Mutex:

```
import java.util.Scanner;

public class Ass_6_2 {
    public final static int NUMTHREADS = 3;
    public static int sharedData = 0;
    public static int sharedData2 = 0;

    static class theLock extends Object {
    }

    static public theLock lockObject = new theLock();

    static class theThread extends Thread {
        public void run() {
            System.out.print("Thread " + getName() + ": Entered\n");
            synchronized (lockObject) {
                /********** Critical Section ******************/
                System.out.println("Thread " + getName() + ": Start
critical section, insynchronized block\n");
                ++sharedData;
                --sharedData2;
                System.out.print("Thread " + getName() + ": End critical
section, leavesynchronized block\n");
                /********** Critical Section ******************/
            }
        }
    }

    public static void main(String argv[]) {
        theThread threads[] = new theThread[NUMTHREADS];
```

```
        System.out.print("Entered the testcase\n");
        System.out.print("Synchronize to prevent access to shared data
\n");
        synchronized (lockObject) {
            System.out.print("Create/start the thread\n");
            for (int i = 0; i < NUMTHREADS; ++i) {
                threads[i] = new theThread();
                threads[i].start();
            }
            System.out.print("Wait a bit until we're 'done' with the
shared data\n");
            try {
                Thread.sleep(3000);
            } catch (InterruptedException e) {
                System.out.print("sleep interrupted\n");
            }
            System.out.print("Unlock shared data\n");
        }
        System.out.print("Wait for the threads to complete\n");
        try {
            for (int i = 0; i < NUMTHREADS; ++i) {
                threads[i].join();
                System.out.print("Testcase completed\n");
                System.exit(0);
            }
        } catch (InterruptedException e) {
            System.out.print("Join interrupted\n");
        }
    }
}
```

Output:

```
Entered the testcase
Synchronize to prevent access to shared data
Create/start the thread
Wait a bit until we're 'done' with the shared data
Thread Thread-0: Entered
Thread Thread-2: Entered
Thread Thread-1: Entered
Unlock shared data
Wait for the threads to complete
Thread Thread-1: Start critical section, insynchronized block

Thread Thread-1: End critical section, leavesynchronized block
Thread Thread-2: Start critical section, insynchronized block

Thread Thread-2: End critical section, leavesynchronized block
Thread Thread-0: Start critical section, insynchronized block

Thread Thread-0: End critical section, leavesynchronized block
Testcase completed
PS E:\LP-1\LP-1 Codes>
```