Experiment 4

Student Name:Hardhik K **UID:** 22BCS10990
Branch: BE CSE **Section/Group:** 642/A
Semester: 06 **Date of Performance:** 17-2-25
Subject Name: Project Based Learning in Java **Subject Code:** 22CSH-359

1. Aim-

Easy: Problem Statement: Write a Java program to implement an ArrayList that stores employee details (ID, Name, and Salary). Allow users to add, update, remove, and search employees.

Medium: Create a program to collect and store all the cards to assist the users in finding all the cards in a given symbol using Collection interface.

Hard: Problem Statement: Develop a ticket booking system with synchronized threads to ensure no double booking of seats. Use thread priorities to simulate VIP bookings being processed first.

2. Code-

EASY:

```
import java.util.HashMap;
import java.util.Scanner;

class Employee {
int id;
String name;
double salary;

Employee(int id, String name, double salary) {
this.id = id;
this.name = name;
this.salary = salary;
}

public String toString() {
return "ID: " + id + ", Name: " + name + ", Salary: " + salary;
}
}

public class EmployeeManagement {
```

```
static HashMap<Integer, Employee> employees = new HashMap<>();
static Scanner sc = new Scanner(System.in);

public static void main(String[] args) {
while (true) {
System.out.println("\n1. Add Employee\n2. Update Employee\n3. Remove Employee\n4.
Search Employee\n5. Exit");
System.out.print("Enter choice: ");
int choice = sc.nextInt();

switch (choice) {
case 1: addEmployee(); break;
case 2: updateEmployee(); break;
case 3: removeEmployee(); break;
case 4: searchEmployee(); break;
case 5: System.exit(0);
default: System.out.println("Invalid choice!");
}
}
}

static void addEmployee() {
System.out.print("Enter ID: ");
int id = sc.nextInt();
sc.nextLine();
System.out.print("Enter Name: ");
String name = sc.nextLine();
System.out.print("Enter Salary: ");
double salary = sc.nextDouble();
employees.put(id, new Employee(id, name, salary));
System.out.println("Employee added!");
}

static void updateEmployee() {
System.out.print("Enter Employee ID to update: ");
int id = sc.nextInt();
if (employees.containsKey(id)) {
sc.nextLine();
System.out.print("Enter new Name: ");
String name = sc.nextLine();
System.out.print("Enter new Salary: ");
double salary = sc.nextDouble();
employees.put(id, new Employee(id, name, salary));
System.out.println("Employee updated!");
```

```
} else {
System.out.println("Employee not found!");
}
}

static void removeEmployee() {
System.out.print("Enter Employee ID to remove: ");
int id = sc.nextInt();
if (employees.remove(id) != null) {
System.out.println("Employee removed!");
} else {
System.out.println("Employee not found!");
}
}

static void searchEmployee() {
System.out.print("Enter Employee ID to search: ");
int id = sc.nextInt();
if (employees.containsKey(id)) {
System.out.println(employees.get(id));
} else {
System.out.println("Employee not found!");
}
}
}
```

MEDIUM:

```
import java.util.*;

class CardCollection {
    HashMap<String, List<String>> cardMap = new HashMap<>();

    void addCard(String symbol, String cardName) {
        cardMap.putIfAbsent(symbol, new ArrayList<>());
        cardMap.get(symbol).add(cardName);
    }

    void displayCards(String symbol) {
        if (cardMap.containsKey(symbol)) {
            System.out.println("Cards with symbol '" + symbol + "': " + cardMap.get(symbol));
        } else {
            System.out.println("No cards found for symbol: " + symbol);
        }
    }

    public static void main(String[] args) {
        CardCollection collection = new CardCollection();
        collection.addCard("Heart", "Ace of Hearts");
        collection.addCard("Heart", "King of Hearts");
        collection.addCard("Spade", "Queen of Spades");
        collection.addCard("Diamond", "Jack of Diamonds");

        Scanner sc = new Scanner(System.in);
        System.out.print("Enter symbol to search for cards: ");
        String symbol = sc.nextLine();
        collection.displayCards(symbol);
    }
}
```

HARD:

```
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

class TicketBookingSystem {
    private int availableSeats = 5;
    private final Lock lock = new ReentrantLock();

    void bookSeat(String passenger, boolean isVIP) {
        if (isVIP) Thread.currentThread().setPriority(Thread.MAX_PRIORITY);
        lock.lock();
        try {
            if (availableSeats > 0) {
                System.out.println(passenger + " successfully booked a seat. Seats left: " + (--
availableSeats));
            } else {
                System.out.println(passenger + " booking failed. No seats available.");
            }
        } finally {
            lock.unlock();
        }
    }
}

public class TicketBooking {
    public static void main(String[] args) {
        TicketBookingSystem system = new TicketBookingSystem();

        Runnable regularBooking = () -> system.bookSeat(Thread.currentThread().getName(),
false);
        Runnable vipBooking = () -> system.bookSeat(Thread.currentThread().getName(), true);

        Thread[] threads = new Thread[7];

        threads[0] = new Thread(vipBooking, "VIP-1");
        threads[1] = new Thread(vipBooking, "VIP-2");
        threads[2] = new Thread(regularBooking, "Passenger-1");
        threads[3] = new Thread(regularBooking, "Passenger-2");
        threads[4] = new Thread(regularBooking, "Passenger-3");
        threads[5] = new Thread(regularBooking, "Passenger-4");
        threads[6] = new Thread(regularBooking, "Passenger-5");

        for (Thread t : threads) t.start();
    }}
```

OUTPUT:

EASY:

```
1. Add Employee
2. Update Employee
3. Remove Employee
4. Search Employee
5. Exit
Enter choice: 1
Enter ID: 1234
Enter Name: Hardhik
Enter Salary: 123456
Employee added!

1. Add Employee
2. Update Employee
3. Remove Employee
4. Search Employee
5. Exit
Enter choice: 4
Enter Employee ID to search: 1234
ID: 1234, Name: Hardhik, Salary: 123456.0

1. Add Employee
2. Update Employee
3. Remove Employee
4. Search Employee
5. Exit
Enter choice: 2
Enter Employee ID to update: 1234
Enter new Name: Hardhikk
Enter new Salary: 10990
Employee updated!

1. Add Employee
2. Update Employee
3. Remove Employee
4. Search Employee
5. Exit
Enter choice: 4
Enter Employee ID to search: 1234
ID: 1234, Name: Hardhikk, Salary: 10990.0
```

MEDUIM:

```
Enter symbol to search for cards: Heart
Cards with symbol 'Heart': [Ace of Hearts, King of Hearts]
```

HARD:

```
VIP-1 successfully booked a seat. Seats left: 4
VIP-2 successfully booked a seat. Seats left: 3
Passenger-1 successfully booked a seat. Seats left: 2
Passenger-2 successfully booked a seat. Seats left: 1
Passenger-3 successfully booked a seat. Seats left: 0
Passenger-4 booking failed. No seats available.
Passenger-5 booking failed. No seats available.
```

3. Learning Outcomes-

Master Java Collections – Efficient use of ArrayList and HashMap for data management.
Implement CRUD Operations – Perform add, update, delete, and search efficiently.
Handle User Input & Validation – Ensure accurate data entry and processing. Apply Multithreading & Synchronization – Prevent data inconsistencies in concurrent tasks.
Enhance Problem-Solving Skills – Develop logical thinking for efficient data handling and decision-making.