|  |  |
| --- | --- |
| **KHOA CÔNG NGHỆ THÔNG TIN** | **BÀI 08 – MULTITHREADING** |

**Phần thông tin sinh viên:**

Họ & tên: Ngô Thái Nguyên MSSV: 2180607804

Lớp: 21DTHD3

**Phần tự đánh giá:**

Các bài tập **ĐÃ** hoàn thành: 11/11

Các bài tập **CHƯA** hoàn thành: Không

Đánh giá mức độ hoàn thành (thang điểm 10): 10 /10

**Phần 1 – Lý thuyết**

**Câu 1 :** Đa tuyến là gì ?

**Đa tuyến (Multithreading)** là kỹ thuật cho phép một chương trình thực hiện nhiều luồng (thread) xử lý độc lập nhau, chia sẻ cùng một tài nguyên trong bộ nhớ chung. Mỗi luồng là một đơn vị xử lý của chương trình và có thể chạy song song với các luồng khác. Đa tuyến giúp tăng hiệu suất xử lý của chương trình, đặc biệt trong các hệ thống đa nhân.

**Câu 2 :** Vai trò của đa tuyến trong các ứng dụng client – server ?

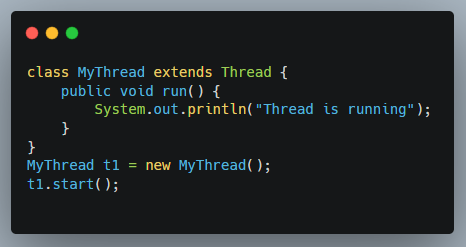
**Đa tuyến trong các ứng dụng client-server** giúp quản lý nhiều yêu cầu từ client cùng lúc. Server có thể tạo ra một luồng riêng cho mỗi yêu cầu từ client, giúp xử lý song song nhiều yêu cầu mà không cần chờ hoàn thành một yêu cầu trước khi bắt đầu yêu cầu khác. Điều này giúp cải thiện hiệu suất và khả năng đáp ứng của server.

**Câu 3 :** Ưu điểm của lập trình đa tuyến so với đa tiến trình ?

* **Tiết kiệm tài nguyên hệ thống**: Các luồng trong cùng một tiến trình chia sẻ bộ nhớ và tài nguyên, giúp giảm chi phí khởi tạo và quản lý so với việc tạo ra các tiến trình mới.
* **Tốc độ giao tiếp nhanh**: Các luồng giao tiếp trực tiếp với nhau thông qua bộ nhớ chia sẻ, trong khi các tiến trình cần cơ chế IPC (Inter-Process Communication) để giao tiếp, thường phức tạp và chậm hơn.
* **Tối ưu hiệu suất**: Đa tuyến tối ưu hiệu suất trên các CPU đa nhân do nhiều luồng có thể thực thi đồng thời.

**Câu 4 :** Để sử dụng tuyến, có những cách nào ? Cho ví dụ cụ thể về cách sử dụng ?

* Cách 1: Kế thừa từ lớp Thread



* Cách 2: Implement interface Runnable



**Câu 5 :** Mô tả các trạng thái của tuyến ?

* **New (Mới)**: Luồng được tạo nhưng chưa bắt đầu thực thi.
* **Runnable (Có thể chạy)**: Luồng đã được khởi động và đang chờ tài nguyên để chạy.
* **Blocked (Bị chặn)**: Luồng đang chờ một khóa hoặc tài nguyên để tiếp tục.
* **Waiting (Chờ đợi)**: Luồng chờ điều kiện cụ thể (ví dụ: chờ tín hiệu từ một luồng khác).
* **Timed Waiting (Chờ có thời gian)**: Luồng chờ một khoảng thời gian cụ thể trước khi tiếp tục.
* **Terminated (Kết thúc)**: Luồng đã hoàn thành thực thi.

**Câu 6 :** Các phương thức để quản lý tuyến ?

* **start()**: Khởi chạy luồng.
* **run()**: Được gọi khi luồng bắt đầu chạy.
* **sleep(long millis)**: Tạm dừng luồng trong một khoảng thời gian.
* **join()**: Chờ một luồng khác hoàn thành trước khi tiếp tục.
* **interrupt()**: Ngắt luồng đang chạy.
* **wait(), notify(), notifyAll()**: Đồng bộ hóa giữa các luồng

**Câu 7 :** Các thuật toán lập lịch cho tuyến ?

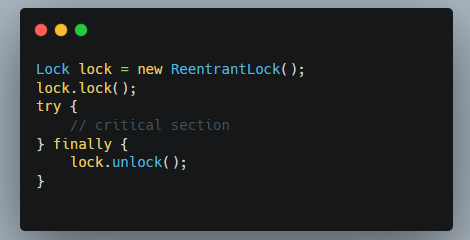
* **FIFO (First In First Out)**: Luồng được xử lý theo thứ tự được tạo ra.
* **Round Robin**: Mỗi luồng được cấp một khoảng thời gian cụ thể để thực thi, sau đó chuyển đến luồng tiếp theo.
* **Priority Scheduling**: Luồng có độ ưu tiên cao hơn sẽ được thực thi trước.
* **Shortest Job Next**: Luồng có thời gian thực thi ngắn nhất sẽ được thực thi trước.

**Câu 8 :** Các phương pháp để đồng bộ hóa trong trường hợp có nhiều tuyến truy cập cùng một dữ liệu hoặc tài nguyên ?

1. **Synchronized block/method**: Đảm bảo rằng chỉ một luồng có thể truy cập tài nguyên chia sẻ tại một thời điểm.



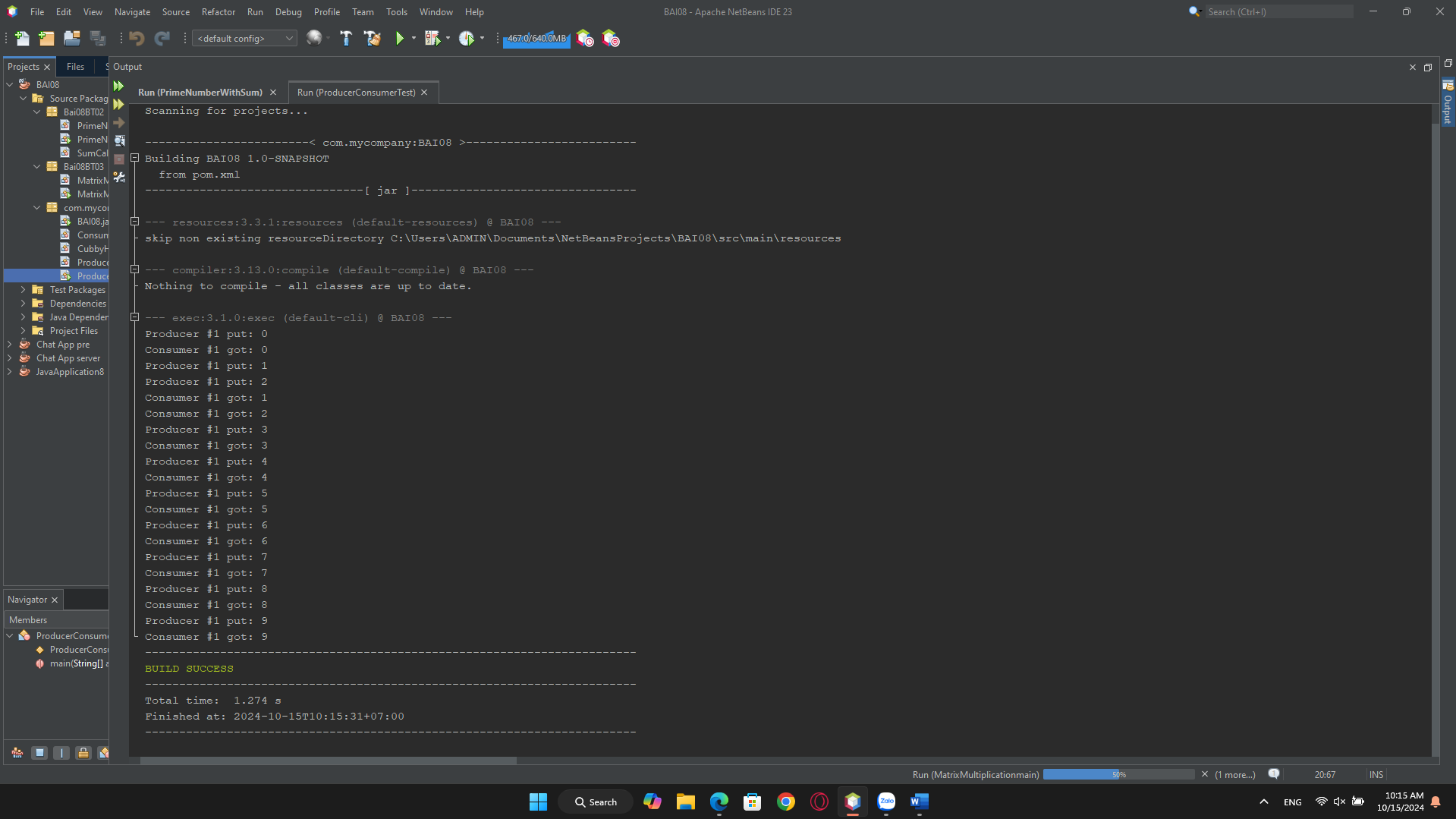
1. **Lock**: Cung cấp khả năng kiểm soát truy cập tài nguyên phức tạp hơn.



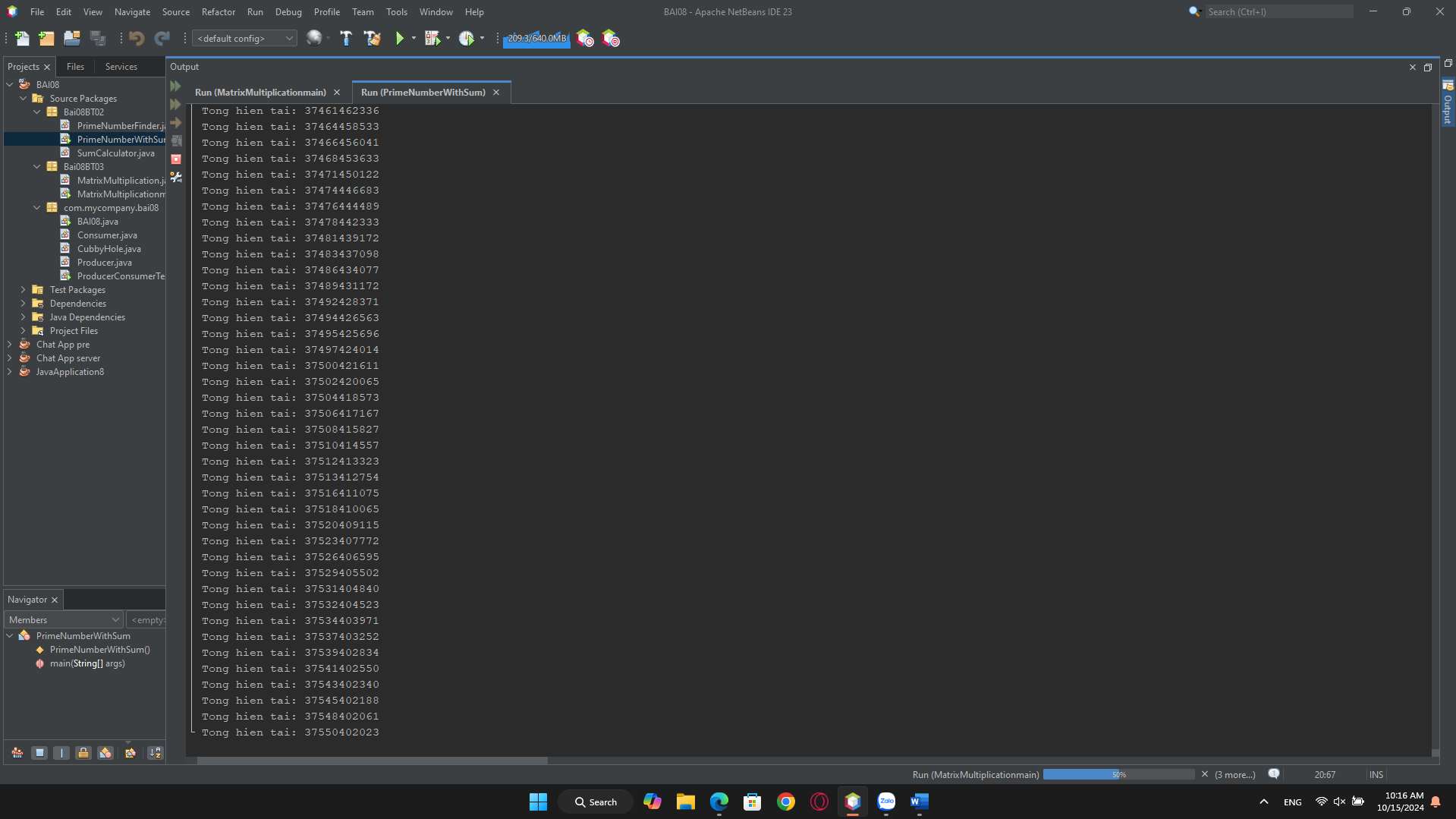
1. **Atomic Variables**: Các biến nguyên tử như AtomicInteger, đảm bảo các hoạt động trên biến là không chia cắt.
2. **Semaphore**: Giới hạn số luồng có thể truy cập tài nguyên cùng lúc.
3. **Monitor**: Cơ chế đồng bộ tích hợp trong Java để quản lý truy cập tài nguyên.

**Phần 2 – Bài tập**

* BAI08–BT01:



* BAI08–BT02:



* BAI08–BT03:

