Programación de Servicios y procesos.

Tarea para PSP07.

Enunciado.

Ejercicio 1.

De igual manera a lo visto en el tema, ahora te proponemos un ejercicio que genere una cadena de texto y la deje almacenada en un fichero encriptado, en la raíz del proyecto hayas creado, con el nombre **`fichero.cifrado`**.

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

La clave, la debes generar de la siguiente forma:

- A partir de un número aleatorio con semilla la cadena del nombre de usuario + password.
- Con una longitud o tamaño 128 bits.

Para probar el funcionamiento, el mismo programa debe acceder al fichero encriptado para desencriptarlo e imprimir su contenido.

Como una constante a lo largo del curso, vuelvo a utilizar los materiales que nos proporciona el temario para la resolución del ejercicio.

Si bien es cierto, que del algoritmo propuesto no hay un ejemplo concreto, creo que es más que suficiente como para adaptarlos a lo pedido y resolver, comprendiendo los conceptos mostrado en el tema.

CifradoAES.java

```
/*
 * TAREA PSP07. EJERCICIO 1.
 * De igual manera a lo visto en el tema, ahora te proponemos un
ejercicio
 * que genere una cadena de texto y la deje almacenada en un fichero
encriptado,
 * en la raíz del proyecto hayas creado, con el nombre
fichero.cifrado.
 * Para encriptar el fichero, utilizarás el algoritmo Rijndael o AES,
con
 * las especificaciones de modo y relleno siguientes:
Rijndael/ECB/PKCS5Padding.
 * La clave, la debes generar de la siguiente forma:
 *  - A partir de un número aleatorio con semilla la cadena del
 *    nombre de usuario + password.
 *  - Con una longitud o tamaño 128 bits.
 * Para probar el funcionamiento, el mismo programa debe acceder al
 * fichero encriptado para desencriptarlo e imprimir su contenido.

 *
 * RECORDAR  COMENTAR EL PACKAGE SI SE QUIERE COMPILAR FUERA DE
NETBEANS.
 */
package cifradoaes;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
```

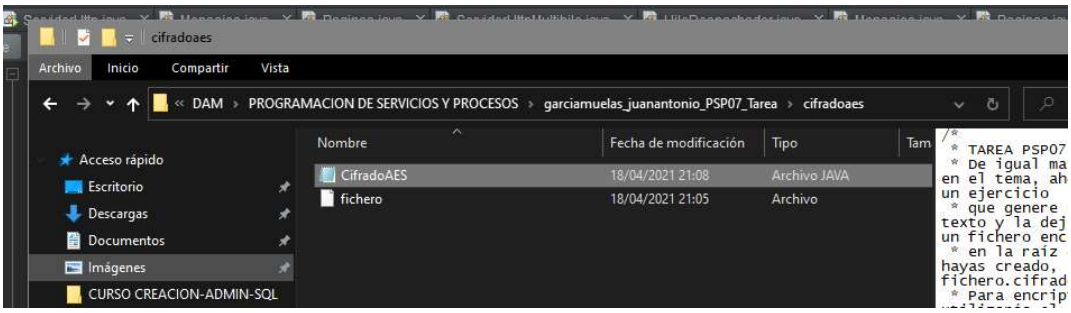
```java
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.IOException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;
import java.util.Scanner;
import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.KeyGenerator;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.SecretKey;


/**
 * @author juang <juangmuelas@gmail.com>
 * @since 18/04/2021
 * @version 1
 */
public class CifradoAES {

    /**
     * Para la realización de este ejercicio y a falta de ejemplos algo más
     * concretos sobre el algoritmo solicitado, uso parte del ejemplo
     * que nos aportan en el apartado 4.7 del tema, para encriptaciones de
     * clave simétrica, intentando seguir así los conceptos del tema y
     * no desviarnos con ejemplos encontrados en webs externas, que puedan
     * llegar a confundir más en el aprendizaje.
     */
    public static void main(String[] args) {

System.out.println("============================================");
        System.out.println("PROGRAMA PARA ENCRIPTAR ARCHIVOS MEDIANTE
AES");

System.out.println("============================================");
        /**
         * Solicitar mediante el uso de Scanner usuario y pass
         * @param  usuario String que recoge el nombre de usuario
         * @param  pass String que recoge pass de usuario
         * @param  concatenados String que recoge semilla para clave
         */
        String usuario;
        String pass;
        String concatenados;

        System.out.println("INDIQUE SU USUARIO: ");
        Scanner teclado = new Scanner (System.in);
        usuario = teclado.nextLine ();
        System.out.println("INDIQUE SU CONTRASEÑA: ");
        pass = teclado.nextLine ();

        //Usamos los datos recibidos para nuestra clave
        concatenados = usuario+pass;
        System.out.println(concatenados);

        /**
         * @param  clave objeto SecretKey inicializado
         */
        SecretKey clave = null;
```

```
        //llama a los métodos que encripta/desencripta un fichero
        try {
           /**
            * Llama al método que encripta el fichero que se pasa como
            * parámetro y se añade la variable con la nueva linea
            */
            clave = cifrarFichero("fichero",concatenados);
            //Llama al método que desencripta el fichero pasado como primer
parámetro
            descifrarFichero("fichero.cifrado",
clave,"fichero.descifrado");
        } catch (Exception e) {
            e.printStackTrace();
        }
        muestraContenido("fichero.descifrado");
    } //Fin main

    /**
     * Seguimos manteniendo el guión del ejemplo del tema, pero adapatandolo
     * al algoritmo pedido.
     *
     * @param file
     * @param password
     * @return
     * @throws NoSuchAlgorithmException
     * @throws NoSuchPaddingException
     * @throws FileNotFoundException
     * @throws IOException
     * @throws IllegalBlockSizeException
     * @throws BadPaddingException
     * @throws InvalidKeyException
     */

    //método que encripta el fichero que se pasa como parámetro
    //devuelve el valor de la clave privada utilizada en encriptación
    //El fichero encriptado lo deja en el archivo de nombre fichero.cifrado
    //en el mismo directorio
    private static SecretKey cifrarFichero(String file, String
password) throws NoSuchAlgorithmException, NoSuchPaddingException,
FileNotFoundException, IOException, IllegalBlockSizeException,
BadPaddingException, InvalidKeyException {
        FileInputStream fe = null; //fichero de entrada
        FileOutputStream fs = null; //fichero de salida
        int bytesLeidos;

        //1. Crear e inicializar clave
        System.out.println("1.- Genera clave Rijndael o AES");
        //crea un objeto para generar la clave usando algoritmo Rijndael o AES
        KeyGenerator keyGen = KeyGenerator.getInstance("AES");
        /**
        * Utilizamos el objeto SecureRandom (la documentación es abundante)
        * para poder usar la semilla y gracias a un ejemplo, usaremos el
        * algoritmo SHA1PRNG,mostrado en un ejemplo de la junta de Andalucia,
        * donde nos muestra como generar números aleatorios fuertes en Java.
        * Luego el método setSeed recoge nuestro parámetro.
        */

        SecureRandom secureRandom =
SecureRandom.getInstance("SHA1PRNG");
        secureRandom.setSeed(password.getBytes());
        keyGen.init(128,secureRandom); //se indica el tamaño de la clave
        SecretKey clave = keyGen.generateKey(); //genera la clave privada
```

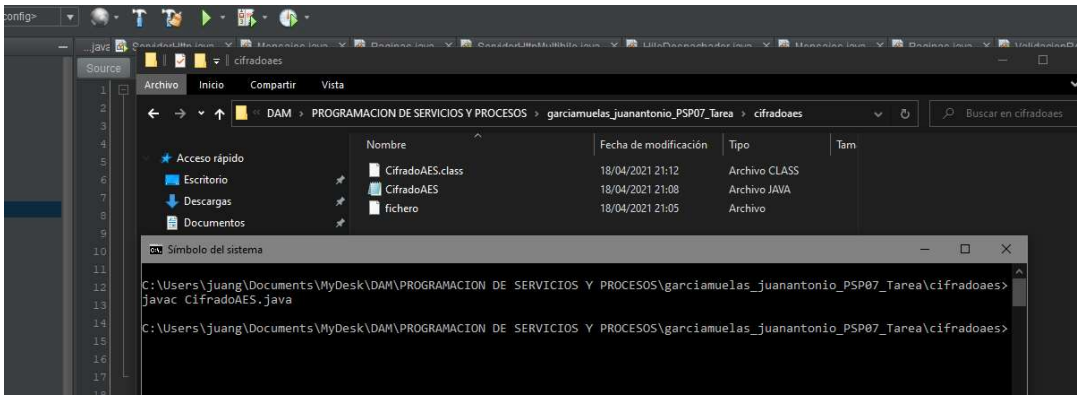
```
        //Puede comentarse, no es necesario para la ejecución pero es útil:
        System.out.println("Clave");
        mostrarBytes(clave.getEncoded()); //muestra la clave
        System.out.println();

        //Se Crea el objeto Cipher para cifrar, utilizando el algoritmo AES
        Cipher cifrador = Cipher.getInstance("AES/ECB/PKCS5Padding");
        //Se inicializa el cifrador en modo CIFRADO o ENCRIPTACIÓN
        cifrador.init(Cipher.ENCRYPT_MODE, clave);
        System.out.println("2.- Cifrar con AES el fichero: " + file
                + ", y dejar resultado en " + file + ".cifrado");
        //declaración  de objetos
        byte[] buffer = new byte[1000]; //array de bytes
        byte[] bufferCifrado;
        fe = new FileInputStream(file); //objeto fichero de entrada
        fs = new FileOutputStream(file + ".cifrado"); //fichero de
salida
       //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 claro al cifrador y lo cifra, asignándolo a
bufferCifrado
            bufferCifrado = cifrador.update(buffer, 0, bytesLeidos);
            fs.write(bufferCifrado); //Graba el texto cifrado en
fichero
            bytesLeidos = fe.read(buffer, 0, 1000);
        }
        bufferCifrado = cifrador.doFinal(); //Completa el cifrado
        fs.write(bufferCifrado); //Graba el final del texto cifrado,
si lo hay
        //Cierra ficheros
        fe.close();
        fs.close();
        return clave;
    }  //Fin método cifrarFichero
    //método que desencripta el fichero pasado como primer parámetro file1
    //pasándole también la clave privada que necesita para desencriptar, key
    //y deja el fichero desencriptado en el tercer parámetro file2

    private static void descifrarFichero(String file1, SecretKey key,
String file2) throws NoSuchAlgorithmException, NoSuchPaddingException,
FileNotFoundException, IOException, IllegalBlockSizeException,
BadPaddingException, InvalidKeyException {
        FileInputStream fe = null; //fichero de entrada
        FileOutputStream fs = null; //fichero de salida
        int bytesLeidos;
        Cipher cifrador = Cipher.getInstance("AES/ECB/PKCS5Padding");
        //3.- Poner cifrador en modo DESCIFRADO o DESENCRIPTACIÓN
        cifrador.init(Cipher.DECRYPT_MODE, key);
        System.out.println("3.- Descifrar con AES el fichero: " +
file1
                + ", y dejar en  " + file2);
        fe = new FileInputStream(file1);
        fs = new FileOutputStream(file2);
        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);
        }
        bufferClaro = cifrador.doFinal(); //Completa el descifrado
        fs.write(bufferClaro); //Graba el final del texto claro, si lo
hay
        //cierra archivos
        fe.close();
        fs.close();
    } //Fin método descifrarFichero

    //método que muestra bytes
    public static void mostrarBytes(byte[] buffer) {
      System.out.write(buffer, 0, buffer.length);
    }

    /**
     * Hasta aquí hemos seguido el ejemplo para clave simétrica adaptado,
     * pero nos queda poder leer el fichero.
     * Para ello reutilizo un método usado en un curso anterior.
     *
     * @param archivo
     */

    private static void muestraContenido(String archivo){
        File fichero = null;
        FileReader fr = null;
        BufferedReader br = null;

        System.out.println("4.- Leer el archivo descifrado "+
archivo);
        try {

            fichero = new File (archivo);
            fr = new FileReader (fichero);
            br = new BufferedReader(fr);

            String linea;
            while((linea=br.readLine())!=null)
                System.out.println(linea);
        }catch(Exception e){
            e.printStackTrace();
        }finally{
            /**
             * Cerramos el fichero
             */
            try{
                if( null != fr ){
                    fr.close();
                }
            }catch (Exception e2){
                e2.printStackTrace();
            }
        }
    } //Fin metodo leerFichero
} //Fin clase CifradoAES
```

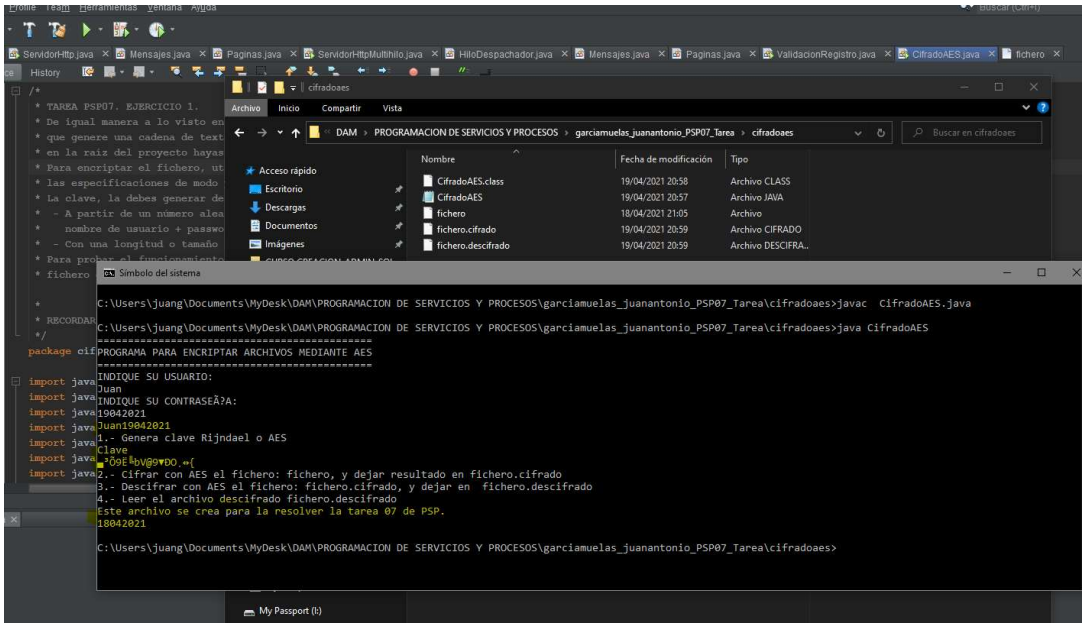
Creados los archivos fichero y CifradoAES.java, se compila este mediante `javac CifradoAES.java`



Ejecutamos en nuestra consola mediante `java CifradoAES`



Tras incluir nombre de usuario y contraseña, genera los archivos solicitados y muestra en la misma consola el contenido finalmente:



Como podemos comprobar, coincide con el contenido guardado.