Actividad 4. Cifrador simétrico y asimétrico

Grupo:

Frida Abella Fernández

Jaime Aranda Congil

David Matías Mota

Hemos creado una pequeña aplicación que sirve para realizar cifrado simétrico o asimétrico tanto de frases como de objetos, en nuestro caso de la clase Coche (ver Coche.java).

La idea es ofrecer al usuario un menú que le permita varias opciones, introducir la frase a encriptar, mostrar la frase encriptada, desencriptarla (mostrando la original y la encriptada) o bien introducir los datos del objeto coche a encriptar.

Los resultados serán mostrados por pantalla.

En primer lugar, declaramos o creamos todos los objetos propios del cifrado en Java que nos permitirán operar cifrando, como el KeyGenerator, el SecretKey o el Cipher. En el caso del cifrado simétrico, serían los siguientes:

```
// Primero creamos todos los objetos necesarios para el cifrado simétrico:
// Creamos el generador de claves, elegimos el algoritmo AES
KeyGenerator generadorSimetrica = KeyGenerator.getInstance("AES");

// Creamos la clave simétrica con la que vamos a encriptar:
  SecretKey claveSecreta = generadorSimetrica.generateKey();

  // Creamos el objeto que va a cifrar y descifrar las frases y objetos con la clave,
  // Le pasamos el algoritmo que usa para cifrar, que será el AES
  Cipher cifradorSimetrico = Cipher.getInstance("AES");
  // Lo iniciamos o configuramos para encriptar:
```

```
    cifradorSimetrico.init(Cipher.ENCRYPT_MODE, claveSecreta);

    // Creamos un objeto Cipher para desencriptado simétrico:
    Cipher descifradorSimetrico = Cipher.getInstance("AES");
    descifradorSimetrico.init(Cipher.DECRYPT_MODE, claveSecreta);
```

En el caso del cifrado asimétrico serían diferentes, ya que se necesita un objeto que cree un par de llaves (y el consecuente objeto que las contiene, el KeyPair). Usaremos la clave pública para encriptar (.getPublic()) y la privada para desencriptar (.getPrivate() en el descifradorAsimetrico):

```
  // A continuación, creamos los objetos necesarios para el CIFRADO ASIMÉTRICO:
  // El generador del par de claves, indicándole el algoritmo que usaremos, el RSA
  KeyPairGenerator generadorAsimetricas = KeyPairGenerator.getInstance("RSA");
  // Obtenemos el par de claves, simétrica y asimétrica, con KeyPair
    KeyPair claves = generadorAsimetricas.generateKeyPair();

    // Configuramos el cifrador asimétrico:
    Cipher cifradorAsimetrico = Cipher.getInstance("RSA");
    // Lo inicializamos para que genere la clave pública con la que encriptaremos
    cifradorAsimetrico.init(Cipher.ENCRYPT_MODE, claves.getPublic());

    // Crearemos un objeto para descifrado asimétrico:
    Cipher descifradorAsimetrico = Cipher.getInstance("RSA");
    // Lo configuramos para que descifre con la clave privada:
    descifradorAsimetrico.init(Cipher.DECRYPT_MODE, claves.getPrivate());
```

A continuación declaramos las variables que se usarán a lo largo de todo el programa para realizar las funciones necesarias. Son las mismas para ambos casos, cifrado simétrico o asimétrico:

```
  // Ahorraremos código usando las mismas variables, y mismo menú, para ambos cifrados (si
métrico y asimétrico)
  // Creamos la variable que almacena la frase original
  String frase = null;
  // Creamos la variable que almacena el flujo de bytes de la frase original introducida:
    byte [] bytesFrase = null;

    // Creamos la variable que almacenerá en memoria el flujo de bytes con la frase cifrad
a
```

```
    byte [] bytesFraseCifrada = null;

    // Creamos la variable tipo String donde se almacenará la frase encriptada
    String fraseCifrada = null;

    // Variable que almacena los bytes de la frase tras descifrarla
    byte [] bytesFraseDescifrada = null;

    // Declaramos el objeto tipo Coche con el que trabajaremos en el menú:
    Coche coche1;

    // Declaramos el objeto tipo SealedObject donde se creará o guardará el objeto (tipo c
oche) encriptado
    SealedObject so = null;

    // Creamos una variable boolean para controlar la repetición del bucle.
    boolean continuar = true;
```

También creamos, en una clase aparte, el constructor y los atributos que servirán para crear el objeto de tipo Coche con el que trabajaremos más adelante.

```
    // Creamos la clase Coche para construir objetos de su tipo. Debe implementar Serializ
able para poder ser cifrado

public class Coche implements Serializable{
private String matricula;
private String marca;
private String modelo;
private double precio;

public Coche() {

}

public Coche(String matricula, String marca, String modelo, double precio) {
  super();
  this.matricula = matricula;
  this.marca = marca;
  this.modelo = modelo;
  this.precio = precio;
}



public String getMatricula() {
  return matricula;
}
```

```
public void setMatricula(String matricula) {
  this.matricula = matricula;
}


// ... El código continúa
```

A continuación creamos el menú, que será el mismo para ambos programas. Irá encerrado dentro de una estructura o bucle DO-WHILE para que se repita constantemente hasta que el usuario teclee la opción 5, la de salir. En ese momento, la condición del while dejará de ser 'true' y la aplicación se cerrará. Dentro del menú habrá un Switch-Case que controlará el diferente código a ejecutar según la opción elegida por el usuario.

```
    do {
    // Creamos el manú para que salga por pantalla y permita escoger una opción
    System.out.println("ELIJA UNA OPCIÓN:\n\n"
    + "  1. Encriptar frase \n"
    + "  2. Mostrar frase encriptada \n"
    + "  3. Desencriptar frase \n"
    + "  4. Encriptar un objeto coche \n"
    + "  5. Salir \n");
  // Creamos un objeto Scanner para pedir por pantalla la opcion del menú deseada
      Scanner sc = new Scanner(System.in);
      int opcion = sc.nextInt();
      // Creamos un único Scanner para usar a lo largo del menú que lea Strings
      Scanner scanner = new Scanner(System.in);
    switch (opcion) {
```

Opción 1: Encriptar una frase

Se encripta la frase escrita por consola. Se ofrece al usuario la posibilidad de cifrarla con simétrico (introduciendo número 1 por pantalla) o asimétrico (número 2), y se comprobará la opción elegida con un if-else.

```
  case 1:
System.out.println("----ENCRIPTAR FRASE----");
// Pedimos la frase por consola y la almacenamos en una variable de tipo String:
System.out.println("Escriba la frase que desea encriptar:");
frase = scanner.nextLine();
// Convertimos el String recibido en un array o flujo de bytes para poder cifrar la frase
```

```
bytesFrase = frase.getBytes();
// Pedimos la opción de cifrado deseado por pantalla
System.out.println("Escriba 1 para cifrado simétrico o 2 para cifrado asimétrico");
int opcionCifrado =  Integer.parseInt(scanner.nextLine());
if (opcionCifrado == 1) {
// Guardamos en un array o flujo de bytes el mismo mensaje pero encriptado, usando el méto
do .doFinal() del cifrador para encriptar:
bytesFraseCifrada = cifradorSimetrico.doFinal(bytesFrase);
// Guardamos en la variable tipo String fraseCifrada el valor del flujo de bytes de la fra
se cifrada pasado a String
fraseCifrada = new String(bytesFraseCifrada);
System.out.println("La frase ha sido cifrada con cifrado simétrico");
} else if (opcionCifrado == 2) {
// Guardamos en un array o flujo de bytes el mismo mensaje pero encriptado, usando el méto
do .doFinal() del cifrador para encriptar:
bytesFraseCifrada = cifradorAsimetrico.doFinal(bytesFrase);
// Lo pasamos o guardamos en variable tipo String
fraseCifrada = new String(bytesFraseCifrada);
System.out.println("La frase ha sido cifrada con cifrado asimétrico");
} else {
System.out.println("La opción marcada es incorrecta, no existe");
opcionCifrado = 0;
frase = "";
}
break;
```

Opción 2: Mostrar la frase encriptada

Se mostrarán por pantalla la frase original y la enrciptada, sea cual sea el método que se haya elegido:

case 2:
System.out.println("----MOSTRAR FRASE ENCRIPTADA----");
// Mostramos la frase original:
System.out.println("La frase original era: " + frase);
// Mostramos por pantalla la frase encriptada almacenada en memoria
System.out.println("La frase encriptada es la siguiente: " + fraseCifrada);
break;

Opción 3: Descifrar la frase cifrada:

Se descifrará la frase que se ha encriptado o cifrado previamente en la opción 1 del menú. De nuevo se ofrecerá la opción de desencriptado que el usuario desee, simétrico o asimétrico. Deberá elegirse la misma que se ha elegido para encriptar:

```
case 3:
System.out.println("----DESCIFRAR FRASE----");
System.out.println("Escriba 1 para descifrado simétrico o 2 para descifrado asimétrico");
int opcionDescifrado = Integer.parseInt(scanner.nextLine());
if (opcionDescifrado == 1) {
// Guardamos en un flujo de bytes la frase desencriptada:
bytesFraseDescifrada = descifradorSimetrico.doFinal(bytesFraseCifrada);
// Mostramos la frase descifrada por pantalla, pasándola a String
System.out.println("La frase desencriptada es: " + new String(bytesFraseDescifrada));
} else if (opcionDescifrado == 2) {
// Guardamos en un array o flujo de bytes el mismo mensaje pero encriptado, usando el méto
do .doFinal() del cifrador para encriptar:
bytesFraseDescifrada =  descifradorAsimetrico.doFinal(bytesFraseCifrada);
System.out.println("La frase desencriptada es: " + new String(bytesFraseDescifrada));
} else {
System.out.println("La opción escogida es incorrecta");
opcionDescifrado = 0;
}
break;
```

Opción 4: Encriptar un objeto de tipo Coche

Primero se crea el objeto y se piden por pantalla los atributos que conforman el objeto y se le asignan, para poder crearlo:

coche1 = new Coche();
System.out.println("Introduzca los datos del coche:");
System.out.println("Matrícula: ");
coche1.setMatricula(scanner.nextLine());
System.out.println("Marca: ");
coche1.setMarca(scanner.nextLine());
System.out.println("Modelo: ");
coche1.setModelo(scanner.nextLine());
System.out.println("Precio: ");
Scanner precio = new Scanner(System.in);
coche1.setPrecio(precio.nextInt());

A continuación se pide por pantalla el tipo de cifrado y se realiza. Se capturan las excepciones que saltarían:

```
      System.out.println("Escriba 1 para cifrado simétrico o 2 para cifrado asimétrio");
int cifradoCoche = Integer.parseInt(scanner.nextLine());
if (cifradoCoche == 1) {
try {
// Usamos un objeto de tipo SealedObject para cifrar el objeto Coche:
so = new SealedObject(coche1, cifradorSimetrico);
System.out.println("Coche original: " + coche1);
System.out.println("Coche cifrado: " + so.toString());
    } catch (IOException ioe) {
        System.out.println("Error. No se ha podido cifrar el objeto");
      }
    }
    else if (cifradoCoche == 2){
      try {
        // Usamos un objeto de tipo SealedObject para cifrar el objeto Coche:
        so = new SealedObject(coche1, cifradorAsimetrico);
        System.out.println("Coche original: " + coche1);
        System.out.println("Coche cifrado: " + so.toString());

      } catch (IOException ioe) {
        System.out.println("Error. No se ha podido cifrar el objeto");
      }
    }

  break;
```

Opción 5: Salir

```
case 5:
  System.out.println("----SALIR----");
  System.out.println("La aplicación ha terminado ");
  continuar = false;
  break;
  }

  } while (continuar);
```

Por último, todo el programa va encerrado dentro de un try-catch que atrapa todas las excepciones posibles, incluidas las de seguridad que suelen saltar con el cifrado

```
  } catch (Exception e) {
  System.out.println("Un error ha ocurrido... " + e.getMessage());
  e.printStackTrace();
  }

}
```