Hypothesis & Testing – Bug 03

# Bug Reported

*“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.”*

Contents

[Bug Reported 1](#_Toc432782214)

[Analysis 2](#_Toc432782215)

[Localising the bug 3](#_Toc432782216)

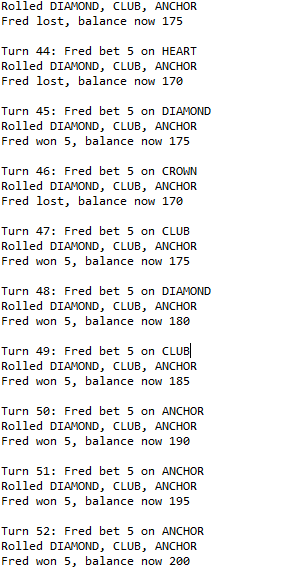
[Hypothesis 4](#_Toc432782217)

[Results 5](#_Toc432782218)

[Conclusion 6](#_Toc432782219)

# Analysis

Initial inspection of program reveals similar behaviour as to what was described in the bug report. The roll value of SPADE never appears.



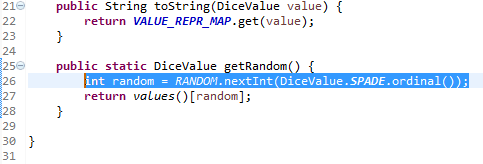
SPADE should occur randomly, yet never occurs.

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

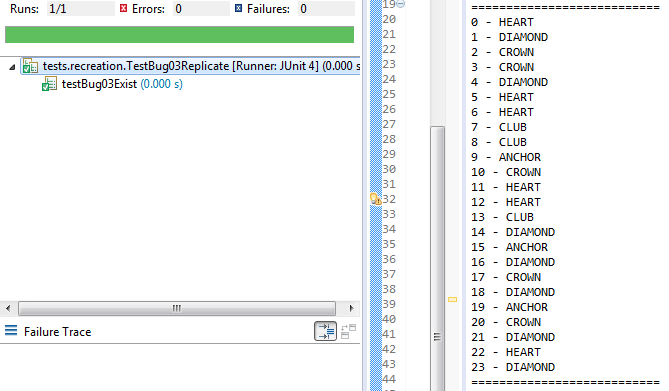
# Localising the bug

The bug is related to calculating roll odds, and applying the players choice. Closer inspection reveals that only DiceValue and Main.java should be inspected. At first another bug, causing the die to always be the same, was suspected however it was not the cause of this bug, due to it never being able to roll a SPADE. Further racing allowed the discovery of this bugs cause.

Code analysis reveals that the bug is likely within DiceValue.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 DiceValue.java, the process is as follows for getRandom method:

1. **Method creates a new int, between 0, and the ordinal value of SPADES in the map.**
2. The method returns a value of the map, the value is determined by its ordinal position, chosen in step 1.

Step 1 is the suspected cause of the bug, the use of the .ordinal() method is incorrect. Researching the oracle userdoc we find that when oridinal is used it “*returns the ordinal of this enumeration constant (its position in its enum declaration, where the initial constant is assigned an ordinal of zero).*” This means in this case, SPADES has returned an enum of 5 (because it begins at 0), meaning the random int will only generate ints from 1 to 5.

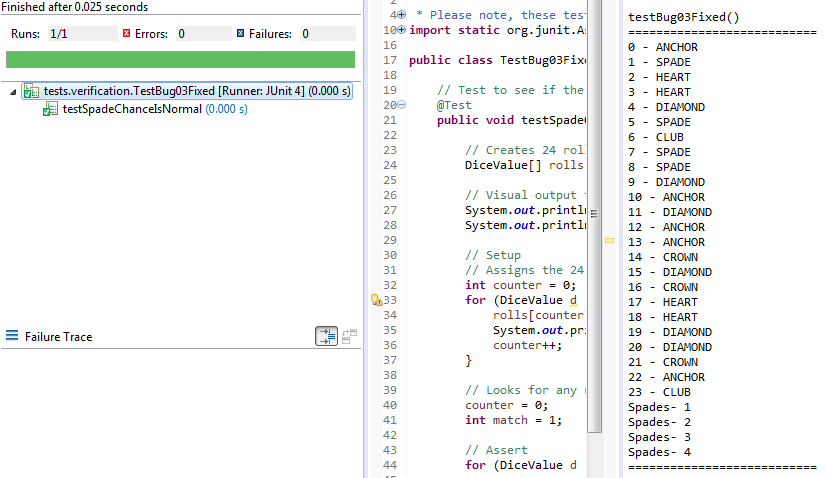
Replication has revealed that this will always occur, and also affects the roll of the 3 die, alongside the players pick (generated in main.java).

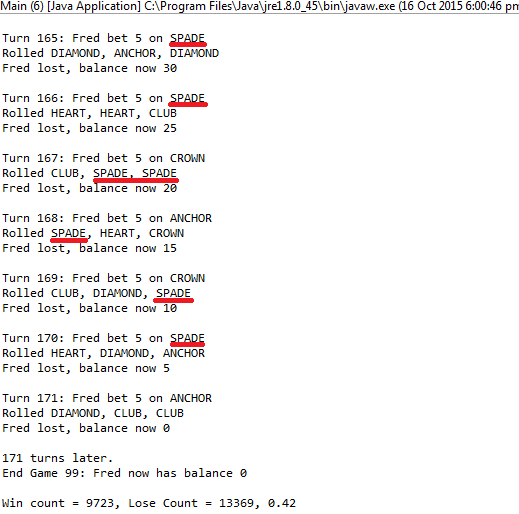
The proposed solution is to alter DiceValue.java to properly use .ordinal, once this solution has been implemented, the chance of rolling a SPADE should be realistic, making the chances normal.

# Results

My hypothesis was correct. The cause of the bug was that the ordinate function was making rolling SPADE impossible. The roll chances should all be even for a fair game.

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:





The test rolled the die 24 times. If at least 1 spade occurred, the test passed. There is a 98.74% that a spade will roll at least once during 24 rolls. So if test fails, the test should be run again to be sure.

# Conclusion

* My hypothesis was correct.
* The cause of the bug was that the ordinate function was making rolling SPADE impossible.
* Bug has been tested and fixed.