Recommended YouTube Topics and Concepts

1. **Java Socket Programming (Client-Server Communication)**
    - **Why: The project requires a server (Server.java) listening on port 2500 and clients (Client1.java, Client2.java) communicating with it using sockets. You need to understand how to set up a server socket, handle multiple client connections, and exchange messages in a specific format (e.g., REQUEST_TASK;ServiceNumber;ClientName;Priority).**
    - **Search Terms:**
        - **"Java socket programming tutorial"**
        - **"Java client-server socket example"**
        - **"Java multithreaded server socket programming"**
    - **Key Concepts to Learn:**
        - **Creating a ServerSocket and accepting client connections.**
        - **Using Socket for client-server communication.**
        - **Sending and receiving messages with BufferedReader and PrintWriter.**
        - **Handling multiple clients using threads (e.g., a new thread per client connection).**
        - **Parsing and formatting messages (e.g., splitting strings by ; for the protocol).**
    - **Example Videos (Search for Updated Content from 2020-2025):**
        - **"Java Socket Programming - Client Server Example" by a channel like Programming with Mosh or Telusko.**
        - **"Multithreaded Client-Server in Java" by Amigoscode or Cave of Programming.**
    - **Relevance to Project: This will help you implement the server's listening loop, client connection handling, and the communication protocol for commands like QUEUE_STATUS and TASK_HISTORY.**
2. **Java PriorityQueue for Task Scheduling**

- **Why: The server uses a PriorityQueue to store tasks sorted by priority (1=High, 2=Medium, 3=Low) and timestamp. You need to understand how to customize a PriorityQueue to handle complex objects (e.g., a Task class with priority and timestamp).**
- **Search Terms:**
  - **"Java PriorityQueue tutorial"**
  - **"Custom comparator in Java PriorityQueue"**
  - **"Java PriorityQueue example with objects"**
- **Key Concepts to Learn:**
  - **Creating a PriorityQueue with a custom Comparator or Comparable interface.**
  - **Defining a Task class with fields like TaskID, ServiceNumber, ClientName, Priority, and Timestamp.**
  - **Sorting tasks by priority first, then by timestamp for same-priority tasks.**
  - **Adding, removing, and peeking tasks in the queue.**
- **Example Videos:**
  - **"Java Priority Queue Tutorial" by Tech With Tim or Coding with John.**
  - **"Custom Comparator in Java" by Telusko or Durgasoft.**
- **Relevance to Project: This is critical for the server's task queue, ensuring tasks are executed in the correct order (e.g., high-priority system_monitor.sh before low-priority file_audit.sh).**

3. **Java Thread Synchronization and Locks**
   - **Why: The server must ensure only one instance of a script (e.g., user_setup.sh) runs at a time, requiring synchronization. Since you've covered synchronization, reinforce your understanding of thread safety for shared resources like the task queue and script execution.**
   - **Search Terms:**
     - **"Java thread synchronization tutorial"**

        - **“Java ReentrantLock example”**
        - **“Java synchronized block vs synchronized method”**
    - **Key Concepts to Learn:**
        - **Using synchronized blocks/methods to protect shared resources.**
        - **Implementing ReentrantLock for fine-grained control (e.g., locking specific scripts).**
        - **Avoiding race conditions when accessing the PriorityQueue or task history.**
        - **Managing thread pools with ExecutorService for task execution (optional but useful for scalability).**
    - **Example Videos:**
        - **“Java Multithreading and Synchronization” by freeCodeCamp or Cave of Programming.**
        - **“ReentrantLock in Java” by Telusko or Amigoscode.**
    - **Relevance to Project: Ensures thread-safe task queuing and script execution, preventing multiple instances of the same script from running concurrently.**
4. **Executing Shell Scripts in Java**
    - **Why: The server executes Linux shell scripts (e.g., dir_perms.sh, MySQL_login_<USER_NAME>.sh) using ProcessBuilder, and clients run local scripts (e.g., ssh_config.sh). You need to learn how to invoke scripts, capture output, and handle errors.**
    - **Search Terms:**
        - **“Java ProcessBuilder tutorial”**
        - **“Run shell script from Java”**
        - **“Capture output of shell script in Java”**
    - **Key Concepts to Learn:**
        - **Using ProcessBuilder to execute shell commands or scripts.**

        - **Passing arguments to scripts (e.g., for MySQL_login_<USER_NAME>.sh).**
        - **Capturing script output and error streams with InputStream.**
        - **Checking exit codes to detect script success or failure (for ERROR status).**
        - **Handling file paths for scripts stored in shared directories.**
    - **Example Videos:**
        - **"Java ProcessBuilder Example" by Coding with John or Telusko.**
        - **"Execute Shell Commands in Java" by ProgrammingKnowledge or Durgasoft.**
    - **Relevance to Project: Essential for executing server-side scripts like system_monitor.sh and client-side scripts like resource_report.sh, with proper logging of results.**
5. **Linux Shell Scripting Basics**
    - **Why: The project requires writing error-free shell scripts (e.g., Network.sh, user_setup.sh) with inline comments. If you're new to shell scripting, learn the basics to create scripts for tasks like user creation, permission setting, and connectivity checks.**
    - **Search Terms:**
        - **"Linux shell scripting tutorial for beginners"**
        - **"Bash script error handling"**
        - **"Write shell script with comments"**
    - **Key Concepts to Learn:**
        - **Writing Bash scripts with #!/bin/bash shebang.**
        - **Using commands like useradd, chmod, chown (for user_setup.sh, dir_perms.sh).**
        - **Implementing connectivity checks with ping or nc (for Network.sh).**
        - **Adding error handling (e.g., set -e, if statements, exit codes).**

        - **Writing inline comments to explain logic (required for submission).**
    - **Example Videos:**
        - **“Bash Scripting Tutorial for Beginners” by freeCodeCamp or TechWorld with Nana.**
        - **“Shell Scripting Crash Course” by Traversy Media or LinuxHint.**
    - **Relevance to Project: Enables you to write scripts like file_audit.sh and quota_check.sh, ensuring they are robust and well-documented.**
6. **Java Timers and Scheduling**
    - **Why: Clients must send task requests every 5 minutes, and the server enforces rate limiting (one task per client every 5 minutes). Learn how to schedule tasks and track time intervals in Java.**
    - **Search Terms:**
        - **“Java ScheduledExecutorService tutorial”**
        - **“Java Timer and TimerTask example”**
        - **“Java time-based task scheduling”**
    - **Key Concepts to Learn:**
        - **Using ScheduledExecutorService to schedule tasks at fixed intervals (e.g., every 5 minutes for client requests).**
        - **Tracking client task submission timestamps in a HashMap for rate limiting.**
        - **Calculating time differences using System.currentTimeMillis() or LocalDateTime.**
    - **Example Videos:**
        - **“Java ScheduledExecutorService Tutorial” by Cave of Programming or Telusko.**
        - **“Java Timer and Scheduling” by ProgrammingKnowledge or Amigoscode.**

- **Relevance to Project: Critical for implementing the 5-minute task submission interval in clients and rate limiting on the server.**

7. **File and Database Logging in Java (Optional but Helpful)**
    - **Why: The server must log task history (completed, cancelled, failed) in a file or MySQL database. While file logging is simpler, understanding basic MySQL integration may be useful if you choose a database for MySQL_login_<USER_NAME>.sh.**
    - **Search Terms:**
        - **"Java file logging tutorial"**
        - **"Java write to file example"**
        - **"Java MySQL JDBC tutorial" (if using MySQL)**
    - **Key Concepts to Learn:**
        - **Writing task logs to a file using FileWriter or BufferedWriter.**
        - **Structuring logs in a readable format (e.g., TaskID=95, Script=file_audit.sh, Status=COMPLETED).**
        - **(Optional) Connecting to MySQL with JDBC, executing queries, and storing task history.**
    - **Example Videos:**
        - **"Java File Logging Tutorial" by Coding with John or Telusko.**
        - **"Java MySQL JDBC Tutorial" by freeCodeCamp or Amigoscode (if needed).**
    - **Relevance to Project: Ensures proper logging for the TASK_HISTORY command and supports MySQL_login_<USER_NAME>.sh if database logging is implemented.**

Suggested Learning Plan

- **Week 1 (Before Coding):**
    - **Day 1-2: Watch videos on Java Socket Programming (2-3 hours). Practice a simple client-server program sending strings.**

    - **Day 3: Watch Java PriorityQueue tutorials (1-2 hours). Write a small program to queue custom objects by priority.**
    - **Day 4: Review Java Thread Synchronization (1-2 hours). Experiment with synchronized blocks or ReentrantLock in a multithreaded program.**
    - **Day 5: Watch Java ProcessBuilder tutorials (1-2 hours). Try running a simple shell command (e.g., ls) from Java.**
- **Week 2:**
    - **Day 1-2: Watch Linux Shell Scripting tutorials (2-3 hours). Write a basic script (e.g., create a user or check connectivity) with error handling.**
    - **Day 3: Watch Java Timers and Scheduling videos (1-2 hours). Implement a program that prints a message every 5 seconds.**
    - **Day 4: (Optional) Watch File Logging or MySQL JDBC tutorials (1-2 hours). Practice writing to a file or querying a database.**
    - **Day 5: Set up the VMs and test basic socket communication between them.**
- **Total Time: ~12-15 hours of video watching and practice, spread over 1-2 weeks, depending on your pace.**

Tips for Finding Quality Videos

- **Channels to Look For: freeCodeCamp, Telusko, Amigoscode, Coding with John, Tech With Tim, Cave of Programming, Traversy Media, TechWorld with Nana.**
- **Filter by Date: Search for videos from 2020-2025 to ensure relevance (Java and Bash haven't changed much, but APIs like ProcessBuilder have modern practices).**
- **Check Video Length: Aim for 10-30 minute tutorials for focused learning, or longer crash courses (1-2 hours) for shell scripting.**
- **Practice Alongside: Pause videos to code examples yourself, as hands-on practice is crucial for retention.**

Relevance to Your Knowledge Level

Since you've covered multithreading, synchronization, and sockets, the socket programming and synchronization topics will reinforce your understanding with practical examples. The new topics (PriorityQueue, ProcessBuilder, shell scripting,

timers) are beginner-friendly and directly map to project requirements. Watching these videos will give you the confidence to:

- **Set up the server-client architecture.**
- **Manage prioritized tasks with thread safety.**
- **Execute and log shell scripts.**
- **Implement timed task submissions and rate limiting.**

Additional Notes

- **Start Small: After watching videos, begin with a minimal client-server program that sends a REQUEST_TASK command and receives a QUEUED response. Gradually add features like the priority queue and script execution.**
- **Collaborate: If working in a group of four, assign each student 1-2 topics to master (e.g., Student 1: Sockets, Student 2: PriorityQueue) and teach the others during team meetings.**
- **Test Early: Use tools like netcat to simulate client-server communication before coding, ensuring your VMs are networked correctly.**
- **Document Learning: Take notes from videos to reference during implementation, especially for syntax (e.g., ProcessBuilder setup or Bash error handling).**

By focusing on these YouTube topics, you'll gain the practical skills needed to implement the project efficiently, building on your existing knowledge of multithreading, synchronization, and sockets. If you need specific video recommendations or help with a particular concept later, let me know!