Semaphore:

```
import java.util.*;
import java.util.concurrent.*;

class SharedResource {
    static int count = 0;
}

class MyThread extends Thread {
    Semaphore semaphore;
    String threadName;

    public MyThread(Semaphore semaphore, String threadName) {
        super(threadName);
        this.semaphore = semaphore;
        this.threadName = threadName;
    }

    public void run() {
        if (this.getName().equals("Thread1")) {
            System.out.println("Starting " + threadName);
            try {
                System.out.println(threadName + " is waiting for a permit.");
                semaphore.acquire();
                System.out.println(threadName + " gets a permit.");
                for (int i = 0; i < 5; i++) {
                    SharedResource.count++;
                    System.out.println(threadName + ": " + SharedResource.count);
                    Thread.sleep(10);
                }
            } catch (InterruptedException exc) {
                System.out.println(exc);
            }
            System.out.println(threadName + " releases the permit.");
            semaphore.release();
        } else {
            System.out.println("Starting " + threadName);

            try {
                System.out.println(threadName + " is waiting for a permit.");
                semaphore.acquire();
                System.out.println(threadName + " gets a permit.");
                for (int i = 0; i < 5; i++) {
                    SharedResource.count--;
                    System.out.println(threadName + ": " + SharedResource.count);
                    Thread.sleep(10);
                }
            } catch (InterruptedException exc) {
                System.out.println(exc);
            }
            System.out.println(threadName + " releases the permit.");
            semaphore.release();
        }
    }
}

public class Ass_6 {
    public static void main(String args[]) throws InterruptedException {
        Semaphore semaphore = new Semaphore(1);
        MyThread t1 = new MyThread(semaphore, "Thread1");
        MyThread t2 = new MyThread(semaphore, "Thread2");
        t1.start();
        t2.start();
        t1.join();
        t2.join();
        System.out.println("count: " + SharedResource.count);
    }
}
```

Output:

```
Starting Thread2
Starting Thread1
Thread2 is waiting for a permit.
Thread2 gets a permit.
Thread1 is waiting for a permit.
Thread2: -1
Thread2: -2
Thread2: -3
Thread2: -4
Thread2: -5
Thread2 releases the permit.
Thread1 gets a permit.
Thread1: -4
Thread1: -3
Thread1: -2
Thread1: -1
Thread1: 0
Thread1 releases the permit.
count: 0
PS E:\LP-1\LP-1 Codes>
```