Bug 01 Log

What's Wrong?

Game does not pay out at correct level. When player wins on 1 match, balance does not increase.

Additional Observations

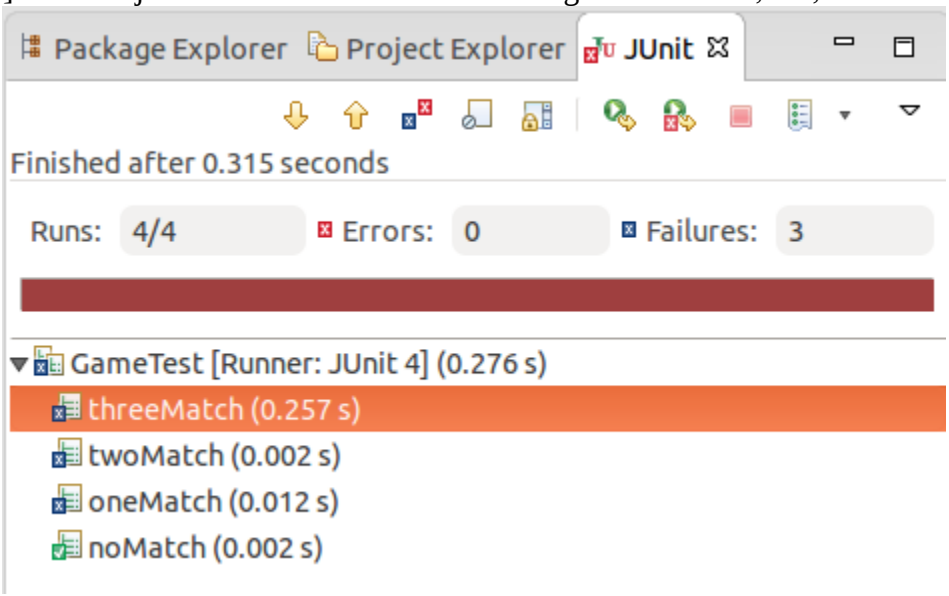
- Seemingly when a player wins one match with a payout of 2:1, the balance increases at 1:1
- Seemingly when a player wins one match with a payout of 3:1, the balance increases at 2:1

Initial Hypothesis

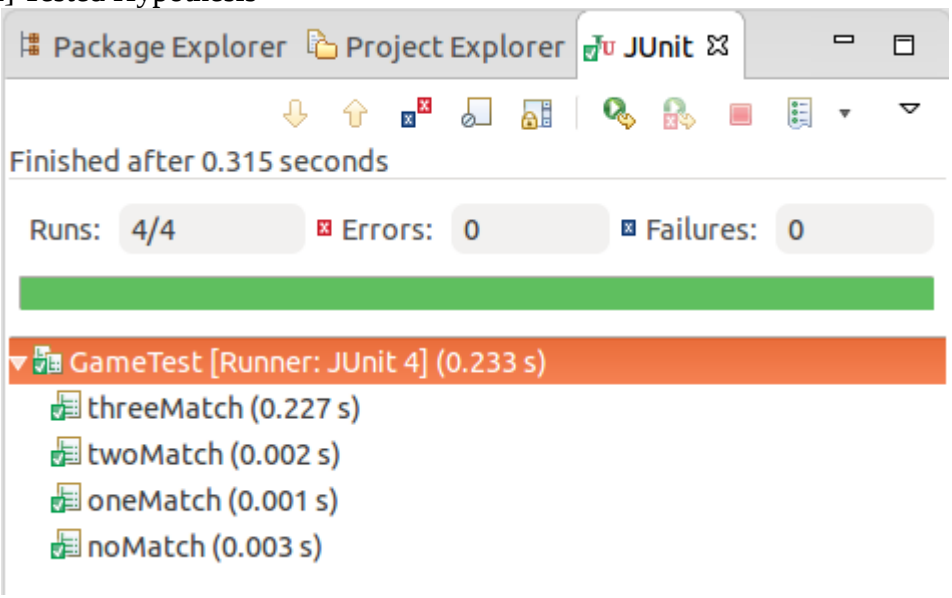
- The math being used to calculate the balance after each round is not taking the bet amount into account.

Process

1. [Action] Read the code to familiarize oneself with methods and what they do.
2. [Observation] The ongoing balance is calculated using the `receiveWinnings function`
3. [Observation] The calculation for `winnings` appears to be correct
4. [Observation] The calculation for receiveW`innings` appears to be correct
5. [Observation – New Hypothesis] The takeBet function appears to subtract the bet from the balance irrespective of whether the player wins or loses.
6. [Action] Define a test case in UAT format that will reliably reproduce the buggy behaviour. Commit the test case.
7. [Action] Define a test script in UAT format that will reliably reproduce the buggy behaviour. Commit the test script.
8. [Action] Create a jUnit test for each of the following scenarios 1:1, 2:1, 3:1.



9. [Action] Save StackTrace Output for examination.
10. [Observation] Player.takeBet and Player.recieveWinnings are both behaving as they should
11. [New Hypothesis] If takeBet and recieveWinnings are enacted using an if/else statement then the bug will be fixed.
12. [Action] Tested Hypothesis



13. [Observation] It worked.

Bug 02 Log

What's Wrong?

The game ends prematurely when the player's balance is near zero but still greater than the betting limit.

Initial Hypothesis

- It's likely that the game is doing a greater than comparison between the balance and the bet or between the balance and the limit, instead of a greater than or equal to comparison.
- Hypothesis → if the balance and the bet are equal, the game ends... yes or no?

Process

1. [Action] Create a UAT that recreates the bug.
2. [Simplification] Create a test scenario where the bet and the balance are isolated and a single round is played at a time.
3. [Action] Test to see if the test scenario can be played when the balance exceeds the bet.

```
    // play round
    Game game = new Game (die1, die2, die3);
    int bet = 5;
    int balance = 10;
    DiceValue pick = DiceValue.HEART;
```

Finished after 0.293 seconds

Runs: 1/1 Errors: 0 Failures: 0

TestBug02 [Runner: JUnit 4] (0.196 s)

4. [Observation] It passes.
5. [Action] Test to see if the test scenario can be played when the balance and the bet are equal.

```
    // play round
    Game game = new Game (die1, die2, die3);
    int bet = 5;
    int balance = 5;
```

Finished after 0.295 seconds

Runs: 1/1 Errors: 0 Failures: 1

TestBug02 [Runner: JUnit 4] (0.251 s)

testBetEqualsBalance (0.251 s)

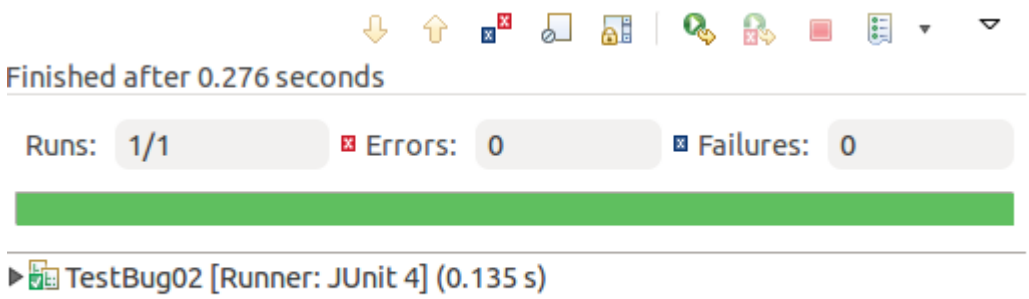
6. [Action] The test fails. Save stack trace output.
7. [Observation] The method in question is `player.balanceExceedsLimitBy()`

```
public boolean balanceExceedsLimitBy(int amount) {
    return (balance - amount > limit);
}
```

8. [Narrow Hypothesis] The bug occurs when the balance minus the bet equals zero and the limit equals zero, so the game ends.
9. [Action] Test new hypothesis by changing the balanceExceedsLimitBy method.

```
anceExceedsLimitBy(i
- amount >= limit);
```

10. [Action] Rerun automated test.



11. [Observation] It worked. Bug resolved.

Bug 03 Log

What's Wrong?

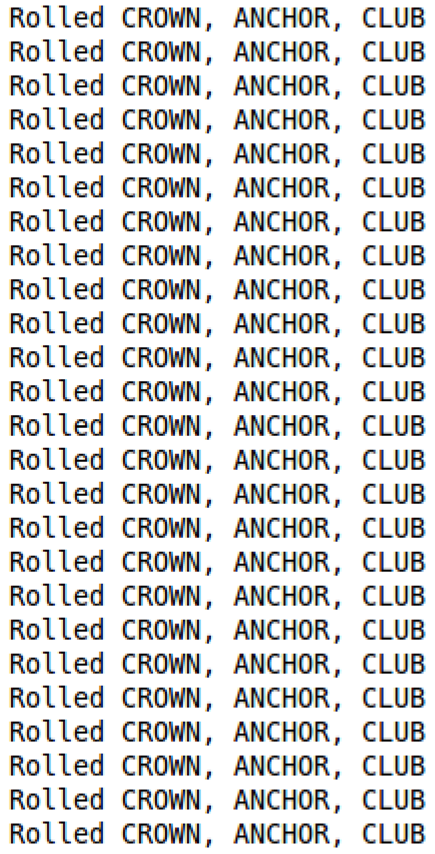
Crown and Anchor games have an approximate 8% bias to the house. So the win / (win+lose) ratio should approximately equal 0.42. This does not appear to be the case.

Initial Hypothesis

- It appears as if the dice results are not random. I believe this would skew the results.
- Hypothesis → the randomness of the dice results is skewing the win count... yes or no?

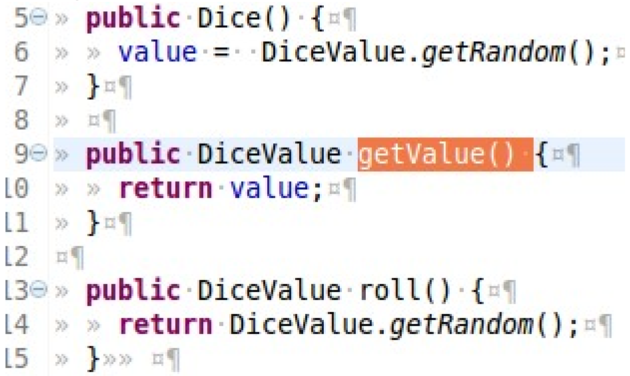
Process

1. [Observation] The dice results appear to repeat themselves in a non-random way.
2. [Question] But first let's check that the win ratio is being calculated correctly as-is. i.e., is the math behind `winCount, loseCount, (float) winCount/(winCount+loseCount))` behaving correctly?
3. [Observation] After inspecting the code in Main.java it appears that both winCount and loseCount are incrementing correctly and the ratio appears to be calculating correctly.
4. [Observation] If the math calculations are sound, then the randomness of the game itself is now suspect. (see initial observation above).
5. [Action] Create a UAT to recreate the bug.
6. [Action] Create automated test to recreate bug.
7. [Observation] It is curious that not only are the dice results NOT random, they appear to be the same round after round. Is the problem with the sequential dice role or the initial randomness itself?



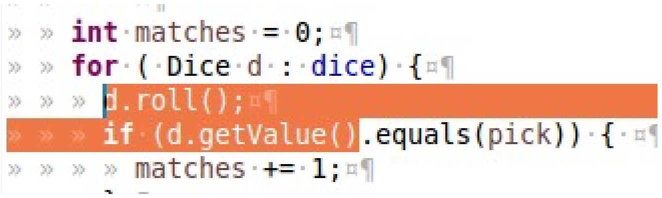
```
Rolled CROWN, ANCHOR, CLUB
Rolled CROWN, ANCHOR, CLUB
Rolled CROWN, ANCHOR, CLUB
Rolled CROWN, ANCHOR, CLUB
Rolled CROWN, ANCHOR, CLUB
Rolled CROWN, ANCHOR, CLUB
Rolled CROWN, ANCHOR, CLUB
Rolled CROWN, ANCHOR, CLUB
Rolled CROWN, ANCHOR, CLUB
Rolled CROWN, ANCHOR, CLUB
Rolled CROWN, ANCHOR, CLUB
Rolled CROWN, ANCHOR, CLUB
Rolled CROWN, ANCHOR, CLUB
Rolled CROWN, ANCHOR, CLUB
Rolled CROWN, ANCHOR, CLUB
Rolled CROWN, ANCHOR, CLUB
Rolled CROWN, ANCHOR, CLUB
Rolled CROWN, ANCHOR, CLUB
Rolled CROWN, ANCHOR, CLUB
Rolled CROWN, ANCHOR, CLUB
Rolled CROWN, ANCHOR, CLUB
Rolled CROWN, ANCHOR, CLUB
Rolled CROWN, ANCHOR, CLUB
Rolled CROWN, ANCHOR, CLUB
Rolled CROWN, ANCHOR, CLUB
```

8. [Observation] `getValue()` returns the dice value but doesn't initiate a reroll of the dice.



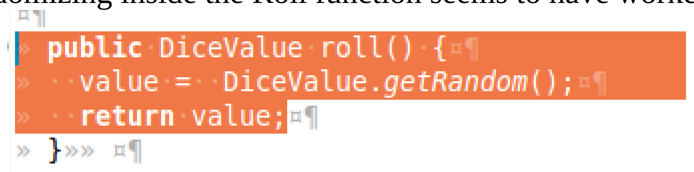
```
	public Dice() {
		value =  DiceValue.getRandom();
	}
	
	public DiceValue getValue() {
		return value;
	}

	public DiceValue roll() {
		return DiceValue.getRandom();
	}
```

9. [Question] Is Game.java initiating a reroll or consistently calling getValuewithout a reroll?



```
		int matches = 0;
		for ( Dice d : dice) {
			d.roll();
			if (d.getValue().equals(pick)) {
				matches += 1;
```

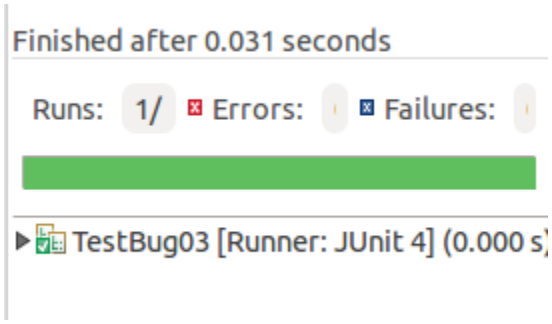
10. [Observation] I am not sure how to create a test to figure this out. Falling back to trial and error.
11. [Action] Re-Randomizing inside the Roll function seems to have worked.



```
	public DiceValue roll() {
	  value =  DiceValue.getRandom();
	  return value;
	}
```

```
Rolled DIAMOND, CROWN, HEART
Rolled HEART, CROWN, CROWN
Rolled ANCHOR, CROWN, CLUB
Rolled DIAMOND, CLUB, CLUB
Rolled ANCHOR, HEART, HEART
Rolled CLUB, CLUB, CLUB
Rolled DIAMOND, CROWN, ANCHOR
Rolled CLUB, DIAMOND, ANCHOR
Rolled DIAMOND, ANCHOR, CROWN
Rolled CLUB, DIAMOND, HEART
Rolled ANCHOR, HEART, CLUB
Rolled CROWN, DIAMOND, DIAMOND
Rolled HEART, CROWN, ANCHOR
Rolled DIAMOND, CLUB, CLUB
Rolled HEART, CROWN, DIAMOND
Rolled ANCHOR, CLUB, CROWN
Rolled CLUB, DIAMOND, CLUB
Rolled CROWN, HEART, DIAMOND
Rolled HEART, CROWN, DIAMOND
Rolled CROWN, HEART, HEART
Rolled HEART, DIAMOND, HEART
Rolled ANCHOR, ANCHOR, CLUB
163 turns later.
End Game 99: Fred now has balance 200
```

12. [Question] But does this solve the ratio problem?
13. [Action] Rerunning the initial test to find out.



14. [Observation] Ratio is still incorrect.
15. [Observation] Ratio appears to be consistently in the .48-.49 range.
16. [Action] Changing the parameters on the test to find if this is consistent.
17. [Observation] The test is producing false negatives – the test itself might be broken.