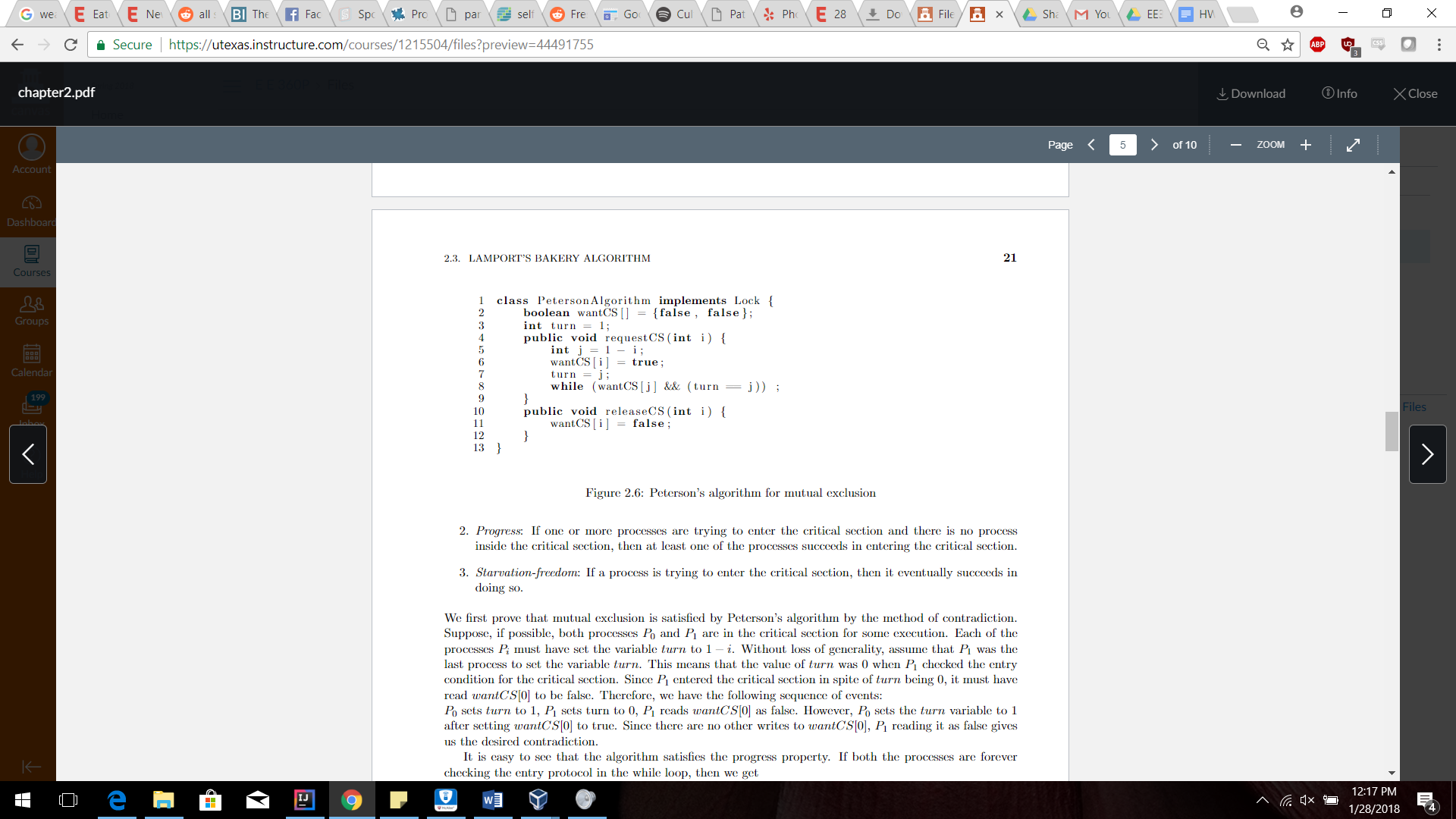
Kevin Chau, kc28535

Drew Poovey, ahp473

HW1

1. See source code on Canvas
2. See source code on Canvas
3. TACC UserID: kchau UserID: apoovey



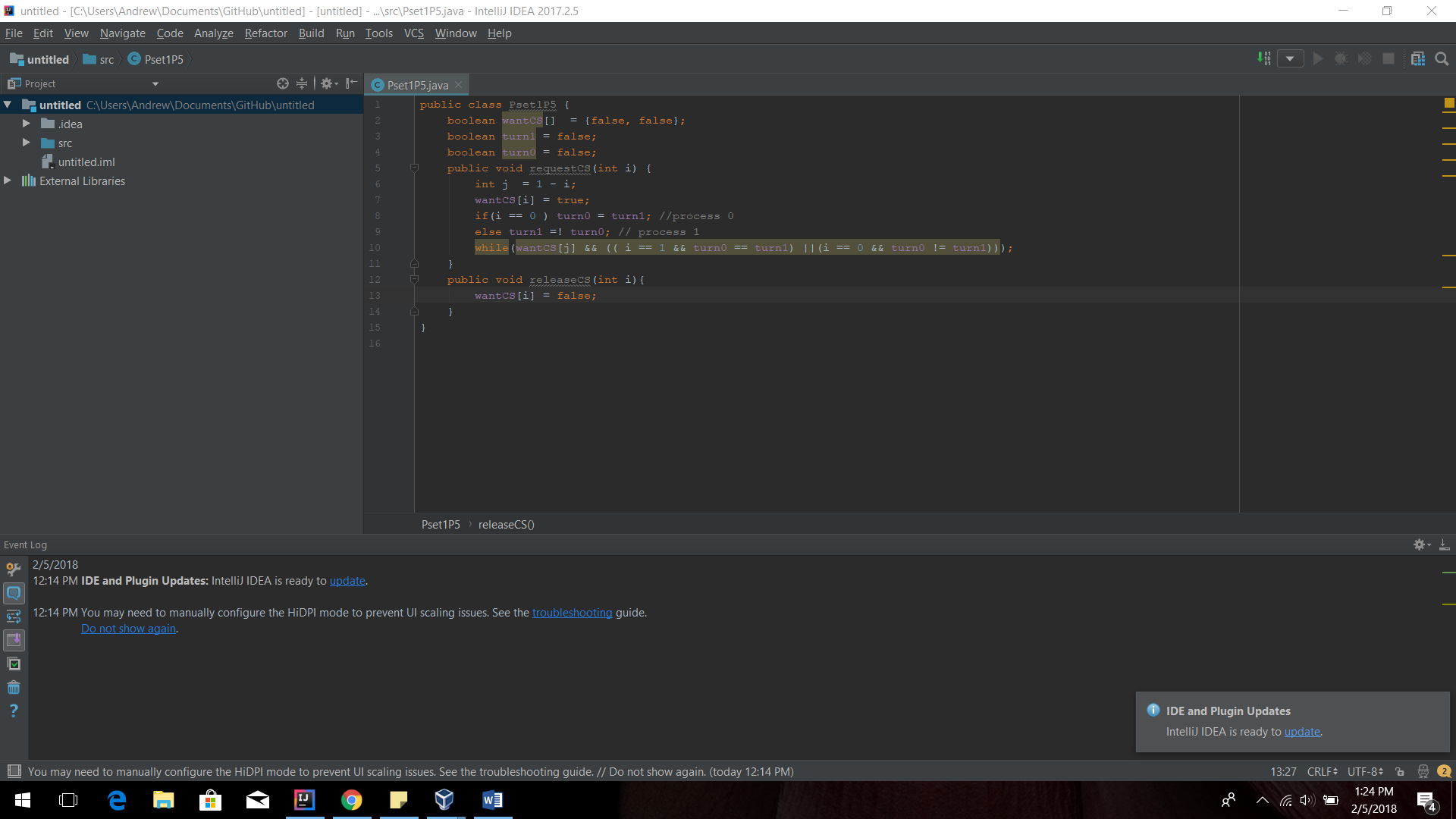
1. Incorrect Peterson’s Algorithm:
   1. Changing turn to self

Assume that the algorithm works when the turn changes to itself. Then *wantCS[0]=true* for P0 and also sets *turn=0*. P0 passes the busy-wait statement and enters the critical section. Then before P0 exits the critical section, P1 enters the *requestCS* method and sets its *wantCS[1]=true* and sets *turn* to itself (1). P1 also enters the critical section. Therefore this is a contradiction because both threads cannot be in the critical section at the same time.

* 1. Set turn before wantCS

Assume that the algorithm works when the turn is set before wantCS. The order of execution for the threads is the following: P0 sets *turn=1*, P1 sets *turn=0*, P1 sets *wantCS[1]=true*, P1 passes the busy-wait since *wantCS[0]=false*, then P0 sets *wantCS[0]=true* and also passes the busy-wait since *turn=0*. This would have P0 and P1 both skip the busy-wait and in the critical section. Therefore this is a contradiction because both P0 and P1 cannot be in the critical section at the same time.

1. Modified Peterson’s Algorithm:



1. Bakery Algorithm does not work without choosing variable
   1. Assume that the Bakery algorithm does not use choosing variables. Assume there are two processes (P0 and P1). P0 goes through the doorway and get the next largest variable. P1 also enters the doorway while P0 is checking for the largest number and gets the same numbers as P0. P0  enters the critical section after checking if it is the largest number. P1 then also enters the critical section since it is also the largest number. Therefore, there is a contradiction because two processes are able to enter the critical section when only one of them should be able to.