```
package ass4Mutex;
import java.util.concurrent.Semaphore;

public class ass4Mutex {


  // max 1 people
  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 Semaphore 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 Semaphore permits : "
         + semaphore.availablePermits());

       // sleep 1 second
       Thread.sleep(1000);

      }

     } finally {

      // calling release() after a successful acquire()
      System.out.println(name + " : releasing lock...");
      semaphore.release();
      System.out.println(name + " : available Semaphore permits now: "
         + semaphore.availablePermits());

     }

    } catch (InterruptedException e) {

     e.printStackTrace();
   }

  }

 }

 public static void main(String[] args) {

  System.out.println("Total available Semaphore 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();

 }
}
```