Experiment 5

Student Name: Shaurya Anand | **UID: 22BCS15952**
Branch: CSE | **Section/Group:639/B**
Semester: 6th | **DOP:5/3/2025**
Subject: Java Lab | **Subject Code: 22CSH-359**

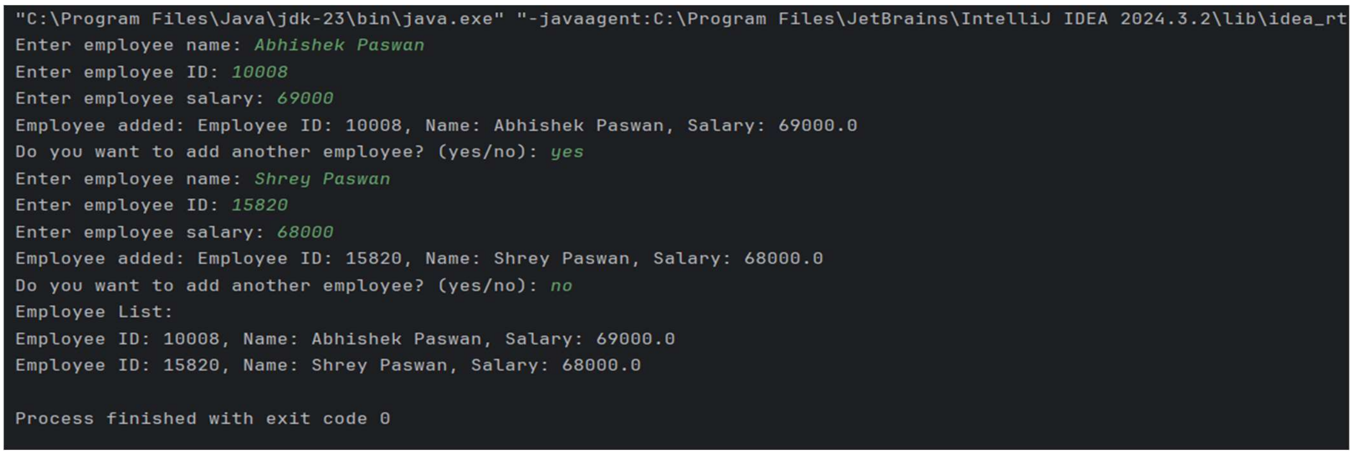
Aim:

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.

Code:

```
import java.util.ArrayList;
import java.util.Scanner;

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

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

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

public class EmployeeManagement {
    private ArrayList<Employee> employees = new ArrayList<>();

    public void addEmployee(String name, String id, double salary) {
        Employee employee = new Employee(name, id, salary);
        employees.add(employee);
        System.out.println("Employee added: " + employee);
    }

    public void displayEmployees() {
        if (employees.isEmpty()) {
            System.out.println("No employees to display.");
        } else {
            System.out.println("Employee List:");
            for (Employee employee : employees) {
                System.out.println(employee);
            }
        }
    }

        do {
            System.out.print("Enter employee name: ");
            String name = scanner.nextLine();
            System.out.print("Enter employee ID: ");
            String id = scanner.nextLine();
            System.out.print("Enter employee salary: ");
            double salary = scanner.nextDouble();
            scanner.nextLine();

            em.addEmployee(name, id, salary);

            System.out.print("Do you want to add another employee? (yes/no): ");
            choice = scanner.nextLine();
        } while (choice.equalsIgnoreCase("yes"));

        em.displayEmployees();
        scanner.close();
    }
}
```

Output:



```
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.3.2\lib\idea_rt
Enter employee name: Abhishek Paswan
Enter employee ID: 10008
Enter employee salary: 69000
Employee added: Employee ID: 10008, Name: Abhishek Paswan, Salary: 69000.0
Do you want to add another employee? (yes/no): yes
Enter employee name: Shrey Paswan
Enter employee ID: 15820
Enter employee salary: 68000
Employee added: Employee ID: 15820, Name: Shrey Paswan, Salary: 68000.0
Do you want to add another employee? (yes/no): no
Employee List:
Employee ID: 10008, Name: Abhishek Paswan, Salary: 69000.0
Employee ID: 15820, Name: Shrey Paswan, Salary: 68000.0

Process finished with exit code 0
```

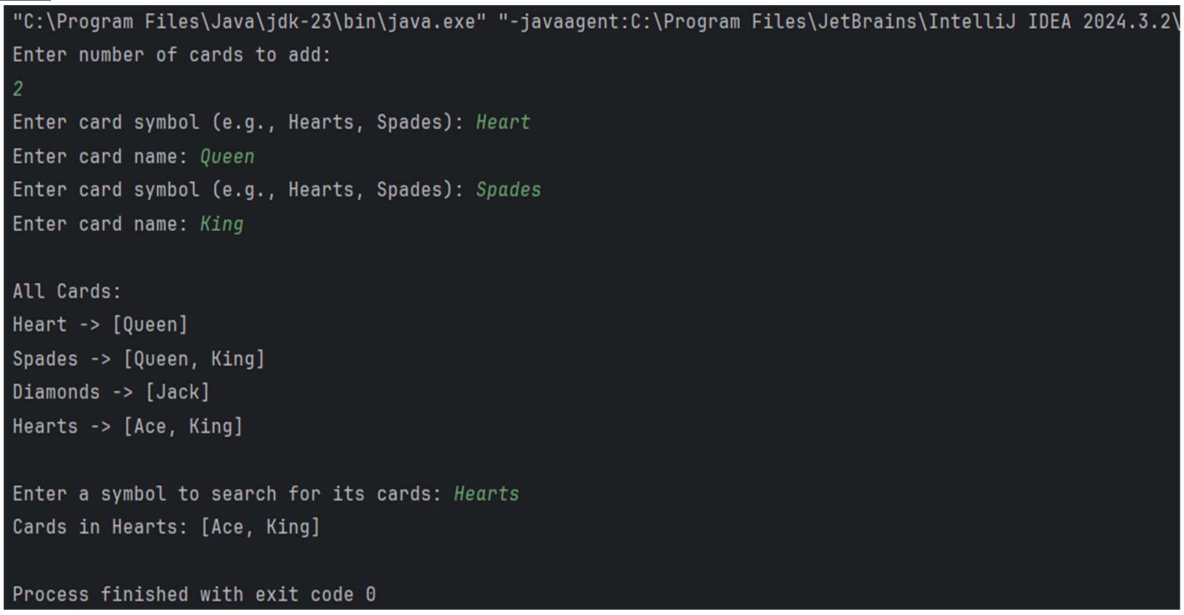
Aim:

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.

Code:

```
import java.util.*;

public class CardCollection {
    private Map<String, List<String>> cardMap;

    public CardCollection() {
        cardMap = new HashMap<>();
    }

    public void addCard(String symbol, String cardName) {
        cardMap.computeIfAbsent(symbol, k -> new ArrayList<>()).add(cardName);
    }

    public List<String> getCardsBySymbol(String symbol) {
        return cardMap.getOrDefault(symbol, Collections.emptyList());
    }

    public void displayAllCards() {
        for (Map.Entry<String, List<String>> entry : cardMap.entrySet()) {
            System.out.println(entry.getKey() + " -> " + entry.getValue());
        }
    }

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

        cardCollection.addCard("Hearts", "Ace");
        cardCollection.addCard("Hearts", "King");
        cardCollection.addCard("Spades", "Queen");
        cardCollection.addCard("Diamonds", "Jack");

        System.out.println("Enter number of cards to add:");
        int n = scanner.nextInt();
        scanner.nextLine();

        for (int i = 0; i < n; i++) {
            System.out.print("Enter card symbol: ");
            String symbol = scanner.nextLine();
            System.out.print("Enter card name: ");
            String cardName = scanner.nextLine();
            cardCollection.addCard(symbol, cardName);
        }

        System.out.println("\nAll Cards:");
        cardCollection.displayAllCards();

        System.out.print("\nEnter a symbol to search for its cards: ");
        String searchSymbol = scanner.nextLine();
        List<String> cards = cardCollection.getCardsBySymbol(searchSymbol);

        if (!cards.isEmpty()) {
            System.out.println("Cards in " + searchSymbol + ": " + cards);
        } else {
            System.out.println("No cards found for symbol: " + searchSymbol);
        }

        scanner.close();
    }
}
```

Output:



```
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.3.2\
Enter number of cards to add:
2
Enter card symbol (e.g., Hearts, Spades): Heart
Enter card name: Queen
Enter card symbol (e.g., Hearts, Spades): Spades
Enter card name: King

All Cards:
Heart -> [Queen]
Spades -> [Queen, King]
Diamonds -> [Jack]
Hearts -> [Ace, King]

Enter a symbol to search for its cards: Hearts
Cards in Hearts: [Ace, King]

Process finished with exit code 0
```

Aim:

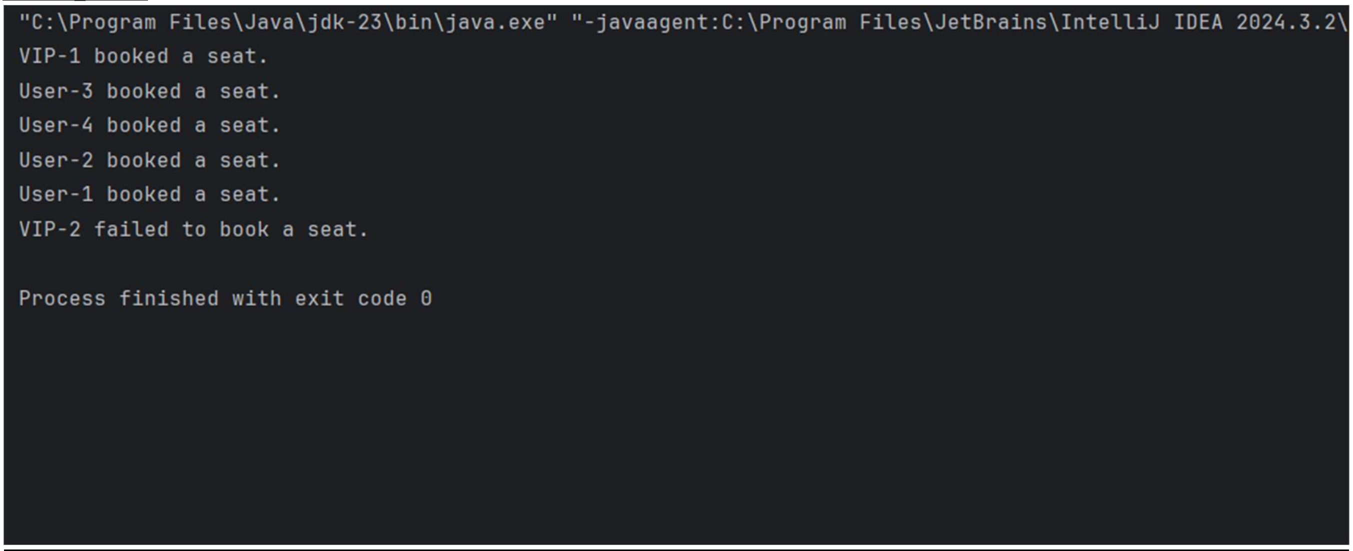
c) Develop a ticket booking system with synchronized threads to ensure no double booking of seats. Use thread priorities to simulate VIP bookings being processed first.

Code:

```
import java.util.*;

class TicketBookingSystem {
    private int availableSeats;

    public TicketBookingSystem(int seats) {
        this.availableSeats = seats;
    }

    public synchronized boolean bookSeat(String name) {
        if (availableSeats > 0) {
            System.out.println(name + " booked a seat.");
            availableSeats--;
            return true;
        } else {
            System.out.println(name + " failed to book a seat.");
            return false;
        }
    }
}

class BookingThread extends Thread {
    private TicketBookingSystem system;
    private String userName;

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

    public void run() {
        system.bookSeat(userName);
    }
}

public class TicketBookingApp {
    public static void main(String[] args) {
        TicketBookingSystem system = new TicketBookingSystem(5);

        BookingThread vip1 = new BookingThread(system, "VIP-1", Thread.MAX_PRIORITY);
        BookingThread vip2 = new BookingThread(system, "VIP-2", Thread.MAX_PRIORITY);
        BookingThread user1 = new BookingThread(system, "User-1", Thread.NORM_PRIORITY);
        BookingThread user2 = new BookingThread(system, "User-2", Thread.NORM_PRIORITY);
        BookingThread user3 = new BookingThread(system, "User-3", Thread.MIN_PRIORITY);
        BookingThread user4 = new BookingThread(system, "User-4", Thread.MIN_PRIORITY);

        vip1.start();
        vip2.start();
        user1.start();
        user2.start();
```

```
            user3.start();
            user4.start();
        }
    }
```

Output:



```
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.3.2\
VIP-1 booked a seat.
User-3 booked a seat.
User-4 booked a seat.
User-2 booked a seat.
User-1 booked a seat.
VIP-2 failed to book a seat.

Process finished with exit code 0
```