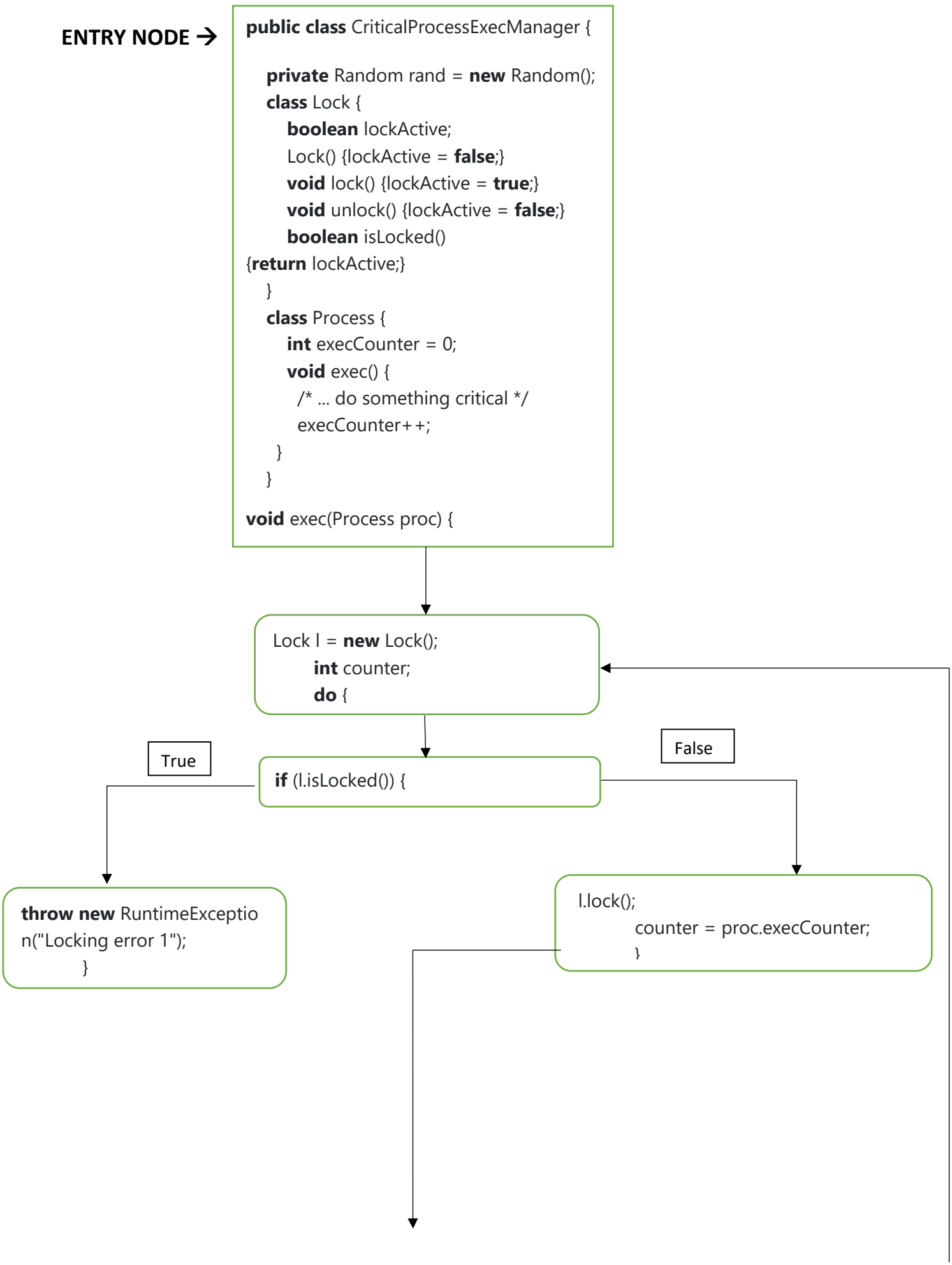
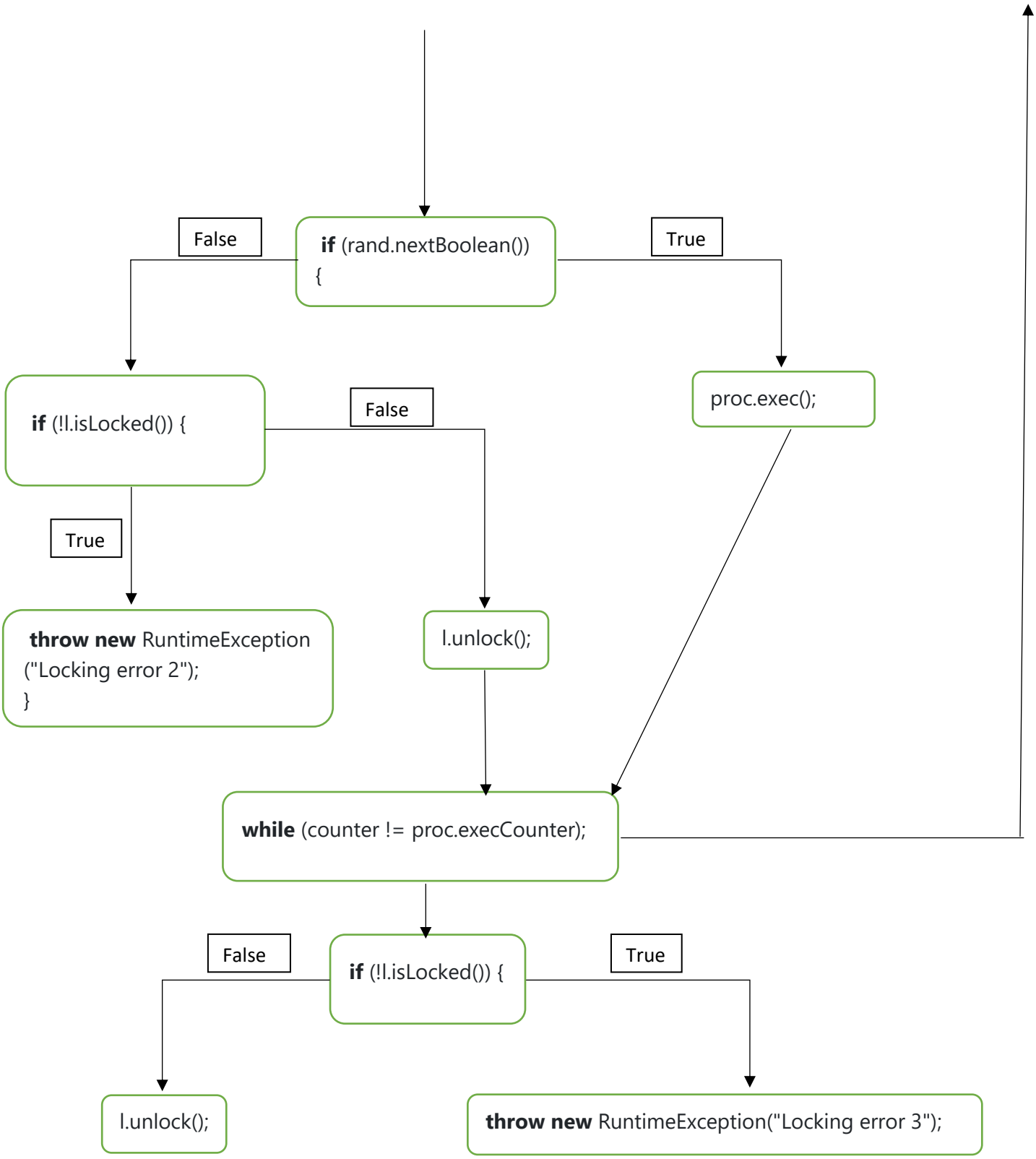
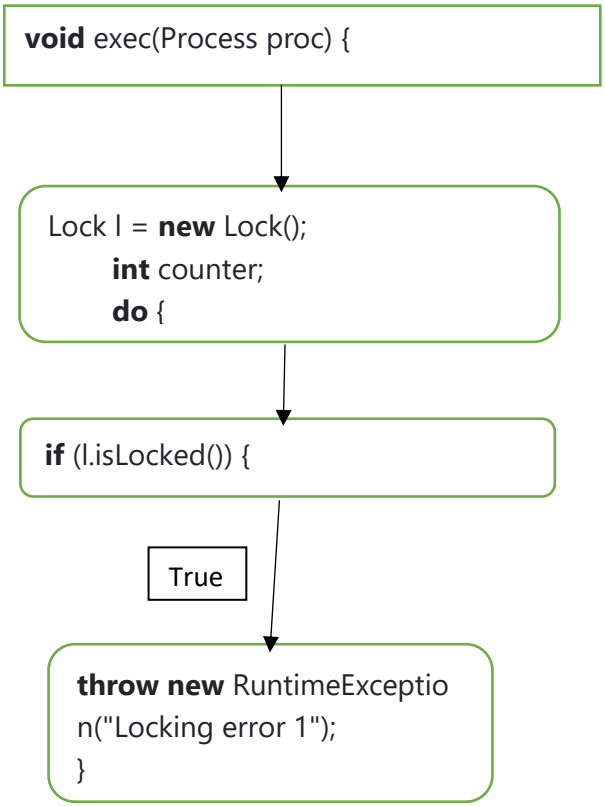

ENTRY NODE →
public class CriticalProcessExecManager {
private Random rand = new Random();
class Lock {
boolean lockActive;
Lock() {lockActive = false;}
void lock() {lockActive = true;}
void unlock() {lockActive = false;}
boolean isLocked()
{return lockActive;}
}
class Process {
int execCounter = 0;
void exec() {
/* ... do something critical */
execCounter++;
}
}
void exec(Process proc) {
Lock l = new Lock();
int counter;
do {
if (l.isLocked()) {
True
False
throw new RuntimeExceptio
n("Locking error 1");
}
l.lock();
counter = proc.execCounter;
}


if (rand.nextBoolean())
{
False
True
if (!l.isLocked()) {
False
True
proc.exec();
throw new RuntimeException
("Locking error 2");
}
l.unlock();
while (counter != proc.execCounter);
if (!l.isLocked()) {
False
True
l.unlock();
throw new RuntimeException("Locking error 3");

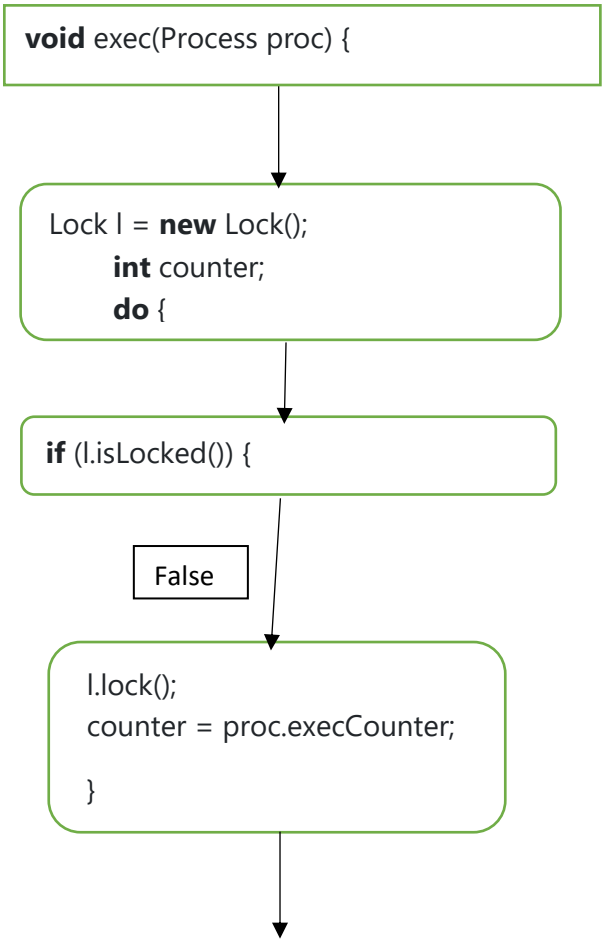
Question 1:

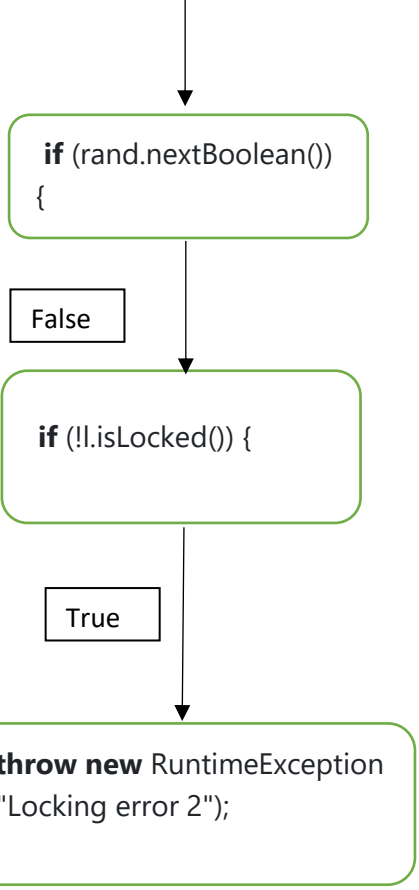
There are CFG paths that leads to throwing the exceptions "Locking error 1", "Locking error 2" and "Locking error 3". Their CFG path i sas follows:

Locking error 1

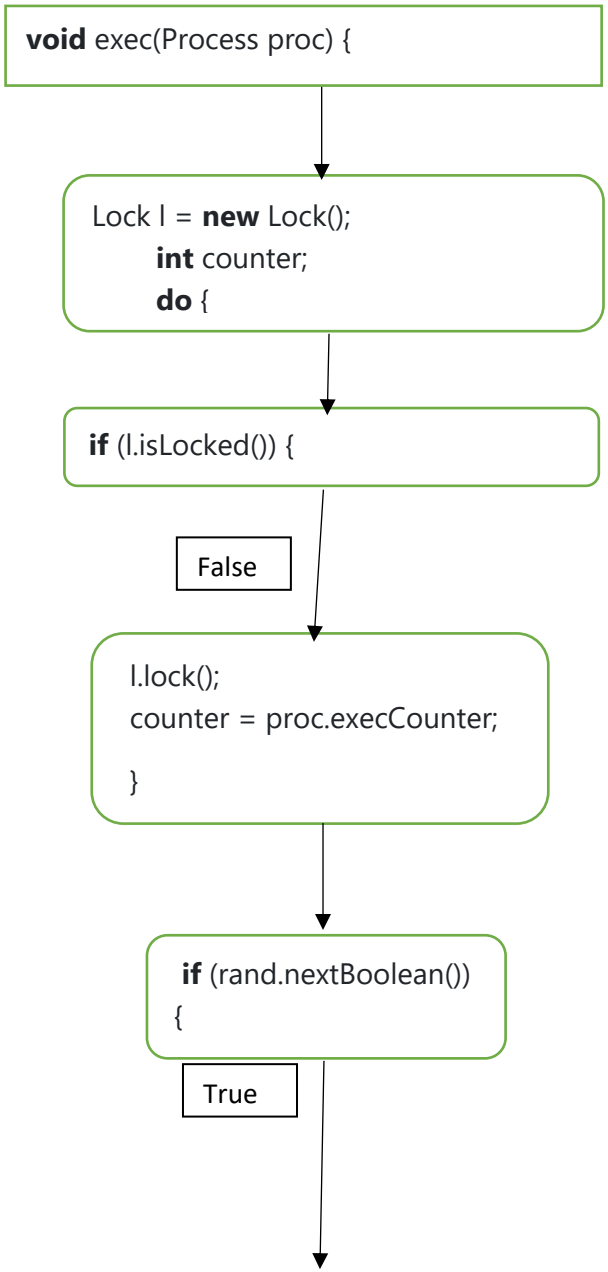


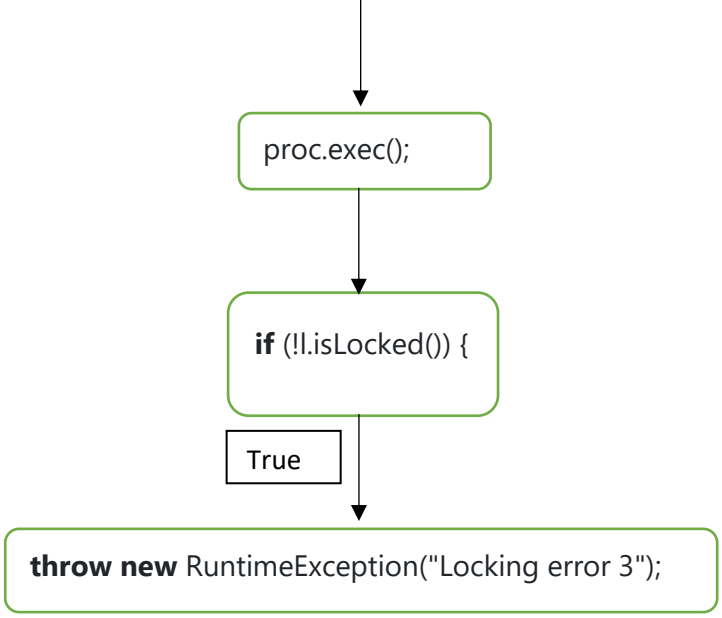
Locking error 2





Locking error 3





Question 2:

Considering the CFG paths of question 1 we can say that for both Locking error 1 and Locking error 2 it is possible to have executions that lead to them. This is not possible when we want to get a Locking error 3 because in order to get this type of error we should have the lock active after the ending of the do-while cycle which is not possible.