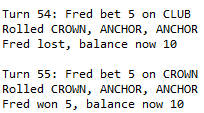
**Proof:**

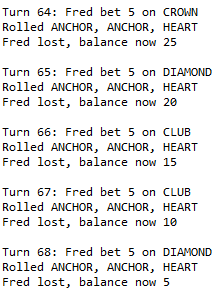
We can see in the below figure that on turn 54 the player Fred had a balance of 10, then on turn 55 he bet on Crown which was rolled, however his balance is still set at 10.



**Hypothesis:** There are a number of different areas where the bug could be occurring, Appropriate Junit tests / Debug Proof will be setup to test each one.

|  |  |  |
| --- | --- | --- |
| Theory | Junit Test / Debug Proof | Result |
| The players balance is not being set correctly |  | pass |
| The receiveWinnings method is not functioning correctly | pass |
| The playRound method is not functioning correctly (**NOTE:** This method was incorrect for testing and was modified with the correct calculations) | (incorrect test)  (correct test) | Pass (incorrect result)  Failed (correct result) |

Seeing that the three main theories are working correctly there might be something wrong with the integration. Further inspection of the program output I noticed that for every single turn the same dice are rolled. Now it could be possible for it to happen maybe once, however the below figure shows that it is happening all the time.



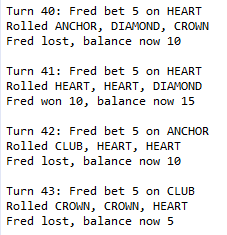
With Anchor, Anchor and Heart always being rolled for this snippet. This indicates there is potentially a bug with the dice list. An integration test will need to be performed.

|  |  |  |
| --- | --- | --- |
| Theory | Junit Test | Result |
| The list of dice are not modified even after playRound is called. Test run 3 times to attempt to avoid potential same rolls. |  | confirmed |
| Check to see that Dice.roll() is being called from within Game.playRound() |  | Pass |

Now knowing that roll is being called and inspecting the Dice class it is clear that the value inside Dice is never modified.

Modifying the roll method to modify the value and then return the value should solve the output issue.

Running the program now outputs a different Rolled value for each turn



With the above modification testPlayRoundDiceListNotModified correctly fails



It will now be updated to check that the list has been modified.

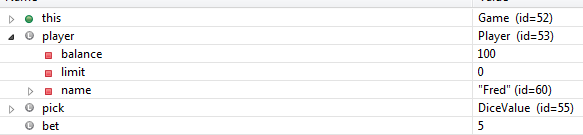


Returning to the original issue it occoured to me that whilst playRound appeared to be working correctly in a unit test this is because the Player.takeBet() method was only being called on a mocked object.

|  |  |  |  |
| --- | --- | --- | --- |
| Theory | Expected result | Junit result | Result |
| When doing an intergration test on game.playRound() it will fail as the player balance is not increased | Player.getBalance() after game.playRound() does not get moddified correctly |  | Confirmed |

The issue is within the playRound method by debugging the method I was able to find the issue

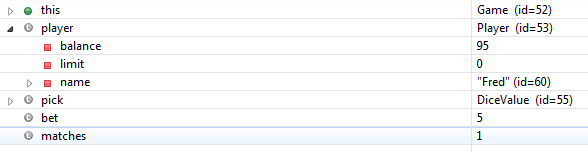
The players starts with their default balance and bet



The player then is deducted their bet amount from their balance



The amount of matches are correct



However the winnings are not correct with only a value of 5 where it should be 10



The winnings are correctly added to the player



In order to fix this bug the bet amount will need to be added to the winnings amount as well as the multiplication of the number of matches. In order to ensure that this is done in the correct scope this should be calculated inside the (matches > 0) conditional with winnings being initilised to 0.

|  |  |  |  |
| --- | --- | --- | --- |
| Theory | Expected result | Junit result | Result |
| When doing an intergration test on game.playRound() it will calculate the winnings correctly | Player.getBalance() after game.playRound() provides the correct amount |  | pass |

Going back and checking the unit testing I did on game.testPlayRound() I realised that my calculation was incorrect but this test has now been updated.

The bug has now been resolved.