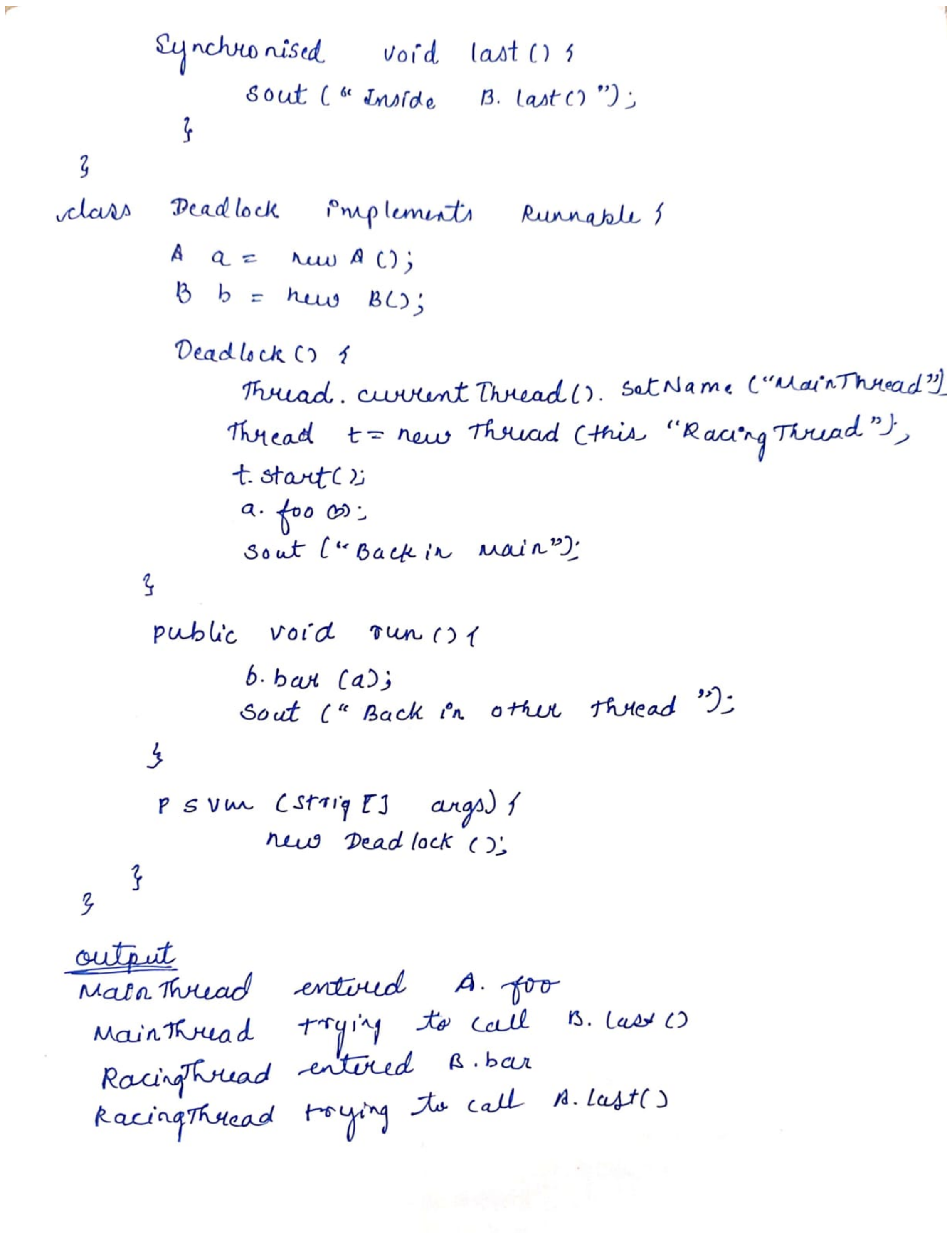
Lab Program 10

Q] Demonstrate Inter process Communication and deadlock

10] Demonstrate Inter Process communication and deadlock

```
class A {
    Synchronized void foo(B b) {
        String name = thread. currentThread(). getName
        System.out. println (name+ "entered A.foo");
        try {
            Thread. Sleep (1000);
        }catch (Exception e) {
            System. out. println ("A Interrupted");
        }
        System out. println(name+"trying to call B.last()");
        b.last();
    }
    Synchronised void last() {
        sout (" Inside A.last()");
    }
}
class B {
    Synchronised void bar (A a) {
        String name = Thread. currentThread (). getName
        Sout (name+ "entered B. bar");
        try {
            Thread. Sleep (1000);
        }
        catch (Exception e) {
            Sout (" B Interrupted");
        }
        Sout (name + " trying to call A. last()");
        a. last();
    }
```



```
        Synchronised   void   last () {
                Sout (" Inside   B. last() ");
        }
}
class   Deadlock   implements   Runnable {
        A   a =   new A ();
        B   b =   new   B();

        Deadlock () {
                Thread. currentThread(). setName ("MainThread")
                Thread   t = new Thread (this "RacingThread");
                t. start();
                a. foo (b);
                Sout ("Back in   main");
        }
        public   void   run () {
                b. bar (a);
                Sout (" Back in   other   thread ");
        }
        P S V M (String [ ]   args) {
                new   Dead lock ();
        }
}
```

output

```
MainThread   entered   A. foo
MainThread   trying   to call   B. last ()
RacingThread   entered   B.bar
RacingThread   trying   to call   A. last( )
```

```
class A {
    synchronized void foo(B b) {
        String name = Thread.currentThread().getName();
        System.out.println(name + " entered A.foo");
        try {
            Thread.sleep(1000);
        } catch (Exception e) {
            System.out.println("A interrupted");
        }
        System.out.println(name + " trying to call B.last()");
        b.last();
    }

    synchronized void last() {
        System.out.println("Inside A.last()");
    }
}

class B {
    synchronized void bar(A a) {
        String name = Thread.currentThread().getName();
        System.out.println(name + " entered B.bar");
        try {
            Thread.sleep(1000);
        } catch (Exception e) {
            System.out.println("B interrupted");
        }
        System.out.println(name + " trying to call A.last()");
        a.last();
    }

    synchronized void last() {
        System.out.println("Inside B.last()");
    }
}

class Deadlock implements Runnable {
    A a = new A();
    B b = new B();

    Deadlock() {
        Thread.currentThread().setName("MainThread");
        Thread t = new Thread(this, "RacingThread");
        t.start();
        a.foo(b); // Main thread locks A and tries to call B.last()
        System.out.println("Back in main");
    }

    public void run() {
        b.bar(a); // Racing thread locks B and tries to call A.last()
        System.out.println("Back in other thread");
    }

    public static void main(String[] args) {
        new Deadlock();
    }
}
```

//OUTPUT

```
MainThread entered A.foo
MainThread trying to call B.last()
RacingThread entered B.bar
RacingThread trying to call A.last()
```