Hypothesis & Testing – Bug 01

# Bug Reported

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

Contents

[Bug Reported 1](#_Toc432782182)

[Analysis 2](#_Toc432782183)

[Localising the bug 3](#_Toc432782184)

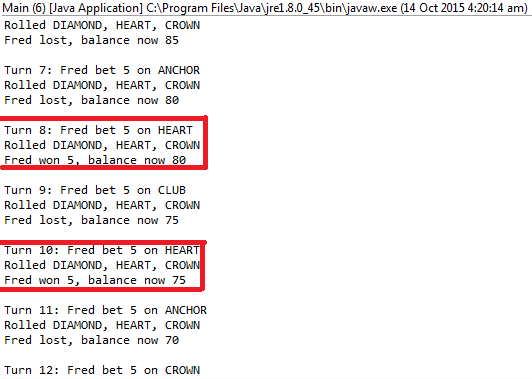
[Hypothesis 4](#_Toc432782185)

[Results 5](#_Toc432782186)

[Conclusion 6](#_Toc432782187)

# Analysis

Initial inspection of program reveals similar behaviour as to what was described in the bug report. The balance was not being increased when the player won.



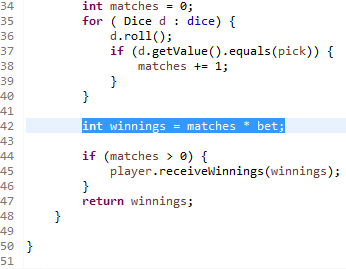
Balance should have increased, due to winning

Additional runs reveal the same results, the bug is consistent throughout all runs.

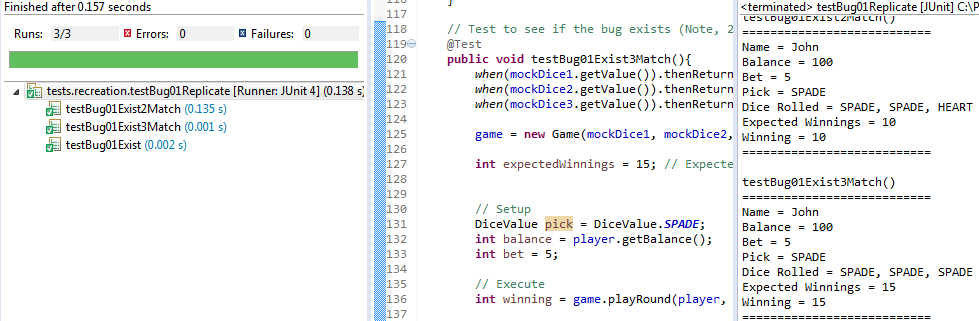
# Localising the bug

The bug is related to calculating winnings/passing winnings between classes and altering the players balance. Closer inspection reveals that only Game and Player.java should be inspected.

Code analysis reveals that the bug is likely within Game.java. Further code analysis reveals that one methods is likely to produce the bug:



Automated test reveals that the bug is created in this method. See Bug01Replicate.java for test.



The above image is testing for if the bug exists, the bugs exist, so the tests pass. The above example is also proof of localising and recreating the bugs.

When testing if the code is fixed (separate test), if a bug exists, the tests will not pass.

# Hypothesis

An error in the code is creating the bug. In Game.java, the process is as follows for playRound method:

1. Method receives player object, dice value object, and int bet. These are checked for errors
2. The ‘bet’ amount is deducted from players balance
3. The die are rolled. If any of the 3 die match the players DiceValue picks, the value of matches is increased by 1
4. **Winnings are calculated by multiplying the matches by the bet.**
5. If value of ‘matches’ is greater than 0, then the player will receive their winnings
6. Return ‘winnings’

Step 4 is the suspected cause of the bug, the winnings calculated in the method are simply what has been won from the roll alone. The winnings should consist of the money won from the roll, plus the players bet itself.

Replication has revealed that if only 1 match occurs, then the players balance does not appear to increase, the reason is suspected that the matches(value of 1) is multiplied by the bet(value of 5), which equals the winnings (value of 5). This is then returned as the winnings and gets added to the players balance. If the bet had have been added to the winnings, which is what SHOULD happen, the bet (value of 5) would be added to the calculated winnings (value of 5) before it is passed to the player object. This would result in the players balance increasing normally.

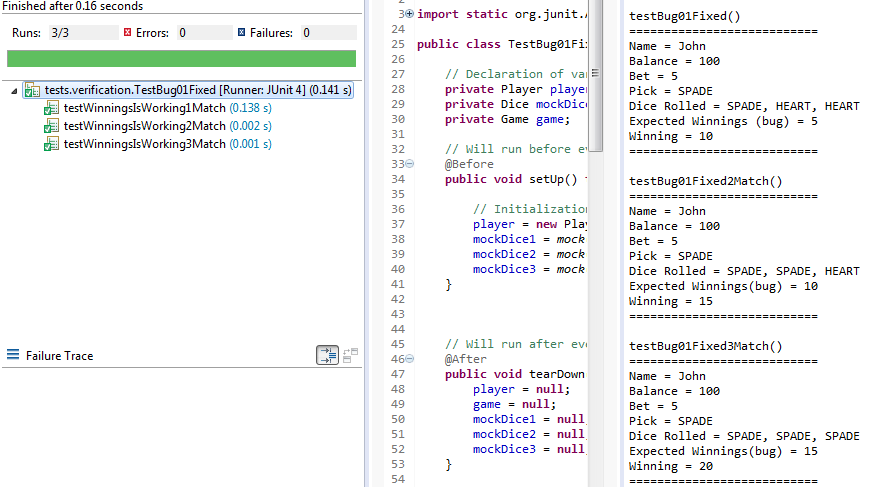
Replication has revealed that if 2 or matches occur, the balance will appear to increase (due to the matches being higher, and then being multiplied by the bet). However the bet is not added to the calculated winnings, and so still adds an incorrect amount of money back into the players balance.

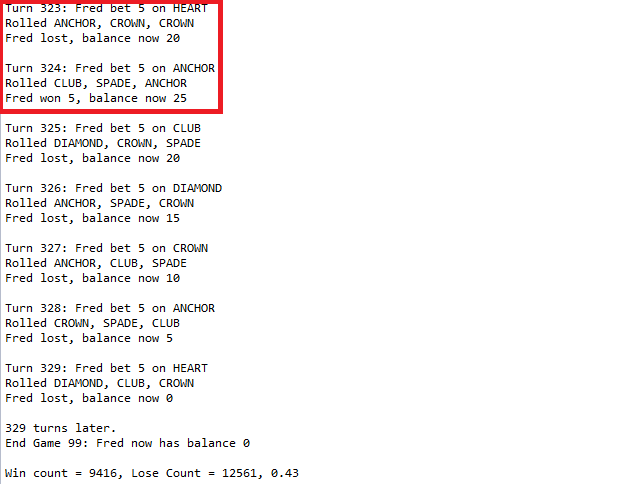
The proposed solution is to alter Game.java to add the bet amount to the calculated winnings, before it is passed to Player.java, once this solution has been implemented, the balance of the player should increase properly.

# Results

My hypothesis was correct. The cause of the bug was that the winnings calculated in the method are simply what has been won from the roll alone. The winnings should consist of the money won from the roll, plus the players bet itself.

Testing had confirmed both that the bug exists and can be replicated, and that it is now removed from the fixed code. Test results below:





# Conclusion

* My hypothesis was correct.
* The cause of the bug was that the winnings calculated in the method are simply what has been won from the roll alone. The winnings should consist of the money won from the roll, plus the players bet itself.
* Bug has been tested and fixed.