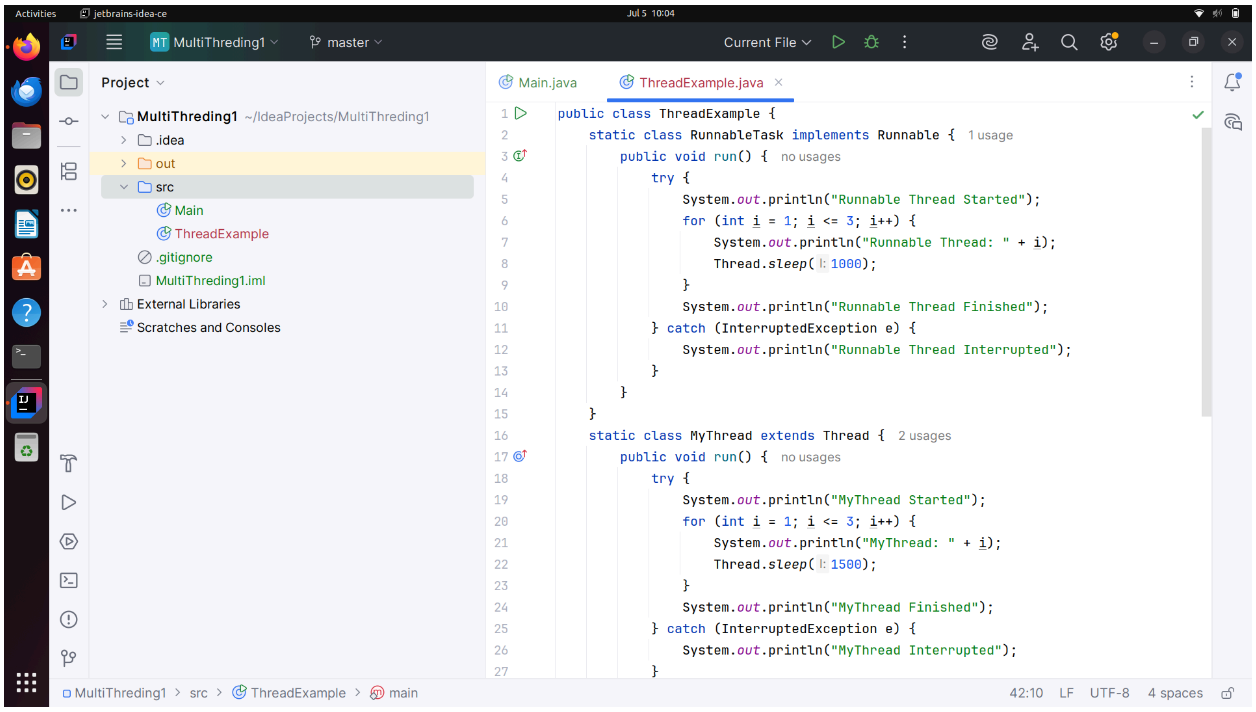
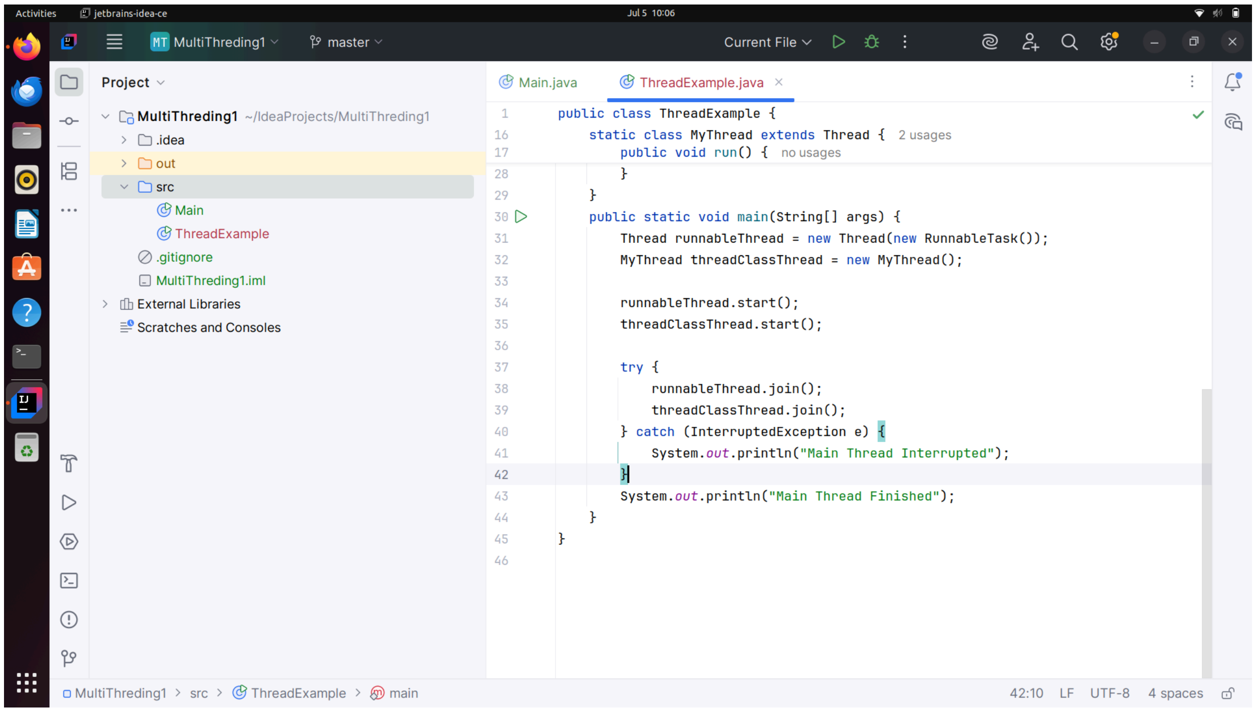
1. Create and Run a Thread using Runnable Interface and Thread class and show usage of sleep and join methods in the created threads.

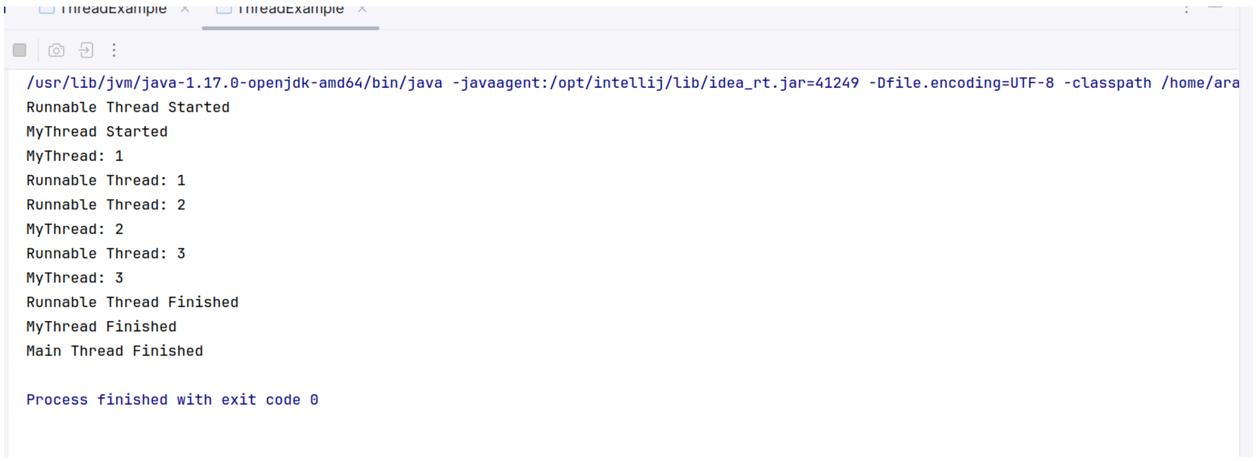


```java
public class ThreadExample {
    static class RunnableTask implements Runnable {
        public void run() {
            try {
                System.out.println("Runnable Thread Started");
                for (int i = 1; i <= 3; i++) {
                    System.out.println("Runnable Thread: " + i);
                    Thread.sleep(1000);
                }
                System.out.println("Runnable Thread Finished");
            } catch (InterruptedException e) {
                System.out.println("Runnable Thread Interrupted");
            }
        }
    }
    static class MyThread extends Thread {
        public void run() {
            try {
                System.out.println("MyThread Started");
                for (int i = 1; i <= 3; i++) {
                    System.out.println("MyThread: " + i);
                    Thread.sleep(1500);
                }
                System.out.println("MyThread Finished");
            } catch (InterruptedException e) {
                System.out.println("MyThread Interrupted");
            }
```



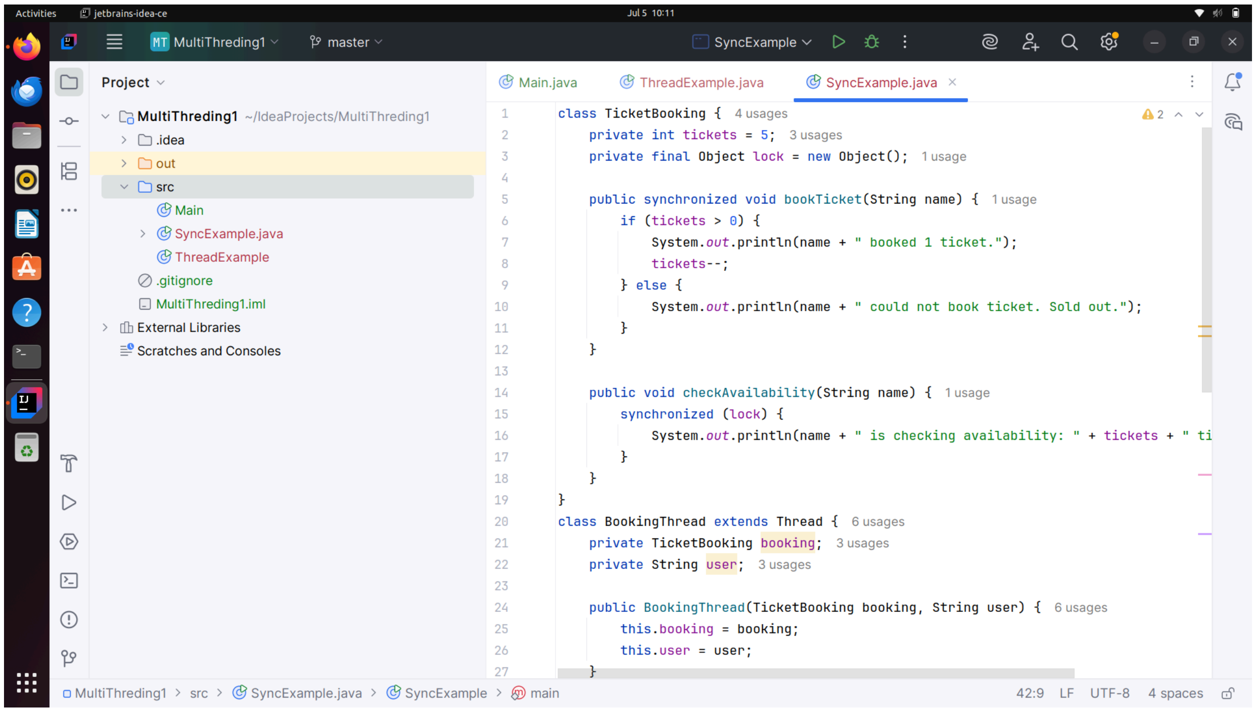
```java
public class ThreadExample {
    static class MyThread extends Thread {
        public void run() {
        }
    }
    public static void main(String[] args) {
        Thread runnableThread = new Thread(new RunnableTask());
        MyThread threadClassThread = new MyThread();

        runnableThread.start();
        threadClassThread.start();

        try {
            runnableThread.join();
            threadClassThread.join();
        } catch (InterruptedException e) {
            System.out.println("Main Thread Interrupted");
        }
        System.out.println("Main Thread Finished");
    }
}
```

Output:

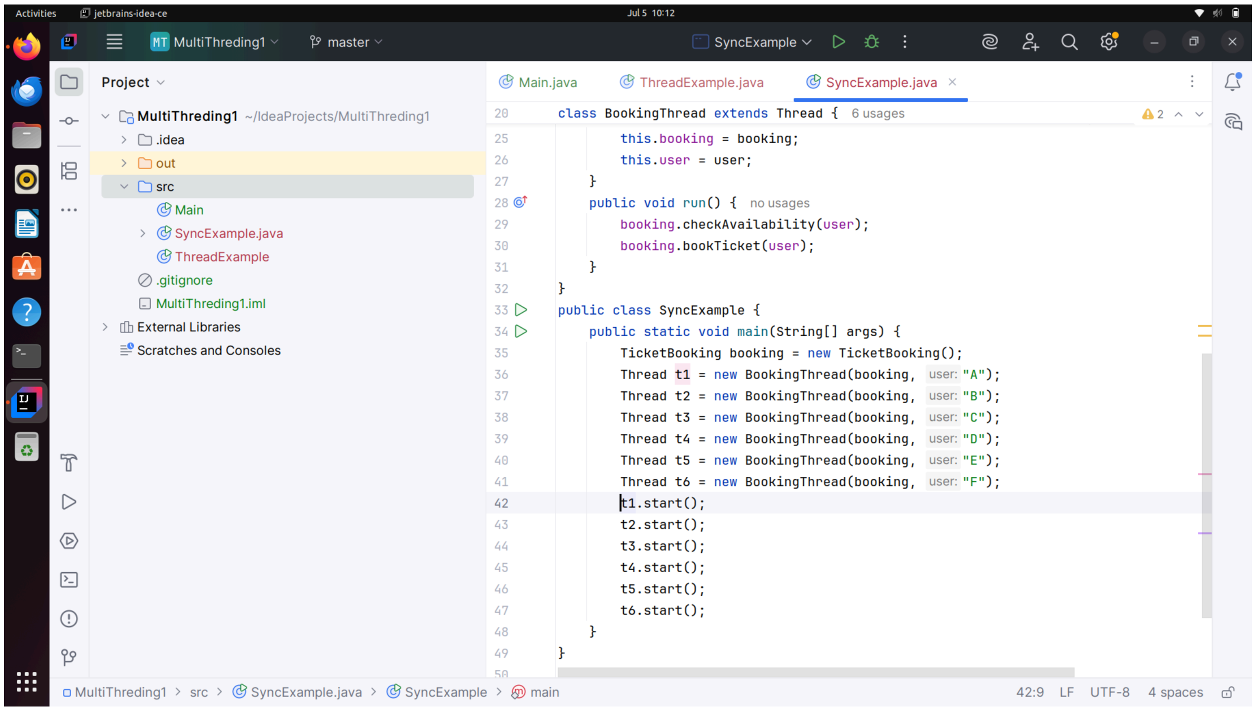


```
/usr/lib/jvm/java-1.17.0-openjdk-amd64/bin/java -javaagent:/opt/intellij/lib/idea_rt.jar=41249 -Dfile.encoding=UTF-8 -classpath /home/ara
Runnable Thread Started
MyThread Started
MyThread: 1
Runnable Thread: 1
Runnable Thread: 2
MyThread: 2
Runnable Thread: 3
MyThread: 3
Runnable Thread Finished
MyThread Finished
Main Thread Finished

Process finished with exit code 0
```

2. Use Synchronize method and synchronize block to enable synchronization between multiple threads trying to access method at same time.

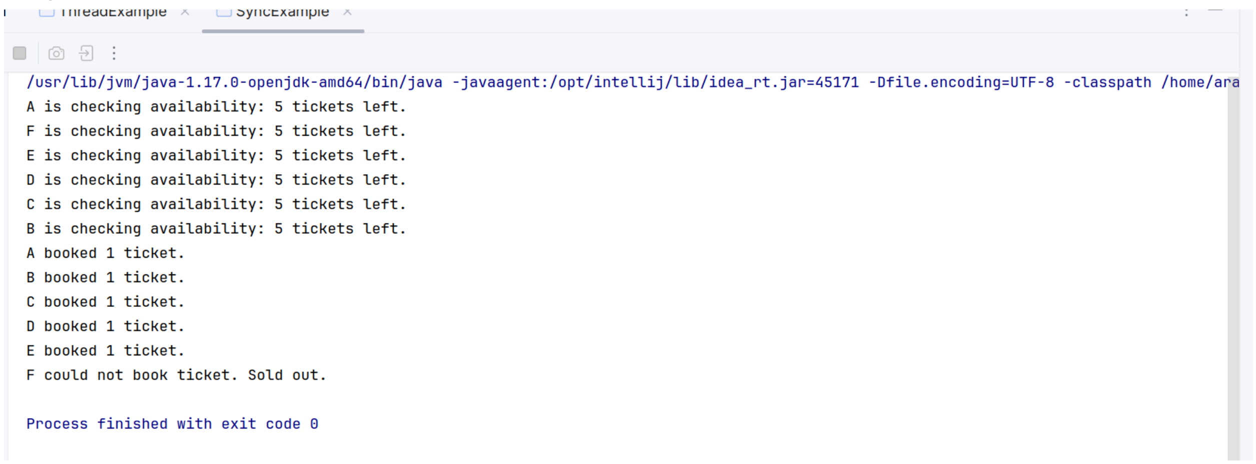


```java
class TicketBooking {  4 usages
    private int tickets = 5;  3 usages
    private final Object lock = new Object();  1 usage

    public synchronized void bookTicket(String name) {  1 usage
        if (tickets > 0) {
            System.out.println(name + " booked 1 ticket.");
            tickets--;
        } else {
            System.out.println(name + " could not book ticket. Sold out.");
        }
    }

    public void checkAvailability(String name) {  1 usage
        synchronized (lock) {
            System.out.println(name + " is checking availability: " + tickets + " ti
        }
    }
}
class BookingThread extends Thread {  6 usages
    private TicketBooking booking;  3 usages
    private String user;  3 usages

    public BookingThread(TicketBooking booking, String user) {  6 usages
        this.booking = booking;
        this.user = user;
    }
```



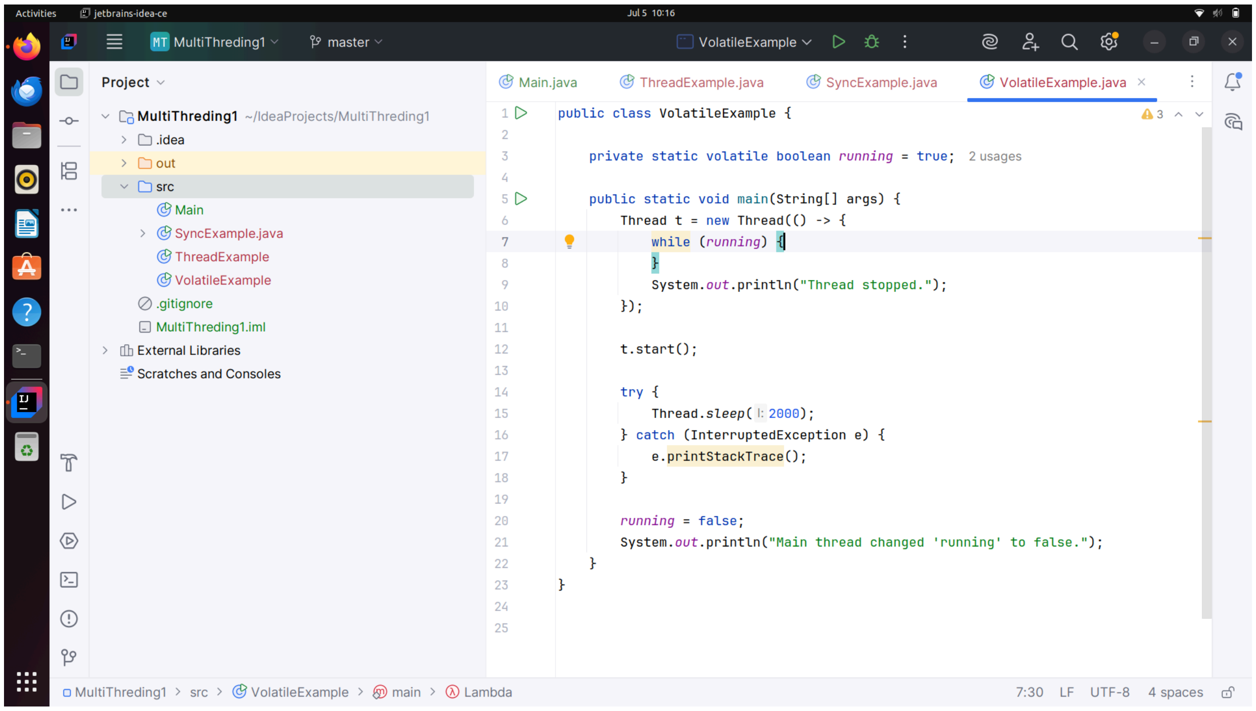
```java
class BookingThread extends Thread {  6 usages
        this.booking = booking;
        this.user = user;
    }
    public void run() {  no usages
        booking.checkAvailability(user);
        booking.bookTicket(user);
    }
}
public class SyncExample {
    public static void main(String[] args) {
        TicketBooking booking = new TicketBooking();
        Thread t1 = new BookingThread(booking,  user: "A");
        Thread t2 = new BookingThread(booking,  user: "B");
        Thread t3 = new BookingThread(booking,  user: "C");
        Thread t4 = new BookingThread(booking,  user: "D");
        Thread t5 = new BookingThread(booking,  user: "E");
        Thread t6 = new BookingThread(booking,  user: "F");
        t1.start();
        t2.start();
        t3.start();
        t4.start();
        t5.start();
        t6.start();
    }
}
```

Output:



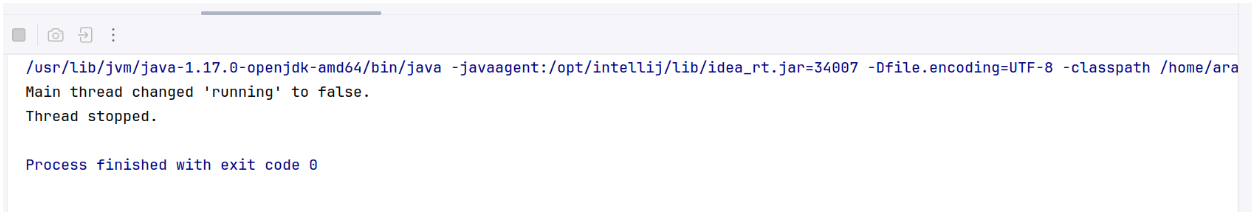
```
/usr/lib/jvm/java-1.17.0-openjdk-amd64/bin/java -javaagent:/opt/intellij/lib/idea_rt.jar=45171 -Dfile.encoding=UTF-8 -classpath /home/ara
A is checking availability: 5 tickets left.
F is checking availability: 5 tickets left.
E is checking availability: 5 tickets left.
D is checking availability: 5 tickets left.
C is checking availability: 5 tickets left.
B is checking availability: 5 tickets left.
A booked 1 ticket.
B booked 1 ticket.
C booked 1 ticket.
D booked 1 ticket.
E booked 1 ticket.
F could not book ticket. Sold out.

Process finished with exit code 0
```

3. WAP to showcase the usage of volatile in java.



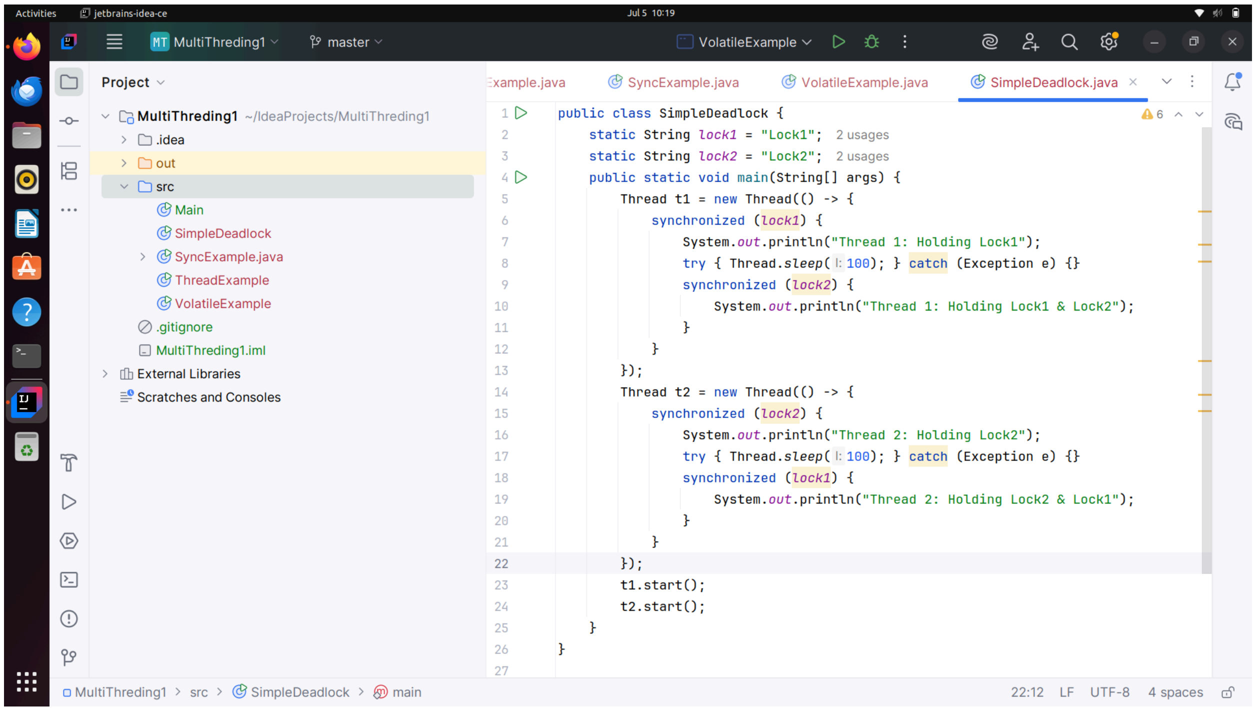
```java
public class VolatileExample {

    private static volatile boolean running = true;

    public static void main(String[] args) {
        Thread t = new Thread(() -> {
            while (running) {
            }
            System.out.println("Thread stopped.");
        });

        t.start();

        try {
            Thread.sleep(2000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        running = false;
        System.out.println("Main thread changed 'running' to false.");
    }
}
```

Output:



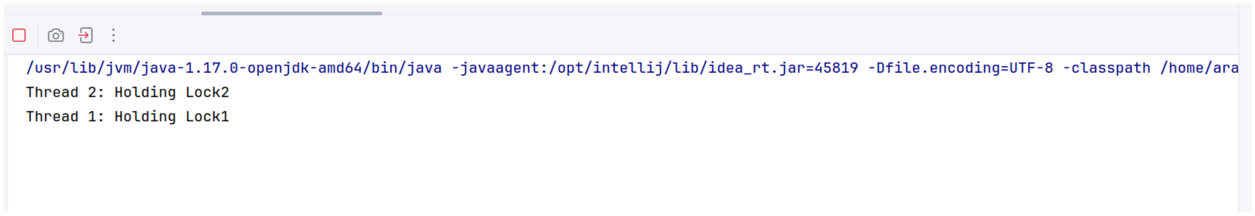
```
/usr/lib/jvm/java-1.17.0-openjdk-amd64/bin/java -javaagent:/opt/intellij/lib/idea_rt.jar=34007 -Dfile.encoding=UTF-8 -classpath /home/ara
Main thread changed 'running' to false.
Thread stopped.

Process finished with exit code 0
```

4. Write a code to simulate a deadlock in java



```java
public class SimpleDeadlock {
    static String lock1 = "Lock1";   2 usages
    static String lock2 = "Lock2";   2 usages
    public static void main(String[] args) {
        Thread t1 = new Thread(() -> {
            synchronized (lock1) {
                System.out.println("Thread 1: Holding Lock1");
                try { Thread.sleep( l: 100); } catch (Exception e) {}
                synchronized (lock2) {
                    System.out.println("Thread 1: Holding Lock1 & Lock2");
                }
            }
        });
        Thread t2 = new Thread(() -> {
            synchronized (lock2) {
                System.out.println("Thread 2: Holding Lock2");
                try { Thread.sleep( l: 100); } catch (Exception e) {}
                synchronized (lock1) {
                    System.out.println("Thread 2: Holding Lock2 & Lock1");
                }
            }
        });
        t1.start();
        t2.start();
    }
}
```

Output:



```
/usr/lib/jvm/java-1.17.0-openjdk-amd64/bin/java -javaagent:/opt/intellij/lib/idea_rt.jar=45819 -Dfile.encoding=UTF-8 -classpath /home/ara
Thread 2: Holding Lock2
Thread 1: Holding Lock1
```