| **Test Name** | | | Player is unable to win betting limit | | | |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case Tested:** | | | Crown and Anchor Game | | | |
| **Test Description:** | | | The betting limit of the game is set to 0, the game will terminate when the players balance is still 5 | | | |
| **Pre-conditions** | | | Default game setup | | | |
| **Post-conditions** | | | At the end of the game the players balance should be 5 | | | |
| **Notes:** | | **Because the game terminates if the players balance is 200, the only way to see the bug is to keep playing until the game has terminated on the balance of 5. A pass for the test will indicate that the bug does actually exist** | | | | |
| **Result (Pass/Fail/Warning/Incomplete)** | |  | | | | |
|  | **TEST STEP** | | | **EXPECTED TEST RESULTS** | P | F |
|  | Run program | | |  | X |  |
|  | Check the players balance once the game has finished | | | The players balance should be 5 | x |  |
|  |  | | |  |  |  |

# Test Result

Output from Game

Turn 114: Fred bet 5 on HEART

Rolled DIAMOND, DIAMOND, DIAMOND

Fred lost, balance now 10

Turn 115: Fred bet 5 on HEART

Rolled DIAMOND, DIAMOND, DIAMOND

Fred lost, balance now 5

115 turns later.

End Game 99: Fred now has balance 5

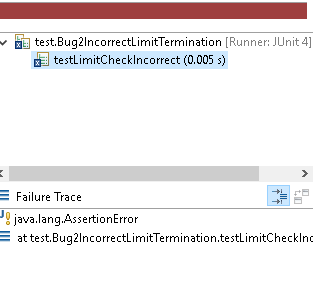
Win count = 1674, Lose Count = 6567, 0.20

Diagnose

From the above output we can see that the game is terminating to early. The game appears to be stopping once the players balance is 5 and not 0. This seems to indicate that the termination code is off, and is terminating on limit +bet amount (0+5)

# Automated Test with Junit to show the bug

The screenshot below shows the bug through an automated test. The test has been designed to fail if the bug is present. For the test to fail I forced the players balance to be 5 and the bet amount to be 5. The player should still be able to play one more game before the game ends. The test indicates that would not happen



# Tracing the code back to find the bug

From the Junit test the bug seems to be a result of the balanceExceedsLimitBy function. But before we look at what this function does let’s trace it back from Main.java. If we open Main.java and go to line 41 we can see that we have a while loop that has two checks, the first check is the balanceExceedsLimitBy and the second to check if the player has won 200. We know the bug isn’t from 200 as the game terminates on this correctly so that just leaves the balanceExceedsLimitBy function. Now let’s take a look inside this function and see what it is actually doing.

We can find this function on line 30 of Player.java at first glance the function appears to be correct it is a very simple function that checks the balance against the limit

**return** (balance - amount > limit);

The easiest way to check if this function is returning the correct values will be to place values into the variables and see what gets returned by hand. Let’s start with the test data that we used for Junit. For that test we had a balance of 5 and bet amount of 5 and a limit of 0. With that said that makes the function look like the following

**return** (5 - 5 > 0);

This gives us

**return** (0 > 0);

Because 0 is not greater than 0 the function returns false. This is actually incorrect as the player should still be able to play on as they have not reached the limit yet but are still able to play one more game. Just to confirm that this is the case let’s look at the function when we pass in a higher balance. For this we will say that we have a balance of 15, a bet amount of 5 and a limit of 0. This gives us the following

**return** (15 - 5 > 0);

This gives us

**return** (10 > 0);

Now because 10 is greater than 0 the function returns true. This is correct. So with that said the issue seems to be the check when balance is equal to bet amount, or 0 >0.

The solution to fixing this bug is actually very simple instead of check if balance –amount greater than 0 we can change the check to check if balance – amount is greater or equal to 0. Before we do a check to see if this would fix the issue in code let’s do it by hand. So this would now make the new function look like the following

**return** (balance - amount >= limit);

Again we will use the values balance of 5, bet amount 5, and limit of 0. So now the function looks like

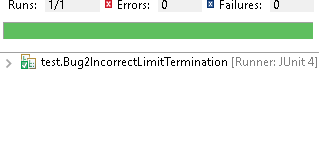
**return** (5 - 5 >= 0);

This gives us

**return** (0 >= 0);

Because 0 is equal to 0 the function will return true, thus given us one last chance to win

After correcting the code to reflect this change the Junit test now succeds, which is an indication that we fixed the bug. The screen shot below shows the successful test



| **Test Name** | | | Player is able to bet the remaining amount | | | |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case Tested:** | | | Crown and Anchor Game | | | |
| **Test Description:** | | | The game should only stop if the player is unable to bet there remaining amount | | | |
| **Pre-conditions** | | | Default game setup | | | |
| **Post-conditions** | | | At the end of the game the players balance should be 5 | | | |
| **Notes:** | | **Because the game terminates if the players balance is 200, the only way to see if player can bet the remaining amount is to keep playing until the game has terminated on the balance of 0. pass for the test will indicate that the bug is fixed** | | | | |
| **Result (Pass/Fail/Warning/Incomplete)** | |  | | | | |
|  | **TEST STEP** | | | **EXPECTED TEST RESULTS** | P | F |
|  | Run program | | |  | X |  |
|  | Check the players balance once the game has finished | | | The players balance should be 0 | x |  |
|  |  | | |  |  |  |

# Test Result

The game is now terminating on the correct limit, as the below output demonstrates

Turn 388: Fred bet 5 on CLUB

Rolled ANCHOR, CROWN, CROWN

Fred lost, balance now 5

Turn 389: Fred bet 5 on DIAMOND

Rolled ANCHOR, CROWN, CROWN

Fred lost, balance now 0

389 turns later.

End Game 99: Fred now has balance 0

Win count = 11462, Lose Count = 17463, 0.40