**Intern Exercise: Airport Check-in System Using PriorityQueue**

**Problem Statement:**

You are tasked with simulating an **airport check-in system**:

* Passengers arrive at the airport to check in for their flight.
* Some passengers have **priority status** (VIP, Business, Economy).
* The airport wants to serve passengers based on **priority class**:
  + **VIP → Business → Economy**
* If two passengers have the same class, they can be served in **any order**.
* Input should be taken from the **user**.

**Requirements:**

1. **Passenger Class:**
   * Attributes:
     + name → String
     + priorityLevel → int (3 = VIP, 2 = Business, 1 = Economy)
   * Create a toString() method to display passenger details.
2. **PriorityQueue:**
   * Use a **PriorityQueue** to store passengers.
   * **Highest priority class (VIP)** should be served first.
   * Use a **Comparator** to define the order.
3. **Input from User:**
   * Ask how many passengers to add.
   * For each passenger, take **name** and **priority class** (VIP/Business/Economy) as input.
4. **Process Passengers:**
   * Poll passengers from the queue one by one.
   * Display the **check-in order**.

**Example Input/Output:**

**Input:**

Number of passengers: 4

Passenger 1: Ravi Economy

Passenger 2: Meena VIP

Passenger 3: Amit Business

Passenger 4: Sara VIP

**Output:**

Next passenger: Meena (VIP)

Next passenger: Sara (VIP)

Next passenger: Amit (Business)

Next passenger: Ravi (Economy)

**Hints for Interns:**

* Use a **Comparator** to prioritize VIP > Business > Economy.
* Remember: **PriorityQueue does not maintain insertion order**.
* Only the **head element** is guaranteed to have the highest priority.

**Full Solution Code:**

import java.util.\*;

// Step 1: Passenger class

class Passenger {

String name;

int priorityLevel; // 3 = VIP, 2 = Business, 1 = Economy

Passenger(String name, int priorityLevel) {

this.name = name;

this.priorityLevel = priorityLevel;

}

public String toString() {

String type = switch (priorityLevel) {

case 3 -> "VIP";

case 2 -> "Business";

default -> "Economy";

};

return name + " (" + type + ")";

}

}

// Step 2: Comparator class to define priority

class PriorityComparator implements Comparator<Passenger> {

@Override

public int compare(Passenger p1, Passenger p2) {

return p2.priorityLevel - p1.priorityLevel; // higher priorityLevel first

}

}

// Step 3: Main class

public class AirportCheckIn {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

PriorityQueue<Passenger> pq = new PriorityQueue<>(new PriorityComparator());

System.out.print("How many passengers to add? ");

int n = sc.nextInt();

sc.nextLine(); // consume newline

for (int i = 0; i < n; i++) {

System.out.print("Enter passenger name: ");

String name = sc.nextLine();

System.out.print("Enter passenger class (VIP/Business/Economy): ");

String cls = sc.nextLine();

int priority = switch (cls.toLowerCase()) {

case "vip" -> 3;

case "business" -> 2;

default -> 1;

};

pq.add(new Passenger(name, priority));

}

System.out.println("\nServing passengers based on priority:");

while (!pq.isEmpty()) {

System.out.println("Next passenger: " + pq.poll());

}

sc.close();

}

}

**✅ Key Learning Points:**

* How to use **PriorityQueue** with **custom objects**.
* How **Comparator** determines the serving order.
* How **real-world priority-based processing** works.
* Input from user and processing in **priority order**.