10-Java_Assignment

🧵 Part 1: Multithreading Assignments

1️⃣ Task: Simulating a YouTube Video Buffering System

🎯 **Scenario:** You are building a **YouTube-like** platform where videos buffer in **chunks**. Each chunk loads with a delay, simulating network speed.

📌 **Goal:** Create **3 threads**, each responsible for buffering **one-third** of a video. Use `sleep()` to simulate the delay.

🔷 **Expected Output:**

```
Buffering Part 1...
(2 seconds delay)
Buffering Part 2...
(2 seconds delay)
Buffering Part 3...
Video Fully Buffered!
```

2️⃣ Task: Managing a Ride-Sharing Queue (Thread Priority)

🎯 **Scenario:** A **ride-sharing app** prioritizes rides based on user category:

- **VIP Rider** 🏆 (Highest priority)
- **Regular Rider** 🚗 (Medium priority)
- **New User** 🆕 (Lowest priority)

📌 **Goal:** Create 3 threads, set priority levels accordingly, and observe the execution order.

🔷 **Expected Output:**

```
VIP Rider is getting a ride!
Regular Rider is getting a ride!
```

```
New User is getting a ride!
```

3️⃣ Task: Countdown Timer for an Online Exam (Join Method)

🎯 **Scenario:** A student takes an **online exam** with a **countdown timer** running in the background. The exam submission waits until the countdown finishes.

📌 **Goal:** Implement **two threads**:

- **Thread 1:** Countdown from **10 seconds** (use `sleep()` & `join()`)
- **Thread 2:** Submits the test **only after** the countdown ends.

🔷 **Expected Output:**

```
Exam starting in: 10, 9, 8, 7... (countdown continues)
Time's up! Submitting the test...
Test submitted successfully!
```

📂 Part 2: File Handling Assignments

4️⃣ Task: Writing and Reading a To-Do List

🎯 **Scenario:** Create a **to-do list** application where users can:

1. **Add** tasks to a file (`write`)
2. **Read** all tasks (`read`)

📌 **Goal:** Use **File Handling** to store and retrieve tasks from a file.

🔷 **Expected Output:**

```
User adds:
- "Complete Java Assignment"
- "Submit Resume"

Stored tasks in file:
1. Complete Java Assignment
2. Submit Resume
```

5️⃣ Task: Updating a User Profile System

🎯 **Scenario:** A **User Profile System** stores user data in a file. Users should be able to:

1. **Create** a profile (`write`)
2. **View** the profile (`read`)
3. **Update** the profile (`update` a specific line)

📌 **Goal:** Implement file handling to modify user data.

🔷 **Expected Output:**

```
Initial Profile:
Name: John Doe
Age: 22
City: New York

User updates age → 23

Updated Profile:
Name: John Doe
Age: 23
City: New York
```

🚀 Submission Guidelines:

✔️ Implement **multithreading** and **file handling** in real-world scenarios.

✔️ Compare the execution order of **priority threads**.

✔️ Use **file reading/writing** efficiently.

Source: https://github.com/gurukannan22/Java-Learning