CO – 3 COLLECTION ASSESSMENT

Scenario 6 :

Hospital emergency room management system

You are developing a hospital emergency room management system where patients are treated based on the severity of their condition.

Each patient has a Patient ID, Name, Condition Severity (1: Critical, 2: Serious, 3: Stable), and Arrival Time.

Your task is to use a **Priority Queue** to manage the flow of patients such that:

(1) New patients are added to the queue, prioritizing those in critical condition

(2) Patients are treated in the order of their condition severity, ensuring that those with the same severity are treated based on their arrival time

(3) Once treated, a patient is removed from the queue

(4) The system should allow checking the next patient in line for treatment without removing them from the queue. Additionally, implement functionality to **filter** patients based on severity and **map** them to a list of their names and conditions, allowing staff to quickly view all patients in a specific severity category.

Create methods to add patients, process and remove patients, peek at the next patient, and filter and map patients based on severity for efficient emergency room management

Program :

```
import java.util.*;

class Patient implements Comparable<Patient> {
    private int patientId;
    private String name;
    private int severity;
    private int arrivalTime;
    public Patient(int patientId, String name, int severity, int arrivalTime) {
        this.patientId = patientId;
        this.name = name;
        this.severity = severity;
        this.arrivalTime = arrivalTime;
    }
    public int getPatientId() {
```

```
        return patientId;
    }
    public String getName() {
        return name;
    }
    public int getSeverity() {
        return severity;
    }
    public int getArrivalTime() {
        return arrivalTime;
    }
    @Override
    public int compareTo(Patient other) {
        if (this.severity != other.severity) {
            return Integer.compare(this.severity, other.severity);
        } else {
            return Integer.compare(this.arrivalTime, other.arrivalTime);
        }
    }
    @Override
    public String toString() {
        return "Patient[ID=" + patientId + ", Name=" + name + ", Severity=" + severity + ",
ArrivalTime=" + arrivalTime + "]";
    }
}
class EmergencyRoom {
    private PriorityQueue<Patient> patientQueue;
    private Map<Integer, List<Patient>> severityMap;
    private int arrivalCounter;
    public EmergencyRoom() {
        this.patientQueue = new PriorityQueue<>();
```

```
    this.severityMap = new HashMap<>();
    this.arrivalCounter = 0;
}
public void addPatient(int patientId, String name, int severity) {
    arrivalCounter++;
    Patient newPatient = new Patient(patientId, name, severity, arrivalCounter);
    patientQueue.offer(newPatient);
    severityMap.computeIfAbsent(severity, k -> new ArrayList<>()).add(newPatient);
    System.out.println("Added Patient: " + newPatient);
}
public Patient processPatient() {
    if (!patientQueue.isEmpty()) {
        Patient nextPatient = patientQueue.poll();
        severityMap.get(nextPatient.getSeverity()).remove(nextPatient);
        System.out.println("Processed Patient: " + nextPatient);
        return nextPatient;
    } else {
        System.out.println("No patients to process.");
        return null;
    }
}
public Patient nextPatient() {
    if (!patientQueue.isEmpty()) {
        Patient nextPatient = patientQueue.peek();
        System.out.println("Next Patient: " + nextPatient);
        return nextPatient;
    } else {
        System.out.println("No patients in the queue.");
        return null;
    }
```

```
    }
    public List<Patient> filterPatientsBySeverity(int severity) {
        if (severityMap.containsKey(severity)) {
            return severityMap.get(severity);
        } else {
            System.out.println("No patients with severity: " + severity);
            return Collections.emptyList();
        }
    }
}
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        EmergencyRoom er = new EmergencyRoom();
        boolean running = true;
        while (running) {
            System.out.println("\nHospital Emergency Room Management System");
            System.out.println("1. Add Patient");
            System.out.println("2. Process Patient");
            System.out.println("3. Peek Next Patient");
            System.out.println("4. Filter Patients by Severity");
            System.out.println("5. Exit");
            System.out.print("Choose an option: ");
            int option = scanner.nextInt();
            switch (option) {
                case 1:
                    System.out.print("Enter Patient ID: ");
                    int id = scanner.nextInt();
                    scanner.nextLine();  // Consume newline
                    System.out.print("Enter Patient Name: ");
```

```
                String name = scanner.nextLine();
                System.out.print("Enter Severity (1-Critical, 2-Serious, 3-Stable): ");
                int severity = scanner.nextInt();
                er.addPatient(id, name, severity);
                break;
            case 2:
                er.processPatient();
                break;
            case 3:
                er.nextPatient();
                break;
            case 4:
                System.out.print("Enter severity level to filter by (1-Critical, 2-Serious, 3-
Stable): ");
                int filterSeverity = scanner.nextInt();
                List<Patient> filteredPatients = er.filterPatientsBySeverity(filterSeverity);
                if (!filteredPatients.isEmpty()) {
                    System.out.println("Patients with Severity " + filterSeverity + ": " +
filteredPatients);
                }
                break;
            case 5:
                System.out.println("Exiting...");
                running = false;
                break;
            default:
                System.out.println("Invalid option. Please try again.");
        }
    }
    scanner.close();
  }
```

}

OUTPUT:

```
D:\java_program>path="D:\jdk-21\bin"

D:\java_program>javac main.java

D:\java_program>java main

Hospital Emergency Room Management System
1. Add Patient
2. Process Patient
3. Peek Next Patient
4. Filter Patients by Severity
5. Exit
Choose an option: 1
Enter Patient ID: 45
Enter Patient Name: swetha
Enter Severity (1-Critical, 2-Serious, 3-Stable): 1
Added Patient: Patient[ID=45, Name=swetha, Severity=1, ArrivalTime=1]

Hospital Emergency Room Management System
1. Add Patient
2. Process Patient
3. Peek Next Patient
4. Filter Patients by Severity
5. Exit
Choose an option: 1
Enter Patient ID: 46
Enter Patient Name: Jaiz
Enter Severity (1-Critical, 2-Serious, 3-Stable): 2
Added Patient: Patient[ID=46, Name=Jaiz, Severity=2, ArrivalTime=2]
```

```
Hospital Emergency Room Management System
1. Add Patient
2. Process Patient
3. Peek Next Patient
4. Filter Patients by Severity
5. Exit
Choose an option: 1
Enter Patient ID: 47
Enter Patient Name: Harshini
Enter Severity (1-Critical, 2-Serious, 3-Stable): 1
Added Patient: Patient[ID=47, Name=Harshini, Severity=1, ArrivalTime=3]

Hospital Emergency Room Management System
1. Add Patient
2. Process Patient
3. Peek Next Patient
4. Filter Patients by Severity
5. Exit
Choose an option: 1
Enter Patient ID: 48
Enter Patient Name: princy
Enter Severity (1-Critical, 2-Serious, 3-Stable): 3
Added Patient: Patient[ID=48, Name=princy, Severity=3, ArrivalTime=4]
```

```
Hospital Emergency Room Management System
1. Add Patient
2. Process Patient
3. Peek Next Patient
4. Filter Patients by Severity
5. Exit
Choose an option: 1
Enter Patient ID: 49
Enter Patient Name: aksh
Enter Severity (1-Critical, 2-Serious, 3-Stable): 2
Added Patient: Patient[ID=49, Name=aksh, Severity=2, ArrivalTime=5]

Hospital Emergency Room Management System
1. Add Patient
2. Process Patient
3. Peek Next Patient
4. Filter Patients by Severity
5. Exit
Choose an option: 1
Enter Patient ID: 50
Enter Patient Name: sakthi
Enter Severity (1-Critical, 2-Serious, 3-Stable): 2
Added Patient: Patient[ID=50, Name=sakthi, Severity=2, ArrivalTime=6]
```

```
Hospital Emergency Room Management System
1. Add Patient
2. Process Patient
3. Peek Next Patient
4. Filter Patients by Severity
5. Exit
Choose an option: 2
Processed Patient: Patient[ID=45, Name=swetha, Severity=1, ArrivalTime=1]

Hospital Emergency Room Management System
1. Add Patient
2. Process Patient
3. Peek Next Patient
4. Filter Patients by Severity
5. Exit
Choose an option: 3
Next Patient: Patient[ID=47, Name=Harshini, Severity=1, ArrivalTime=3]

Hospital Emergency Room Management System
1. Add Patient
2. Process Patient
3. Peek Next Patient
4. Filter Patients by Severity
5. Exit
Choose an option: 4
Enter severity level to filter by (1-Critical, 2-Serious, 3-Stable): 1
Patients with Severity 1: [Patient[ID=47, Name=Harshini, Severity=1, ArrivalTime=3]]
```

```
Hospital Emergency Room Management System
1. Add Patient
2. Process Patient
3. Peek Next Patient
4. Filter Patients by Severity
5. Exit
Choose an option: 4
Enter severity level to filter by (1-Critical, 2-Serious, 3-Stable): 2
Patients with Severity 2: [Patient[ID=46, Name=Jaiz, Severity=2, ArrivalTime=2], Patient[ID=49, Name=aksh, S
everity=2, ArrivalTime=5], Patient[ID=50, Name=sakthi, Severity=2, ArrivalTime=6]]

Hospital Emergency Room Management System
1. Add Patient
2. Process Patient
3. Peek Next Patient
4. Filter Patients by Severity
5. Exit
Choose an option: 4
Enter severity level to filter by (1-Critical, 2-Serious, 3-Stable): 3
Patients with Severity 3: [Patient[ID=48, Name=princy, Severity=3, ArrivalTime=4]]

Hospital Emergency Room Management System
1. Add Patient
2. Process Patient
3. Peek Next Patient
4. Filter Patients by Severity
5. Exit
Choose an option: 5
Exiting...
```