	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorios de docencia

Laboratorio de Computación Salas A y B

Profesor(a): René Adrián Dávila Pérez

Asignatura: POO

Grupo: 01

No de Práctica(s): Practica 3

Integrante(s): 322125337

322080869

322059179

323629814

322113536

No. de lista o brigada: 6

Semestre: 2025

Fecha de entrega: 12/09/2025

Observaciones:

CALIFICACIÓN: _____

Índice

1. Introducción

- **Planteamiento del problema**: La práctica consiste en simular lo que es una función digestiva hash. Ya que la función para generar el hash la dio el docente nosotros tendremos que analizar como es que podemos utilizarla con las especificaciones dadas (como usar la función, que los argumentos se ingresen por main, que muestre la palabra y el valor hash, etc)
- **Motivación**: Las funciones digestivas hash son fundamentales para la seguridad informática e integridad de datos. Gracias a esta práctica podemos entender estos términos y podemos saber lo que es la "huella digital"de un programa y como cuidarnos
- **Objetivos**: El objetivo principal es implementar un programa que simule una función digestiva hash. Deberá ingresar los datos por main y mostrar la equivalencia en Hash

2. Marco Teórico

Funcion Hash:Es un algoritmo matemático que mapea datos de tamaño arbitrario a un valor de tamaño fijo. Este valor, conocido como valor hash o simplemente hash, actúa como una representación compacta y única de los datos originales. La función es determinista, lo que significa que la misma entrada siempre producirá la misma salida. Las funciones hash son cruciales para garantizar la integridad de los datos, ya que un cambio mínimo en la entrada resultará en un hash completamente diferente.

Argumentos de Línea de Comandos: : Es una técnica que permite enviar datos al programa al momento de su ejecución, directamente desde la consola o terminal. En Java, estos argumentos se reciben en el método principal main(String[] args) como un arreglo de cadenas de texto. Esta funcionalidad permite que el programa pueda adaptarse a distintas entradas sin necesidad de modificar el código fuente. Se utiliza para facilitar la interacción del usuario con el programa, permitiendo ingresar valores o parámetros desde la línea de comandos al momento de ejecutar el archivo .class que compilemos.

ArrayList: Es una clase de la biblioteca estándar de Java que implementa una lista dinámica, permitiendo almacenar elementos de forma ordenada y acceder a ellos por su índice. A diferencia de los arreglos tradicionales, un ArrayList puede cambiar su tamaño automáticamente al agregar o eliminar elementos.[1]

HashMap:Es una estructura de datos que implementa una tabla de dispersión (hash table) para almacenar pares clave-valor. En Java, la clase HashMap permite asociar una clave única con un valor, y acceder rápidamente a este último utilizando su clave. Internamente, utiliza funciones hash para determinar la posición de almacenamiento

de cada par, lo que permite una búsqueda, inserción y eliminación eficientes tiene similitudes con la estructura de datos (diccionario) en python.[1]

3. Desarrollo

Primero nos documentamos al respecto de las herramientas que se tenian que implementar en la practica, tales como que es un HashMap, ArrayList, Funcion Hash y el uso de los mismos.
Una vez estudiado eso implementamos lo aprendido ‘para la resolucion del problema.

Iniciamos decidiendo que las cadenas de entrada sean recibidas vía los argumentos del método main (String[] args), asi cuando se ejecute el programa desde consola se puedan pasar varias palabras. Después almacenamos estas cadenas en un ArrayList<String>y mediante una prueba nos aseguramos que estubieramos guardando los valores correctos. Una vez que verificamos el ArrayList definimos la función generaHash(String texto), que simula un resumen (hash) de 32 caracteres hexadecimales en donde primero calcula una semilla sumando los códigos numéricos de cada carácter, con esa semilla se crea un objeto Random, el cual sea determinista en donde la misma cadena produce siempre la misma salida.

Despues se genera un número aleatorio entre 0 y 15, 32 veces, convirtiendo cada número a su representación hexadecimal, acumulándolos para formar la cadena hash de longitud fija. Usamos un HashMap<String, String>para asociar cada cadena con su hash generado. Se usa put para insertar las parejas clave-valor.

Por ultimo utilizando otro for-each se recorre de nuevo la lista de entradas originales, se recupera cada hash con get del mapa, y se imprime en consola

4. Resultados

El ejercicio se realizo correctamente tomando la cadena desde el método main y simulando una función digestiva hash

Resultado en terminal

```
PS C:\Users\edgar\OneDrive\Documentos\Practicas POO> javac Practica3.java
PS C:\Users\edgar\OneDrive\Documentos\Practicas POO> java Practica3 Hola Mundo
Entrada: Hola   Salida: c9d3ef43eaa48e1e0a8fde0a176039ac
Entrada: Mundo  Salida: a4ef0e40c9220e0eabc2b7dc7ef06ddf
PS C:\Users\edgar\OneDrive\Documentos\Practicas POO>
```

-Calcula la suma de los códigos ASCII de cada carácter de la cadena de entrada.
-Usa esa suma como semilla para el generador aleatorio.
-Luego genera 32 valores aleatorios entre 0 y 15, y los convierte a hexadecimal.
-Devuelve esos 32 caracteres como una cadena de texto.
-Cada hash será siempre el mismo para la misma palabra, porque la semilla depende solo de los caracteres de la palabra.

5. Conclusiones

La práctica nos permitió comprender e implementar el funcionamiento de una función digestiva hash de manera simulada con la funcion que se nos brindo, utilizando los conceptos vistos en clase como uso de argumentos desde la línea de comandos, las estructuras de datos como ArrayList y HashMap, y la generación determinista de valores a partir de una semilla. Y quedo claro como una entrada puede transformarse en una representación fija y única(firma), reflejando el principio básico de las funciones hash reales.
Sobre el problema se logró que cada palabra ingresada desde la consola produjera siempre el mismo hash, cumpliendo así con la propiedad de determinismo que se esperaba de la funcion. Además, se reforzó el uso práctico de estructuras de datos para almacenar y asociar información de forma eficiente.

6. Referencias bibliográficas

[1] Oracle. (s.f.). List (Java Platform, SE 8) Documentation.
Recuperado de https://docs.oracle.com/javase/8/docs/api/java/util/List.html

[2] Universitat Carlemany, "¿Qué es la iteración en programación?,"29 de nov. de 2023. [En línea]. Disponible en: https://www.universitatcarlemany.com/actualidad/blog/que-es-la-iteracion/.