Answer

If there are two threads T1 and T2, which simultaneously try to change the value of highestBid, then the race condition can occur.. Consider an example with values, where currentHighestBid is 100. T1 tries to place amount 150 as bid, while T2 tries to place 200 as the bid..

Now lets assume execution starts with thread T1 and T1 only executes the statement **if(amount > highestBid)**
This statement is true, as amount = 150 for T1, hence T1 assumes its condition is met and will set the highestBid. however before it could set the variable, the thread is preempted by OS and execution is handed over to Thread T2.. So right now current highestBid is still at 100.

now T2 starts execution and T2 checkes for **if(amount > highestBid)**
This statement is true again for T2, as amount = 200 for T2, hence T2 sets the highest bid to 200..

now execution is back to T1 and T1 tries to finish the remaining part of the code. as it has done the comparison before itself, hence it now just sets its amount(150) to the highest bid..
So **highestBid becomes 150 now..**

As we have seen, two thread place 150 and 200 as bids, and bid should have becomes 200, while here because of the race condition, the result comes out to be 150// which is wrong.

To prevent the race condition, The method should be marked synchronized and should be executed atomically.