out.println( x:"\n... Aquí lee el fichero y lo muestra por pantalla");

        int bytesLeidos;
        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( b:buffer,  off:0,  len:1000);

        while (bytesLeidos != -1) {

            //pasa texto cifrado al cifrador y lo descifra, asignándolo a buffer
            arrayDescif = cifrador.update( input:buffer,  inputOffset:0,  inputLen:bytesLeidos);
            bytesLeidos = fe.read( b:buffer,  off:0,  len:1000);
        }

        descifrar = new String( bytes:arrayDescif) + new String( bytes:cifrador.doFinal());
        System.out.println("\n3. Leemos el fichero descifrado:\n\n" + descifrar);

        //cierra archivos
        fe.close();

        //contorl de posibles excepciones
    } catch (FileNotFoundException ex) {
        ex.getMessage();
    } catch (NoSuchAlgorithmException ex) {
        ex.getMessage();
    } catch (NoSuchPaddingException ex) {
        ex.getMessage();
    } catch (InvalidKeyException ex) {
        ex.getMessage();
    } catch (IOException ex) {
        ex.getMessage();
    } catch (IllegalBlockSizeException ex) {
        ex.getMessage();
    } catch (BadPaddingException ex) {
        ex.getMessage();
    }

}
```

Este método desencripta el fichero que le pasamos por parámetro junto con la clave, realizando operaciones similares al método encriptar, lo que hace es crear un objeto “Cipher” el cual

utilizaremos para descifrar el contenido del fichero cifrado mediante el modo “DECRYPT_MODE” y la clave que le pasamos por parámetro, leemos el fichero en fragmentos de 1000bytes, los cuales se le pasan al cifrador mediante un buffer para que lo descifre. Por último se crea un string para poder sacar por pantalla el contenido del fichero ya descifrado.

```
skeySpec = crearClave(resumen); //Creamos la clave.
File ficheroEncriptado = cifrar(skeySpec, fichero); //Ciframos el contenido del fichero y a la vez el método nos regresa el fichero cifrado.
descifrar(skeySpec, ficheroEncriptado); //Desciframos el contenido del fichero.
```

Finalmente en el método “main” llamamos al método “crearClave” para que nos genere la clave secreta y la guardamos en el objeto de la clase “SecretKeySpec”, luego creamos un objeto de la clase “File” donde guardaremos el fichero encriptado que se nos creará al llamar al método “cifrar”, para concluir hacemos la llamada al método “descifrar” que nos mostrar el contenido ya descifrado del fichero encriptado.