	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: René Adrián Dávila Pérez

Asignatura: Programación Orientada a Objetos

Grupo: 01

No. de práctica(s): 03

Integrante(s): 322276824
322258516
425037384
320108116
322221415

No. de brigada: 01

Semestre: 2026-1

Fecha de entrega: 12 de septiembre de 2025

Observaciones: _____

CALIFICACIÓN: _____

Índice

1. Introducción	2
2. Marco Teórico	2
3. Desarrollo	4
4. Resultados	5
5. Conclusiones	5

1. Introducción

- **Planteamiento del problema:**

Conocer el principio de las funciones digestivas resulta relevante pues son un pilar de la seguridad informática. Para ello, se busca crear un programa para simular el funcionamiento de una función digestiva, particularmente el algoritmo MD5. Que este reciba cadenas de texto como entrada y luego se procesen mediante un mecanismo pseudoaleatorio para devolver un valor hexadecimal de longitud fija asociado a la entrada (*hash*).

- **Motivación:**

El estudio de las funciones digestivas es fundamental en el ámbito de la seguridad informática, ya que constituyen la base de numerosos sistemas de autenticación y resguardo de información. Esta simulación es útil para comprender la naturaleza determinista y unidireccional de estas funciones. Además, conocer clases de uso general relativas a colecciones de datos como `ArrayList` o `HashMap` es crucial para facilitar el manejo de datos y la generación de resultados.

- **Objetivos:**

Introducir los principios de las funciones digestivas y su importancia en la informática representando el objetivo de estas: transformar información de manera unidireccional, de forma que obtener la salida sea sencillo, mientras que revertir el proceso sea computacionalmente inviable. Así como reforzar el uso de colecciones y métodos de la biblioteca estándar de Java.

2. Marco Teórico

Para la elaboración del programa se emplearon diferentes clases de la biblioteca estándar de Java para el manejo de cadenas, colecciones y generación de valores pseudoaleatorios.

Clases de Java

StringBuilder: Es una clase de Java utilizada para la manipulación de cadenas de texto mutables. Permite modificar, insertar o eliminar caracteres sin necesidad de crear nuevos objetos en memoria. De acuerdo con [4] “Las operaciones principales en un `StringBuilder` son los métodos `append` e `insert`. Cada uno de ellos convierte eficazmente un dato dado en una cadena y, a continuación, añade o inserta los caracteres de esa cadena en el generador de cadenas.”.

Random: La clase `Random` proporciona un generador de números pseudoaleatorios. Se basa en una semilla inicial y los valores generados mantienen un comportamiento determinista, es decir, la misma semilla produce siempre la misma secuencia de resultados. [3]

HashMap: Un `HashMap` es una estructura de datos basada en tablas hash que permite almacenar pares clave-valor, con operaciones de búsqueda, inserción y recu-

peración. La naturaleza de esta estructura no incluye orden alguno, de hecho, “Esta clase no ofrece ninguna garantía en cuanto al orden del mapa; en particular, no garantiza que el orden se mantenga constante a lo largo del tiempo.”. [2]

ArrayList: Un **ArrayList** es una colección dinámica que permite almacenar elementos en una estructura de tamaño variable. A diferencia de los arreglos convencionales, un **ArrayList** puede crecer o reducirse según sea necesario. Entre sus métodos de uso común están: **size**, **get**, **set**, **add**, en su mayoría operaciones consideradas constantes. [1]

Función digestiva

También conocidas como funciones hash criptográficas, son algoritmos que transforman una entrada de longitud arbitraria en una salida de tamaño fijo, normalmente expresada en forma de cadena hexadecimal. Su característica fundamental es ser unidireccionales: calcular la salida es sencillo, pero revertir el proceso para obtener la entrada original es computacionalmente inviable. Además, poseen propiedades como la sensibilidad a pequeñas variaciones en la entrada y la baja probabilidad de colisiones. Entre sus aplicaciones están la emisión de certificados, las firmas digitales, la generación de llaves y la verificación de contraseñas. [5]

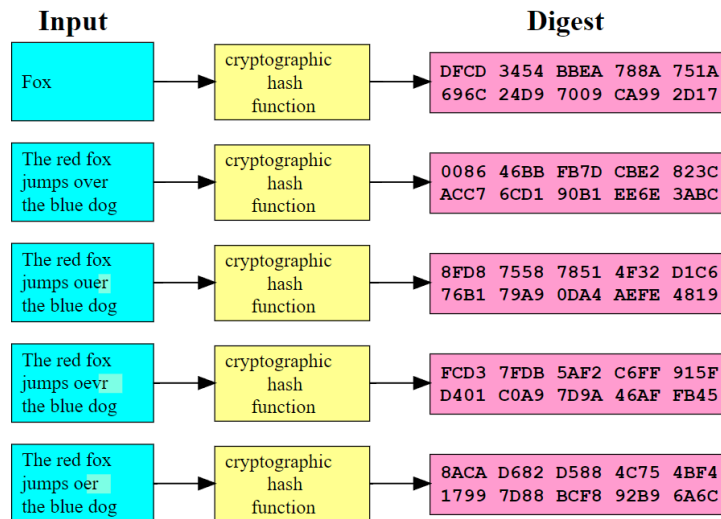


Figura 1: Función hash criptográfica. [7]

MD5

El algoritmo Message Digest 5 (MD5) es una función hash criptográfica ampliamente utilizada en la década de 1990 y principios de 2000. Genera un valor de 128 bits (32 dígitos hexadecimales) a partir de un mensaje de entrada. Si bien actualmente se considera inseguro debido a vulnerabilidades que permiten generar colisiones, MD5 sigue siendo un referente didáctico para comprender la naturaleza y el funcionamiento

de las funciones digestivas. Una de sus principales ventajas es su velocidad y eficiencia para generar valores hash. [6]

3. Desarrollo

El programa consistió en la implementación del método `generaHash` y el uso de las clases `HashMap` y `ArrayList` para el manejo de las entradas como argumentos al momento de ejecutar el programa. Con el propósito de efectuar un mejor desglose de la lógica y las herramientas usadas en la elaboración de la práctica dividiremos la explicación en las siguientes secciones.

main

Dentro del método principal `main` se utilizó `ArrayList` para generar el objeto `entradas`, en él se guardaron los argumentos recibidos de `args` al ejecutar el programa, añadiendo cada elemento a `entradas` con la ayuda de una estructura `for` que itera sobre `args`.

Obtenidas las entradas, con la ayuda de la clase `HashMap` se instancia un nuevo objeto `map` con valor y clave de tipo `String`. Se crea un `for` que itera sobre `entradas` y se usa una variable auxiliar para guardar el hash asociado a cada elemento que se genera con el método `generaHash` para enviarse como valor. Buscando la estructura en la que la cadena inicial es la clave y la cadena devuelta por `generaHash` el valor.

Por último se utiliza otro `for` iterando sobre `entradas` para imprimir cada cadena y su hash generado, utilizando `get` para obtener de `map` cada valor (hash) ligado a la clave (cadena).

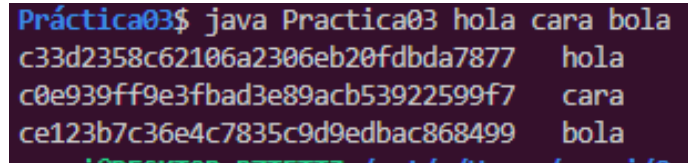
generaHash

El método `generaHash` usa el modificador de acceso `public` y el modificador `static`, lo que le permite ser invocado desde cualquier parte del programa sin la necesidad de crear un objeto. Funciona recibiendo una variable de tipo `String` la cual convierte en un arreglo de caracteres con ayuda de `toCharArray` y suma los valores ASCII de cada carácter para generar una semilla que se usará junto al objeto `random`, creado con la ayuda de la clase `Random`.

Así pues, se crea un objeto `sb` con `StringBuilder` para modificar y manipular las cadenas libremente, se usa un ciclo `for` que va de 0 a 31, dentro de cada iteración se genera un valor aleatorio entre 0 y 16 mediante el método `nextInt` del objeto `random`, luego se pasa cada valor a su forma hexadecimal con `toHexString` y se agrega todo a `sb` usando el método `append` para finalmente retornar la cadena formada.

4. Resultados

Probando el programa con varias cadenas como entrada.



```
Práctica03$ java Practica03 hola cara bola
c33d2358c62106a2306eb20fdbda7877 hola
c0e939ff9e3fbad3e89acb53922599f7 cara
ce123b7c36e4c7835c9d9edbac868499 bola
```

Figura 2: Ejecución del programa.

5. Conclusiones

La práctica sirvió para poder comprender mejor el funcionamiento de las funciones digestivas, particularmente la lógica detrás de la generación de valores hash como en el caso del algoritmo MD5. A través del uso de colecciones de Java como `ArrayList` y `HashMap`, así como clases de apoyo como `StringBuilder` y `Random`, se logró simular un proceso que transforma entradas en salidas de longitud fija. Este ejercicio mostró la relevancia de las funciones hash en la seguridad evidenciando su utilidad para proteger y autenticar los datos.

Referencias

- [1] *Class ArrayList<E>*. Sep. de 2025. URL: <https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html>.
- [2] *Class HashMap<K, V>*. Sep. de 2025. URL: <https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html>.
- [3] *Class Random*. Sep. de 2025. URL: <https://docs.oracle.com/javase/8/docs/api/java/util/Random.html>.
- [4] *Class StringBuilder*. Sep. de 2025. URL: <https://docs.oracle.com/javase/8/docs/api/java/lang/StringBuilder.html>.
- [5] *Funciones Hash*. Sep. de 2025. URL: https://repositorio-uapa.cuaed.unam.mx/repositorio/moodle/pluginfile.php/2186/mod_resource/content/4/contenido-uapa/index.html.
- [6] *Resumen del mensaje MD5*. Jun. de 2025. URL: <https://wraycastle.com/es/blogs/glossary/md5-message-digest-5>.
- [7] Jorge Stolfi. «Cryptographic Hash Function». En: (2008).