08-Java_Assignment

1️⃣ HashSet vs LinkedHashSet vs TreeSet – Unique Usernames

🎯 **Scenario:** You are developing a **login system** where usernames must be unique. However, you need to store them in different ways based on usage:

🔷 **Use** `HashSet` – Store usernames without any specific order.

🔷 **Use** `LinkedHashSet` – Maintain insertion order of usernames.

🔷 **Use** `TreeSet` – Store usernames in **sorted order**.

📝 **Task:**

- Add usernames to all three sets.
- Print the sets and observe how they store elements differently.

🔍 **Example Output:**

```
HashSet: [zoe, charlie, alex, bob, mike]
LinkedHashSet: [zoe, alex, mike, bob, charlie]
TreeSet: [alex, bob, charlie, mike, zoe]
```

2️⃣ HashMap vs LinkedHashMap vs TreeMap – Student Marks System

🎯 **Scenario:** You need to **store student names and their scores** and retrieve them in different ways:

🔷 **Use** `HashMap` – Store data without any specific order.

🔷 **Use** `LinkedHashMap` – Maintain the insertion order of student names.

🔷 **Use** `TreeMap` – Automatically sort students alphabetically.

📝 **Task:**

- Add 5 student names and scores in all three maps.
- Print the maps and observe the ordering differences.

🔍 **Example Output:**

```
HashMap: {Zara=85, Arun=92, Mia=78, Liam=90, Ben=88}
LinkedHashMap: {Zara=85, Arun=92, Mia=78, Liam=90, Ben=88}
TreeMap: {Arun=92, Ben=88, Liam=90, Mia=78, Zara=85}
```

3️⃣ Queue – Customer Support Ticket System

🎯 **Scenario:** You are developing a **customer support system** where users submit queries, and they are **processed in FIFO order** (First In, First Out).

🔷 **Use** `Queue` **(LinkedList implementation)** – Customers are served in the order they raise tickets.

📝 **Task:**

- Add 5 customers to the queue.
- Process each customer one by one (remove from the queue).

🔍 **Example Output:**

```
Support Queue: [User1, User2, User3, User4, User5]
Serving: User1
Serving: User2
Serving: User3
Serving: User4
Serving: User5
```

4️⃣ PriorityQueue – Task Management System

🎯 **Scenario:** You are developing a **task scheduling system** where tasks have **different priorities** (higher priority tasks should be processed first).

🔷 **Use** `PriorityQueue` – Store tasks with their priority levels. The highest-priority task should be processed first.

📝 **Task:**

- Create a class `Task` with a **name** and a **priority**.
- Use `PriorityQueue<Task>` to store tasks in **ascending order of priority**.
- Process and remove tasks based on priority.

🔍 **Example Output:**

```
Processing: Pay Bills (Priority: 1)
Processing: Complete Assignment (Priority: 2)
Processing: Prepare for Exam (Priority: 3)
Processing: Watch Movie (Priority: 5)
```

🚀 Submission Guidelines:

✔️ Implement each program separately and test it.

✔️ Observe how different collections store and retrieve elements.

✔️ Comment on your code where necessary.