Computer Science 441: Assignment 4

Name: Jack Yang UCID: 30062393

No, there is not going to be a race condition in my program. When a timeout occurs, the task will be executed to resend another packet that has already been initialized and stored in the TimerTask task object. In any consequent methods before the break, the shared data of the TimerTask is never changed until the correct ACK has been received and the task is cancelled.

So if the timer task occurs at the same time an ACK is received, the task will retransmit the last packet regardless even though it has already been successfully received at the server side, the server will drop it harmlessly. Also the ACK number is only accessed inside main so there is no shared data accessed between main and run at the same time.

```
timer.scheduleAtFixedRate(task, timeout, timeout);

while(true) {
    // Wait for ACK, when correct ACK arrives stop the timer
    clientSocket.receive(receivePacket);

    // get ACK from recieved packet
    reccSeg = new FtpSegment(receivePacket);
    reccSegNum = reccSeg.getSeqNum();
    System.out.println("ack     " + reccSegNum);
    if (reccSegNum == seqNum + 1) {
        task.cancel();
        break;
    }
```