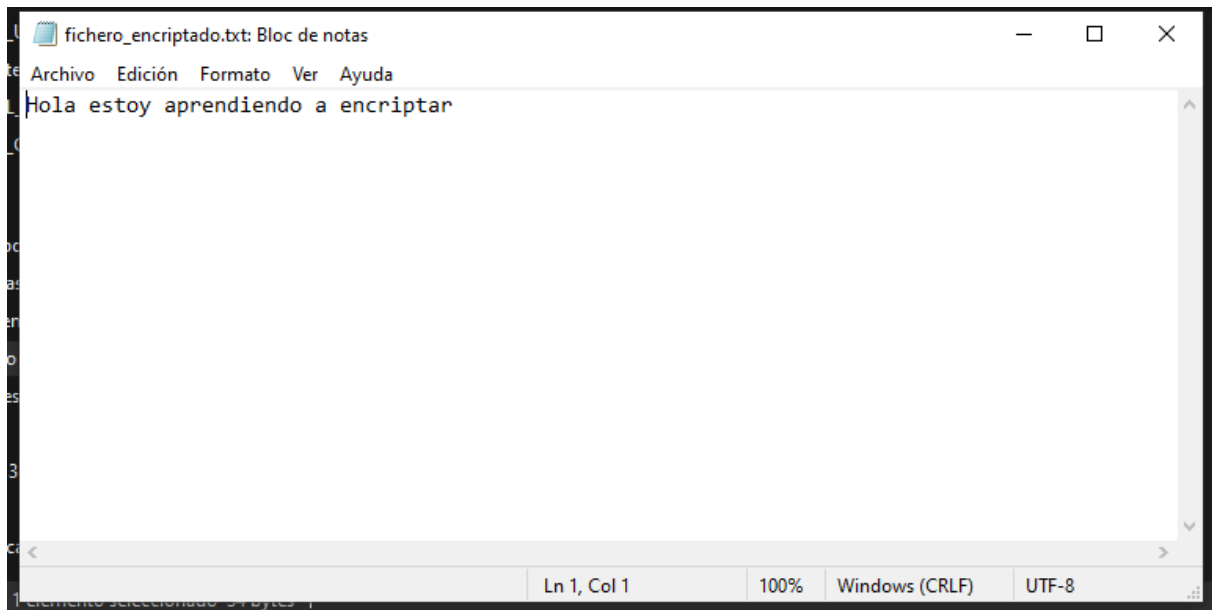


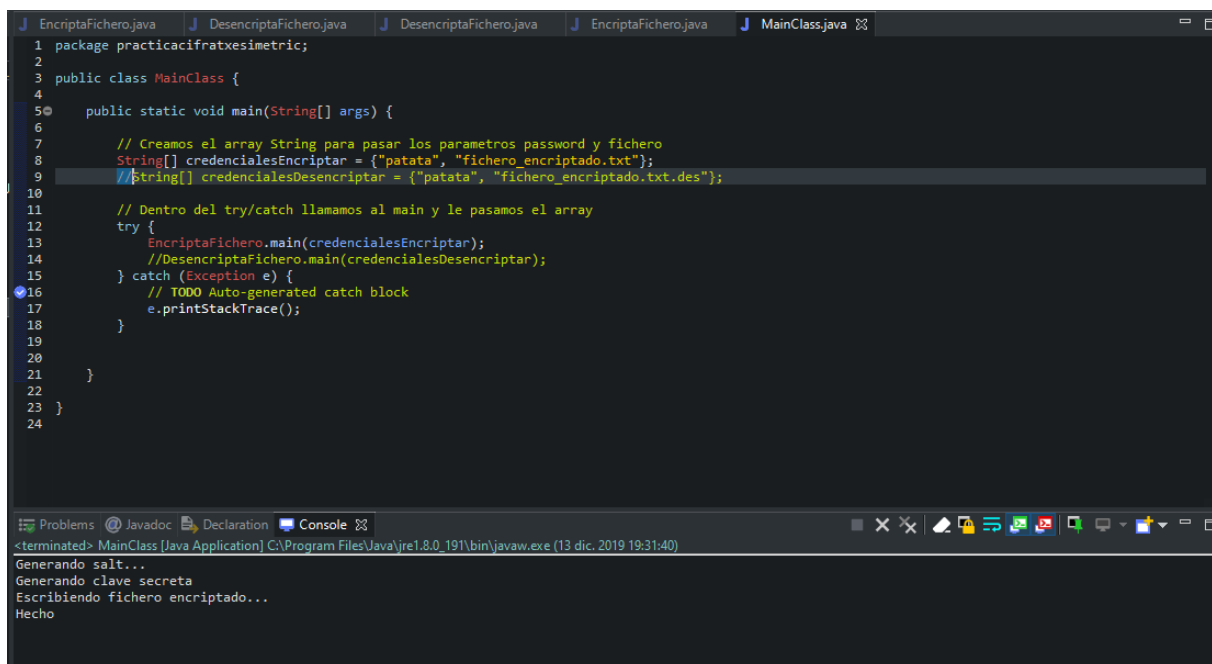
## PRÀCTICA D'ENCRIPTACIÓ

En aquesta primera part provarem les funcions encriptar i desencriptar en Java per encriptar un fitxer. Cal cridar les classes tot passant per paràmetre el password i el fitxer a encriptar.

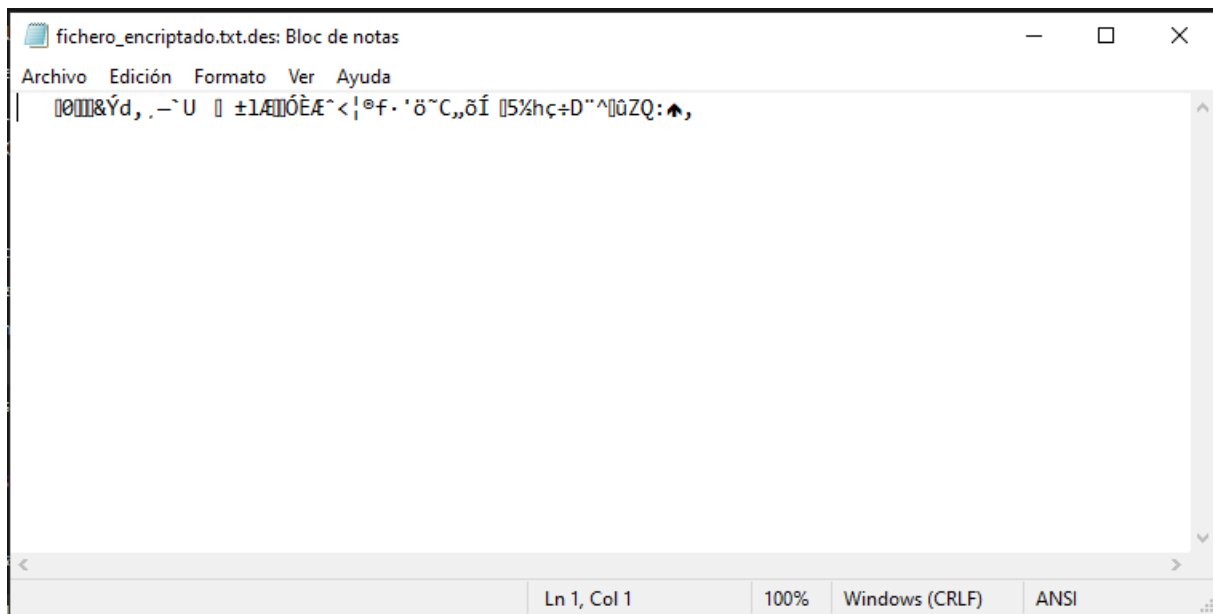
1. Mostra el contingut del fitxer a encriptar.



2. Crea la classe i executa EncriptaFichero. Indica els paràmetres que passes.

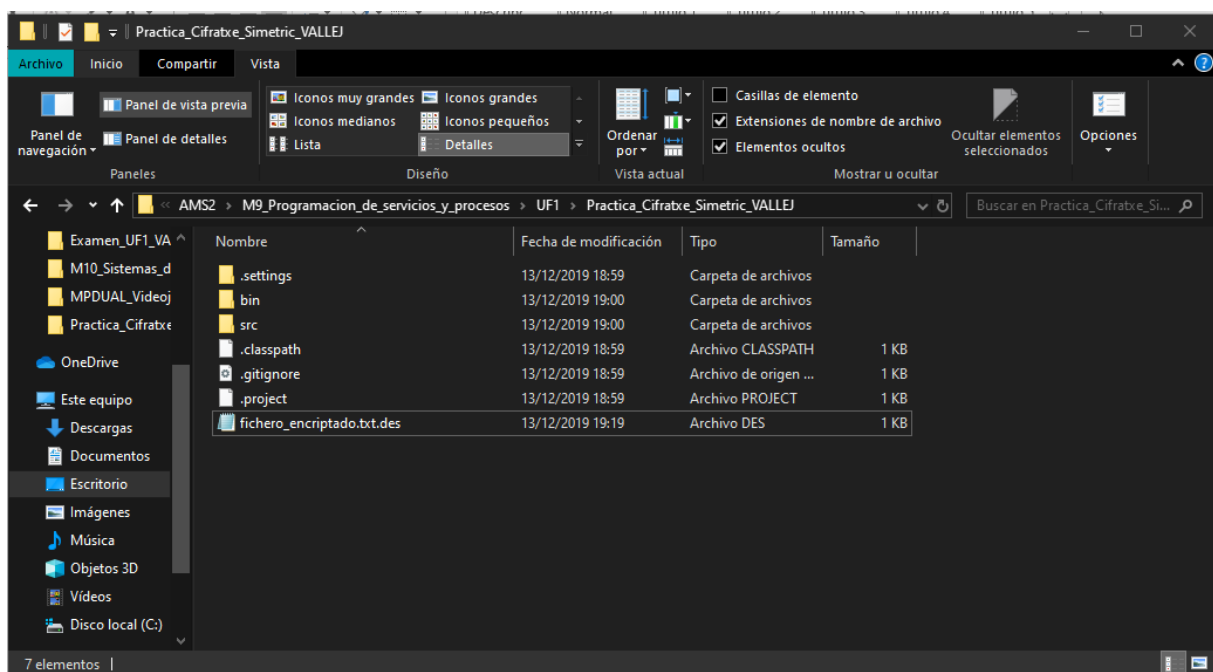


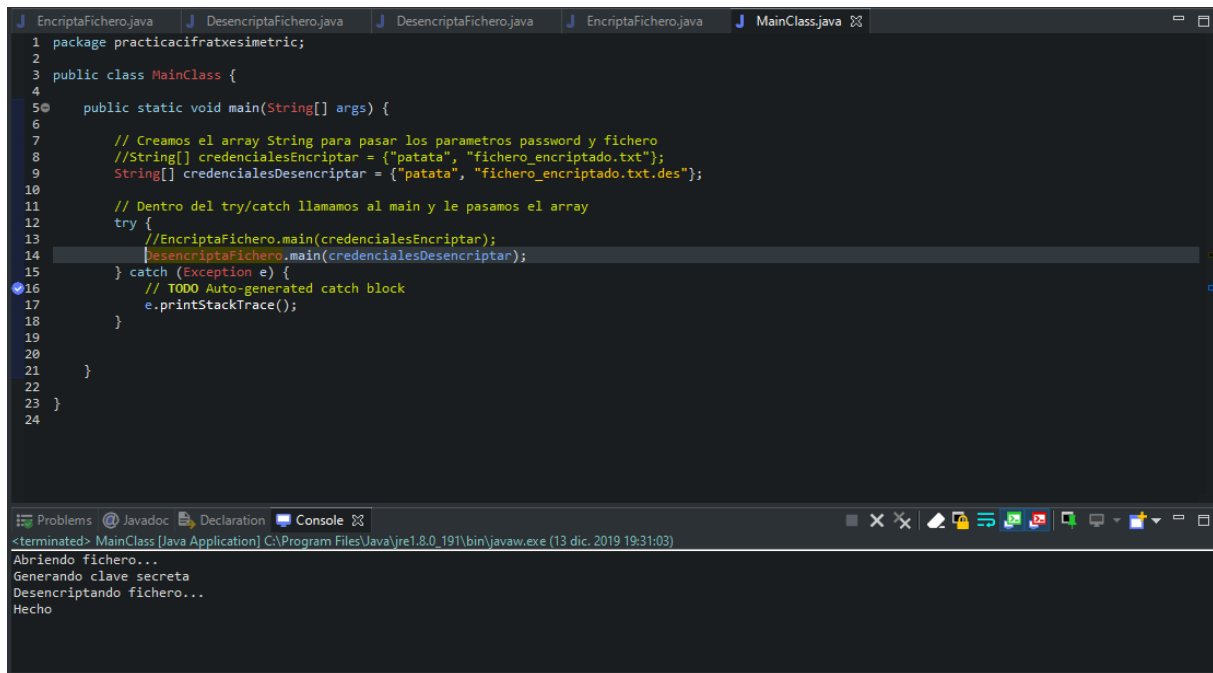
3. Mostra el resultat del fitxer encriptat.



4. Crea la classe i executa DesencriptaFichero. Indica els paràmetres que passes.

- Primer esborrem el fitxer natiu o que no esta encriptat perquè al desencriptar generarà un de nou que serà com el natiu.

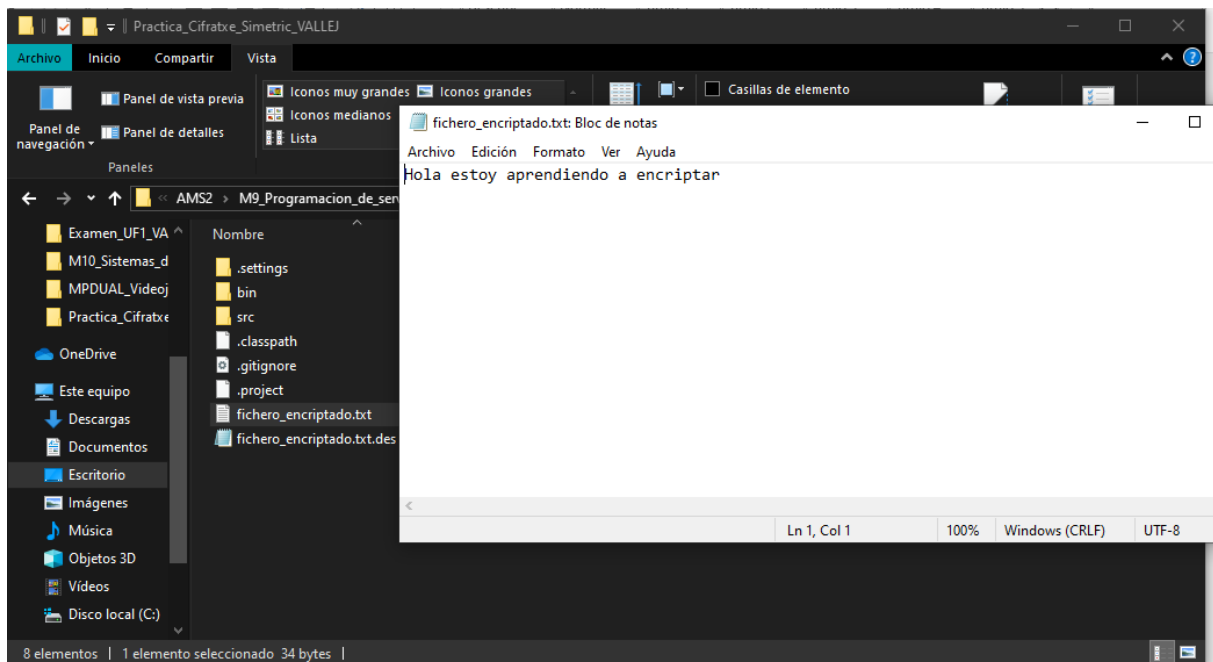




```
1 package practiacifratxesimetric;  
2  
3 public class MainClass {  
4  
5     public static void main(String[] args) {  
6  
7         // Creamos el array String para pasar los parametros password y fichero  
8         //String[] credencialesEncriptar = {"patata", "fichero_encriptado.txt"};  
9         String[] credencialesDesencriptar = {"patata", "fichero_encriptado.txt.des"};  
10  
11         // Dentro del try/catch llamamos al main y le pasamos el array  
12         try {  
13             //EncriptaFichero.main(credencialesEncriptar);  
14             DescriptaFichero.main(credencialesDesencriptar);  
15         } catch (Exception e) {  
16             // TODO Auto-generated catch block  
17             e.printStackTrace();  
18         }  
19  
20     }  
21 }  
22  
23 }  
24
```

Problems Javadoc Declaration Console  
<terminated> MainClass [Java Application] C:\Program Files\Java\jre1.8.0\_191\bin\javaw.exe (13 dic, 2019 19:31:03)  
Abriendo fichero...  
Generando clave secreta  
Descriptando fichero...  
Hecho

5. Mostra el resultat del fitxer desencriptat.



6. Indica el mètode l'algorisme de xifratge que estan emprant i si es clau simètrica o asimètrica.

- Utilitza el algorisme Password Based Encryption con MD5 i DES. La clau es simètrica, perquè els dos utilitzen la mateixa clau.

```
51         fichero_encriptado = new DataOutputStream(  
52             new FileOutputStream(args[2]));  
53  
54         //Generamos un salt aleatorio  
55         System.out.print("Generando salt...");  
56         SecureRandom sr = new SecureRandom();  
57         byte[] salt = new byte[8];  
58         sr.nextBytes(salt);  
59  
60         // Generamos una clave secreta a  
61         // partir del password  
62         System.out.print("\rGenerando clave secreta");  
63         PBEKeySpec objeto_password =  
64             new PBEKeySpec(args[0].toCharArray());  
65         SecretKeyFactory skf =  
66             SecretKeyFactory.getInstance("PBESWithMD5AndDES");  
67         SecretKey clave_secreta =  
68             skf.generateSecret(objeto_password);  
69  
70         // Generamos los parametros de PBESParameterSpec  
71         PBESParameterSpec pbeSpec =  
72             new PBESParameterSpec(salt, ITERACIONES);  
73  
74         // Generamos el cifrador  
75         Cipher cifrador =  
76             Cipher.getInstance("PBESWithMD5AndDES");  
77         cifrador.init(Cipher.ENCRYPT_MODE, clave_secreta, pbeSpec);
```

Codigo Fuente Encriptar:

```
package practiacifratxesimetric;
```

```
import java.io.*;  
import java.security.*;  
import javax.crypto.*;  
import javax.crypto.spec.*;
```

```
public class EncriptaFichero {  
    public static final int ITERACIONES = 1024;  
    public static final int TAMANO_SALT_BYTES = 8;  
    public static final int TAMANO_BUFFER = 1024;  
  
    public static void main(String args[]) throws Exception {  
        // Comprobacion de argumentos  
        if (args.length < 2 || args.length > 3) {  
            System.out.println(  
                "Para encriptar indique " + "<password> <fichero_plano> " +  
                "<fichero_encriptar> como argumento");  
            return;  
        }  
    }  
}
```

```
}
if (args.length > 2 && args[2].endsWith(".des")) {
    System.out.println("Los ficheros encriptados" + " deben tener la extension
.des");

    return;
}

// Abrimos los ficheros
FileInputStream fichero_plano = new FileInputStream(args[1]);
DataOutputStream fichero_encriptado;
if (args.length == 2)
    fichero_encriptado = new DataOutputStream(new FileOutputStream(args[1]
+ ".des"));
else
    fichero_encriptado = new DataOutputStream(new
FileOutputStream(args[2]));

// Generamos un salt aleatorio
System.out.print("Generando salt...");
SecureRandom sr = new SecureRandom();
byte[] salt = new byte[8];
sr.nextBytes(salt);

// Generamos una clave secreta a partir del password
System.out.print("\rGenerando clave secreta");
PBEKeySpec objeto_password = new PBEKeySpec(args[0].toCharArray());
SecretKeyFactory skf = SecretKeyFactory.getInstance("PBEWithMD5AndDES");
SecretKey clave_secreta = skf.generateSecret(objeto_password);

// Generamos los parametros de PBEPParameterSpec
PBEPParameterSpec pbeps = new PBEPParameterSpec(salt, ITERACIONES);

// Generamos el cifrador
Cipher cifrador = Cipher.getInstance("PBEWithMD5AndDES");
cifrador.init(Cipher.ENCRYPT_MODE, clave_secreta, pbeps);

// Escribimos en el fichero encriptado los parametros encoded
System.out.print("\rEscribiendo fichero" + " encriptado... ");
AlgorithmParameters ap = cifrador.getParameters();
byte[] encoded = ap.getEncoded();
fichero_encriptado.writeInt(encoded.length);
fichero_encriptado.write(encoded);

// Escribimos en el fichero encriptado los datos del fichero plano
byte[] buffer_plano = new byte[TAMANO_BUFFER];
int leidos = fichero_plano.read(buffer_plano);
while (leidos > 0) {
    byte[] buffer_encriptado = cifrador.update(buffer_plano, 0, leidos);
    fichero_encriptado.write(buffer_encriptado);
    leidos = fichero_plano.read(buffer_plano);
}
```

```
    }  
    fichero_encriptado.write(cifrador.doFinal());  
  
    // Cerramos los ficheros  
    fichero_plano.close();  
    fichero_encriptado.close();  
    System.out.println("\rHecho");  
}  
}
```

### Codigo Fuente Desencriptar:

```
package practicacifratxesimetric;  
  
import java.io.*;  
import java.security.*;  
import javax.crypto.*;  
import javax.crypto.spec.*;  
  
public class DescriptaFichero {  
    public static final int ITERACIONES = 1024;  
    public static final int TAMANO_BUFFER = 1024;  
  
    public static void main(String args[]) throws Exception {  
        // Comprobacion de argumentos  
        if (args.length < 2 || args.length > 3) {  
            System.out.println("Indique <password> " + " <fichero_encriptado>  
[<fichero_plano>]" + " como argumento");  
            return;  
        }  
        if (!args[1].endsWith(".des")) {  
            System.out.println("Los ficheros encriptados" + " deben tener la extension .des");  
            return;  
        }  
  
        // Abrimos los ficheros  
        System.out.print("Abriendo fichero...");  
        DataInputStream fichero_encriptado = new DataInputStream(new FileInputStream(args[1]));  
        FileOutputStream fichero_plano;  
        if (args.length == 2)  
            fichero_plano = new FileOutputStream(args[1].substring(0, args[1].length() - 4));  
        else  
            fichero_plano = new FileOutputStream(args[2]);  
  
        // Generamos una clave secreta a partir del password  
        System.out.print("\rGenerando clave secreta");  
        PBEKeySpec objeto_password = new PBEKeySpec(args[0].toCharArray());  
        SecretKeyFactory skf = SecretKeyFactory.getInstance("PBEWithMD5AndDES");  
        SecretKey clave_secreta = skf.generateSecret(objeto_password);
```

```
// Leemos los parametros encoded
int longitud_encoded = fichero_encriptado.readInt();
byte[] encoded = new byte[longitud_encoded];
fichero_encriptado.read(encoded);
AlgorithmParameters ap =
AlgorithmParameters.getInstance("PBEWithMD5AndDES");
ap.init(encoded);

// Creamos el cifrador
Cipher cifrador = Cipher.getInstance("PBEWithMD5andDES");
cifrador.init(Cipher.DECRYPT_MODE, clave_secreta, ap);

// Desencriptamos el contenido del fichero encriptado y lo pasamos al fichero
// plano
System.out.print("\rDesencriptando fichero...");
byte[] buffer = new byte[TAMANO_BUFFER];
int bytes_leidos = fichero_encriptado.read(buffer);
while (bytes_leidos > 0) {
    fichero_plano.write(cifrador.update(buffer, 0, bytes_leidos));
    bytes_leidos = fichero_encriptado.read(buffer);
}
fichero_plano.write(cifrador.doFinal());

// Cerramos los ficheros
fichero_encriptado.close();
fichero_plano.close();
System.out.println("\rHecho");
}
}
```