Experiment-4

Student Name: Premprakash Motwani
UID: 22BCS15179
Branch: BE-CSE
Section/Group: KPIT-902/B
Semester: 6th
Date of Performance: 21/02/25
Subject Name: PBLJ-Lab
Subject Code: 22CSH-359

1. Aim:

Develop Java programs using core concepts such as data structures, collections, and multithreading to manage and manipulate data.

a.) 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.

b.) 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.

c.) 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. Objective:

The objective of this Java program is to develop applications utilizing core Java concepts such as data structures, collections, and multithreading to efficiently manage and manipulate data. The program includes:

a.) Employee Management System: Implementing an ArrayList to store employee details (ID, Name, and Salary) while providing functionalities to add, update, remove, and search employees.

b.) Card Collection System: Utilizing the Collection interface to store and organize cards, enabling users to efficiently locate all cards of a given symbol.

c.) Ticket Booking System: Implementing synchronized threads to prevent double booking of seats while leveraging thread priorities to ensure VIP bookings are processed first.

This program demonstrates key Java features such as data organization using collections, efficient data retrieval, and safe concurrent execution through multithreading.

3. Implementation/Code:

a.) Employee Management

```
import java.util.*;

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

    public 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 EmployeeManagement {
    private static List<Employee> employees = new ArrayList<>();
    private static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        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("Choose an option: ");

            int choice = scanner.nextInt();
            scanner.nextLine(); // Consume newline

            switch (choice) {
                case 1: addEmployee(); break;
                case 2: updateEmployee(); break;
                case 3: removeEmployee(); break;
                case 4: searchEmployee(); break;
                case 5: displayEmployees(); break;
                case 6: System.exit(0);
                default: System.out.println("Invalid choice. Please try again.");
            }
        }
    }

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

    private static void updateEmployee() {
        System.out.print("Enter Employee ID to update: ");
        int id = scanner.nextInt();
        scanner.nextLine();
        for (Employee emp : employees) {
            if (emp.id == id) {
                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.");
                return;
            }
        }
        System.out.println("Employee not found.");
    }

    private static void removeEmployee() {
        System.out.print("Enter Employee ID to remove: ");
        int id = scanner.nextInt();
        scanner.nextLine();
        employees.removeIf(emp -> emp.id == id);
        System.out.println("Employee removed successfully.");
    }

    private static void searchEmployee() {
        System.out.print("Enter Employee ID to search: ");
        int id = scanner.nextInt();
        scanner.nextLine();
        for (Employee emp : employees) {
            if (emp.id == id) {
                System.out.println(emp);
                return;
            }
        }
        System.out.println("Employee not found.");
    }

    private static void displayEmployees() {
        if (employees.isEmpty()) {
            System.out.println("No employees to display.");
            return;
        }
        for (Employee emp : employees) {
            System.out.println(emp);
        }
    }
}
```

Output :

```
Employee Management System
1. Add Employee
2. Update Employee
3. Remove Employee
4. Search Employee
5. Display All Employees
6. Exit
Choose an option: 11
Invalid choice. Please try again.

Employee Management System
1. Add Employee
2. Update Employee
3. Remove Employee
4. Search Employee
5. Display All Employees
6. Exit
Choose an option: 1
Enter ID: 13765
Enter Name: Akif
Enter Salary: 70000
Employee added successfully.

Employee Management System
1. Add Employee
2. Update Employee
3. Remove Employee
4. Search Employee
5. Display All Employees
6. Exit
Choose an option: 5
ID: 13765, Name: Akif, Salary: 70000.0
```

b.) Card Collection

```
import java.util.*;

class Card {
    private String symbol;
    private String value;

    public Card(String symbol, String value) {
        this.symbol = symbol;
        this.value = value;
    }

    public String getSymbol() {
        return symbol;
    }

    @Override
    public String toString() {
        return value + " of " + symbol;
    }
}

public class CardCollection {
    private static Collection<Card> cards = new ArrayList<>();
    private static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        //addDefaultCards();

        while (true) {
            System.out.println("\nCard Collection System");
            System.out.println("1. Add Card");
            System.out.println("2. Find Cards by Symbol");
            System.out.println("3. Display All Cards");
            System.out.println("4. Exit");
            System.out.print("Choose an option: ");

            int choice = scanner.nextInt();

            scanner.nextLine();
```

```
            switch (choice) {
                case 1: addCard(); break;
                case 2: findCardsBySymbol(); break;
                case 3: displayCards(); break;
                case 4: System.exit(0);
                default: System.out.println("Invalid choice. Please try again.");
            }
        }
    }

    private static void addDefaultCards() {
        String[] symbols = {"Hearts", "Diamonds", "Clubs", "Spades"};
        String[] values = {"Ace", "2", "3", "4", "5", "6", "7", "8", "9", "10", "Jack",
"Queen", "King"};
        for (String symbol : symbols) {
            for (String value : values) {
                cards.add(new Card(symbol, value));
            }
        }
    }

    private static void addCard() {
        System.out.print("Enter Symbol (Hearts, Diamonds, Clubs, Spades): ");
        String symbol = scanner.nextLine();
        System.out.print("Enter Value (Ace, 2-10, Jack, Queen, King): ");
        String value = scanner.nextLine();
        cards.add(new Card(symbol, value));
        System.out.println("Card added successfully.");
    }

    private static void findCardsBySymbol() {
        System.out.print("Enter Symbol to search: ");
        String symbol = scanner.nextLine();
        boolean found = false;
        for (Card card : cards) {
            if (card.getSymbol().equalsIgnoreCase(symbol)) {
                System.out.println(card);
                found = true;
            }
        }
```

```
            if (!found) {
                System.out.println("No cards found for the given symbol.");
            }
        }

        private static void displayCards() {
            if (cards.isEmpty()) {
                System.out.println("No cards to display.");
                return;
            }
            for (Card card : cards) {
                System.out.println(card);
            }
        }
    }
```

Output:

```
Card Collection System
1. Add Card
2. Find Cards by Symbol
3. Display All Cards
4. Exit
Choose an option: 1
Enter Symbol (Hearts, Diamonds, Clubs, Spades): Hearts
Enter Value (Ace, 2-10, Jack, Queen, King): 3
Card added successfully.

Card Collection System
1. Add Card
2. Find Cards by Symbol
3. Display All Cards
4. Exit
Choose an option: 3
3 of Hearts

Card Collection System
1. Add Card
2. Find Cards by Symbol
3. Display All Cards
4. Exit
Choose an option: 4


...Program finished with exit code 0
Press ENTER to exit console.
```

c.) Booking System

```
import java.util.Scanner;

class TicketBookingSystem {
    private final boolean[] seats;

    public TicketBookingSystem(int totalSeats) {
        seats = new boolean[totalSeats];
    }

    public synchronized boolean bookSeat(int seatNumber, String customer) {
        if (seatNumber < 0 || seatNumber >= seats.length || seats[seatNumber]) {
            System.out.println(customer + " failed to book seat " + seatNumber + ".
Seat is already booked.");
            return false;
        }
        seats[seatNumber] = true;
        System.out.println(customer + " successfully booked seat " + seatNumber);
        return true;
    }



    public void displaySeats() {
        System.out.print("Seat Status: ");
        for (boolean seat : seats) {
            System.out.print(seat ? "[Booked] " : "[Available] ");
        }
        System.out.println();
    }
}

class Customer extends Thread {
    private final TicketBookingSystem system;
    private final int seatNumber;
    private final String customerName;

    public Customer(TicketBookingSystem system, int seatNumber, String
customerName, int priority) {
        this.system = system;
        this.seatNumber = seatNumber;
```

```
        this.customerName = customerName;
        setPriority(priority);
    }

    @Override
    public void run() {
        system.bookSeat(seatNumber, customerName);
    }
}

public class TicketBookingApp {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        TicketBookingSystem system = new TicketBookingSystem(10);

        System.out.print("Enter number of customers: ");
        int numCustomers = scanner.nextInt();
        scanner.nextLine();

        Customer[] customers = new Customer[numCustomers];


        for (int i = 0; i < numCustomers; i++) {
            System.out.print("Enter name of customer " + (i + 1) + ": ");
            String name = scanner.nextLine();


            System.out.print("Enter seat number (0-9) for " + name + ": ");
            int seatNumber = scanner.nextInt();

            System.out.print("Enter priority (1 for VIP, 2 for Normal, 3 for Low): ");
            int priorityLevel = scanner.nextInt();
            scanner.nextLine();

            int priority = (priorityLevel == 1) ? Thread.MAX_PRIORITY :
(priorityLevel == 2 ? Thread.NORM_PRIORITY : Thread.MIN_PRIORITY);
            customers[i] = new Customer(system, seatNumber, name, priority);
        }

        for (Customer customer : customers) {
            customer.start();
        }
```

```
        for (Customer customer : customers) {
            try {
                customer.join();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }

        system.displaySeats();
        scanner.close();
    }
```

Output :

```
Enter number of customers: 3
Enter name of customer 1: ak
Enter seat number (0-9) for ak: 7
Enter priority (1 for VIP, 2 for Normal, 3 for Low): 1
Enter name of customer 2: sudhish
Enter seat number (0-9) for sudhish: 5
Enter priority (1 for VIP, 2 for Normal, 3 for Low): 1
Enter name of customer 3: Aryan
Enter seat number (0-9) for Aryan: 3
Enter priority (1 for VIP, 2 for Normal, 3 for Low): 1
ak successfully booked seat 7
Aryan successfully booked seat 3
sudhish successfully booked seat 5
Seat Status: [Available] [Available] [Available] [Booked] [Available] [Booked] [Available] [Booked] [Available] [Available]


...Program finished with exit code 0
Press ENTER to exit console.
```

4. Learning Outcome:

1. Learned thread synchronization to prevent multiple users from booking the same seat at the same time.
2. Understood thread priorities to ensure VIP bookings are processed before normal bookings.
3. Gained experience with Java Collections by using ArrayList and Collection interfaces for data storage and retrieval.
4. Practiced user input handling to dynamically take and process booking requests.
5. Implemented data management operations such as adding, updating, removing, and searching employee records.
6. Used object-oriented programming concepts like classes, objects, and encapsulation for a structured approach.
7. Developed problem-solving skills by designing and handling real-world scenarios like ticket booking and employee management.
8. Ensured data consistency by preventing duplicate seat bookings through synchronized methods.
9. Improved logic-building abilities by implementing control structures like loops and conditional statements for efficient processing.