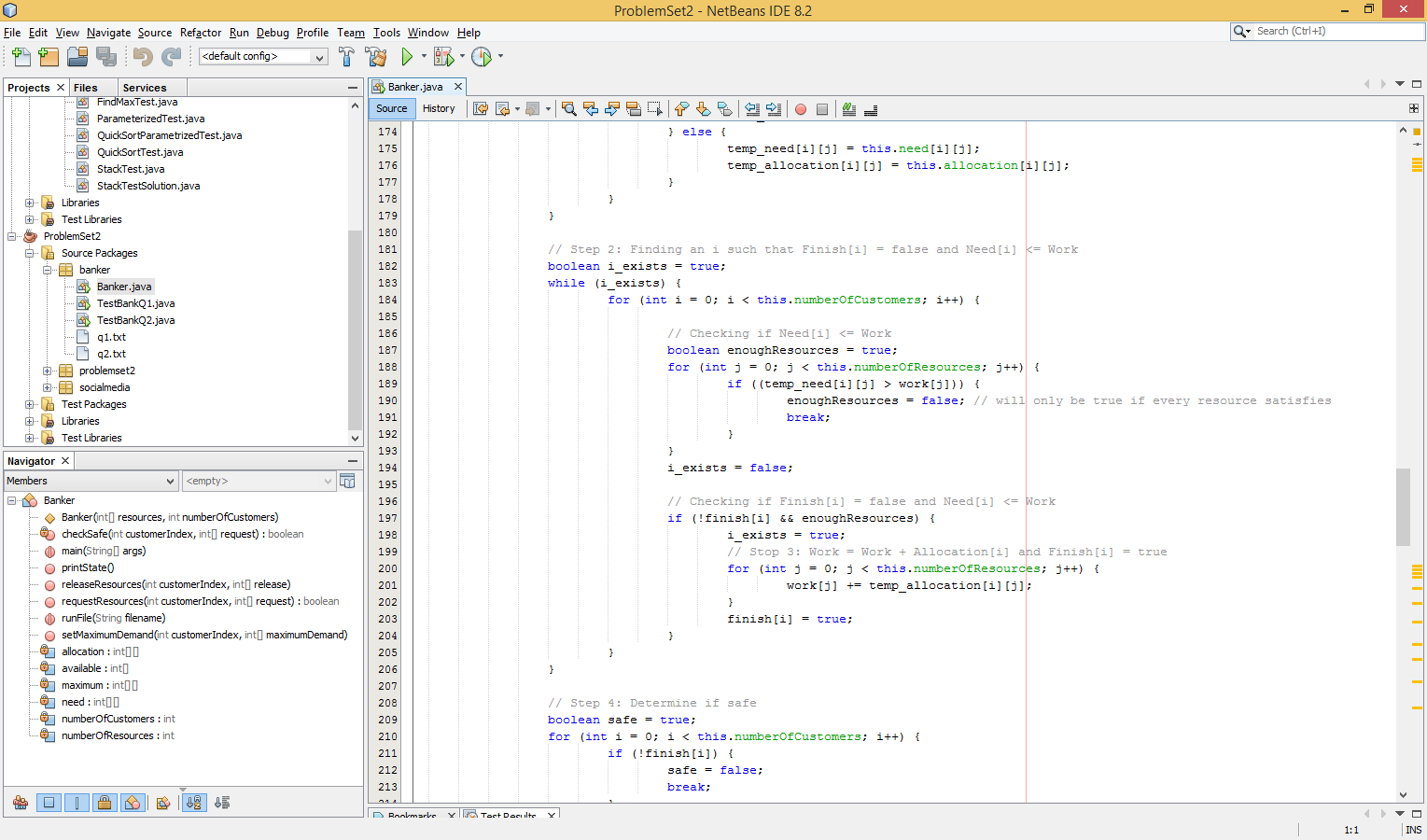
The Banker’s Algorithm:

1. Find a row in the Need matrix which is less than the available vector. If such a row exists, then the process represented by that row may complete with those additional resources. If no such row exists, eventual deadlock is possible
2. Pretend that the process has acquired all its needed resources, executed, terminated and returned resources to the Available vector. Now value of the Available vector should be greater than or equal to the value it was previously
3. Repeat steps 1 and 2 until:
   1. All processes have successfully reached pretended termination
   2. Deadlock as reached (initial state unsafe)

By following the algorithm above,

Looking at Step 2, the for loop from I = 1 to n, n being the number of customers is repeated n times. Furthermore, since the two for loops for j = 1 to m, m being the number of resources available, the time complexity contained within step 2 is O(2\* m\*n) = O(mn). (The reason why Step 3 is still considered O(m) is because this work = work + allocation[i] is also an O(m) operation. But O(m+m) = O(m), so the internal time complexity of this for loop is still O(nm)).

Since this for loop is nested in a while loop that only stops once i\_exist = false and iFinish = true and need[i] = work, the total time complexity is O(m\*n^2).

However, this while loop is called at line 118, nested beneath in the request resource method which is in turn determined by the number of lines. Therefore, total time complexity is O(m\*n^3)