| Experiment# | | Student ID | |
|---|---|---|---|
| Date | | Student Name | |

## 13. Implementation of Sets and Maps

**Aim/Objective:** To understand the concept and implementation of concepts of sets and maps.

**Description:** The student will understand the concepts of sets and maps.

**Pre-Requisites:** Classes and Objects, HashMap, TreeMap, HashSet and TreeSet in JAVA.

**Tools:** Eclipse IDE for Enterprise Java and Web Developers

**Pre-Lab:**

1) Explain the concept of a set in Java. What is the main characteristic of a set? Provide an example code snippet demonstrating the usage of a set.

In Java, a Set is a collection that does not allow duplicate elements. It models the mathematical set abstraction and provides methods for operations such as union, intersection, and difference. One key characteristic of a set is that the elements are unordered, and duplicate elements are automatically ignored.

Java Provides several implementations of the Set interface, the most common ones being:

Hash set: Stores elements in a hash table, allowing fast access but no ordering.

LinkedHash Set: Maintains insertion order.

TreeSet: Stores elements in a sorted tree structure.

Main Characteristics of a Set:

No duplicate elements: A set cannot contain duplicate entries.

Unordered collection: The elements have no specific order, except in the case of TreeSet or LinkedHashSet.

| Experiment# | | Student ID | |
|---|---|---|---|
| Date | | Student Name | |

**In-Lab:**

1) You are tasked with developing a contact management application where you need to store a collection of contacts. Each contact has a name and a phone number. Implement a class called ContactManager that uses a map to store the contacts, with the contact's name as the key and the phone number as the value. Write code to demonstrate the implementation of the ContactManager class.

Requirements

1. ContactManager Class:
   - Use a map to store contacts.
   - The key of the map is the contact's name (String).
   - The value of the map is the phone number (String).
   - Provide methods to add, remove, and retrieve contacts.
   - Provide a method to list all contacts.
2. Demonstration:
   - Create an instance of ContactManager.
   - Add several contacts.
   - Retrieve and display a contact's phone number.
   - Remove a contact.
   - List all contacts.

Procedure/Program:

```
import java.util.HashMap;
import java.util.Map;
class ContactManager{
    private Map<String, String> contacts;
    public ContactManager() {
        contacts = new HashMap<>();
    }
public void addContact(String name, String phoneNumber) {
    contacts.put(name, phoneNumber);
    System.out.println("Contact added: "+name+" -> "+phoneNumber);
}
```

```
public void removeContact(String name) {
    if(contacts.containsKey(name)) {
        contacts.remove(name);
        System.out.println("Contact removed: "+name);
    } else {
        System.out.println("Contact not found: "+name);
    }
}public String getContact(String name){
    return contacts.getOrDefault(name, "Contact not found.");
}public void listAllContacts(){
    if(contacts.isEmpty()){
        System.out.println("No Contacts Available");
    } else {
        System.out.println("Contact List:");
        for(Map.Entry<String, String> entry: contacts.entrySet()){
            System.out.println(entry.getkey()+"->"+entry.getValue());
        }
    }
}
public class ContactManagerDemo{
    public static void main(String[] args){
        ContactManager contactManager = new ContactManager();
        ContactManager.addContact("John Doe", "123-456-7890");
```

```
        contactManager.addContact("Jane Smith", "987-654-3210");
        contactManager.addContact("Emily Davis", "555-123-4567");
        System.out.println("John Doe's phone number: " + contactManager.getContact
                                                        ("John Doe"));

        contactManager.removeContact("Jane Smith");
        contactManager.listAllContacts();
    }
}
```

**VIVA-VOCE Questions (In-Lab):**

1) Explain the difference between a set and a map in programming.

Set: A collection that stores unique elements without any key-value pairing. It only stores values, ensuring no duplicates.

Map: A collection of key-value pairs, where each key is unique. It allows you to associate a specific value with a unique key.

2) What is the difference between a Set and a HashSet?

Set: An interface in Java that defines a collection of unique elements. It can be implemented in different ways.

HashSet: A concrete implementation of the Set Interface. that stores elements in a hash table, ensuring uniqueness and providing average constant-time performance for basic operations like add, remove and contains.

3) What is the difference between a Map and a HashMap?

Map: An interface in Java that defines a collection of key-value pairs. It can be implemented by different classes.

HashMap: A concrete implementation of the Map interface that uses a hash table to store key-value pairs. It allows fast access to data using the hash of the keys, with O(1) average time complexity for basic operations.

4) How does a map differ from an array or a list?

Array/List: Stores a collection of elements, accessed by their index. They store values in a linear fashion and allow duplicates.

Map: Stores key-value pairs, where each key maps to a value. The key is used for accessing the value instead of an index, keys are unique, and each key maps to exactly one value.

5) What is the purpose of the key-value pairs in a map? Can the same key have multiple values?

* The purpose of key-value pairs is to allow quick retrieval of values by using a unique key, similar to a dictionary lookup.

* Can the Same key have Multiple values? : No, in a Map, each key can only have one associated value. However, the value can be a collection (e.g., List or Set), allowing multiple values to be stored for the same key indirectly.

| Experiment# | | Student ID | |
|---|---|---|---|
| Date | | Student Name | |

2) You are working on a program that stores student grades. Each student has a unique ID assigned to them. Design a data structure using sets and maps to efficiently store and retrieve student grades based on their ID. Write the code for adding a student's grade to the data structure given their ID and grade.

Procedure/Program:

```
import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;
class StudentGradesManager{
    private Map<Integer, Set<Integer>> studentGrades = new HashMap<>();
    public void addGrade(int studentId, int grade) {
        studentGrades.computeIfAbsent(studentId, k->new HashSet<>()).
                                                    add(grade);
    }
    public Set<Integer> getGrades(int studentId) {
        return studentGrades.getOrDefault(studentId, new HashSet<>());
    }
    public void listAllStudentGrades() {
        studentGrades.forEach((id, grades) -> System.out.println("Student ID :"
                                                    +id+"-> Grades: "+grades));
    }
}
public class StudentGradesDemo {
    public static void main(String[] args) {
        StudentGradesManager manager = new StudentGradesManager();
        manager.addGrade(101,85);
        manager.addGrade(101,90);
        manager.addGrade(102,78);
```

| Experiment# | | Student ID | |
|---|---|---|---|
| Date | | Student Name | |

✓ **Data and Results:**

Grades for Student ID 101: [85, 90]

Student ID : 101 -> Grades : [85, 90]

Student ID : 102 -> Grades : [78]

✓ **Analysis and Inferences:**

Efficiency: The HashMap allows quick access to student grades with O(1) time complexity.

CorrectNess: The program prevents duplicate grades using a HashSet.

Scalability: Easily handles more students and grades without affecting performance.

Inference: The combination of Map and Set effectively stores and retrieves unique student grades.