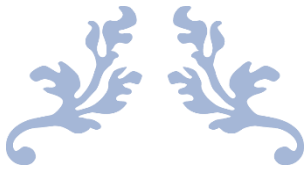

TAREA PSP07

Módulo de Programación de Servicios y Procesos en modalidad a distancia del I.E.S. Augusto González de Linares.

10 DE FEBRERO DE 2023

DIEGO GONZÁLEZ GARCÍA

Índice

Enunciado.

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:

- Obtener un hash de un password (un String) con el algoritmo "SHA-256".
- Copiar con el método Arrays.copyOf los 192 bits a un array de bytes (192/8 bytes)
- Utilizar la clase SecretKeySpec para generar una clave a partir del array de bytes.

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

1. Generación de clave de encriptación.

```java
/**
 * Método que genera una clave con encriptación AES
 *
 * @param password Clave original
 * @return Clave de encriptación
 */
private static SecretKeySpec crearClaveAES(String password) {
    MessageDigest messageDigest;
    byte[] digest;
    byte[] key;
    SecretKeySpec secretKeySpec;
    try {
        messageDigest = MessageDigest.getInstance( algorithm: "SHA-256");   //Instancia con el algoritmo SHA-256
        digest = messageDigest.digest( input: password.getBytes( charset: StandardCharsets.UTF_8));   //Hash con el algoritmo SHA-256
        key = Arrays.copyOf( original: digest,  newLength: 24);    //Copia los bits de la clave
        secretKeySpec = new SecretKeySpec(key,  algorithm: "AES");  //Crea el objeto SecretKeySpec
        return secretKeySpec;
    } catch (NoSuchAlgorithmException ex) {
        ex.printStackTrace();
    }
    return null;
}
```

El método estático *crearClaveAES*, recibe como parámetro una contraseña para generar la clave de encriptación.

Se utiliza un algoritmo *SHA-256*, copiando los 192 bits a un array de bytes y se utiliza la clase *SecretKeySpec* para generar una clave de encriptación privada a partir del array generado previamente.

Una vez generada la clave, se devuelve como objeto *SecretKeySpec*.

2. Cifrado del mensaje.

```java
/**
 * Método que encripta un mensaje y lo guarda en un fichero
 *
 * @param secretKeySpec Clave de encriptación
 * @param file Fichero de salida para el mensaje
 * @param mensaje Texto a cifrar
 * @return Mensaje cifrado
 */
private static String cifrarMensaje(SecretKeySpec secretKeySpec, File file, String mensaje) {
    byte[] mensajeCifrado;
    Cipher cipher;
    FileOutputStream fos = null;
    try {
        cipher = Cipher.getInstance( transformation:"AES/ECB/PKCS5Padding"); //Crea el objeto Cipher para cifrar
        cipher.init( opmode:Cipher.ENCRYPT_MODE,  key:secretKeySpec);    //Se inicializa el cifrador en modo encriptación
        mensajeCifrado = cipher.doFinal( input:mensaje.getBytes()); //Cifra el mensaje
        fos = new FileOutputStream(file);  //Crea el objeto FileOutputStream para el mensaje cifrado en un fichero
        fos.write( b:mensajeCifrado);  //Escribe el fichero
        fos.close();    //Cierra la comunicación
        return new String( bytes:mensajeCifrado);
    } catch (NoSuchAlgorithmException ex) {
        ex.printStackTrace();
    } catch (NoSuchPaddingException ex) {
        ex.printStackTrace();
    } catch (InvalidKeyException ex) {
        ex.printStackTrace();
    } catch (IllegalBlockSizeException ex) {
        ex.printStackTrace();
    } catch (BadPaddingException ex) {
        ex.printStackTrace();
    } catch (FileNotFoundException ex) {
        ex.printStackTrace();
    } catch (IOException ex) {
        ex.printStackTrace();
    } finally {
        try {
            fos.close();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
    return null;
}
```

El método estático *cifrarMensaje*, recibe como parámetro la clave de encriptación, el fichero de destino para el mensaje encriptado y el texto del mensaje a encriptar.

Se utiliza un algoritmo *AES*, con las especificaciones *AES/ECB/PKCS5Padding*, con lo que se crea un cifrador que se encarga de encriptar el mensaje recibido. A continuación, el mensaje encriptado es guardado en el archivo indicado en el parámetro.

Una vez almacenado el mensaje cifrado en el archivo, se devuelve el contenido del mensaje cifrado.

3. Descifrado del mensaje.

```java
/**
 * Método que desencripta un fichero y devuelve su mensaje
 *
 * @param secretKeySpec Clave de encriptación
 * @param file Fichero encriptado
 * @return Mensaje descifrado
 */
private static String descifrarMensaje(SecretKeySpec secretKeySpec, File file) {
    byte[] mensajeCifrado;
    byte[] mensajeDescifrado;
    Cipher cipher;
    FileInputStream fis = null;
    int tamanoFichero;
    try {
        cipher = Cipher.getInstance( transformation:"AES/ECB/PKCS5Padding"); //Crea el objeto Cipher para descifrar
        cipher.init( opmode:Cipher.DECRYPT_MODE,  key:secretKeySpec);    //Se inicializa el cifrador en modo desencriptación
        fis = new FileInputStream(file);    //Crea el objeto FileInputStream para leer el fichero cifrado
        tamanoFichero = (int) file.length();    //Lee el tamaño del fichero
        mensajeCifrado = new byte[tamanoFichero];   //Crea un array de bytes
        fis.read( b:mensajeCifrado);   //Lee el archivo cifrado
        mensajeDescifrado = cipher.doFinal( input:mensajeCifrado);   //Descifra el mensaje
        return new String( bytes:mensajeDescifrado);
    } catch (FileNotFoundException ex) {
        ex.printStackTrace();
    } catch (NoSuchAlgorithmException ex) {
        ex.printStackTrace();
    } catch (NoSuchPaddingException ex) {
        ex.printStackTrace();
    } catch (InvalidKeyException ex) {
        ex.printStackTrace();
    } catch (IllegalBlockSizeException ex) {
        ex.printStackTrace();
    } catch (BadPaddingException ex) {
        ex.printStackTrace();
    } catch (IOException ex) {
        ex.printStackTrace();
    } finally {
        try {
            fis.close();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
    return null;
}
```

El método estático *descifrarMensaje*, recibe como parámetro la clave de encriptación y el fichero que contiene el mensaje encriptado.

Utilizando la clave de encriptación que se utilizó previamente para encriptar el mensaje, se crea un descifrador que se encarga de desencriptar el mensaje que contiene el archivo indicado en el parámetro.

Una vez que el mensaje es descifrado, se devuelve su contenido.

4. Método main.

```java
/**
 * Método main
 *
 * @param args the command line arguments
 */
public static void main(String[] args) {
    String password = "Mi clave supersecreta";
    String mensaje = "Este es un texto de prueba, que va a ser cifrado, "
            + "guardado en un fichero, leido el fichero y mostrado su "
            + "contenido por pantalla.";

    String mensajeCifrado;
    String mensajeDescifrado;

    SecretKeySpec secretKeySpec = null;
    File file = new File( pathname: "fichero.cifrado");    //Fichero cifrado

    System.out.println("\tMensaje original:\n" + mensaje); //Muestra mensaje cifrado
    secretKeySpec = crearClaveAES(password);    //Genera la clave de encriptación
    mensajeCifrado = cifrarMensaje(secretKeySpec, file, mensaje);   //Cifra el mensaje
    System.out.println("\n\tMensaje cifrado:\n" + mensajeCifrado); //Muestra mensaje cifrado
    mensajeDescifrado = descifrarMensaje(secretKeySpec, file);  //Descifra el mensaje
    System.out.println("\n\tMensaje descifrado:\n" + mensajeDescifrado);  //Muestra mensaje descifrado

}
```

El método *main* se encarga de definir el mensaje a cifrar, la clave original para generar el cifrado y el fichero donde se guarda el mensaje encriptado.

Tras esto, gestiona las llamadas a los distintos métodos explicados anteriormente y muestra el resultado de cada uno de ellos por pantalla.