Experiment 5

Student Name: Abhishek Raj **UID: 22BCS10008**
Branch: BE-C.S.E **Section/Group: 639-B**
Semester: 6th **Date of Performance: 05-03-2025**
Subject Name: Project Based Learning In Java with Lab **Subject Code: 22CSH-359**

1. Problem 5.1: 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.

Implementation/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);
            }
        }
    }

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

        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 :

```
PS D:\Code\JAVA CODE> cd "d:\Code\JAVA CODE\" ; if ($?) { javac EmployeeManagement.java } ;
Enter employee name: Shaurya Anand
Enter employee ID: 12510
Enter employee salary: 80000
Employee added: Employee ID: 12510, Name: Shaurya Anand, Salary: 80000.0
Do you want to add another employee? (yes/no): yes
Enter employee name: Jassi
Enter employee ID: 823520
Enter employee salary: 65000
Employee added: Employee ID: 823520, Name: Jassi , Salary: 65000.0
Do you want to add another employee? (yes/no): no
Employee List:
Employee ID: 12510, Name: Shaurya Anand, Salary: 80000.0
Employee ID: 823520, Name: Jassi , Salary: 65000.0
PS D:\Code\JAVA CODE>
```

Figure 5.1

2. Problem 5.2 :- 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

Implementation Code :-

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

// Class to represent a Card
class Card {
    private String suit;
    private String rank;
    public Card(String suit, String rank) {
        this.suit = suit;
        this.rank = rank;
    }
    public String getSuit() {
        return suit;
    }
    @Override
    public String toString() {
```

```
        return rank + " of " + suit;
    }
}
public class CardCollection {
    private Collection<Card> cards;
    public CardCollection() {
        cards = new ArrayList<>();
    }
    public void addCard(String suit, String rank) {
        Card card = new Card(suit, rank);
        cards.add(card);
        System.out.println("Card added: " + card);
    }
    public void findCardsBySuit(String suit) {
        System.out.println("Cards of suit " + suit + ":");
        boolean found = false;
        for (Card card : cards) {
            if (card.getSuit().equalsIgnoreCase(suit)) {
                System.out.println(card);
                found = true;
            }
        }
        if (!found) {
            System.out.println("No cards found for suit " + suit);
        }
    }
    public static void main(String[] args) {
        CardCollection cardCollection = new CardCollection();
        Scanner scanner = new Scanner(System.in);
        String choice;
        do {
            System.out.print("Enter card suit (e.g., Hearts, Diamonds, Clubs, Spades): ");
            String suit = scanner.nextLine();
            System.out.print("Enter card rank (e.g., 2, 3, 4, ..., 10, Jack, Queen, King, Ace): ");
            String rank = scanner.nextLine();
            cardCollection.addCard(suit, rank);
            System.out.print("Do you want to add another card? (yes/no): ");
            choice = scanner.nextLine();
        }
    while (choice.equalsIgnoreCase("yes"));
```

```
        System.out.print("Enter a suit to find all cards of that suit: ");
        String searchSuit = scanner.nextLine();
        cardCollection.findCardsBySuit(searchSuit);
        scanner.close();
    }
}
```

OUTPUT :-

```
PROBLEMS 24   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS D:\Code\JAVA CODE> cd "d:\Code\JAVA CODE\" ; if ($?) { javac CardCollection.java } ;
Enter card suit (e.g., Hearts, Diamonds, Clubs, Spades): clubs
Enter card rank (e.g., 2, 3, 4, ..., 10, Jack, Queen, King, Ace): 3
Card added: 3 of clubs
Do you want to add another card? (yes/no): yes
Enter card suit (e.g., Hearts, Diamonds, Clubs, Spades): Hearts
Enter card rank (e.g., 2, 3, 4, ..., 10, Jack, Queen, King, Ace): 9
Card added: 9 of Hearts
Do you want to add another card? (yes/no): no
Enter a suit to find all cards of that suit: Heart
Cards of suit Heart:
No cards found for suit Heart
PS D:\Code\JAVA CODE>
```

Figure 5.2

3. Problem 5.3 : 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.

Implementation Code :-

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

class TicketBookingSystem {
    private final boolean[] seats;
    private final Lock lock = new ReentrantLock();

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

```
 public void bookSeat(int seatNumber, String customerType) {
        lock.lock();
        try {
            if (seatNumber < 0 || seatNumber >= seats.length) {
                System.out.println("Invalid seat number: " + seatNumber);
                return;
            }
            if (!seats[seatNumber]) {
                seats[seatNumber] = true;
                System.out.println(customerType + " booked seat number: " + seatNumber);
            } else {
                System.out.println("Seat number " + seatNumber + " is already booked.");
            }
        } finally {
            lock.unlock();
        }
    }
}

class BookingThread extends Thread {
    private final TicketBookingSystem bookingSystem;
    private final int seatNumber;
    private final String customerType;

 public BookingThread(TicketBookingSystem bookingSystem, int seatNumber, String customerType) {
        this.bookingSystem = bookingSystem;
        this.seatNumber = seatNumber;
        this.customerType = customerType;
    }

    @Override
    public void run() {
        bookingSystem.bookSeat(seatNumber, customerType);
    }
}
public class TicketBookingApp {
    public static void main(String[] args) {
        TicketBookingSystem bookingSystem = new TicketBookingSystem(10);
```

```
        BookingThread vipBooking1 = new BookingThread(bookingSystem, 2, "VIP");
        BookingThread regularBooking1 = new BookingThread(bookingSystem, 2, "Regular");
        BookingThread vipBooking2 = new BookingThread(bookingSystem, 3, "VIP");
        BookingThread regularBooking2 = new BookingThread(bookingSystem, 3, "Regular");

        vipBooking1.setPriority(Thread.MAX_PRIORITY);
        regularBooking1.setPriority(Thread.NORM_PRIORITY);
        vipBooking2.setPriority(Thread.MAX_PRIORITY);
        regularBooking2.setPriority(Thread.NORM_PRIORITY);

        vipBooking1.start();
        regularBooking1.start();
        vipBooking2.start();
        regularBooking2.start();
    }
}
```

OUTPUT :-

```
PROBLEMS 24   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS D:\Code\JAVA CODE> cd "d:\Code\JAVA CODE\" ; if ($?) { javac TicketBookingApp.java } ; if ($?)
VIP booked seat number: 2
VIP booked seat number: 3
Seat number 2 is already booked.
Seat number 3 is already booked.
PS D:\Code\JAVA CODE>
```

Figure 5.3