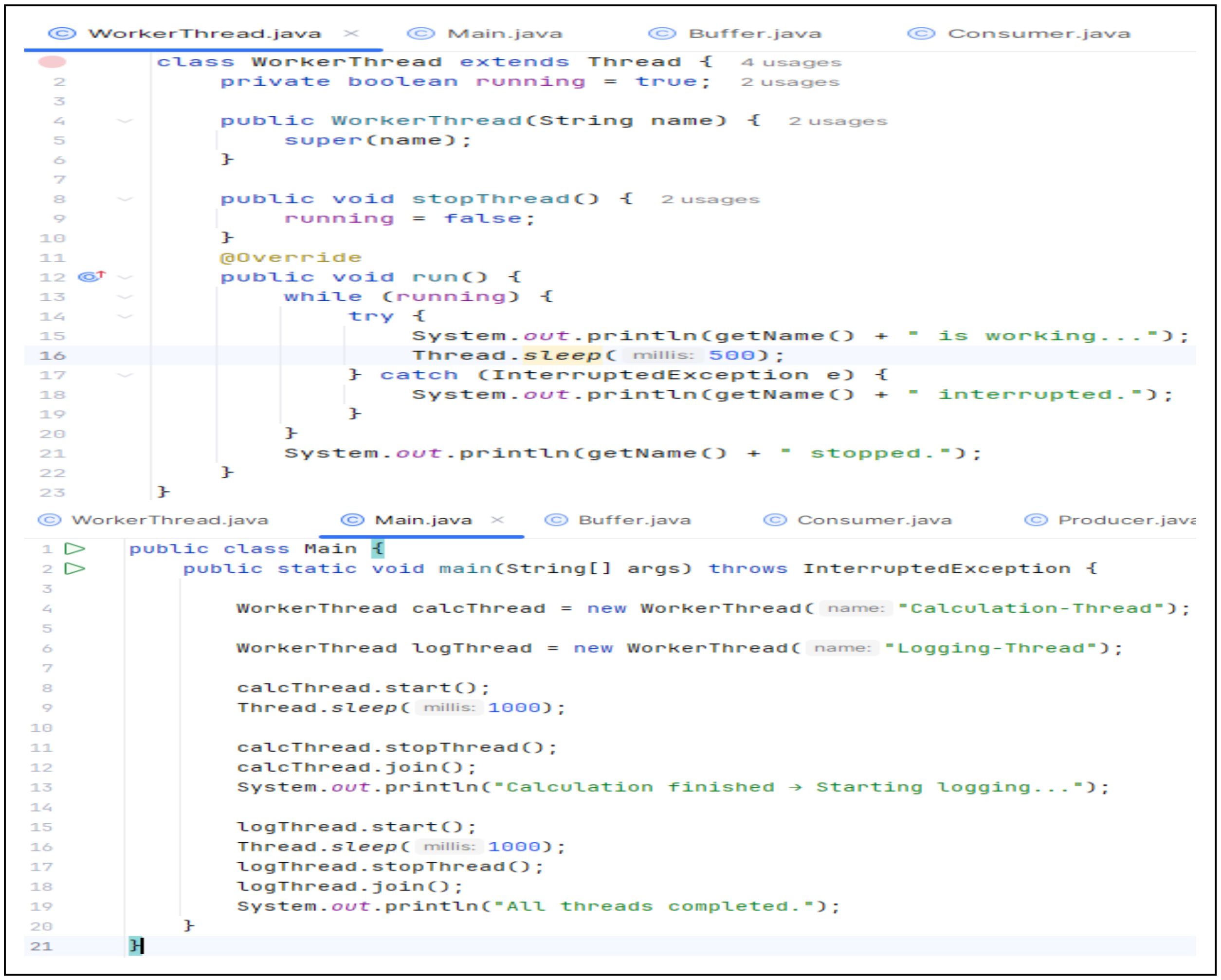
LAB # 8

Open Ended Lab

Lab Tasks:-

1. **Concurrency with Multithreading Mechanisms**
   - Create a class with multiple threads that perform different tasks (e.g., one thread handles calculations, another handles data logging).
   - Implement the `start`, `sleep`, and `stop` functionalities to demonstrate different thread lifecycle states.
   - Use `join()` where necessary to ensure one thread completes before another begins, simulating dependency between threads.
   - **Deliverable**: Code implementing multithreading with clear comments explaining the use of each concurrency method (`start`, `sleep`, `stop`, and `join`).

Code:-



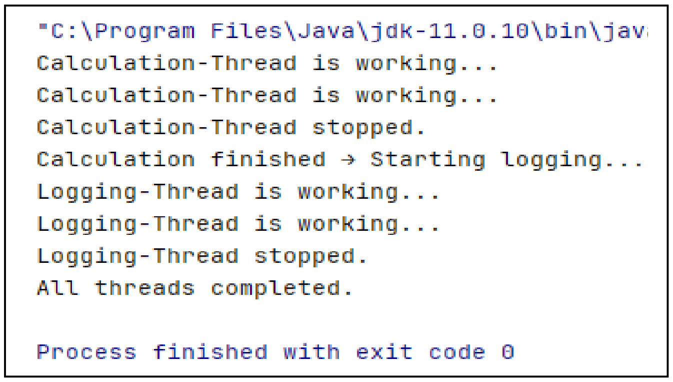
WorkerThread.java   Main.java   Buffer.java   Consumer.java

```
class WorkerThread extends Thread {  4 usages
    private boolean running = true;  2 usages

    public WorkerThread(String name) {  2 usages
        super(name);
    }

    public void stopThread() {  2 usages
        running = false;
    }
    @Override
    public void run() {
        while (running) {
            try {
                System.out.println(getName() + " is working...");
                Thread.sleep( millis: 500);
            } catch (InterruptedException e) {
                System.out.println(getName() + " interrupted.");
            }
        }
        System.out.println(getName() + " stopped.");
    }
}
```

WorkerThread.java   Main.java   Buffer.java   Consumer.java   Producer.java

```
public class Main {
    public static void main(String[] args) throws InterruptedException {

        WorkerThread calcThread = new WorkerThread( name: "Calculation-Thread");

        WorkerThread logThread = new WorkerThread( name: "Logging-Thread");

        calcThread.start();
        Thread.sleep( millis: 1000);

        calcThread.stopThread();
        calcThread.join();
        System.out.println("Calculation finished → Starting logging...");

        logThread.start();
        Thread.sleep( millis: 1000);
        logThread.stopThread();
        logThread.join();
        System.out.println("All threads completed.");
    }
}
```

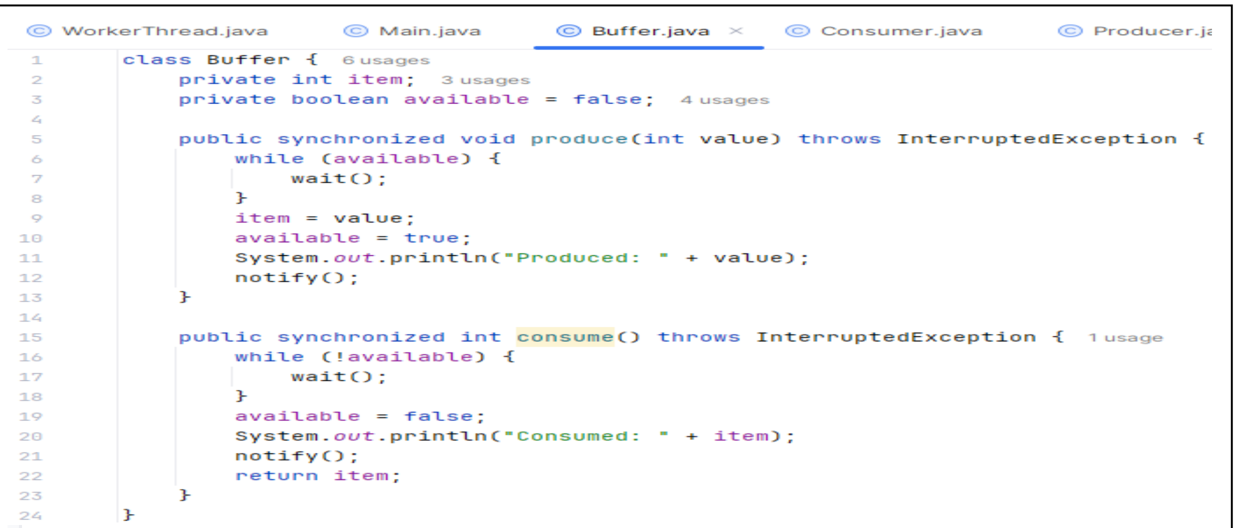
Output:-



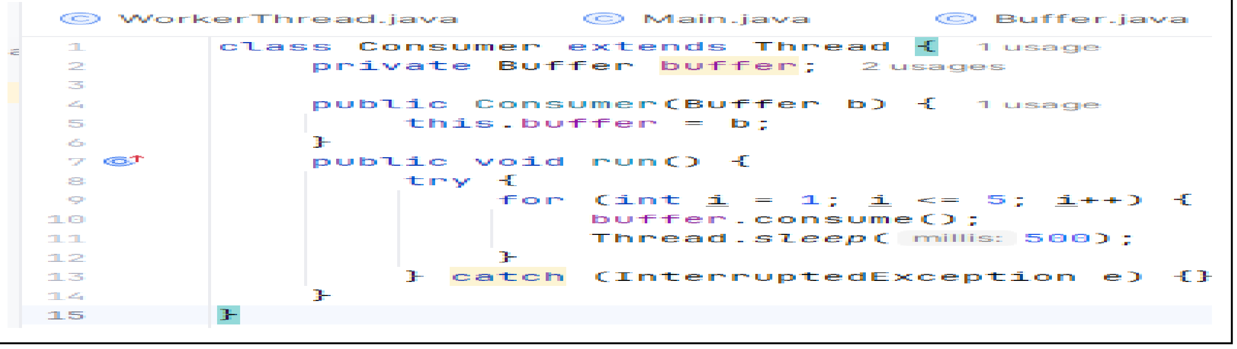
```
"C:\Program Files\Java\jdk-11.0.10\bin\jav
Calculation-Thread is working...
Calculation-Thread is working...
Calculation-Thread stopped.
Calculation finished → Starting logging...
Logging-Thread is working...
Logging-Thread is working...
Logging-Thread stopped.
All threads completed.

Process finished with exit code 0
```

2. **Inter-Thread Communication Using Synchronization**
   - Develop a program where two threads communicate via shared resources.
   - Use synchronization techniques (such as synchronized methods or blocks) to ensure that shared resources are accessed safely by each thread.
   - Example scenario: Implement a "Producer-Consumer" pattern, where one thread (Producer) adds items to a buffer, and another thread (Consumer) removes them, using `wait()` and `notify()` methods.
   - **Deliverable**: A fully functional inter-thread communication program demonstrating correct use of synchronization.

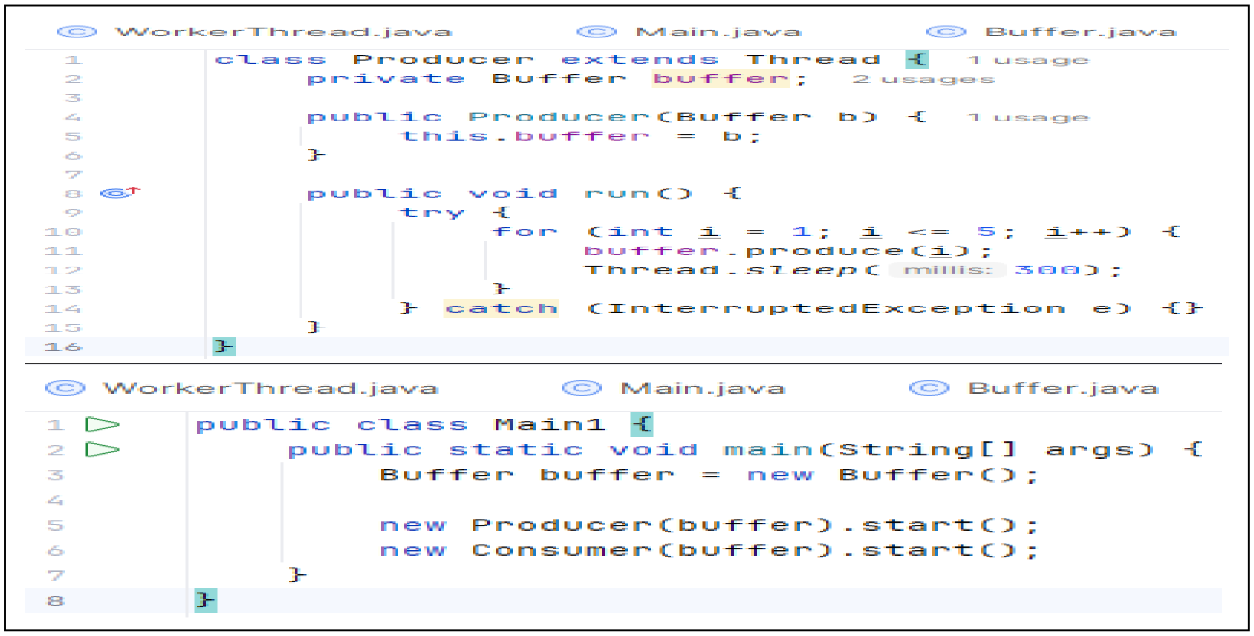
Code:-



```java
class Buffer {
    private int item;
    private boolean available = false;

    public synchronized void produce(int value) throws InterruptedException {
        while (available) {
            wait();
        }
        item = value;
        available = true;
        System.out.println("Produced: " + value);
        notify();
    }

    public synchronized int consume() throws InterruptedException {
        while (!available) {
            wait();
        }
        available = false;
        System.out.println("Consumed: " + item);
        notify();
        return item;
    }
}
```



```java
class Consumer extends Thread {
    private Buffer buffer;

    public Consumer(Buffer b) {
        this.buffer = b;
    }
    public void run() {
        try {
            for (int i = 1; i <= 5; i++) {
                buffer.consume();
                Thread.sleep( millis: 500);
            }
        } catch (InterruptedException e) {}
    }
}
```



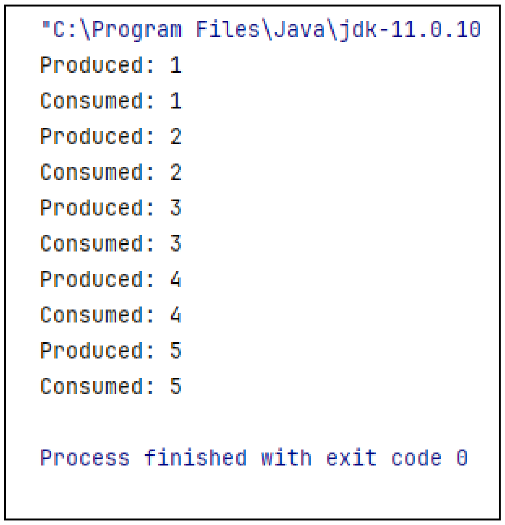
WorkerThread.java Main.java Buffer.java

```java
class Producer extends Thread {  1 usage
    private Buffer buffer;  2 usages

    public Producer(Buffer b) {  1 usage
        this.buffer = b;
    }

    public void run() {
        try {
            for (int i = 1; i <= 5; i++) {
                buffer.produce(i);
                Thread.sleep( millis: 300);
            }
        } catch (InterruptedException e) {}
    }
}
```

WorkerThread.java Main.java Buffer.java

```java
public class Main1 {
    public static void main(String[] args) {
        Buffer buffer = new Buffer();

        new Producer(buffer).start();
        new Consumer(buffer).start();
    }
}
```

Output:-



```
"C:\Program Files\Java\jdk-11.0.10
Produced: 1
Consumed: 1
Produced: 2
Consumed: 2
Produced: 3
Consumed: 3
Produced: 4
Consumed: 4
Produced: 5
Consumed: 5

Process finished with exit code 0
```