SPOS Practical\Mutex Semaphore\Mutex.java

```java
import java.util.concurrent.Semaphore;
public class Mutex {
    static Semaphore semaphore = new Semaphore(1);
    static class MyLockerThread extends Thread {
        String name = "";
        MyLockerThread(String name) {
            this.name = name;
        }
        public void run() {
            try {
                System.out.println(name + " : acquiring lock...");
                System.out.println(name + " : available Mutex permits now: " + semaphore.availablePermits());
                semaphore.acquire();
                System.out.println(name + " : got the permit!");
                try {
                    for (int i = 1; i <= 5; i++) {

                        System.out.println(name + " : is performing operation " + i + ", available Mutex permits : "+ semaphore.availablePermits());
                        // sleep 1 second
                        Thread.sleep(1000);

                    }
                } finally {
                    System.out.println(name + " : releasing lock...");
                    semaphore.release();
                    System.out.println(name + " : available Mutex permits now: " + semaphore.availablePermits());
                }
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }

    public static void main(String[] args) {
        System.out.println("Total available Mutex permits : " + semaphore.availablePermits());
        MyLockerThread t1 = new MyLockerThread("A");
        t1.start();
        MyLockerThread t2 = new MyLockerThread("B");
        t2.start();
        MyLockerThread t3 = new MyLockerThread("C");
        t3.start();

        MyLockerThread t4 = new MyLockerThread("D");
        t4.start();

        MyLockerThread t5 = new MyLockerThread("E");
        t5.start();

        MyLockerThread t6 = new MyLockerThread("F");
        t6.start();

```

```
52        }
53    }
```