static hashing refers to a method of organizing and storing data in a hash table where the number of buckets (or partitions) remains fixed. Each bucket is associated with a specific range of hash values, and a hash function is used to map data items to these buckets. This technique is commonly used in the implementation of hash-based indexing in database systems.

Code

import java.util.ArrayList;

import java.util.Scanner;

class StaticHashing {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

boolean keepGoing = true;

ArrayList<Integer>[] list = new ArrayList[8];

for (int i = 0; i < 8; i++) {

list[i] = new ArrayList<>(3);

}

while (keepGoing) {

System.out.println("1) Enter into Database ");

System.out.println("else Exit");

int choice = scanner.nextInt();

if (choice == 1) {

System.out.print("Enter Data: ");

String input = scanner.next();

long ans = calculateBinarySum(input);

list[(int) (ans % 8)].add((int) ans);

}

else {

keepGoing = false;

}

}

System.out.println("Buckets after insertion :");

for (int i = 0; i < 8; i++) {

System.out.println(list[i]);

}

scanner.close();

}

private static long calculateBinarySum(String name) {

long binarySum = 0;

for (char c : name.toCharArray()) {

// Convert each character to binary and add to the sum

String binaryRepresentation = String.format("%8s",

Integer.toBinaryString(c)).replace(' ', '0');

binarySum += Long.parseLong(binaryRepresentation, 2);

}

return binarySum;

}

}