**Bug**: Overall odds of the game are 0.49 when they should be 0.42

**Hypothesis 1**: Origin of winCount (line 72 main class) is sane.

Test: Check winCount is being incremented properly and is a sane value before line 72.

Prediction: winCount properly increments each time a profit is made. Value is sane before line 72.

Result: winCount is incremented with each win. Value is sane.   
 Hypothesis confirmed.

**Hypothesis 2**: Origin of loseCount (line 72 main class) is sane.

Test: Check loseCount is being incremented properly and is a sane value before line 72.

Prediction: loseCount properly increments each time a profit is made. Value is sane before line 72.

Result: loseCount is incremented with each win. Value is sane.   
 Hypothesis confirmed.

**Hypothesis 3**: Calculation on line 72 produces correct result.

Test: Break at line 72 and use the debuggers evaluate expression tool to check calculation.

Prediction: Ratio will be calculated correctly.

Result: Ratio is calculated correctly as 0.49.   
 Calculation is not infected.   
 Hypothesis confirmed.

**Hypothesis 4**: The Pick variable on line 44 of the main method is not a truly random choice between the 6 suits.

Test: Check the pick variable through multiple loops.

Prediction: Pick variable will adhere to some pattern.

Result: Pick appears to be random however the Spade suit is never selected.  
 Hypothesis confirmed.

**Hypothesis 5**: The getRandom method of the DiceValue class never returns the spade suit.

Test: Automated Junit test used to call the method a large number of times and survey results.

Prediction: The Spade suit will never be returned.

Result: The SPADE value was not returned in over 200 calls.   
 getRandom method is infected.   
 Hypothesis confirmed.

**Hypothesis 6**: Origins of line 25 in the DiceValue class are sane.

Test: Check all enum values of the class are working.

Prediction: Origins are sane.

Result: All enum values are working as intended. Line 25 is infected.  
 Hypothesis confirmed.

**Hypothesis 7**: The parameters used when calling the Random.nextInt() method are insane.

Test: check the value returned by DiceValue.SPADE.ordinal()

Prediction: The value will be 5.

Result: Value was 5.   
 Parameter is insane.   
 Hypothesis confirmed.

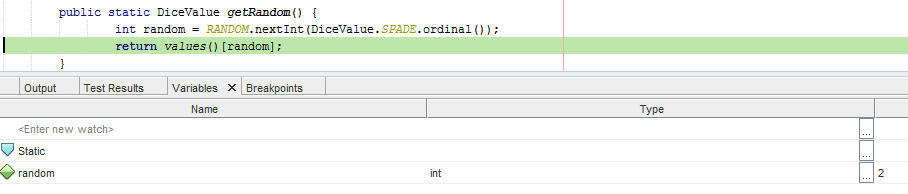
**Hypothesis 8**: Adding a 1 to the Random.NextInt() methods parameter will resolve the bug.

Test: Add one to DiceValue.SPADE.ordinal()

Prediction: Bug will be resolved and spade will be added to the selection of possible suits.

Result: Bug resolved.   
 Win-Lose Ratio is 0.42  
 Hypothesis Confirmed.

Before:



After

