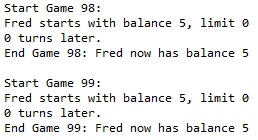
**Proof:**

We can see that when the player starts with a balance of 5 they are not able to play a round



**Hypothesis:**

By looking at the output we can see that the turn is not incremented, however it is set to 0 implying that the following while loop does not get entered.

**while** (player.balanceExceedsLimitBy(bet) && player.getBalance() < 200)

Most likely leaving us with the possibility that one of the methods in the while loop is returning the wrong answer. Given that balanceExeedsLimitBy(bet) is to deal with going below the limit this is most likely the cause.

Looking at the method balanceExceedsLimitBy(bet) the problem seems to be that (balance – amount > limit) this will never be true when balance is 5 amount is 5 and limit is 0 as it is saying (5 – 5 > 0) = (0 > 0) which is false.

**Replication utilising Junit Testing**



As expected the test fails. In order to solve the issues the notion (balance – amount > limit) will need to be changed to (balance – amount >= limit) resulting in (5 – 5 >= 0) = (0 >= 0) which will return true.

**After fixed code**

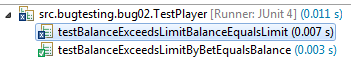


As expected the code now returns the correct result of true.

Further investigation of the Player class in relation to limit there is one more possible bug in the following method

BalanceExceedsLimit() this method has the same issue in that (balance > limit) will return false when balance = limit.

**Utilising Junit Testing**



When the player limit is set to the balance this returns false when it should return true. Modifying the code to read (balance >= limit) should resolve the bug.

**After fixed code**



As expected the extra bug has now been fixed.