



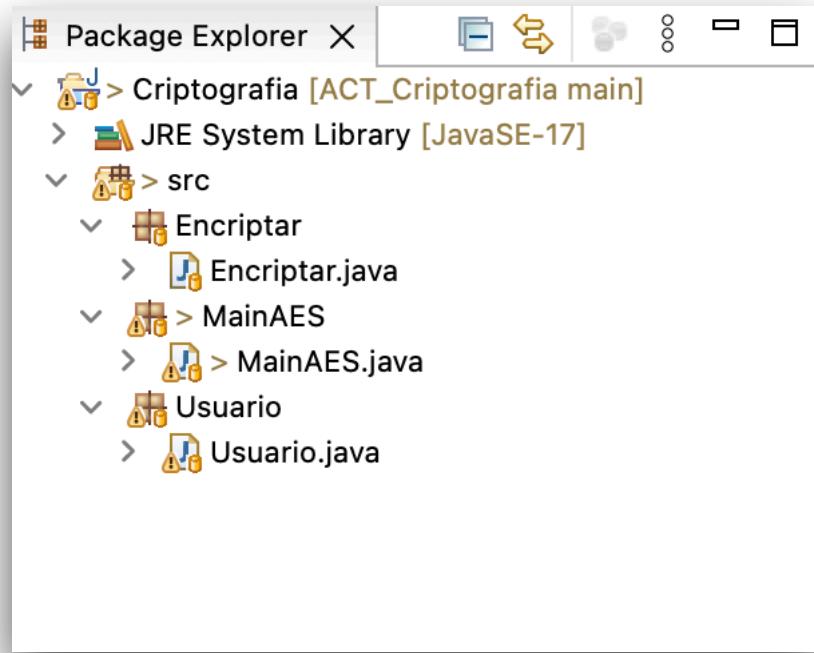
## AE-3. Criptografía

Sergio Palmero—Christian Casado

Url del proyecto:

[https://github.com/sergiopalmero87/Criptografia\\_PSP](https://github.com/sergiopalmero87/Criptografia_PSP):

## Estructura del proyecto con sus paquetes y clases.



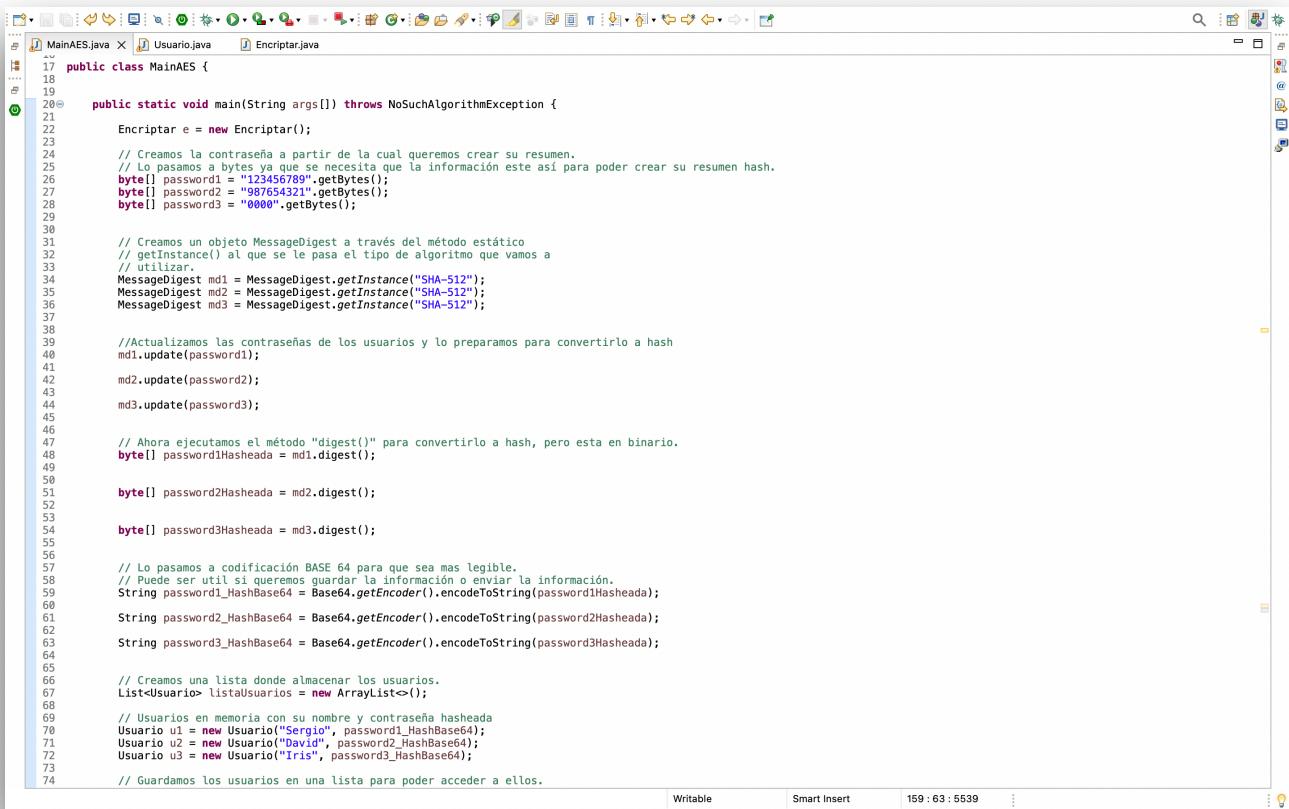
## Clase Usuario.

```
1 package MainAES;
2 import java.util.Objects;
3 import javax.persistence.Entity;
4 import javax.persistence.GeneratedValue;
5 import javax.persistence.GenerationType;
6 import javax.persistence.Id;
7
8 @Entity
9 @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
10 public class Usuario {
11     private String nombre;
12     private String password;
13
14     public Usuario(String nombre, String password) {
15         super();
16         this.nombre = nombre;
17         this.password = password;
18     }
19
20     //Getters and Setters
21     public String getNombre() {
22         return nombre;
23     }
24
25     public void setNombre(String nombre) {
26         this.nombre = nombre;
27     }
28
29     public String getPassword() {
30         return password;
31     }
32
33     public void setPassword(String password) {
34         this.password = password;
35     }
36
37     //Equals and hashCode
38     @Override
39     public int hashCode() {
40         return Objects.hash(nombre, password);
41     }
42
43     @Override
44     public boolean equals(Object obj) {
45         if (this == obj)
46             return true;
47         if (obj == null)
48             return false;
49         if (getClass() != obj.getClass())
50             return false;
51         Usuario other = (Usuario) obj;
52         return Objects.equals(nombre, other.nombre) && Objects.equals(password, other.password);
53     }
54
55     @Override
56     public String toString() {
57         return "Usuario [nombre=" + nombre + ", password=" + password + "]";
58     }
59 }
```

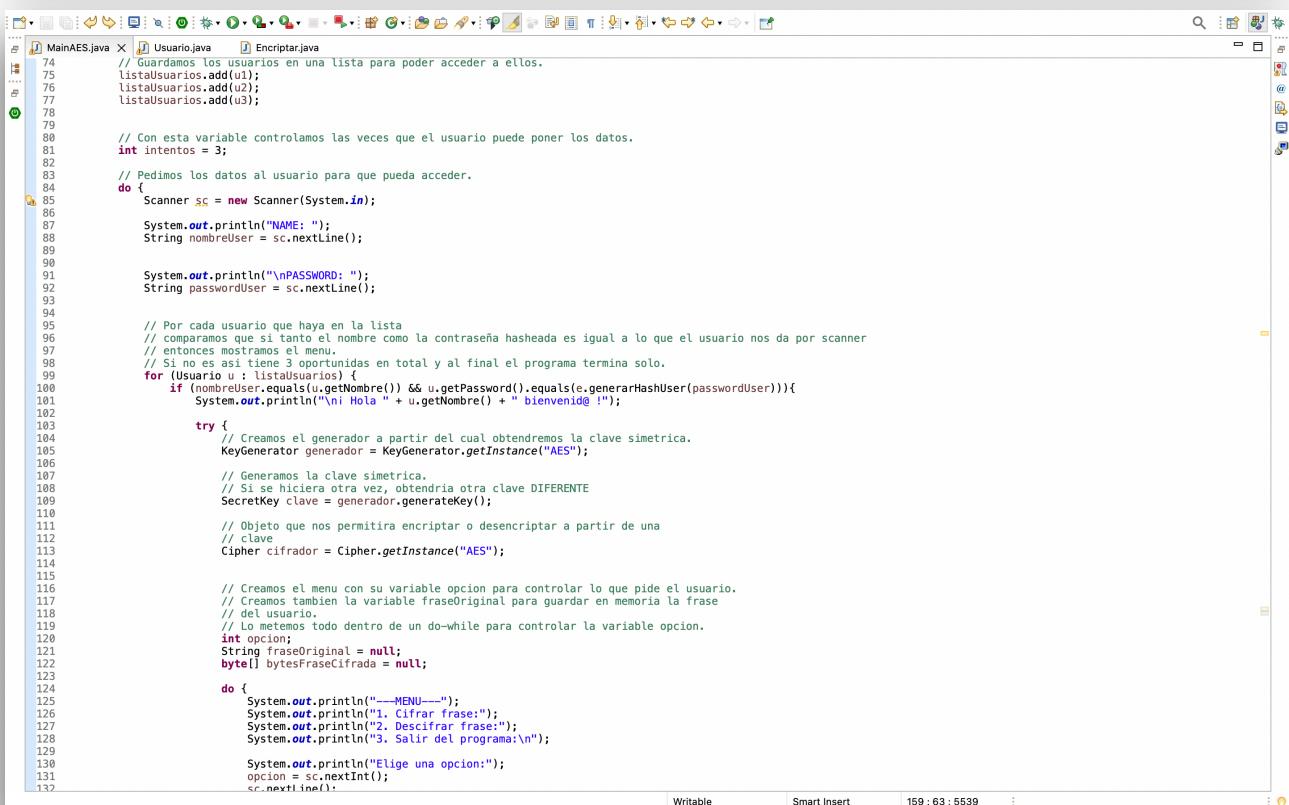
## Clase Encriptar.

```
1 package Encriptar;
2 import java.security.MessageDigest;
3 import java.security.NoSuchAlgorithmException;
4 import java.util.Base64;
5
6
7 public class Encryptar {
8
9     // Generamos un hash a partir de una contraseña que nos da el usuario.
10    public String generarHashUser(String passwordUser) throws NoSuchAlgorithmException {
11        byte[] password = passwordUser.getBytes();
12        MessageDigest mdUser = MessageDigest.getInstance("SHA-512");
13        mdUser.update(password);
14
15        byte[] passwordHasheada = mdUser.digest();
16
17        String passwordUserHashBase64 = Base64.getEncoder().encodeToString(passwordHasheada);
18
19        return passwordUserHashBase64;
20    }
21 }
```

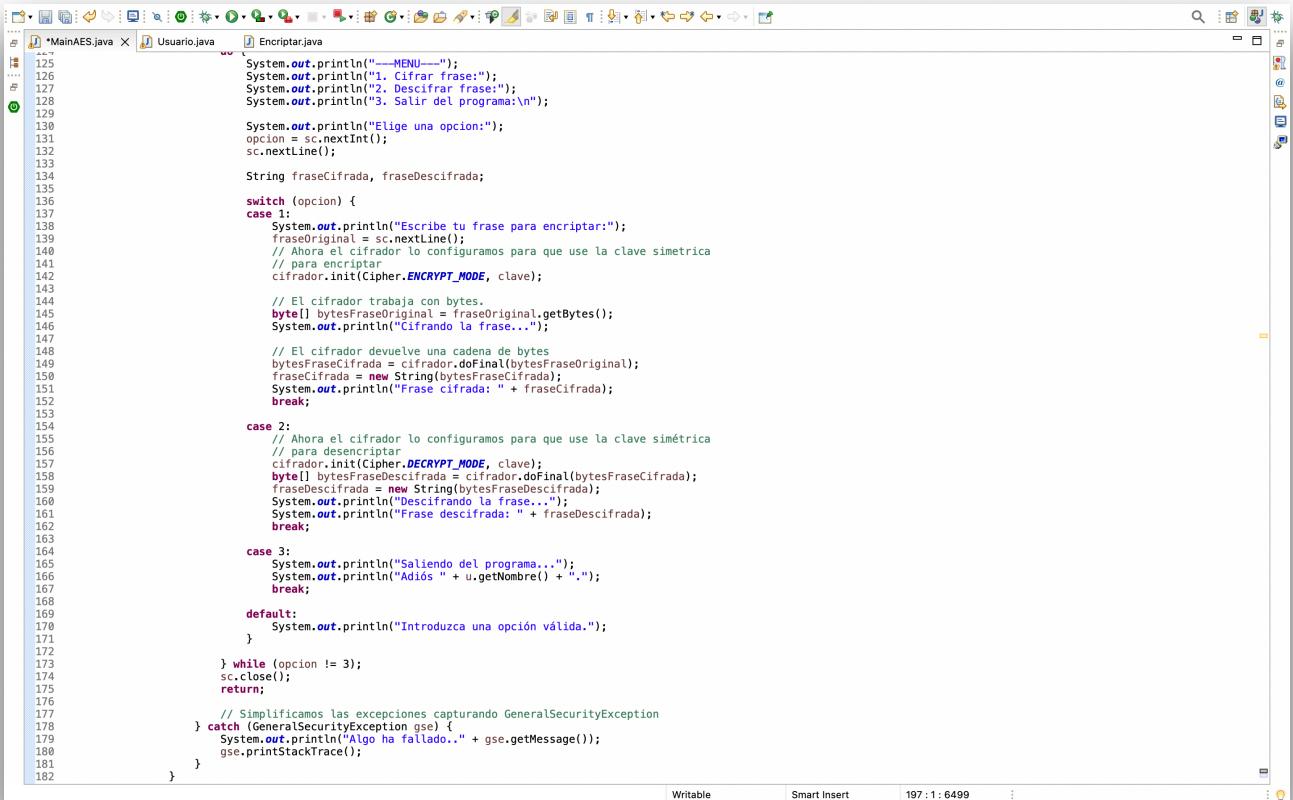
# Clase MainAES.



```
17 public class MainAES {
18
19     public static void main(String args[]) throws NoSuchAlgorithmException {
20         Encryptar e = new Encryptar();
21
22         // Creamos la contraseña a partir de la cual queremos crear su resumen.
23         // Lo pasamos a bytes ya que se necesita que la información este así para poder crear su resumen hash.
24         byte[] password1 = "123456789".getBytes();
25         byte[] password2 = "987654321".getBytes();
26         byte[] password3 = "0000".getBytes();
27
28
29         // Creamos un objeto MessageDigest a través del método estático
30         // getMessageDigest() al que se le pasa el tipo de algoritmo que vamos a
31         // utilizar.
32         MessageDigest md1 = MessageDigest.getInstance("SHA-512");
33         MessageDigest md2 = MessageDigest.getInstance("SHA-512");
34         MessageDigest md3 = MessageDigest.getInstance("SHA-512");
35
36
37         // Actualizamos las contraseñas de los usuarios y lo preparamos para convertirlo a hash
38         md1.update(password1);
39         md2.update(password2);
40
41         md3.update(password3);
42
43
44         // Ahora ejecutamos el método "digest()" para convertirlo a hash, pero esta en binario.
45         byte[] password1Hasheada = md1.digest();
46
47         byte[] password2Hasheada = md2.digest();
48
49         byte[] password3Hasheada = md3.digest();
50
51
52         // Lo pasamos a codificación BASE 64 para que sea mas legible.
53         // Puede ser útil si queremos guardar la información o enviar la información.
54         String password1_HashBase64 = Base64.getEncoder().encodeToString(password1Hasheada);
55
56         String password2_HashBase64 = Base64.getEncoder().encodeToString(password2Hasheada);
57
58         String password3_HashBase64 = Base64.getEncoder().encodeToString(password3Hasheada);
59
60
61         // Creamos una lista donde almacenar los usuarios.
62         List<Usuario> listaUsuarios = new ArrayList<>();
63
64
65         // Usuarios en memoria con su nombre y contraseña hasheada
66         Usuario u1 = new Usuario("Sergio", password1_HashBase64);
67         Usuario u2 = new Usuario("David", password2_HashBase64);
68         Usuario u3 = new Usuario("Iris", password3_HashBase64);
69
70
71         // Guardamos los usuarios en una lista para poder acceder a ellos.
72 }
```



```
73
74         // Guardamos los usuarios en una lista para poder acceder a ellos.
75         listaUsuarios.add(u1);
76         listaUsuarios.add(u2);
77         listaUsuarios.add(u3);
78
79
80         // Con esta variable controlamos las veces que el usuario puede poner los datos.
81         int intentos = 3;
82
83         // Pedimos los datos al usuario para que pueda acceder.
84         do {
85             Scanner sc = new Scanner(System.in);
86
87             System.out.println("NAME: ");
88             String nombreUser = sc.nextLine();
89
90             System.out.println("\nPASSWORD: ");
91             String passwordUser = sc.nextLine();
92
93
94             // Por cada usuario que haya en la lista
95             // comparamos que si tanto el nombre como la contraseña hasheada es igual a lo que el usuario nos da por scanner
96             // entonces mostramos el menu.
97             // Si no es así tiene 3 oportunidades en total y al final el programa termina solo.
98             for (Usuario u : listaUsuarios) {
99                 if (nombreUser.equals(u.getNombre()) && u.getPassword().equals(e.generarHashUser(passwordUser))) {
100                     System.out.println("\ni Hola " + u.getNombre() + " bienvenid@ !");
101
102                     try {
103                         // Creamos el generador a partir del cual obtendremos la clave simétrica.
104                         KeyGenerator generador = KeyGenerator.getInstance("AES");
105
106                         // Generamos la clave simétrica.
107                         // Si se hiciera otra vez, obtendría otra clave DIFERENTE
108                         SecretKey clave = generador.generateKey();
109
110                         // Objeto que nos permitirá encriptar o desencriptar a partir de una
111                         // clave
112                         Cipher cifrador = Cipher.getInstance("AES");
113
114
115                         // Creamos el menú con su variable opción para controlar lo que pide el usuario.
116                         // Creamos también la variable fraseOriginal para guardar en memoria la frase
117                         // del usuario.
118                         // Lo metemos todo dentro de un do-while para controlar la variable opción.
119                         int opcion;
120                         String fraseOriginal = null;
121                         byte[] bytesFraseCifrada = null;
122
123                         do {
124                             System.out.println("----MENU----");
125                             System.out.println("1. Cifrar frase:");
126                             System.out.println("2. Descifrar frase:");
127                             System.out.println("3. Salir del programa:\n");
128
129                             System.out.println("Elige una opción:");
130                             opcion = sc.nextInt();
131                             sc.nextLine();
132
133                         } while (opcion != 3);
```



The screenshot shows a Java IDE interface with the MainAES.java file open. The code implements a menu-based application for encrypting and decrypting strings using AES encryption. It uses standard input and output streams and the Cipher class from the Java Cryptography Extension.

```
125 System.out.println("----MENU----");
126 System.out.println("1. Cifrar frase:");
127 System.out.println("2. Descifrar frase:");
128 System.out.println("3. Salir del programa:\n");
129
130 System.out.print("Elige una opcion:");
131 opcion = sc.nextInt();
132 sc.nextLine();
133
134 String fraseCifrada, fraseDescifrada;
135
136 switch (opcion) {
137     case 1:
138         System.out.println("Escribe tu frase para encriptar:");
139         fraseOriginal = sc.nextLine();
140         // Ahora el cifrador lo configuramos para que use la clave simetrica
141         // para encriptar
142         cifrador.init(Cipher.ENCRYPT_MODE, clave);
143
144         // El cifrador trabaja con bytes.
145         byte[] bytesFraseOriginal = fraseOriginal.getBytes();
146         System.out.println("Cifrando la frase...");
```

## Funcionamiento del proyecto:

Aquí vemos por consola como se pide el nombre y contraseña del usuario.

Si los datos son correctos, se muestra el menú con las diferentes opciones para que el usuario elija alguna, así como lo que ocurre con cada punto del programa.

```
NAME:  
Sergio  
  
PASSWORD:  
123456789  
  
i Hola Sergio bienvenid@ !  
---MENU---  
1. Cifrar frase:  
2. Descifrar frase:  
3. Salir del programa:  
  
Elige una opcion:  
1  
Escribe tu frase para encriptar:  
pepeluis  
Cifrando la frase...  
Frase cifrada: 0N9000Kw0|0 V 0  
---MENU---  
1. Cifrar frase:  
2. Descifrar frase:  
3. Salir del programa:  
  
Elige una opcion:  
2  
Descifrando la frase...  
Frase descifrada: pepeluis  
---MENU---  
1. Cifrar frase:  
2. Descifrar frase:  
3. Salir del programa:  
  
Elige una opcion:  
3  
Saliendo del programa...  
Adiós Sergio.
```

Aquí vemos lo mismo con los otros 2 usuarios.

NAME:  
**David**

PASSWORD:  
**987654321**

i Hola David bienvenid@ !  
---MENU---  
1. Cifrar frase:  
2. Descifrar frase:  
3. Salir del programa:

Elige una opcion:  
**1**  
Escribe tu frase para encriptar:  
**qwertyuiop**  
Cifrando la frase...  
Frase cifrada: **ØØ'R9CØØø{ØØ@Ø**  
---MENU---  
1. Cifrar frase:  
2. Descifrar frase:  
3. Salir del programa:

Elige una opcion:  
**2**  
Descifrando la frase...  
Frase descifrada: **qwertyuiop**  
---MENU---  
1. Cifrar frase:  
2. Descifrar frase:  
3. Salir del programa:

Elige una opcion:  
**3**  
Saliendo del programa...  
Adiós David.

NAME:  
**Iris**

PASSWORD:  
**0000**

i Hola Iris bienvenid@ !  
---MENU---  
1. Cifrar frase:  
2. Descifrar frase:  
3. Salir del programa:

Elige una opcion:  
**1**  
Escribe tu frase para encriptar:  
**s<dghadzbzfh**  
Cifrando la frase...  
Frase cifrada: {ش|û[ÛUiLUmÛÛ  
---MENU---  
1. Cifrar frase:  
2. Descifrar frase:  
3. Salir del programa:

Elige una opcion:  
**2**  
Descifrando la frase...  
Frase descifrada: s<dghadzbzfh  
---MENU---  
1. Cifrar frase:  
2. Descifrar frase:  
3. Salir del programa:

Elige una opcion:  
**3**  
Saliendo del programa...  
Adiós Iris.

Aquí vemos como si el usuario no introduce los datos correctamente 3 veces, el programa se para solo y acaba.

```
NAME:  
sergio  
  
PASSWORD:  
qwafgagg  
Usuario no encontrado  
Te quedan 2 intentos  
NAME:  
setrgfhd  
  
PASSWORD:  
sfghsafg  
Usuario no encontrado  
Te quedan 1 intentos  
NAME:  
asag  
  
PASSWORD:  
dsfh<dhf  
Usuario no encontrado  
Te quedan 0 intentos  
Fin del programa. Adios
```