Lab 10. (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();
}
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();
}
```

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

    Thread.currentThread().setName("Main Thread");

    Thread t = new Thread(this, "RacingThread");
    t.start();
    a.foo(b);    // get a lock on a in this thread
    System.out.println("Back in main thread");
}

public void run() {
    b.bar(a); // get a lock on b in other thread.

System.out.println("Back in other thread");
}
public static void main(String args[]) {
    new Deadlock();
}
}
```

Main Thread entered

Output.

MainThread entered A.foo
RacingThread entered B.bar
MainThread trying to call B.last()
Inside A.last
Back in main thread
RacingThread trying to call A.last()
Inside A.last
Back in other thread