Experiment 4

Student Name: Farhat **UID: 22BCS12854**
Branch: CSE **Section: IOT-642-B**
Semester: 6th **DOS: 20th Feb, 2024**
Subject: AP **Subject Code:22CSP-351**

Program 1: Employee Mnagement System

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.

Program/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;
    }

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

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

    public static void addEmployee(int id, String name, double salary) {
      for (Employee emp : employees) {
        if (emp.id == id) {
          System.out.println("Error: Employee with ID " + id + " already exists.");
          return;
```

```
            }
        }
        employees.add(new Employee(id, name, salary));
        System.out.println("Employee Added: ID=" + id + ", Name=" + name + ", Salary=" +
salary);
    }

    public static void updateEmployee(int id, double newSalary) {
        for (Employee emp : employees) {
            if (emp.id == id) {
                emp.salary = newSalary;
                System.out.println("Employee ID " + id + " updated successfully.");
                return;
            }
        }
        System.out.println("Error: Employee ID " + id + " not found.");
    }

    public static void removeEmployee(int id) {
        for (Employee emp : employees) {
            if (emp.id == id) {
                employees.remove(emp);
                System.out.println("Employee ID " + id + " removed successfully.");
                return;
            }
        }
        System.out.println("Error: Employee ID " + id + " not found.");
    }

    public static void searchEmployeeById(int id) {
        for (Employee emp : employees) {
            if (emp.id == id) {
                System.out.println("Employee Found: " + emp);
                return;
            }
        }
```

```
        System.out.println("Error: Employee ID " + id + " not found.");
    }

    public static void displayAllEmployees() {
        if (employees.isEmpty()) {
            System.out.println("No employees found.");
        } else {
            for (Employee emp : employees) {
                System.out.println("ID: " + emp.id + ", Name: " + emp.name + ", Salary: " +
emp.salary);
            }
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        while (true) {
            System.out.println("\n1. Add Employee\n2. Update Employee\n3. Remove
Employee\n4. Search Employee by ID\n5. Display All Employees\n6. Exit");
            System.out.print("Choose an option: ");
            int choice = scanner.nextInt();
            switch (choice) {
                case 1:
                    System.out.print("Enter ID: ");
                    int id = scanner.nextInt();
                    scanner.nextLine(); // Consume newline
                    System.out.print("Enter Name: ");
                    String name = scanner.nextLine();
                    System.out.print("Enter Salary: ");
                    double salary = scanner.nextDouble();
                    addEmployee(id, name, salary);
                    break;
                case 2:
                    System.out.print("Enter Employee ID: ");
                    int updateId = scanner.nextInt();
                    System.out.print("Enter New Salary: ");
                    double newSalary = scanner.nextDouble();
```

```
                updateEmployee(updateId, newSalary);
                break;
            case 3:
                System.out.print("Enter Employee ID: ");
                int removeId = scanner.nextInt();
                removeEmployee(removeId);
                break;
            case 4:
                System.out.print("Enter Employee ID: ");
                int searchId = scanner.nextInt();
                searchEmployeeById(searchId);
                break;
            case 5:
                displayAllEmployees();
                break;
            case 6:
                System.out.println("Exiting...");
                scanner.close();
                return;
            default:
                System.out.println("Invalid choice. Please try again.");
        }
    }
}
}
```

Output:

```
input

1. Add Employee
2. Update Employee
3. Remove Employee
4. Search Employee by ID
5. Display All Employees
6. Exit
Choose an option: 1
Enter ID: 12854
Enter Name: Farhat
Enter Salary: 1000000
Employee Added: ID=12854, Name=Farhat, Salary=1000000.0

1. Add Employee
2. Update Employee
3. Remove Employee
4. Search Employee by ID
5. Display All Employees
6. Exit
Choose an option: 1
Enter ID: 10019
Enter Name: Asya
Enter Salary: 10000
Employee Added: ID=10019, Name=Asya, Salary=10000.0
```

Program 2: Card Collection System

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.

Program:

```
import java.util.*;

class Card {
    private String suit;
    private String rank;

    public Card(String rank, String suit) {
        this.rank = rank;
        this.suit = suit;
    }

    public String getSuit() {
        return suit;
    }

    public String getRank() {
        return rank;
    }

    @Override
    public String toString() {
        return rank + " of " + suit;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null || getClass() != obj.getClass()) return false;
        Card card = (Card) obj;
        return Objects.equals(rank, card.rank) && Objects.equals(suit, card.suit);
```

```
    }

    @Override
    public int hashCode() {
        return Objects.hash(rank, suit);
    }
}

class CardCollection {
    private HashSet<Card> cardSet;
    private HashMap<String, List<Card>> suitMap;

    public CardCollection() {
        cardSet = new HashSet<>();
        suitMap = new HashMap<>();
    }

    public void addCard(String rank, String suit) {
        Card card = new Card(rank, suit);
        if (cardSet.contains(card)) {
            System.out.println("Error: Card \"" + card + "\" already exists.");
            return;
        }
        cardSet.add(card);
        suitMap.putIfAbsent(suit, new ArrayList<>());
        suitMap.get(suit).add(card);
        System.out.println("Card added: " + card);
    }

    public void findCardsBySuit(String suit) {
        if (!suitMap.containsKey(suit) || suitMap.get(suit).isEmpty()) {
            System.out.println("No cards found for " + suit + ".");
            return;
        }
        for (Card card : suitMap.get(suit)) {
            System.out.println(card);
```

```
        }
    }

    public void displayAllCards() {
        if (cardSet.isEmpty()) {
            System.out.println("No cards found.");
            return;
        }
        for (Card card : cardSet) {
            System.out.println(card);
        }
    }

    public void removeCard(String rank, String suit) {
        Card card = new Card(rank, suit);
        if (!cardSet.contains(card)) {
            System.out.println("Error: Card \"" + card + "\" does not exist.");
            return;
        }
        cardSet.remove(card);
        suitMap.get(suit).remove(card);
        if (suitMap.get(suit).isEmpty()) {
            suitMap.remove(suit);
        }
        System.out.println("Card removed: " + card);
    }
}

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

        while (true) {
            System.out.println("\n1. Add Card");
            System.out.println("2. Find Cards by Suit");
```

```
System.out.println("3. Display All Cards");
System.out.println("4. Remove Card");
System.out.println("5. Exit");
System.out.print("Choose an option: ");

int choice = scanner.nextInt();
scanner.nextLine();

switch (choice) {
    case 1:
        System.out.print("Enter rank: ");
        String rank = scanner.nextLine();
        System.out.print("Enter suit: ");
        String suit = scanner.nextLine();
        collection.addCard(rank, suit);
        break;
    case 2:
        System.out.print("Enter suit to find: ");
        suit = scanner.nextLine();
        collection.findCardsBySuit(suit);
        break;
    case 3:
        collection.displayAllCards();
        break;
    case 4:
        System.out.print("Enter rank to remove: ");
        rank = scanner.nextLine();
        System.out.print("Enter suit to remove: ");
        suit = scanner.nextLine();
        collection.removeCard(rank, suit);
        break;
    case 5:
        System.out.println("Exiting...");
        scanner.close();
        return;
    default:
```

```
            System.out.println("Invalid option. Please try again.");
        }
    }
  }
}
```

Output:

```
1. Add Card
2. Find Cards by Suit
3. Display All Cards
4. Remove Card
5. Exit
Choose an option: 1
Enter rank: Ace of Spade
Enter suit: 2
Card added: Ace of Spade of 2

1. Add Card
2. Find Cards by Suit
3. Display All Cards
4. Remove Card
5. Exit
Choose an option: 1
Enter rank: King of Hearts
Enter suit: 10
Card added: King of Hearts of 10
```

Program 3: Ticket Booking System

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.

Program/Code:

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

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

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

    public synchronized boolean bookSeat(int seatNumber, String userName, boolean isVIP) {
        if (seatNumber < 1 || seatNumber > seats.length) {
            System.out.println(userName + ": Invalid seat number!");
            return false;
        }

        int index = seatNumber - 1;
        if (seats[index]) {
            System.out.println(userName + ": Seat " + seatNumber + " is already booked!");
            return false;
        }

        seats[index] = true;
        System.out.println(userName + " (" + (isVIP ? "VIP" : "Regular") + ") booked seat " +
seatNumber);
        return true;
    }

    public void displaySeats() {
        System.out.println("Current Seat Status:");
```

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

public class TicketBookingDemo {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of seats: ");
        int totalSeats = scanner.nextInt();
        TicketBookingSystem system = new TicketBookingSystem(totalSeats);

        while (true) {
            System.out.println("\n1. Book Seat\n2. Display Seats\n3. Exit");
            System.out.print("Choose an option: ");
            int choice = scanner.nextInt();

            if (choice == 1) {
                System.out.print("Enter your name: ");
                String name = scanner.next();
                System.out.print("Enter seat number: ");
                int seatNumber = scanner.nextInt();
                System.out.print("Are you a VIP? (true/false): ");
                boolean isVIP = scanner.nextBoolean();
                system.bookSeat(seatNumber, name, isVIP);
            } else if (choice == 2) {
                system.displaySeats();
            } else if (choice == 3) {
                System.out.println("Exiting...");
                break;
            } else {
                System.out.println("Invalid choice, try again.");
            }
        }
```

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

Output:

```
1. Book Seat
2. Display Seats
3. Exit
Choose an option: 1
Enter your name: Farhat
Enter seat number: 1
Are you a VIP? (true/false): true
Farhat (VIP) booked seat 1

1. Book Seat
2. Display Seats
3. Exit
Choose an option: 2
Current Seat Status:
Seat 1: Booked
Seat 2: Available
Seat 3: Available
Seat 4: Available
```