Data Structure and Algorithms (JAVA)



**10th Lab**

**Semester: Spring 2025**

**Software Engineering**

**Faculty of Information Technology, UCP Lahore, Pakistan**

**Lab 10: Applications of Hashing**

## 1. What is Hashing?

**Hashing** is a technique to **map data of arbitrary size** to **fixed-size values** using a **hash function**.  
This helps in **fast data access**, usually in **O(1)** time for search, insert, and delete operations (average case).

|  |
| --- |
| int hash = key.hashCode() % size; // Example hash function |

## 2. What is a Hash Table?

A **Hash Table** uses a **hash function** to compute an index into an array (bucket) where data is stored.

* **Keys** are hashed to an index.
* **Values** are stored at that index.
* If two keys hash to the same index → **collision** occurs.

## 3. Collision Resolution Techniques

### a. ****Chaining**** (Used in Java's HashMap)

* Store multiple elements at the same index using a **LinkedList** or a similar structure.

### b. ****Open Addressing****

* If a position is occupied, find the next available one using:
  + Linear Probing
  + Quadratic Probing
  + Double Hashing

## 4. Java's HashMap<K, V>

* Implements **Map interface**
* Stores **key-value pairs**
* Keys must be unique
* Allows one null key and multiple null values

### Example:

|  |
| --- |
| import java.util.HashMap;  public class HashMapExample {  public static void main(String[] args) {  HashMap<String, Integer> marks = new HashMap<>();  // Insert  marks.put("Alice", 85);  marks.put("Bob", 90);  marks.put("Charlie", 75);  // Retrieve  System.out.println("Bob's marks: " + marks.get("Bob"));  // Check existence  System.out.println("Has Alice? " + marks.containsKey("Alice"));  // Loop  for (String name : marks.keySet()) {  System.out.println(name + ": " + marks.get(name));  }  }  } |

## 5. Java's HashSet<E>

* Implements **Set interface**
* Stores **unique elements only**
* Internally uses a HashMap with dummy values

### Example:

|  |
| --- |
| import java.util.HashSet;  public class HashSetExample {  public static void main(String[] args) {  HashSet<String> colors = new HashSet<>();  // Insert  colors.add("Red");  colors.add("Green");  colors.add("Blue");  colors.add("Red"); // Duplicate, will be ignored  // Check existence  System.out.println("Has Green? " + colors.contains("Green"));  // Loop  for (String color : colors) {  System.out.println(color);  }  }  } |

## ****Practice Tasks****

### Lab Task 1: Simple Login System Using Hashing

### Problem:

You are designing a basic login system where users register with a **username** and **password**. These credentials are stored using a HashMap. When a user tries to log in, the system checks if the username exists and whether the entered password matches the stored one.

import java.util.\*;

public class SimpleLoginSystem {

public static void main(String[] args) {

// Create a HashMap to store usernames and passwords

HashMap<String, String> users = new HashMap<>();

// Registering users (username -> password)

users.put("ali123", "passAli");

users.put("fatima", "fatima2025");

users.put("zain", "zainStrong");

// Simulate login

Scanner scanner = new Scanner(System.in);

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

String inputUsername = scanner.nextLine();

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

String inputPassword = scanner.nextLine();

// Check if user exists and password matches

if (users.containsKey(inputUsername)) {

String storedPassword = users.get(inputUsername);

if (storedPassword.equals(inputPassword)) {

System.out.println("Login successful! Welcome, " + inputUsername);

} else {

System.out.println("Incorrect password.");

}

} else {

System.out.println("Username not found.");

}

scanner.close();

}

**Lab Task 2: Student ID Registration System Using HashSet**

**Problem**: You are developing a student registration system. Each student has a unique **ID number**. The system must:

* Allow new student ID entries.
* Prevent duplicate IDs from being added (e.g., by mistake).
* Display all registered IDs.

import java.util.\*;

public class StudentIDRegistry {

public static void main(String[] args) {

// Create a HashSet to store unique student IDs

HashSet<String> studentIDs = new HashSet<>();

// Scanner for input

Scanner scanner = new Scanner(System.in);

// Register 3 student IDs

for (int i = 0; i < 3; i++) {

System.out.print("Enter Student ID: ");

String id = scanner.nextLine();

// Check if ID already exists

if (studentIDs.contains(id)) {

System.out.println("Error: Student ID already registered!");

} else {

studentIDs.add(id);

System.out.println("Student ID registered successfully.");

}

}

// Show all registered IDs

System.out.println("\nRegistered Student IDs:");

for (String id : studentIDs) {

System.out.println(id);

}

scanner.close();

}

}

**Lab Task 3: Country-Capital Lookup Using HashMap**

**Problem:**You are building a program that stores countries and their capitals. The user can:

* Add a country and its capital.
* Search for the capital of a specific country.
* Display all country-capital pairs.

This demonstrates how a HashMap associates keys (countries) with values (capitals).

**How Hashing Works Here:**

* HashMap<String, String> maps each country name to its capital.
* Internally, hashing helps locate the country quickly for fast lookup (get()).
* It avoids duplicates — the same country key cannot be added twice without overwriting.