DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
Discover. Learn. Empower.

Experiment 4

Name: Varun Kumar **UID: 22BCS16117**
Branch: BE-CSE **Section: 22BCS_IOT_639-A**
Semester:6th **Date of Performance: 14/2/2025**
Subject Name: Project Based Learning in Java with Lab **Subject Code: 22CSH-359**

1. **Aim:**

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:**

1. Implement an ArrayList to store, manage, and manipulate employee details (ID, Name, Salary) with CRUD operations.

2. Utilize the Collection interface to efficiently store and retrieve cards based on symbols, enabling easy searching and management.

3. Use synchronized threads to prevent double booking while prioritizing VIP bookings using thread priority.

3. **Implementation/Code:**

a). Arraylist Employee code:

```
import java.util.*;

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

    static void addEmployee() {
        System.out.print("Enter ID: ");
        int id = scanner.nextInt();
        scanner.nextLine();  // Consume the newline character
        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!");
    }
```

```
static void updateEmployee() {
    System.out.print("Enter Employee ID to update: ");
    int id = scanner.nextInt();
    scanner.nextLine(); // Consume newline
    boolean found = false;
    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!");
            found = true;
            break;
        }
    }
    if (!found) {
        System.out.println("Employee not found!");
    }
}

static 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!");
    }
}

static 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) {
            System.out.println("Employee Found: " + emp);
            found = true;
            break;
        }
```

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

    static void displayEmployees() {
        if (employees.isEmpty()) {
            System.out.println("No employees found.");
        } else {
            System.out.println("\nEmployee List:");
            for (Employee emp : employees) {
                System.out.println(emp);
            }
        }
    }

    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("Enter your choice: ");
            int choice = scanner.nextInt();

            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.out.println("Exiting the system. Goodbye!");
                    return;
                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        }
    }
}
```

b). Cards Code:

```
import java.util.*;

class Card {
    String symbol;
    int number;

    Card(String symbol, int number) {
        this.symbol = symbol;
        this.number = number;
    }

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

public class CardGame {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Map<String, List<Card>> cardMap = new HashMap<>();

        System.out.print("Enter Number of Cards: ");
        int n = scanner.nextInt();
        scanner.nextLine();

        for (int i = 1; i <= n; i++) {
```

```
            System.out.println("Enter card " + i + ":");
            String symbol = scanner.nextLine();
            int number = scanner.nextInt();
            scanner.nextLine();

            cardMap.putIfAbsent(symbol, new ArrayList<>());
            cardMap.get(symbol).add(new Card(symbol, number));
        }

        List<String> symbols = new ArrayList<>(cardMap.keySet());
        Collections.sort(symbols);

        System.out.println("\nDistinct Symbols are:");
        for (String symbol : symbols) {
            System.out.print(symbol + " ");
        }
        System.out.println("\n");

          for (String symbol : symbols) {
            List<Card> cards = cardMap.get(symbol);
            int sum = 0;

            System.out.println("Cards in " + symbol + " Symbol");
            for (Card card : cards) {
                System.out.println(card);
                sum += card.number;
            }

            System.out.println("Number of cards: " + cards.size());
            System.out.println("Sum of Numbers: " + sum);
            System.out.println();
        }

        scanner.close();
    }
}
```

c). Ticket booking:

```
import java.util.*;

class TicketBookingSystem {
```

```
    private final boolean[] seats;

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

    public synchronized boolean bookSeat(int seatNumber, String customerName)
{
        if (seatNumber < 0 || seatNumber >= seats.length) {
            System.out.println(customerName + " tried to book an invalid seat.");
            return false;
        }
        if (!seats[seatNumber]) {
            seats[seatNumber] = true;
            System.out.println(customerName + " successfully booked seat " +
seatNumber);
            return true;
        } else {
            System.out.println(customerName + " tried to book seat " + seatNumber +
" but it's already taken.");
            return false;
        }
    }

    public int getTotalSeats() {
        return seats.length;
    }
}

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

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

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

    public class TicketBookingApp {
        public static void main(String[] args) {
            TicketBookingSystem system = new TicketBookingSystem(10); // 10 seats
available

            List<Thread> bookingRequests = new ArrayList<>();
            bookingRequests.add(new BookingThread(system, "Abhishek (VIP)", 1,
Thread.MAX_PRIORITY));
            bookingRequests.add(new BookingThread(system, "Sachin (VIP)", 3,
Thread.MAX_PRIORITY));
            bookingRequests.add(new BookingThread(system, "Ansh (VIP)", 5,
Thread.MAX_PRIORITY));
            bookingRequests.add(new BookingThread(system, "Naitik", 1,
Thread.NORM_PRIORITY));
            bookingRequests.add(new BookingThread(system, "Shubham", 4,
Thread.NORM_PRIORITY));
            bookingRequests.add(new BookingThread(system, "Mayank", 5,
Thread.NORM_PRIORITY));

            Collections.shuffle(bookingRequests);

            for (Thread t : bookingRequests) {
                t.start();
            }

            for (Thread t : bookingRequests) {
                try {
                    t.join();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }

            System.out.println("\nAll bookings completed!");
        }
    }
```

4. **Output:**

a). Arraylist employee output:

```
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\JetBra

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 ID: 102
Enter Name: Abhishek
Enter Salary: 60000
Employee added successfully!
```

b). Cards Output:

```
6

Distinct Symbols are:
c d h s

Cards in c Symbol
c 5
c 5
c 2
Number of cards: 3
Sum of Numbers: 12

Cards in d Symbol
d 4
d 4
d 3
d 6
Number of cards: 4
Sum of Numbers: 17

Cards in h Symbol
h 5
h 9
Number of cards: 2
Sum of Numbers: 14
```

c). Ticket booking output:

```
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\
Naitik successfully booked seat 1
Shubham successfully booked seat 4
Sachin (VIP) successfully booked seat 3
Abhishek (VIP) tried to book seat 1 but it's already taken.
Ansh (VIP) successfully booked seat 5
Mayank tried to book seat 5 but it's already taken.

All bookings completed!

Process finished with exit code 0
```

5. **Learning Outcomes:**

1. Object-Oriented Programming (OOP): Learn how to design and implement custom classes with constructors, attributes, and methods, and override methods like toString() for object representation.

2. Working with Collections: Understand the use of Java collections such as ArrayList, HashMap, and List for storing and manipulating related data.

3. Multithreading and Synchronization: Learn how to create and manage threads in Java, ensuring thread safety with synchronized methods and handling concurrent operations.

4. User Input and Validation: Practice taking user input through the Scanner class and validating data, ensuring proper handling of edge cases.

5. Sorting and Organizing Data: Learn how to sort and organize data efficiently, such as sorting collections and calculating aggregates like sums.

6. Thread Management: Understand thread lifecycle management in Java, including controlling thread execution with start(), join(), and adjusting thread priorities.