Name : Tushar Srivastav **UID : 22BCS17290**

Subject : Java With Lab **Section : 903-B[DL]**

1). Easy Problem

Code :

```
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 ArrayList<Employee> employees = new ArrayList<>();
    private Scanner scanner = new Scanner(System.in);
```

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

public void updateEmployee() {
    System.out.print("Enter Employee ID to update: ");
    int id = scanner.nextInt();
    for (Employee emp : employees) {
        if (emp.id == id) {
            scanner.nextLine();
            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!");
}
```

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

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

public void displayEmployees() {
    if (employees.isEmpty()) {
        System.out.println("No employees found.");
    } else {
        for (Employee emp : employees) {
            System.out.println(emp);
        }
    }
```

```
}

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

    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 Employees");
        System.out.println("6. Exit");
        System.out.print("Enter your choice: ");
        int choice = scanner.nextInt();

        switch (choice) {
            case 1:
                em.addEmployee();
                break;
            case 2:
                em.updateEmployee();
                break;
            case 3:
                em.removeEmployee();
                break;
            case 4:
```

```
                em.searchEmployee();
                break;
            case 5:
                em.displayEmployees();
                break;
            case 6:
                System.out.println("Exiting...");
                scanner.close();
                return;
            default:
                System.out.println("Invalid choice! Try again.");
        }
    }
  }
}
```

2).Medium Problem

Code :

```
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;
    }

    public String getValue() {
        return value;
    }

    @Override
    public String toString() {
        return "Card{Symbol='" + symbol + "', Value='" + value + "'}";
    }
}

class CardCollection {
    private Collection<Card> cards;

    public CardCollection() {
        cards = new ArrayList<>();
    }

    public void addCard(String symbol, String value) {
        cards.add(new Card(symbol, value));
        System.out.println("Card added successfully!");
    }

    public void findCardsBySymbol(String symbol) {
        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 symbol: " + symbol);
        }
    }

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

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

        while (true) {
```

```
System.out.println("\nCard Management 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("Enter your choice: ");
int choice = scanner.nextInt();
scanner.nextLine();

switch (choice) {
    case 1:
        System.out.print("Enter Card Symbol: ");
        String symbol = scanner.nextLine();
        System.out.print("Enter Card Value: ");
        String value = scanner.nextLine();
        collection.addCard(symbol, value);
        break;
    case 2:
        System.out.print("Enter Symbol to Search: ");
        String searchSymbol = scanner.nextLine();
        collection.findCardsBySymbol(searchSymbol);
        break;
    case 3:
        collection.displayAllCards();
        break;
    case 4:
        System.out.println("Exiting... Goodbye!");
        scanner.close();
```

```
                return;
            default:
                System.out.println("Invalid choice. Please try again.");
        }
    }
  }
}
```

```
Output                                                    Clear

Card Management System
1. Add Card
2. Find Cards by Symbol
3. Display All Cards
4. Exit
Enter your choice: 1
Enter Card Symbol: 18
Enter Card Value: 17
Card added successfully!

Card Management System
1. Add Card
2. Find Cards by Symbol
3. Display All Cards
4. Exit
Enter your choice: 3
Card{Symbol='18', Value='17'}

Card Management System
1. Add Card
2. Find Cards by Symbol
3. Display All Cards
4. Exit
Enter your choice: 4
Exiting... Goodbye!
```

3).Hard Problem

Code:

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

class TicketBookingSystem {
    private static final int TOTAL_SEATS = 10;
    private boolean[] seats = new boolean[TOTAL_SEATS];
    private final Lock lock = new ReentrantLock();

    public void bookSeat(int seatNumber, String customerName) {
        lock.lock();
        try {
            if (seatNumber < 0 || seatNumber >= TOTAL_SEATS) {
                System.out.println(customerName + " tried to book an invalid seat.");
                return;
            }
            if (!seats[seatNumber]) {
                seats[seatNumber] = true;
                System.out.println(customerName + " successfully booked seat " + seatNumber);
            } else {
                System.out.println("Seat " + seatNumber + " is already booked. " + customerName
+ " could not book.");
            }
        } finally {
            lock.unlock();
        }
    }
}

class Customer extends Thread {
```

```
    private TicketBookingSystem system;
    private int seatNumber;
    private 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 Main {
    public static void main(String[] args) {
        TicketBookingSystem system = new TicketBookingSystem();

        Customer vip1 = new Customer(system, 3, "VIP Customer 1", Thread.MAX_PRIORITY);
        Customer vip2 = new Customer(system, 5, "VIP Customer 2", Thread.MAX_PRIORITY);
        Customer regular1 = new Customer(system, 3, "Regular Customer 1",
Thread.MIN_PRIORITY);
        Customer regular2 = new Customer(system, 5, "Regular Customer 2",
Thread.MIN_PRIORITY);
```

```
        Customer regular3 = new Customer(system, 7, "Regular Customer 3",
Thread.NORM_PRIORITY);


        vip1.start();
        vip2.start();
        regular1.start();
        regular2.start();
        regular3.start();
    }
}
```