Hypothesis & Testing – Bug 02

# Bug Reported

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

Contents

[Bug Reported 1](#_Toc432782164)

[Analysis 2](#_Toc432782165)

[Post-Test Analysis 2](#_Toc432782166)

[Localising the bug 3](#_Toc432782167)

[Hypothesis 3](#_Toc432782168)

[Results 5](#_Toc432782169)

[Conclusion 6](#_Toc432782170)

# Analysis

Initial inspection of program is ineffective, as running the program normally resulted in player wining every time by reaching 200. Test must be performed in order to recreate bug.

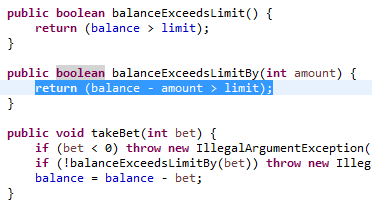
## Post-Test Analysis

The test results reveal that this bug does exist, and causes the program to behave in exactly the way the bug was reported. It appears that the limit does end the game when it reaches 5, as opposed to the pre-defined limit of 0.

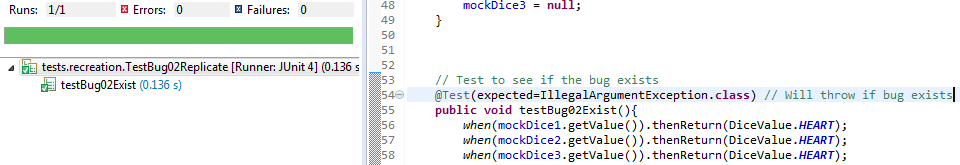
# Localising the bug

The bug is related to creating the limit, enforcing the limit, and checking the player’s balance. Closer inspection reveals that only Main and Player.java should be inspected.

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



Automated test reveals that the bug is created in this method. See Bug02Replicate.java for test.



The above image is testing for if the bug exists, the bugs exist, so the test passes. 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 Player.java, the process is as follows for balanceExceedsLimitBy method:

1. Method receives the ‘bet amount’ from the main method
2. The ‘bet amount’ is deducted from players balance
3. **If the result is ‘greater than’ the limit(value of 0), return true**

Step 3 is the suspected cause of the bug, despite not exceeding the limit (but simply meeting it) results in the game not allowing the player to continue playing, despite having enough for another game. The method should be returning true if the result is equal to or greater than the limit.

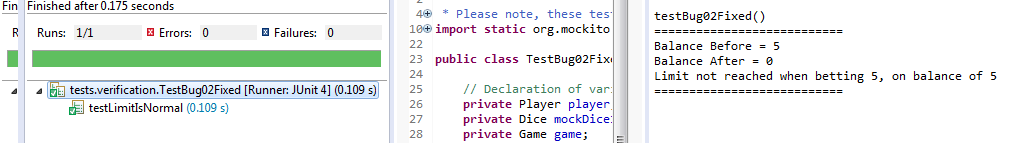
Replication has revealed that under any circumstance that the player has 5 balance remaining, the game will consider it as too low to continue.

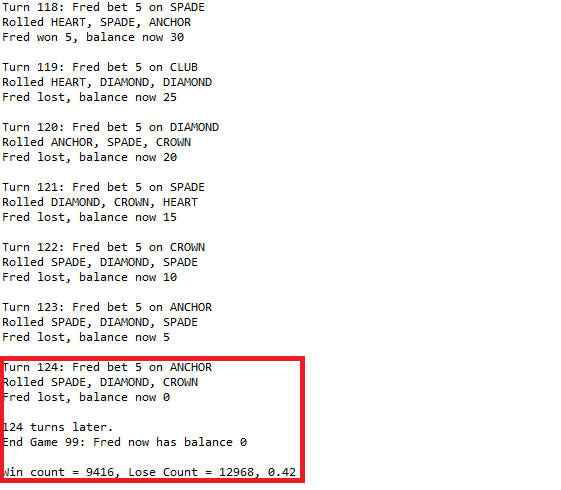
The proposed solution is to alter Player.java, changing the ‘greater than’ to a ‘greater than or equal to’. Once this solution has been implemented, the player should be able to continue until a limit of 0 is reached, not 5.

# Results

My hypothesis was correct. The cause of the bug was that the greater than did not think that a balance of 5 was acceptable. The balance of 5 should be acceptable as it is within the limit.

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.
* My hypothesis was correct. The cause of the bug was that the greater than did not think that a balance of 5 was acceptable.
* Bug has been tested and fixed.