**Proof:**

We can see that when the game is played 100 times the win rate is higher than the expected 0.42

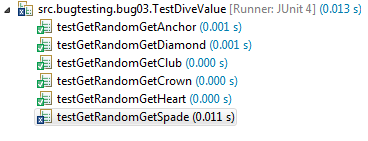


**Hypothesis:**

Considering the win loss ratio is ultimately decided by what dice are rolled this is the first place to look. To back this up scrolling through the last 50 turns that where played not once did spade get rolled. This could possible mean that there is something wrong with the DiceValue.getRandom() method.

In order to test the getRandom() method we will use Junit tests to verify that getRandom() does indeed have the ability to produce every value.

After running the tests we can see that Spade is not returned by getRandom()



Looking into the getRandom() method there is a very subtle issue. The fact that Spade is not called points to some sort of issue where the highest values being retunred by RANDOM.nextInt() is only 4 which is equal to CLUB.

From the JavaDoc of ordinal:

*Returns the ordinal of this enumeration constant (its position in its enum declaration, where the initial constant is assigned an ordinal of zero).*

Meaning that DiceValue.Spade.ordinal() would return a value of 5. This is proved by the Junit test



Therefore we know that ordinal is returning the correct value

However RANDOM.nextInt(bound) JavaDoc attempts to account for this:

*Returns a pseudorandom, uniformly distributed int value between 0 (inclusive) and the specified value (exclusive)*

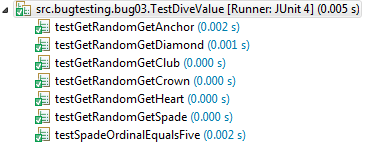
This is where the problem is nextInt(bound) excludes the bounded integer in its random lookup hence in our case it will only ever produce a random number of 0,1,2,3,4 as it is 5 values but it starts counting at 0.

There are two ways to resolve this issue.

**Fix 1**(quick and easy):

RANDOM.nextInt(DiceValue.SPADE.ordinal() + 1);

Junit Result:



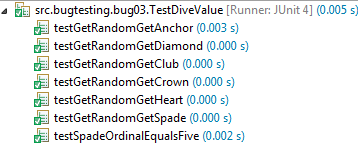
We can see that Spade is now found. And that the win ratio os the program is correct.



**Fix 2**(Better)

Adding a local variable to produce the length of the Enum. This will ensure that no extra human calculation will be required. And that the nextInt() method is not reliant on spade.ordinal() in the event that it is removed, moved or more values are added.

Junit Result:



Program still has a 0.42 ratio



The bug has now been fixed.