

Código Fuente

```
1 package factory.method;
2 /**
3  * @author Victor Lavalle
4  */
5 public interface AlgoritmoEncriptamiento {
6
7     public abstract void configurar();
8
9     public abstract String encriptar(String texto);
10
11     public abstract String desencriptar(String textoEncriptado);
12 }
```

```
1 package factory.method;
2 /**
3  * @author Victor Lavalle
4  */
5 public class FactoryAlgoEncripKHash implements FactoryAlgoEncriptamiento {
6
7     @Override
8     public AlgoritmoEncriptamiento create() {
9         AlgoritmoEncriptamiento algo = new KHash();
10        algo.configurar();
11        return algo;
12    }
13
14 }
```

```
1 package factory.method;
2 /**
3  * @author Victor Lavalle
4  */
5 public class FactoryAlgoEncripGHash implements FactoryAlgoEncriptamiento {
6
7     @Override
8     public AlgoritmoEncriptamiento create() {
9         AlgoritmoEncriptamiento algo = new GHash();
10        algo.configurar();
11        return algo;
12    }
13
14 }
```

```
1 package factory.method;
2 /**
3  * @author Victor Lavalle
4  */
5 public interface FactoryAlgoEncriptamiento {
6
7     public AlgoritmoEncriptamiento create();
8
9 }
```

```

1 package factory.method;
2
3 import javax.crypto.Cipher;
4 import javax.crypto.SecretKey;
5 import javax.crypto.SecretKeyFactory;
6 import javax.crypto.spec.IvParameterSpec;
7 import javax.crypto.spec.PBEKeySpec;
8 import javax.crypto.spec.SecretKeySpec;
9 import java.security.spec.KeySpec;
10 import java.util.Base64;
11
12 public class GHash implements AlgoritmoEncriptamiento {
13     private static String secretKey ;
14     private static String salt ;
15     private byte[] gHashInitVector = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0 };
16     private IvParameterSpec gHashInitVectorHash;
17
18     @Override
19     public void configurar() {
20         secretKey = "GHash!!!!";
21         salt = "GHash!!!!";
22         IvParameterSpec gHashInitVectorHash = new
IvParameterSpec(gHashInitVector);
23     }
24
25     @Override
26     public String encriptar(String str) {
27         System.out.println("Encrypting String with GHash");
28         try {
29             SecretKeyFactory factory =
SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256");
30             KeySpec spec = new PBEKeySpec(secretKey.toCharArray(),
salt.getBytes(), 65536, 256);
31             SecretKey tmp = factory.generateSecret(spec);
32             SecretKeySpec secretKey = new SecretKeySpec(tmp.getEncoded(), "AES");
33             Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
34             cipher.init(Cipher.ENCRYPT_MODE, secretKey, gHashInitVectorHash);
35             return
Base64.getEncoder().encodeToString(cipher.doFinal(str.getBytes("UTF-8")));
36         }
37         catch (Exception e) {
38             System.out.println("Error while encrypting with GHash: " +
e.toString());
39         }
40         return null;
41     }
42
43     @Override
44     public String desencriptar(String str) {
45         System.out.println("Decrypting String with GHash");
46         try {
47             SecretKeyFactory factory =
SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256");
48             KeySpec spec = new PBEKeySpec(secretKey.toCharArray(),
salt.getBytes(), 65536, 256);
49             SecretKey tmp = factory.generateSecret(spec);
50             SecretKeySpec secretKey = new SecretKeySpec(tmp.getEncoded(), "AES");
51             Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5PADDING");
52             cipher.init(Cipher.DECRYPT_MODE, secretKey, gHashInitVectorHash);
53             return new String(cipher.doFinal(Base64.getDecoder().decode(str)));}

```

```

54         catch (Exception e) {
55             System.out.println("Error while decrypting: " + e.toString());
56         }
57         return null;
58     }
59 }

```

```

1  package factory.method;
2
3  import javax.crypto.Cipher;
4  import javax.crypto.SecretKey;
5  import javax.crypto.SecretKeyFactory;
6  import javax.crypto.spec.IvParameterSpec;
7  import javax.crypto.spec.PBEKeySpec;
8  import javax.crypto.spec.SecretKeySpec;
9  import java.security.spec.KeySpec;
10 import java.util.Base64;
11
12 public class KHash implements AlgoritmoEncriptamiento {
13     private static String secretKey;
14     private static String salt ;
15     private byte[] kHashInitVector = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
16     0};
17     private IvParameterSpec kHashInitVectorHash;
18
19     @Override
20     public void configurar() {
21         secretKey = "GHash!!!!";
22         salt = "GHash!!!!";
23         IvParameterSpec kHashInitVectorHash = new
24         IvParameterSpec(kHashInitVector);
25     }
26
27     @Override
28     public String encriptar(String str) {
29         System.out.println("Encrypting String with KHash");
30         try {
31             SecretKeyFactory factory =
32             SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256");
33             KeySpec spec = new PBEKeySpec(secretKey.toCharArray(),
34             salt.getBytes(), 65536, 256);
35             SecretKey tmp = factory.generateSecret(spec);
36             SecretKeySpec secretKey = new SecretKeySpec(tmp.getEncoded(), "AES");
37
38             Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
39             cipher.init(Cipher.ENCRYPT_MODE, secretKey, kHashInitVectorHash);
40             return
41             Base64.getEncoder().encodeToString(cipher.doFinal(str.getBytes("UTF-8")));
42         } catch (Exception e) {
43             System.out.println("Error while encrypting with KHash: " +
44             e.toString());
45         }
46         return null;
47     }
48
49     @Override
50     public String desencriptar(String str) {
51         System.out.println("Decrypting String with KHash");
52         try {

```

```

47         SecretKeyFactory factory =
SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256");
48         KeySpec spec = new PBEKeySpec(secretKey.toCharArray(),
salt.getBytes(), 65536, 256);
49         SecretKey tmp = factory.generateSecret(spec);
50         SecretKeySpec secretKey = new SecretKeySpec(tmp.getEncoded(), "AES");
51
52         Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5PADDING");
53         cipher.init(Cipher.DECRYPT_MODE, secretKey, kHashInitVectorHash);
54         return new String(cipher.doFinal(Base64.getDecoder().decode(str)));
55     } catch (Exception e) {
56         System.out.println("Error while decrypting: " + e.toString());
57     }
58     return null;
59 }
60
61 }

```

```

1 package factory.method;
2 /**
3  * @author Victor Lavalle
4  */
5 public class Cliente {
6
7     public Cliente(){ }
8
9     public String encriptar(String texto, FactoryAlgoEncriptamiento factory) {
10         AlgoritmoEncriptamiento encriptador = factory.create();
11         return encriptador.encriptar(texto);
12     }
13
14     public String desencriptar(String texto, FactoryAlgoEncriptamiento factory) {
15         AlgoritmoEncriptamiento desencriptador = factory.create();
16         return desencriptador.desencriptar(texto);
17     }
18
19     public static void main(String args[]) {
20         String cadenaEncriptada;
21         cadenaEncriptada = new Cliente().encriptar("Hola", new
FactoryAlgoEncripGHash());
22         System.out.println(cadenaEncriptada+"\n");
23
24         String cadenaDesencriptada;
25         cadenaDesencriptada=new
Cliente().desencriptar("0xQM9GGhFwHK4h1TFmFbBw==", new FactoryAlgoEncripGHash());
26         System.out.println(cadenaDesencriptada+"\n");
27
28         System.out.println("-----");
29
30         cadenaEncriptada = new Cliente().encriptar("Hola", new
FactoryAlgoEncripKHash());
31         System.out.println(cadenaEncriptada+"\n");
32
33         cadenaDesencriptada=new
Cliente().desencriptar("jLkPaQStligov95QZ487Zg==", new FactoryAlgoEncripKHash());
34         System.out.println(cadenaDesencriptada+"\n");
35     }
36 }

```

Diagrama de Clases

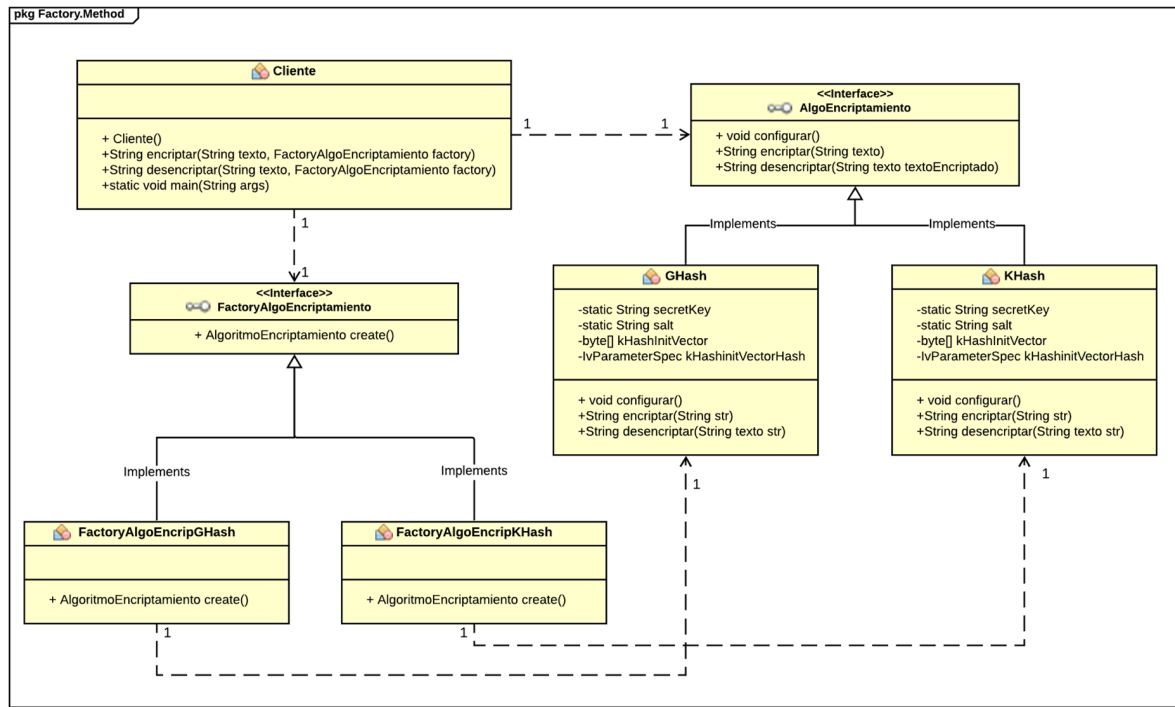
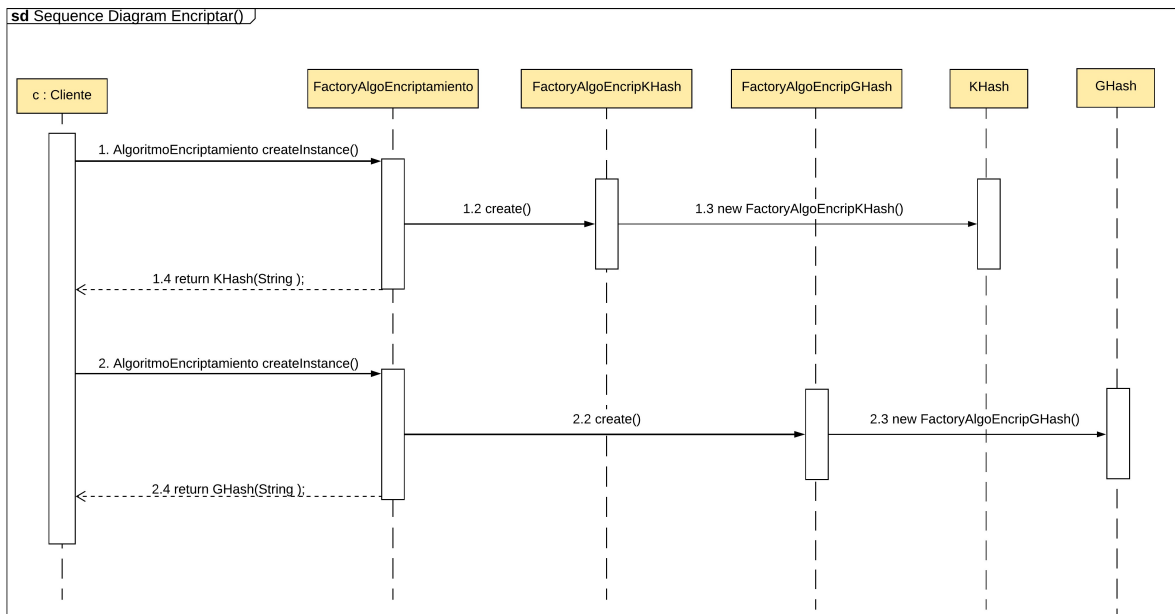


Diagrama de Secuencia



Ejecución del Programa

```
Output - Factory Method (run)

run:
Encrypting String with GHash
cNH0mDJiG2qdbcODKXkp/Q==

Decrypting String with GHash
Hola

-----

Encrypting String with KHash
yfJunroqmLAtGaFpb7AfjA==

Decrypting String with KHash
Hola

BUILD SUCCESSFUL (total time: 1 second)
```