# Bug 3 (plus “Bug 6”) Test Investigation/Fix Log

## Hypothesis 1

That the odds of a player winning the game are too high because there is some issue with the dice rolling module(s), which are somehow “loaded” in favour of the player by around 4-5% (basing this on previous automated tests performed – see “Automated Test – Bug 3”.

## Action 1: Test Hypothesis

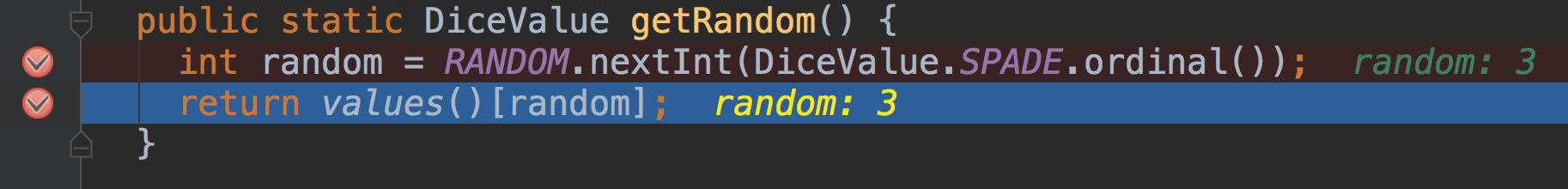
Place breakpoints into the classes Dice and DiceValue (which has method getRandom called by Dice and provides the actual random values) to:

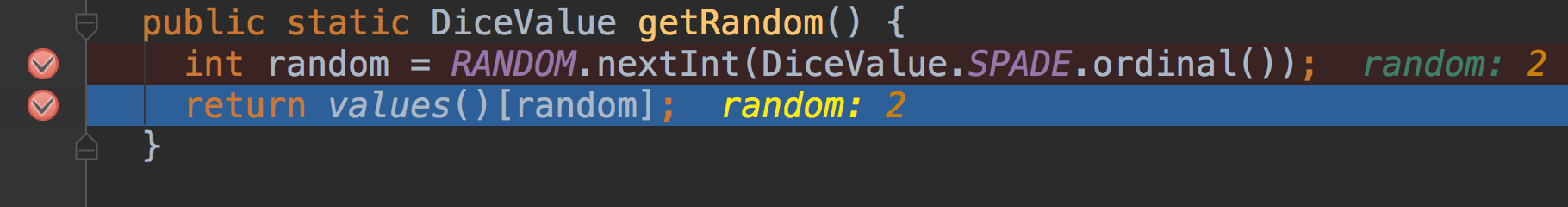
1. Observe the range of values that are produced by the modules when the Automated Test for Bug 3 is run; and
2. Check that DiceValue.getRandom() method is passing one of the randomly selected values of the six possible dice values as integers (0-5) to the Dice class.

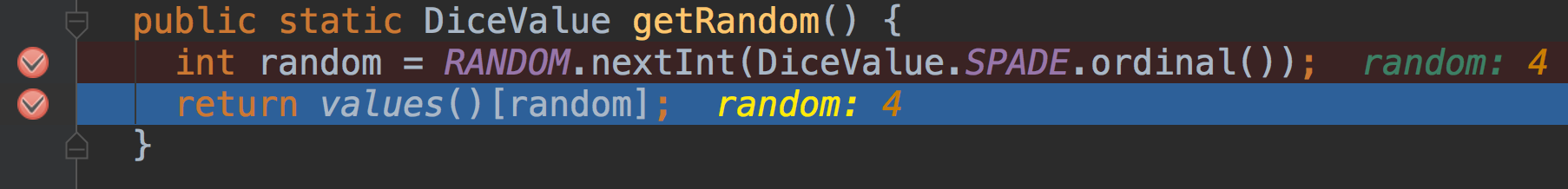
## Results of Test

Investigation into Bug 3 shows that the random values of dice are passed from DiceValue.getRandom to Dice.getValue() – see screenshots below. However, observation shows that there has never been a value of “SPADE” selected by the game. This gives rise to another hypothesis that the game never rolls a SPADE – see Hypothesis 2 below. See screenshots below.

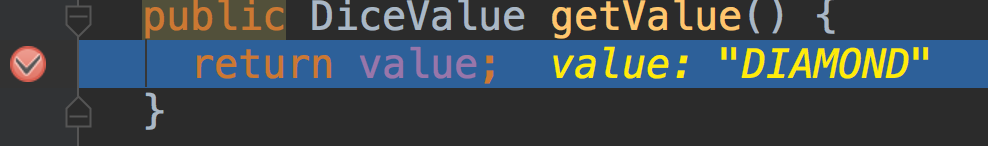
Screenshots of variables from DiceValue.getRandom():

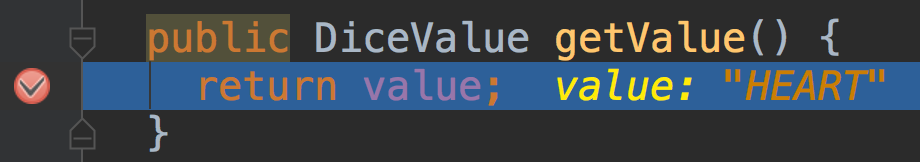


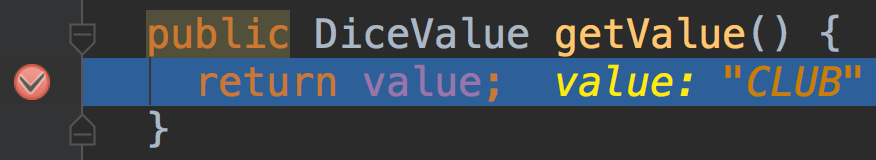




Screenshots of variables from Dice.getValue() – note the dice names correspond to the ordinal values in DiceValue.getRandom():







## Hypothesis 2 (“Bug 6”)

The second hypothesis is that Bug 3 is caused by the game never rolling a SPADE. Another Test program called “TestBug3a” has been created to assert whether the value of “SPADE” is never chosen by the program. **Note that this bug is the same as what was referred to as “Bug 6” in the UAT test No. 1**, however at that point it was not known to be connected with the cause of Bug 3 (incorrect win ratio).

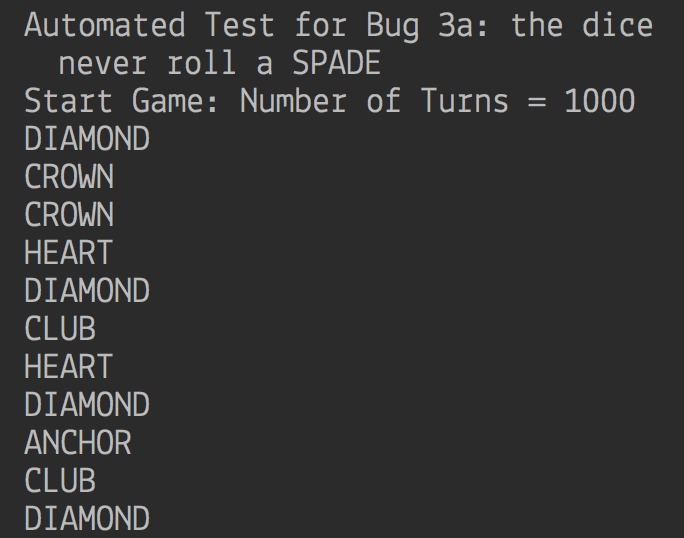
## Action 2: Run “TestBug3a”

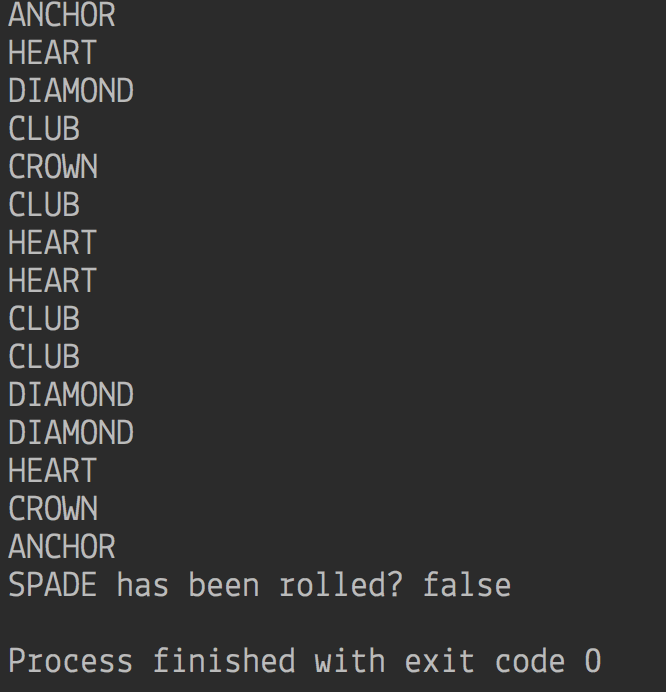
TestBug3a was run to determine whether the value of “SPADE” is never produced by the Dice and DiceValue classes. If the assertion is correct, this will show that a possible cause of the ratio bug is because the range of random values for dice rolls is smaller than it should be, i.e. from 0-4 instead of 0-5. If true, this means that the player will find a match against a die roll more frequently. The test results are below.

## Results of Test of TestBug3a

This test showed that a SPADE is never rolled in the game. See the screenshots below which show the start of the 1000 dice rolls and the last few rolls. The next hypothesis will investigate the cause for this bug.

Screenshots:





## Hypothesis 3

Based on hypothesis 2 being proven, that Bug 3 is caused by the game never rolling a SPADE (“Bug 6”), the next hypothesis (3) is that Bug 3 is caused by the DiceValue.getRandom() method incorrectly creating random values for the dice rolls.

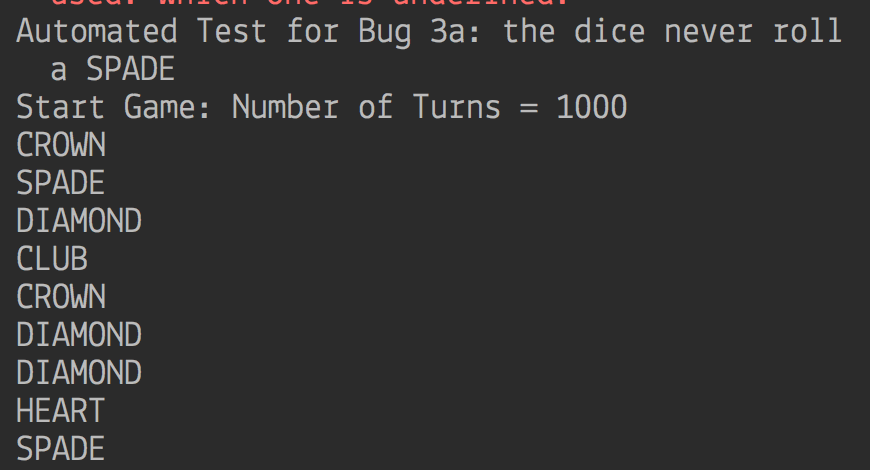
## Action 3: Fix DiceValue.getRandom() and re-run TestBug3a, TestBug3 and main()

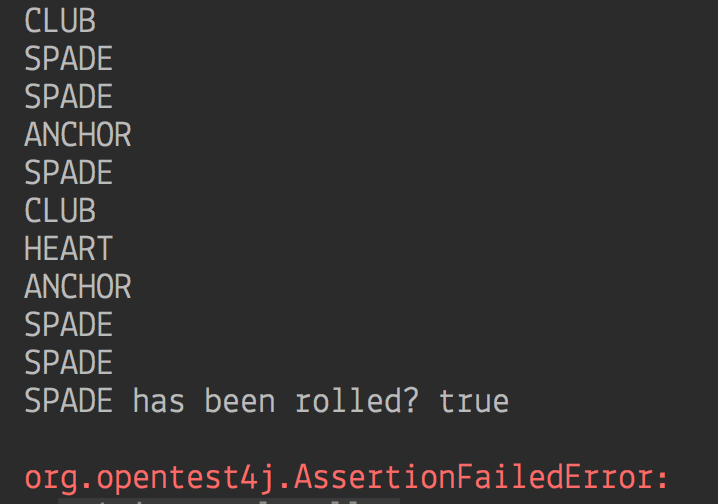
On examination, it appears that the end value in the range for the Random() method is set at “SPADE.ordinal(), which is too low by one as the end value is not included in the range. The code in the method DiceValue.getRandom() will be fixed to allow the full range of dice values (0-5) to be rolled. The TestBug3a and TestBug3 programs will be run again to see if the bug has been fixed. If so, the game (main()) will be run again to see if the ratio of wins has been corrected to approximately 42%.

## Results of Test for Hypothesis 3 (“Bug 6”)

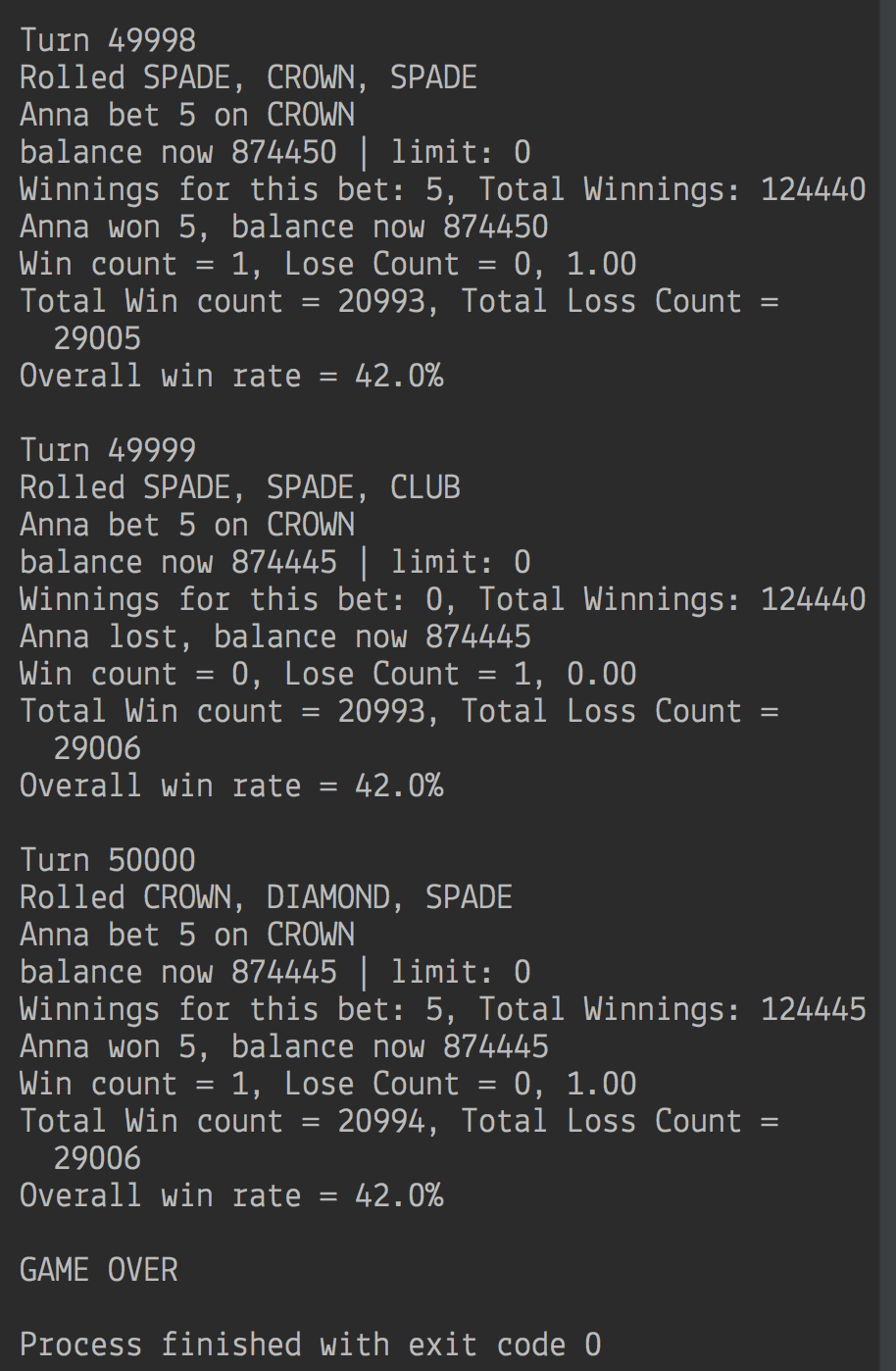
The test results show the bug is now fixed within the DiceValue.getRandom() method - see following screenshots from TestBug3a that show that the SPADE value is now rolled by DiceValue.getRandom(). Also see that when TestBug3 is run, the final ratio is 42% (based on 50,000 turns and a starting balance of $1,000,000). However, when the game is run from main() method, the results do not support the evidence that the bug is fixed – see screenshots below. It appears that because the same three dice are rolled in each turn, the ratio results are skewed. This bug is called Bug 4 – see Fix Log for Bug 4.

Screenshots from TestBug3a after fix was applied to DiceValue.getRandom():





Screenshot from TestBug3 after fix was applied to DiceValue.getRandom():



Screenshots from three runs of the game (main()) after fix was applied:

