	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorios de docencia

Laboratorio de Computación
Salas A y B

Profesor(a): _____

Asignatura: _____

Grupo: _____

No de Práctica(s): Práctica 3

Integrante(s): 322246320

322267567

322012051

322236688

322330872

No. de lista o brigada: _____

Semestre _____

Fecha de entrega: _____

Observaciones: _____

_____

CALIFICACIÓN: _____

Índice

1. Introducción

- **Planteamiento del Problema:** Se busca simular una función **digestiva** en Java, transformando cadenas de entrada (obtenidas desde args()) en una cadena alfanumérica generada pseudoaleatoriamente.
- **Motivación:** Al realizar esta práctica buscamos lograr una correcta comprensión acerca de lo que realiza una función **digestiva** y el uso de los **Hash**. Además de hacer una correcta captura de datos de entrada en el **Array List**.
- **Objetivos:** Comprender la utilidad y las operaciones que realiza una función **Hash**. Así como hacer un manejo adecuado de cadenas almacenadas en un **ArrayList** al momento de ejecutar.

2. Marco Teórico

Hash: Es lo que se obtiene de aplicar una función matemática a una entrada (usualmente una cadena de caracteres), transformando esta en una nueva cadena representativa alfanumérica con una longitud definida. [1]

Otra definición: una función computacionalmente eficiente que asigna cadenas de longitud arbitraria a cadenas binarias. Se conocen también como función solo de ida, función unidireccional o función digestiva. [2]

MD5: Es un protocolo criptográfico que verifica contenido y firmas digitales, basándose en la función Hash para verificar que un archivo es igual que otro, ya que no importa dónde se ejecuten, tendrán la misma clave. [3]

3. Desarrollo

1. **Importación de clases:** Importamos las clases que utilizaremos en este código, las cuales son:
   - **import java.util.ArrayList; :** Ya que se usa un ArrayList para almacenar las entradas
   - **import java.util.HashMap; :** Ya que se utilizará un HashMap para almacenar las parejas clave–valor correspondientes a la cadena de entrada.
   - **import java.util.Random; :** Porque se usa Random para generar números pseudoaleatorios a partir de la semilla obtenida de cada palabra ingresada.
2. **Creación del Hash:**

   El método **generaHash** que nos proporcionó el profesor, obtiene una semilla a partir de la suma de los valores numéricos que tiene cada carácter de la cadena de entrada.

Luego se crea un objeto de tipo **Random**, y usamos el constructor que recibe una semilla, para que siempre genere el mismo tipo de números con la misma semilla.

Después creamos un objeto **StringBuilder**, este nos permite manejar cadenas de forma mas eficiente.

Dentro de un ciclo **for** que itera 32 veces (simulando un **MD5**), se generan valores aleatorios del 0 al 16 con el objeto **Random** ya predefinido con una semilla, esto con el objetivo de solo generar valores **hexadecimales**, y se agregan al objeto **StringBuilder**, pero antes convirtiendo ese valor a su representación **hexadecimal**.

Finalmente se regresa ese valor generado pero en cadena con el método **toString();**.

3. **Desarrollo del main:**

   Dentro del método **main**, creamos un **ArrayList** y un **HashMap**

   Antes de iniciar cualquier operación en el programa, evaluamos si se ingresó alguna cadena en la ejecución, de no ser así, se imprime un mensaje pidiendo ingresar alguna cadenas al ejecutar y termina la ejecución

   Si el programa continua su ejecución, almacenamos en el **ArrayList** las cadenas de **args** ingresadas en la ejecución, y posteriormente iteramos en el **ArrayList**, convirtiendo cada cadena en una llave con el metodo ya creado **generaHash**, y después agregando esa llave junto con su cadena al **hash**.

4. Resultados

```
PS C:\Users\PC\OneDrive\Desktop\VisualCode\Java> java Practica3 hola mundo
Clave: ae49425595be8692c2a1aad1326aadaa Palabra: mundo
Clave: c33d2358c62106a2306eb20fdbda7877 Palabra: hola
```

Manera correcta de ejecución y salida

```
ahmed@ahmedykarim:~/POO-G1-E2/Practica3$ java Practica3 hola mundo
Clave: ae49425595be8692c2a1aad1326aadaa Palabra: mundo
Clave: c33d2358c62106a2306eb20fdbda7877 Palabra: hola
```

```
PS C:\Users\angel\Downloads\POO\Practica3> java Practica3 hola mundo
Clave: ae49425595be8692c2a1aad1326aadaa Palabra: mundo
Clave: c33d2358c62106a2306eb20fdbda7877 Palabra: hola
```

Ejecución en otros equipos

5. Conclusiones

La práctica permitió comprender adecuadamente cómo una función Hash transforma cadenas de texto en claves únicas, las cuales se mantendrán iguales sin importar el equipo donde se realice la ejecución, lo que facilita su uso para identificadores únicos.

También entendimos el uso y la diferencia entre un **Hash** y un **HashMap**, donde en el HashMap se guardan claves únicas con un único valor. Así como también entendimos el uso y las operaciones de un **ArrayList**.

Referencias

[1] *De la verificación de contraseña a la firma electrónica: el secreto está en el 'hash'.* 2025. URL: `https://www.bbva.com/es/innovacion/de-la-verificacion-de-contrasena-a-la-firma-electronica-el-secreto-esta-en-el-hash` (visitado 09-09-2025).

[2] *Funciones hash. Unidades de Apoyo para el Aprendizaje.* 2021. URL: `https://repositorio-uapa.cuaed.unam.mx/repositorio/moodle/pluginfile.php/2186/mod_resource/content/1/contenido-uapa/index.html` (visitado 09-09-2025).

[3] *¿Qué es el algoritmo de hashing MD5 y cómo funciona?* 2022. URL: `https://www.avast.com/es-es/c-md5-hashing-algorithm` (visitado 09-09-2025).

6. Reto para token

Para asegurarnos que el usuario siempre ingrese 3 valores para calcular la formula general desde la ejecución del programa, pusimos un **if** que evaluara si el tamaño del arreglo **args** era diferente de 3, lo que significaba que había ingresado mas o menos valores a los solicitados (a, b , c).

```
if(args.length!=3){
    System.out.println("Ingresa los valores de a, b, c");
    return;
}
```

Condicion inicial

Luego convertimos los valores que recibe args a double con **Double.ParseDouble**, para poder utilizarlos en la operaciones, ya que llegan como cadenas

```
double a = Double.parseDouble(args[0]);
double b = Double.parseDouble(args[1]);
double c = Double.parseDouble(args[2]);
```

Conversión a double

Utilizamos una variable que calculara el determinante para saber si los resultados serian reales o imaginarios, en el caso de los reales, se calcula la formula general directamente y se imprimen los 2 valores de **x**, considerando los signos. En el caso de ser números complejos, calculamos la parte imaginaria y la parte real en variables diferentes, y luego las imprimimos considerando los signos.

```
double det=(b*b)-((4*a*c));
```

Calculo del determinante

Utilizamos los métodos **Math.abs();** para calcular los números imaginarios cuando era el caso y **Math.sqrt();** para la raíz de la formula general