# Debugging the Crown and Anchor Game

***Bug 1: Game doesn’t pay out correct level***

**Hypothesis 1**: May be wining is not calculated properly.

**Experiment:** Inspecting through the classes, I have come to understand the Game class have playRound method which returns the winning, according to matches against user pick. May be the method is not returning right winning amount.

**Test Output:**

There are four conditions to test, in order to ensure that the method doesn’t have bug.

Winning is calculated through bet multiplied by matched numbers.

If there is no match between user picked symbols and system selected symbols then bet multiplied zero which will be zero

If there is one match then bet multiplied by 1

If there is two match then bet multiplied by 2

If all, three match then bet multiplied by 3

Below is the test screenshot from JUnit test, testing these conditions

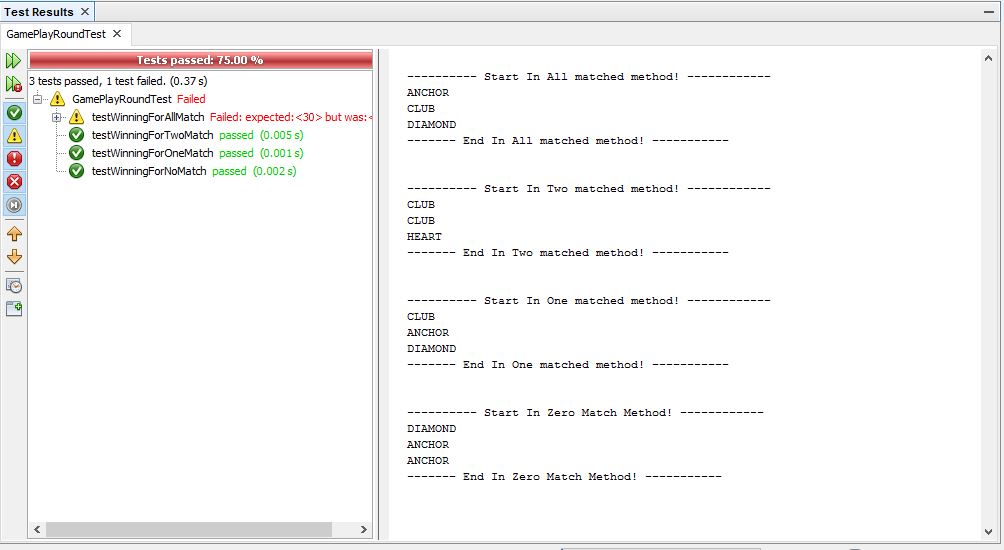


Figure 1: Game class, playRound method test screenshot

As we can see, I had picked and hard coded Club as a dice value, and we can see that, in two matches the winning is same as the expected result, which results in test being passed, and similar cases in one match and zero match, but in all match since there is only one match, which is only one club, that’s why test failed.

**Conclusion:** Since methods are returning correct value, there is no error in this method, thus the hypothesis is wrong.

**Hypothesis 2:** Maybe bet taken from user is not added back to user balance

**Experiment:** While inspecting the playRound method, in Game class, I have noticed that at first bet is being deducted from the user balance and then user picked symbol is matched against the system selected symbols, but if the user wins the match the bet amount is not added to the user balance only returning winning amount. May be there is the error.

**Test Output:**

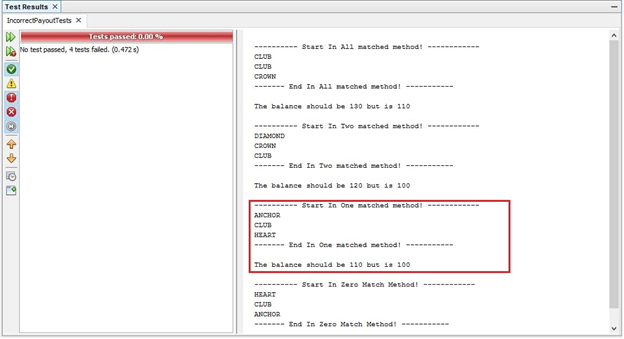


Figure 2: Check player balance after payout test

As we can see in the one match, being matched club exactly one times, the user initial balance was 100 which means after winning the match, user should have 110 but instead it is 100, which means bet taken from the user is not returned to the user.

**Conclusion:**

Hence, I can conclude that the error is in Game class, playRound method, in which user bet is taken from the user balance but upon winning the bet is the added to the balance, and this is the state where the sane to infected transition occurs.

Furthermore actual game error screenshot is below:-

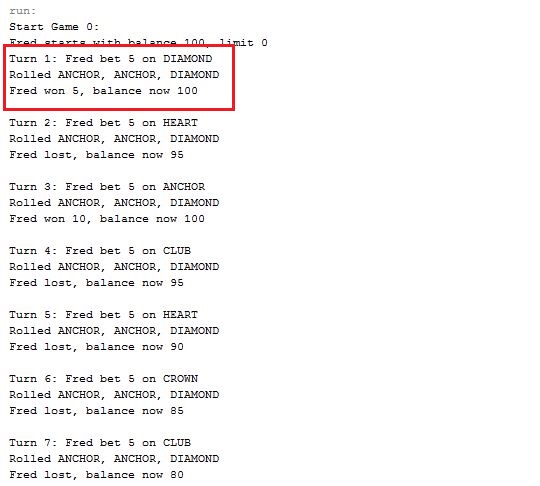


Figure 3: Actual Game payout error

In the highlighted are, as we can see the player Fred , has balance of 100 and won 5, but the balance is still 100.

**Bug 2: Player cannot reach betting limit: Limit set to 0, but game ends with player still with 5 (dollars) remaining.**

**Hypothesis:** Maybe betting limit is incorrectly compared against the player balance.

**Experiment:** Balance should have 0 at the end of the match but it exits early, while still having left 5, which is incorrect. While inspecting the Main class and Player class, balanceExceedsLimitBy, which deducts the bet amount from the balance and checks with limit and ensure that the player balance isn’t negative. May be the bug is incorrect use of comparison operator between player balance and limit.

**Test Output:**

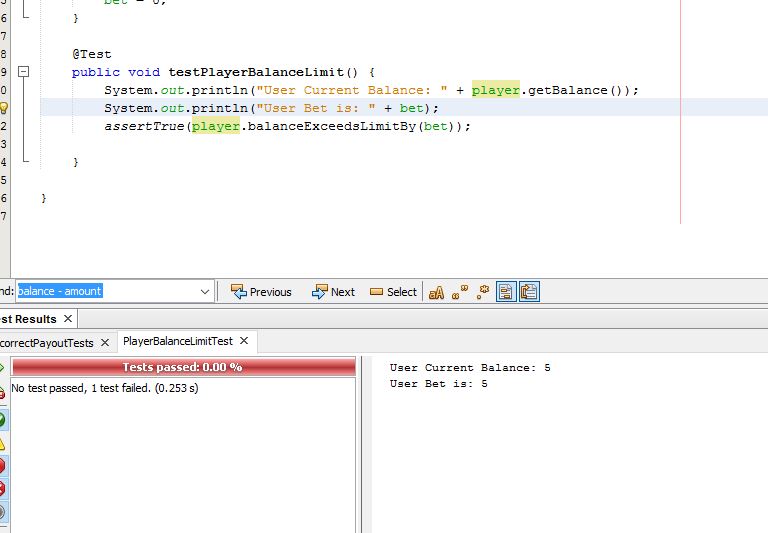


Figure 4: Player balance limit test

As we can see, even though the user balance is 5 and the user bet is 5, player should have got the last chance to play the game, but instead the method returns false, which means user cannot play game and user balance is 0 or less than that, which is not the case.

**Conclusion:**

Hence, the hypothesis is correct and the program transition from sane to infectious state, with incorrect comparison between balance and limit.

Furthermore, Actual game error is below:-

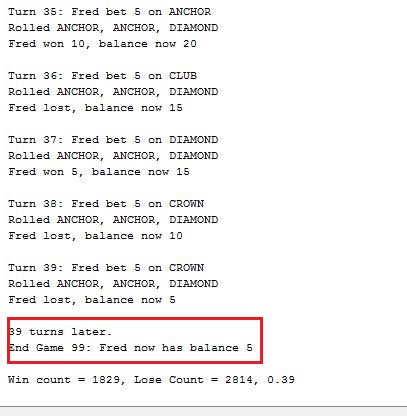


Figure 5: Player cannot reach limit error

As we can see user has still 5 balance left.

**Bug 3: Odds in the game do not appear to be correct. Crown and Anchor games have an approximate 8% bias to the house. So the win : (win+lose) ratio should approximately equal 0.42. This does not appear to be the case.**

**Hypothesis:** At the end of the game, the win/lose ration is not equal to 0.42, but it is random sometimes, it is .40 and sometimes 0.49 not what it should be.

**Experiment:** At this moment I am not sure why the game is behaving differently, may be fixing the other bugs, from giving player right payout, letting player play up to the limit, will automatically solve the problem, I am not sure, but for now I will skip this test and move on to another test, and give update if it is fixed.

**Test Output:**

The actual error is below in screenshot

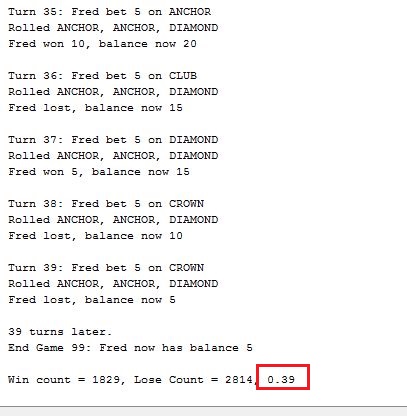


Figure 6: Error in Win/Lose ratio

.

As we can see, the value is 0.39, should be 0.42;

**Bug 4: Combination of three symbols never changes, first combination of symbols is repeated through the whole game.**

**Hypothesis:** Rolled dice Value is not update each time dice is rolled.

**Experiment:** The game should have different symbols in each turn, but while playing game, first turn dice values are used for the entire program, but the payout is not matching the rolled dice value, for example: if the displayed three dice values were Crown, Club and Anchor and user has picked the Anchor, then user should have won, but the prompt is player lost, which means the balance payout working fine, only dice value is not update after rolled each time. Since in Main class uses Game class getDiceValues to get the rolled dice values, and the this method uses Dice class getValue, for that reason I think the bug is in Dice class getValue not being updated.

**Test Output:**

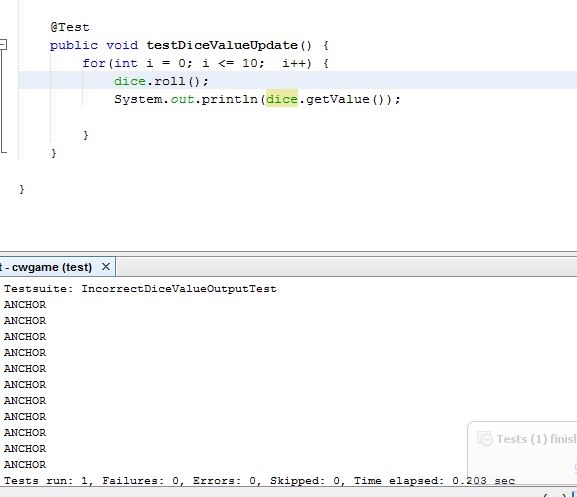


Figure 7: Incorrect Dice Value Update Test

As we can see in the screenshot above that, even though I looped the dice ten times, and each time I call roll, the value is still same.

**Conclusion:**

The hypothesis was correct, the value is not being update in Dice class, thus the values never changes in each turn.

Furthermore actual game error is below:-

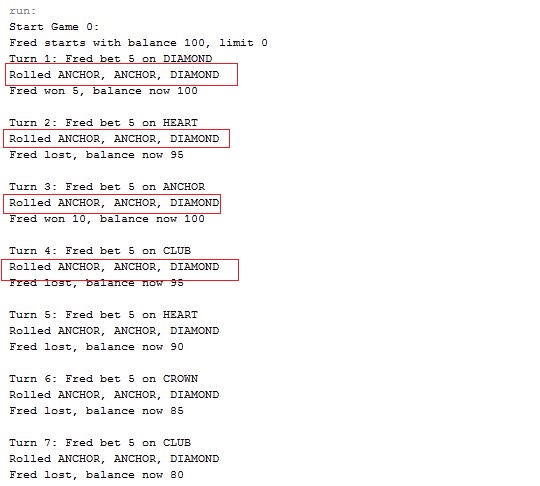


Figure 7: Dice Value doesn't updates error

As we can see below the symbols never changes.

**Bug 5: Spade is not selected.**

**Hypothesis:** Spade is not select out of six symbols

**Experiment:** As through inspection in code, I came to noticed that spade is not select out of six symbols, because of use in Random nextInt method is taking parameter DiceValue.Spade.ordinal() in DiceValue class, which means except spade, select others.

**Test Output:**

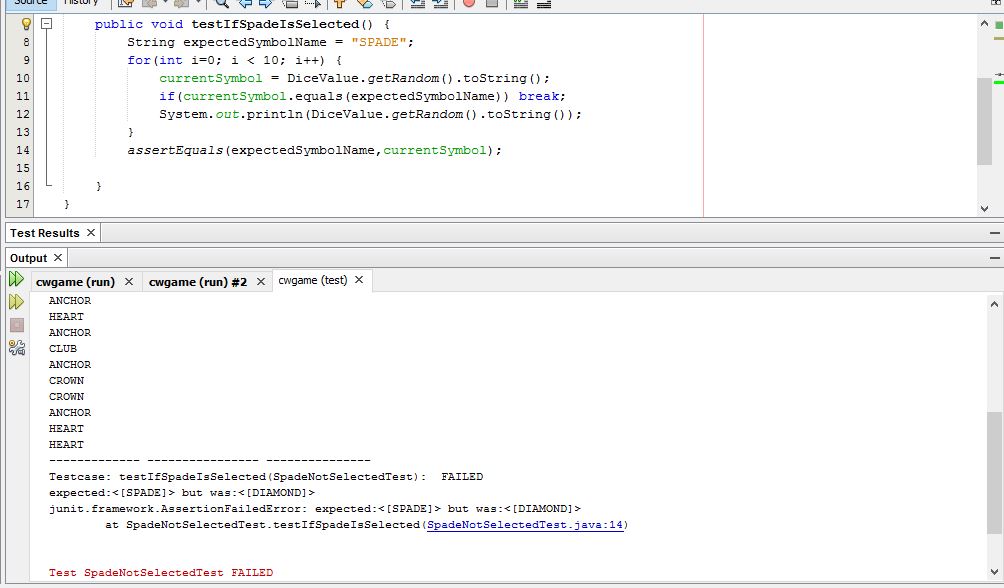


Figure 8: Spade Not Selected Test case

As we can see the all of the five symbols are there but spade is not there.

**Conclusion:**

The hypothesis was right, and spade is not select and sane to infectious transition is it in DiceValue class.

# Resolving the Bug in Crown and Anchor game

**Bug 1: Game does not pay out at correct level.**

**When player wins on 1 match, balance does not increase.**

To fix the incorrect payout bug, the actual error was, when user bet was deducted from the player balance, but not returned when player won, to fix the bug, bet is added while calling Player class recieveWinnings in Game class playRound method.

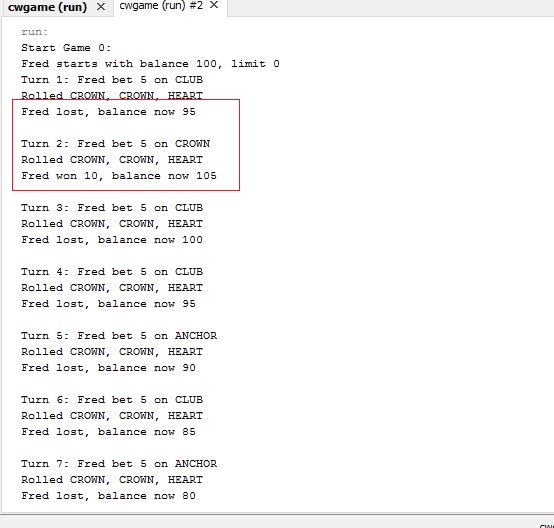


Figure 9: Bug fix for incorrect payout

As we can see, that balance was 95 and player won 10 and the balance is now 105, which means the bug is fixed for incorrect payout.

**Bug 2: Player cannot reach betting limit: Limit set to 0, but game ends with player still with 5 (dollars) remaining.**

The bug for not reaching betting limit, is in Player class, in balanceExceedsLimitBy, when comparing balance and limit. The bug is fixed using correct logical operator.

**Bug 3: Odds in the game do not appear to be correct. Crown and Anchor games have an approximate 8% bias to the house. So the win : (win+lose) ratio should approximately equal 0.42. This does not appear to be the case.**

**Bug 4: Combination of three symbols never changes, first combination of symbols is repeated through the whole game.**

**Bug 5: Spade is never drawn, out of six possible symbols, all five symbols are randomly select except spade**