# Experiment 4

**Student Name:** Aman Kumar
**UID:** 22BCS14634
**Branch:** BE-CSE
**Section/Group:** IOT-618/B
**Semester:** 6th
**Date of Performance:** 21/02/25
**Subject Name:** Project Based Learning in Java
**Subject Code:** 22CSH - 359

**4.1. Aim:** 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.

**4.1. Implementation/Code:**

```
import java.util.ArrayList;

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;

    }


    @Override

    public String toString() {

        return "ID: " + id + ", Name: " + name + ", Salary: " + salary;
```

```
    }
}

public class EmployeeManager {
    private ArrayList<Employee> employees = new ArrayList<>();
    private Scanner scanner = new Scanner(System.in);

    public void addEmployee() {
        System.out.print("Enter Employee ID: ");
        int id = scanner.nextInt();
        scanner.nextLine();
        System.out.print("Enter Employee Name: ");
        String name = scanner.nextLine();
        System.out.print("Enter Employee Salary: ");
        double salary = scanner.nextDouble();
        employees.add(new Employee(id, name, salary));
        System.out.println("Employee Added Successfully.");
    }

    public void updateEmployee() {
        System.out.print("Enter Employee ID to Update: ");
        int id = scanner.nextInt();
        boolean found = false;
```

```
    for (Employee emp : employees) {
        if (emp.id == id) {
            found = true;
            scanner.nextLine();
            System.out.print("Enter New Name: ");
            emp.name = scanner.nextLine();
            System.out.print("Enter New Salary: ");
            emp.salary = scanner.nextDouble();
            System.out.println("Employee Updated Successfully.");
            break;
        }
    }

    if (!found) {
        System.out.println("Employee Not Found.");
    }
}

public void removeEmployee() {
    System.out.print("Enter Employee ID to Remove: ");
    int id = scanner.nextInt();
    boolean removed = employees.removeIf(emp -> emp.id == id);
    if (removed) {
        System.out.println("Employee Removed Successfully.");
```

```
    } else {
        System.out.println("Employee Not Found.");
    }
}

public void searchEmployee() {
    System.out.print("Enter Employee ID to Search: ");
    int id = scanner.nextInt();
    boolean found = false;

    for (Employee emp : employees) {
        if (emp.id == id) {
            found = true;
            System.out.println("Employee Found: " + emp);
            break;
        }
    }

    if (!found) {
        System.out.println("Employee Not Found.");
    }
}

public void displayEmployees() {
```

```
    if (employees.isEmpty()) {
        System.out.println("No Employees to Display.");
    } else {
        for (Employee emp : employees) {
            System.out.println(emp);
        }
    }
}

public static void main(String[] args) {
    EmployeeManager manager = new EmployeeManager();
    Scanner scanner = new Scanner(System.in);
    int choice;

    while (true) {
        System.out.println("\nEmployee Management System");
        System.out.println("1. Add Employee");
        System.out.println("2. Update Employee");
        System.out.println("3. Remove Employee");
        System.out.println("4. Search Employee");
        System.out.println("5. Display All Employees");
        System.out.println("6. Exit");
        System.out.print("Enter Your Choice: ");
        choice = scanner.nextInt();
```

```
    switch (choice) {
        case 1:
            manager.addEmployee();
            break;
        case 2:
            manager.updateEmployee();
            break;
        case 3:
            manager.removeEmployee();
            break;
        case 4:
            manager.searchEmployee();
            break;
        case 5:
            manager.displayEmployees();
            break;
        case 6:
            System.out.println("Exiting Program...");
            return;
        default:
            System.out.println("Invalid Choice. Please Try Again.");
    }
}
```

```
    }
}
```

**4.1. Output:**

```
PROBLEMS 1    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\amank\OneDrive\Desktop\Java 4th> cd "
nager }

Employee Management System
1. Add Employee
2. Update Employee
3. Remove Employee
4. Search Employee
5. Display All Employees
6. Exit
Enter Your Choice: 1
Enter Employee ID: 14634
Enter Employee Name: Aman Kumar
Enter Employee Salary: 15000
Employee Added Successfully.

Employee Management System
1. Add Employee
2. Update Employee
3. Remove Employee
4. Search Employee
5. Display All Employees
6. Exit
Enter Your Choice: 5
ID: 14634, Name: Aman Kumar, Salary: 15000.0
```

**4.2. Aim:** 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.

**4.2. Implementation/Code:**

```
import java.util.*;


public class CardCollection {


    static final String[] SUITS = {"Hearts", "Diamonds", "Clubs", "Spades"};

    static final String[] RANKS = {"2", "3", "4", "5", "6", "7", "8", "9", "10", "Jack", "Queen",
"King", "Ace"};


    static List<String> cards = new ArrayList<>();


    public static void main(String[] args) {

        generateDeck();


        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the symbol to find the cards (Hearts, Diamonds, Clubs,
Spades): ");

        String userSymbol = scanner.nextLine().trim();


        List<String> filteredCards = findCardsBySymbol(userSymbol);


        if (filteredCards.isEmpty()) {

            System.out.println("No cards found with the symbol: " + userSymbol);
```

```
    } else {
        System.out.println("Cards with symbol " + userSymbol + ":");
        for (String card : filteredCards) {
            System.out.println(card);
        }
    }

    scanner.close();
}

public static void generateDeck() {
    for (String suit : SUITS) {
        for (String rank : RANKS) {
            cards.add(rank + " of " + suit);
        }
    }
}

public static List<String> findCardsBySymbol(String symbol) {
    List<String> filteredCards = new ArrayList<>();
    for (String card : cards) {
        if (card.contains(symbol)) {
            filteredCards.add(card);
        }
```

```
        }
        return filteredCards;
    }
}
```

### 4.2. Output:

```
PROBLEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   COMMENTS

PS C:\Users\amank\OneDrive\Desktop\Java 4th> cd "c:\Users\amank\OneDrive\Desktop\Java 4th\"
ion }
Enter the symbol to find the cards (Hearts, Diamonds, Clubs, Spades): Hearts
Cards with symbol Hearts:
2 of Hearts
3 of Hearts
4 of Hearts
5 of Hearts
6 of Hearts
7 of Hearts
8 of Hearts
9 of Hearts
10 of Hearts
Jack of Hearts
Queen of Hearts
King of Hearts
Ace of Hearts
PS C:\Users\amank\OneDrive\Desktop\Java 4th>
```

**4.3. Aim:** 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.

### 4.3. Implementation/Code:

```
import java.util.Scanner;


class TicketBookingSystem {
    private final String[] seats;
```

```
private final Object lock = new Object();

public TicketBookingSystem(int totalSeats) {
    seats = new String[totalSeats];
    for (int i = 0; i < totalSeats; i++) {
        seats[i] = "Available";
    }
}

public boolean bookTicket(String user, boolean isVIP) {
    synchronized (lock) {
        for (int i = 0; i < seats.length; i++) {
            if (seats[i].equals("Available")) {
                seats[i] = user;
                System.out.println("Seat " + (i + 1) + " booked for " + user + ".");
                return true;
            }
        }
        System.out.println("Sorry, " + user + ". No available seats.");
        return false;
    }
}

public void showSeats() {
```

```
        System.out.println("Current seat status:");
        for (int i = 0; i < seats.length; i++) {
            System.out.println("Seat " + (i + 1) + ": " + seats[i]);
        }
    }
}

class BookingThread extends Thread {
    private final TicketBookingSystem bookingSystem;
    private final String user;
    private final boolean isVIP;

    public BookingThread(TicketBookingSystem bookingSystem, String user, boolean
isVIP) {
        this.bookingSystem = bookingSystem;
        this.user = user;
        this.isVIP = isVIP;
    }

    @Override
    public void run() {
        try {
            Thread.sleep(1000); // Simulate processing time
            if (isVIP) {
```

```
            System.out.println("VIP booking for " + user + " is being processed...");
        } else {
            System.out.println("Booking for " + user + " is being processed...");
        }
        bookingSystem.bookTicket(user, isVIP);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
  }
}

public class TicketBookingApp {

  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    // Input for total number of seats
    System.out.print("Enter total number of seats: ");
    int totalSeats = scanner.nextInt();
    scanner.nextLine();  // Consume newline

    TicketBookingSystem bookingSystem = new TicketBookingSystem(totalSeats);

    while (true) {
```

```
        System.out.print("\nEnter your name to book a ticket (or 'exit' to quit): ");
        String user = scanner.nextLine();
        if (user.equalsIgnoreCase("exit")) {
            break;
        }

        System.out.print("Are you a VIP? (yes/no): ");
        String vipResponse = scanner.nextLine();
        boolean isVIP = vipResponse.equalsIgnoreCase("yes");

        BookingThread bookingThread = new BookingThread(bookingSystem, user, isVIP);

        if (isVIP) {
            bookingThread.setPriority(Thread.MIN_PRIORITY);
        } else {
            bookingThread.setPriority(Thread.NORM_PRIORITY);
        }

        bookingThread.start();
    }

    bookingSystem.showSeats();
    scanner.close();
}
```

```
}
```

## 4.3. Output:

```
PROBLEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   COMMENTS

PS C:\Users\amank\OneDrive\Desktop\Java 4th> cd "c:\Users\amank\OneDrive\Desktop\Java 4th\" ; if ($?)
kingApp }
Enter total number of seats: 10

Enter your name to book a ticket (or 'exit' to quit): Aman Kumar
Are you a VIP? (yes/no): no

Enter your name to book a ticket (or 'exit' to quit): Booking for Aman Kumar is being processed...
Seat 1 booked for Aman Kumar.
```