Srivenkatesh Nair (sn6711)
Abhinav Menon (am6176)

Homework 12.1

→ In line 27, we first create an anonymous object of class X and directly call the run method.

```
new X(0).run();
```

→ Within the run method, since initially the id of the anonymous object is 0. Hence, we enter we go into ~~tat~~ the if statement:

```
if (id==0) {
    new X(1).start();
    new X(2).start();
}
```

→ Here, we create two threads with id=1 & 2 and call start() method on both the threads

→ From here on, any of the three threads with id=0, 1 & 2 can enter the synchronized block.

→ Once, one of the three ~~three~~ threads enters the synchronized block, the other two threads wait outside the synchronized block as the key "o" is a static object i.e. there is only one key.

→ The thread which entered ~~at~~ will continue executing till it reaches "o.wait();".

→ Once it reaches wait(), ~~the~~ one of the two threads waiting outside the synchronized block

M T W T F S S
Page No.:
Date:
YOUVA

get the key i.e "static object o" which is assigned by the scheduler.

→ Now, one of the two threads enter the synchronized block.

→ In line 17, "o.notifyAll();" we notify all the threads which are in wait (Initially there is only one thread in wait).

→ Once, the second thread ~~wait~~ goes into wait there are 2 possibilities that can occur:

- Case ① : The ~~thre~~ second thread goes into wait and the third thread $\overset{\text{waiting}}{\text{outside}}$ the synchronized is provided with the key (static object o) by the scheduler, so that it can enter the synchronized block.
- Case ② : Once the second thread goes into wait, the scheduler also has the option to continue the execution of thread which was in wait.

☆ There are places in the code where you can put random sleep to get all the possible outputs:

①
```
public void run() {
if(id==0){
  new X(1).start();
  new X(2).start();
  }
try{ sleep((int)
 (10 * Math.random()));}
catch(Exception e){ }
synchronized(O){
      ⋮
```

②
```
public void run(){
        ⋮
  synchronized(O){
        ⋮
     o.wait();
  try{sleep((int)
  (10 * Math.random()));}
  catch(Exception e){ }
        ⋮
     }
}
```

- For Case ① there are 36 possible outputs

· Right arrow		Left arrow	
0--->	0--->	0<---	0<---
1--->	2--->	1<---	2<---
2--->	1--->	2<---	1<---
1--->	1--->	1<---	1<---
2--->	0--->	2<---	0<---
0--->	2--->	0<---	2<---
2--->	2--->	2<---	2<---
0--->	1--->	0<---	1<---
1--->	0--->	1<---	0<---

→ Since, we have 6 possible outputs for each left and right arrow.
→ Therefore, total no. of outputs = 6×6 = 36

- For Case ② there are 6 possible outputs. In all the outputs the program won't terminate as there will be one thread which will be in wait for infinite time as ~~it wo.~~ there are no other threads which can enter the synchronized block.

①	②	③	④	⑤	⑥
0--->	1--->	0--->	2--->	1--->	2--->
1--->	0--->	2--->	0--->	2--->	1--->
0<---	1<---	0<---	2<---	1<---	2<---
1<---	0<---	2<---	0<---	2<---	1<---
2--->	2--->	1--->	1--->	0--->	0--->

* All these outputs do not terminate.