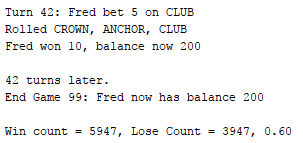
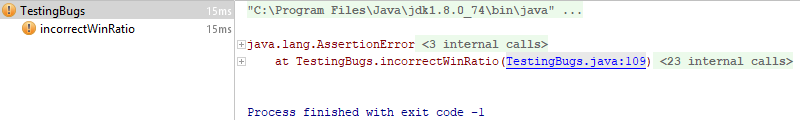
Debug log for bug “Betting Limit Unreachable”

Initial example of bug



Output from automated test



Log:

|  |  |
| --- | --- |
| **Step reasoning** | The origin of the issue starts at where the user interface prints us out an incorrect ratio value. Instead of diving in, we should backtrack from that position |
| **Hypothesis 1** | The UI prints the correct information, and we’re going have to keep backtracking |
| **Test** | Debug mode, review the values in the print statement |
| **Prediction** | The UI prints the correct information, and we’re going have to keep backtracking |
| **Result** | Prediction correct, values are infected in the case that they represent an infected ratio, but the print code is sane |

|  |  |
| --- | --- |
| **Step reasoning** | Where are totalWin and totalLoss are set? |
| **Hypothesis 2** | The values for totalWin and totalLoss aren’t being set/counted correctly |
| **Test** | Debug mode, review the values change |
| **Prediction** | Values aren’t being set properly |
| **Result** | Values are being set fine, issue is further back. The values are infected such that their ratio is infected, but the line of code is sane |

|  |  |
| --- | --- |
| **Step reasoning** | winCount and loseCount set the above values. Winning or losing a round sets winCount and loseCount values. Check if they are being set when winning occurs. |
| **Hypothesis 3** | Issue in code that decides if the game is won or lost |
| **Test** | Debug mode, verify that when wins occur, winCount is incremented, and when a loss occurs that loseCount is incremented |
| **Prediction** | These will be incremented without issue; the issues will be further back |
| **Result** | The values were incremented without issue, values are sane |

|  |  |
| --- | --- |
| **Step reasoning** | The game.playRound code decided if a game is won. It looks like when a match occurs, it increments the ’matchs’ value. Let’s make sure that the comparison for checking if match occur is correct |
| **Hypothesis 4** | The statement which checks if there is a match has a bug in it |
| **Test** | Debug mode, step through, make sure comparison is correct |
| **Prediction** | It won’t be correct |
| **Result** | It was correct, that line is sane, issue must be else where |

|  |  |
| --- | --- |
| **Step reasoning** | d.roll() generates those values. Check this. |
| **Hypothesis 5** | d.roll() does not satisfy its method contract |
| **Test** | Debug mode, check input and output |
| **Prediction** | Method does not meet its method contract |
| **Result** | Method does meet its method contract |

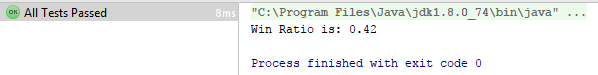
|  |  |
| --- | --- |
| **Step reasoning** | The code inside d.roll(),DiceValue.getRandom(), may not be producing a random value. Check this. |
| **Hypothesis 6** | DiveValue.getRandom does not return a true random value |
| **Test** | Debug mode, step through and confirm it meets its method contract |
| **Prediction** | Method will not produce a random value |
| **Result** | Debug mode shows that DiceValue.SPADE.ordinal() is 5, not the expected 6. Therefore, only 5 of the 6 suits can be returned as a value, this line is infected |

|  |  |
| --- | --- |
| **Step reasoning** | Editing this code to use the value 6 as total would produce different probability stats, let’s change it and return the test |
| **Hypothesis 7** | Incorrect upper limit passed into the random number generator effects the probability |
| **Test** | Change the value 5 to the something which represents the value that should be passed in, 6 (values().length) and rerun the automated test |
| **Prediction** | The automated test will pass and bug will be resolved |
| **Result** | Automated test failed, this is not the full issue |

|  |  |
| --- | --- |
| **Step reasoning** | Automated test failed, did not expect that. Check how the random value goes back into the code. Go forward. Leave the changes from previous step as this must be related. |
| **Hypothesis 8** | Random value is not getting into dice value |
| **Test** | Follow the path of the random generated dice roll into how it evaluates a match |
| **Prediction** | Randomly generated value is not being used to check a match again |
| **Result** | Although d.roll() does generate and return a random value, it does not assign it to the member variable of value, which is checked when the match/comparison check occurs |

|  |  |
| --- | --- |
| **Step reasoning** | Make the method assign the random DiceValue to value, and return value to change as less code as possible. |
| **Hypothesis 9** | Changing Dice.roll() method to set the random value calculated to ‘value’, then returning that value will resolve the issue |
| **Test** | Change the code, rerun automated test |
| **Prediction** | This will resolve the issue |
| **Result** | The automated test passed. |

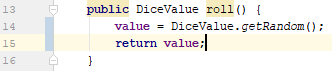
Successful automated test result



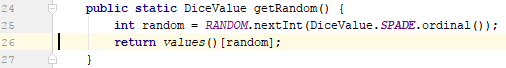
Initial bug, /src/Dice.java, line 14 – returns value before setting it to internal member



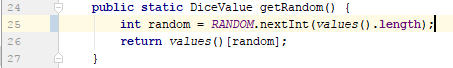
Resolved bug, /src/Dice.java, line 14



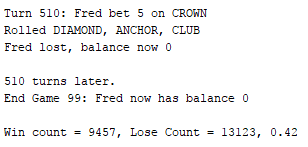
Initial bug, /src/DiceValue.java, line 25 – getting random number 0->5, not 0->6



Resolved bug, /src/DiceValue.java, line 25



Resolved bug, UI output



Regression testing:

Re-running previous tests to ensure no new bugs have been created.

