Emergency Room Patient Management Using PriorityQueue

Objective:

Learn how to use PriorityQueue in Java to manage patients in a hospital based on severity (priority). This example demonstrates:

• The use of PriorityQueue

• Creating a custom comparator

• Processing elements based on priority

• Taking input from the user

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Problem Statement:

In a hospital emergency room:

• Multiple patients arrive with different levels of severity (higher severity = more critical).

• The hospital wants to treat most critical patients first.

• If two patients have the same severity, they can be treated in any order.

Requirements:

1. Maintain a list of patients using PriorityQueue.

2. Treat patients based on severity (highest first).

3. Take patient details as input from the user.

4. Show the order of patients being treated.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Key Concepts:

1. PriorityQueue

• Part of Java Collections Framework.

• Elements are processed based on priority, not insertion order.

• Default priority: natural order (smallest first for numbers, alphabetical for strings).

• Custom priority can be set using Comparator.

2. Comparator

• Used to define custom order for objects in a PriorityQueue.

• Compare two objects and decide which should come first.

3. Patient Class

• Represents a patient with attributes:

o name → String

o severity → int (higher = more critical)

Emergency Room Patient Management Using PriorityQueue

Objective:

Learn how to use PriorityQueue in Java to manage patients in a hospital based on severity (priority). This example demonstrates:

• The use of PriorityQueue

• Creating a custom comparator

• Processing elements based on priority

• Taking input from the user

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Problem Statement:

In a hospital emergency room:

• Multiple patients arrive with different levels of severity (higher severity = more critical).

• The hospital wants to treat most critical patients first.

• If two patients have the same severity, they can be treated in any order.

Requirements:

1. Maintain a list of patients using PriorityQueue.

2. Treat patients based on severity (highest first).

3. Take patient details as input from the user.

4. Show the order of patients being treated.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Key Concepts:

1. PriorityQueue

• Part of Java Collections Framework.

• Elements are processed based on priority, not insertion order.

• Default priority: natural order (smallest first for numbers, alphabetical for strings).

• Custom priority can be set using Comparator.

2. Comparator

• Used to define custom order for objects in a PriorityQueue.

• Compare two objects and decide which should come first.

3. Patient Class

• Represents a patient with attributes:

o name → String

o severity → int (higher = more critical)