10/16/2017

Sarah Froemming - 11587806

Charles Sturt University

Debugging

ITC205

Contents

[GitHub URL 2](#_Toc495970476)

[Bug 1 – Incorrect Winning Amount 3](#_Toc495970477)

[i. UAT demonstrating the bug 3](#_Toc495970478)

[ii. Automated test demonstration of buggy behavior 3](#_Toc495970479)

[iii. Debugging Log 4](#_Toc495970480)

[iv. ‘Before’ and ‘after’ screen shots of bug origin 5](#_Toc495970481)

[v. Automated test demonstration after resolution 5](#_Toc495970482)

[vi. UAT demonstrating the fix 5](#_Toc495970483)

[Bug 2 – Betting Limit Unreachable 6](#_Toc495970484)

[i. UAT demonstrating the bug 6](#_Toc495970485)

[ii. Automated test demonstration of buggy behavior 6](#_Toc495970486)

[iii. Debugging Log 7](#_Toc495970487)

[iv. ‘Before’ and ‘after’ screen shots of bug origin 7](#_Toc495970488)

[v. Automated test demonstration after resolution 7](#_Toc495970489)

[vi. Test results from UAT 8](#_Toc495970490)

[Bug 3 – Incorrect Ratio 9](#_Toc495970491)

[i. UAT demonstrating the bug 9](#_Toc495970492)

[ii. Automated test demonstration of buggy behavior 9](#_Toc495970493)

[iii. Debugging Log 10](#_Toc495970494)

[iv. ‘Before’ and ‘after’ screen shots of bug origin 11](#_Toc495970495)

[v. Automated test demonstration after resolution 12](#_Toc495970496)

[vi. Test results from UAT 12](#_Toc495970497)

[‘Bug 5’ – No ending of application 13](#_Toc495970498)

[i. UAT demonstrating the bug 13](#_Toc495970499)

[ii. Debugging Log 13](#_Toc495970500)

[iii. ‘Before’ and ‘after’ screen shots of bug origin 14](#_Toc495970501)

[iv. Test results from UAT 14](#_Toc495970502)

[User input implementation 15](#_Toc495970503)

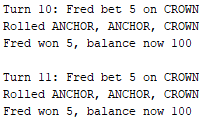
# GitHub URL

https://github.com/ox0Sarah0xo/DiceGame/tree/Sarah

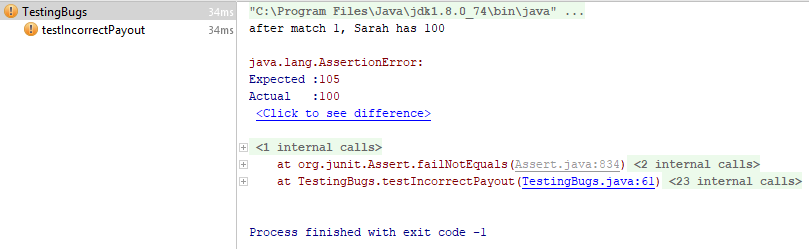
# Bug 1 – Incorrect Winning Amount

## UAT demonstrating the bug

| **Test Name** | | | Winning amount bug | | | |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case Tested:** | | | Play Game | | | |
| **Test Description:** | | | User plays a game but when winning the balance does not increase | | | |
| **Pre-conditions** | | | User must have a balance | | | |
| **Post-conditions** | | | The balance should not change according to winnings | | | |
| **Notes:** | |  | | | | |
| **Result (Pass/Fail/Warning/Incomplete)** | | **Pass** | | | | |
|  | **TEST STEP** | | | **EXPECTED TEST RESULTS** | P | F |
|  | Enter balance and betting amount in code and run the game | | | When winning, the balance should not increase | X |  |



## Automated test demonstration of buggy behavior



## Debugging Log

|  |  |
| --- | --- |
| **Step reasoning** | Issue could be UI where results aren’t displayed correctly |
| **Hypothesis 1** | Automated test has been written, so output of data to screen should be correct -> this is not a UI issue |
| **Test** | Debug mode and verify values that go into the console print command |
| **Prediction** | The values passed in will be the same as printed to the screen |
| **Result** | Correct, values passed into print statement display correctly, values are same |

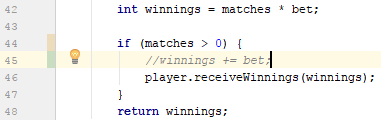
|  |  |
| --- | --- |
| **Step reasoning** | Stepping back through code, game.getDiceValues is only a get method, it should be getting the values from the previous game that run for display. Game.playRound is next. |
| **Hypothesis 2** | player.receiveWinnings in game.playRound is not adding winnings correctly |
| **Test** | Debug mode and step into player. receiveWinnings, see if balance is modified before adding to player |
| **Prediction** | Balance will be modified by recieveWinnings |
| **Result** | Balance was not modified by receiveWinnings, balance is same at this point |

|  |  |
| --- | --- |
| **Step reasoning** | Player.recieveWinnings effectively fulfils its method contract, therefore the value passed into recieveWinnings must be the issue, possibly a calculation issue when generating value. |
| **Hypothesis 3** | When calculating winnings, the number of matches found is incorrect |
| **Test** | Check the raw values of the dice rolls and compare it to the count of matches found |
| **Prediction** | The number of matches found is always 1 short when matches is > 0 (Actually matches the issue) |
| **Result** | This is incorrect, number of matches found is correct, ’matches’ is same at this point |

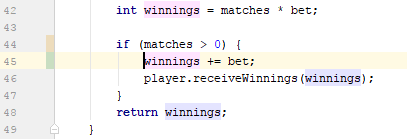
|  |  |
| --- | --- |
| **Step reasoning** | Reading the rules of the games, a player better x amount, when 1 match occurs should be returned their bet + win the value of the bet (1:1). This calculation of winnings is incorrect |
| **Hypothesis 4** | The calculation of winnings is incorrect; the player is not returned their initial bet as well as their winnings, winnings are affected |
| **Test** | Change code to add the initial bet to the winnings calculation only when player has won, run automated test for IncorrectPayoutBugFixed |
| **Prediction** | Automated test will run successfully; bug will be resolved |
| **Result** | Prediction is correct, bug is resolved |

## ‘Before’ and ‘after’ screen shots of bug origin

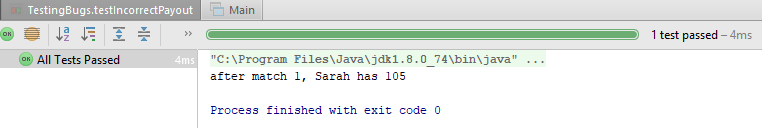
Initial bug, /src/Game.java, line 42



Resolved bug, /src/Game.java, line 45

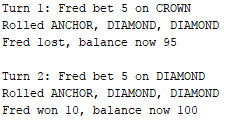


## Automated test demonstration after resolution



## UAT demonstrating the fix

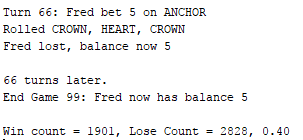
| **Test Name** | | | Winning amount bug | | | |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case Tested:** | | | Play Game | | | |
| **Test Description:** | | | User plays a game and when winning the balance does increase | | | |
| **Pre-conditions** | | | User must have a balance | | | |
| **Post-conditions** | | | The balance should change according to winnings | | | |
| **Notes:** | |  | | | | |
| **Result (Pass/Fail/Warning/Incomplete)** | | **Pass** | | | | |
|  | **TEST STEP** | | | **EXPECTED TEST RESULTS** | P | F |
|  | Enter balance and betting amount in code and run the game | | | When winning, the balance should increase | X |  |



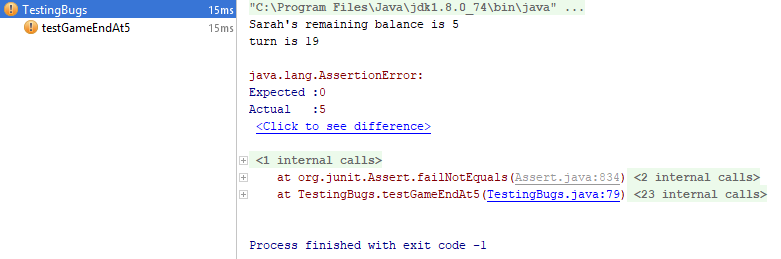
# Bug 2 – Betting Limit Unreachable

## UAT demonstrating the bug

| **Test Name** | | | Betting limit bug | | | |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case Tested:** | | | Play Game | | | |
| **Test Description:** | | | User plays a game with limit set to 0 but when game ends with player still remaining 5 dollars | | | |
| **Pre-conditions** | | | User must have a balance and limit must be set to 0 | | | |
| **Post-conditions** | | | The game should end with 5 dollars remaining in players balance | | | |
| **Notes:** | |  | | | | |
| **Result (Pass/Fail/Warning/Incomplete)** | | **Pass** | | | | |
|  | **TEST STEP** | | | **EXPECTED TEST RESULTS** | P | F |
|  | Enter balance and betting amount in code and run the game | | | When game ends, player balance should be 5 | X |  |



## Automated test demonstration of buggy behavior



## Debugging Log

|  |  |
| --- | --- |
| **Step reasoning** | The origin of the issue is that the game ends without hitting the limit. Let’s start by verifying the UI is printing the correct value. |
| **Hypothesis 1** | The UI is printing the incorrect value of the players remaining balance after the game ends |
| **Test** | Debug mode and view the value of ‘player.getBalance()’ |
| **Prediction** | Value returned by Player.getBalance is different from the one shown in UI |
| **Result** | Prediction incorrect, print statement is correct, value is infected |

|  |  |
| --- | --- |
| **Step reasoning** | The issue is the game is quitting prematurely, what decides when the game ends?  There is a while loop which re-runs the game while the player is above his limit, and below twice their initial balance. There are two conditions, once checks if we’ve going to exceed the limit, and one that check if we’ve made more than the target. It’s going to be an issue with reaching the limit. |
| **Hypothesis 2** | The value passed into player.balanceExceedsLimitBy(bet) in Main is infected |
| **Test** | Debug mode and view the value of ‘bet’ |
| **Prediction** | Bet is incorrect |
| **Result** | Bet’s value is declared only once in the code, then never changed. Bet’s value is always sane. |

|  |  |
| --- | --- |
| **Step reasoning** | Step into balanceExceedsLimitBy(bet) |
| **Hypothesis 3** | Method balanceExceedsLimitBy(bet) is not meeting its method contract. |
| **Test** | Debug mode, go into the method, verify all values going in are correct, and value coming out is correct |
| **Prediction** | Method contract is being violated |
| **Result** | Input value is correct; output value is infected. |

|  |  |
| --- | --- |
| **Step reasoning** | There is only one line in this method! It must be the issue. Thinking about it, in the line ‘(balance - amount > limit)’, if I used (5-5 > 0), it would return false, as 0 is not greater than 0. |
| **Hypothesis 4** | This is a relational operator issue |
| **Test** | Change the greater than symbol to a greater or equal to symbol, and rerun the automated test, it will pass |
| **Prediction** | Issue will be resolved |
| **Result** | Issue is resolved |

## ‘Before’ and ‘after’ screen shots of bug origin

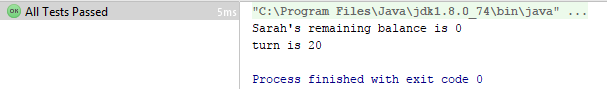
Initial bug, /src/Player.java, line 30



Resolved bug, /src/Player.java, line 30

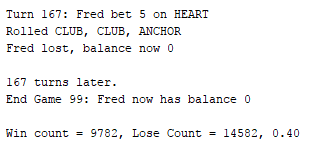


## Automated test demonstration after resolution



## Test results from UAT

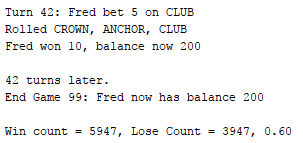
| **Test Name** | | | Betting limit bug | | | |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case Tested:** | | | Play Game | | | |
| **Test Description:** | | | User plays a game with limit set to 0 and when game ends player has 0 balance remaining | | | |
| **Pre-conditions** | | | User must have a balance and limit must be set to 0 | | | |
| **Post-conditions** | | | The game should end with 0 dollars remaining in players balance | | | |
| **Notes:** | |  | | | | |
| **Result (Pass/Fail/Warning/Incomplete)** | | **Pass** | | | | |
|  | **TEST STEP** | | | **EXPECTED TEST RESULTS** | P | F |
|  | Enter balance and betting amount in code and run the game | | | When game ends, player balance should be 0 | X |  |



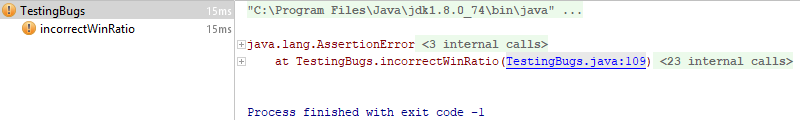
# Bug 3 – Incorrect Ratio

## UAT demonstrating the bug

| **Test Name** | | | Incorrect Winning Ratio | | | |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case Tested:** | | | Play Game | | | |
| **Test Description:** | | | User plays a game and the game win : lose ratio is 0.42 | | | |
| **Pre-conditions** | | | User must have a balance | | | |
| **Post-conditions** | | | The win : lose ratio should be 0.42 | | | |
| **Notes:** | |  | | | | |
| **Result (Pass/Fail/Warning/Incomplete)** | | **Pass** | | | | |
|  | **TEST STEP** | | | **EXPECTED TEST RESULTS** | P | F |
|  | Enter balance and betting amount in code and run the game | | | When game ends, the win : lose ratio is 0.42 | X |  |



## Automated test demonstration of buggy behavior



## Debugging 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 2** | 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** | DiceValue.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 |

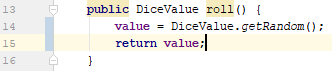
|  |  |
| --- | --- |
| **Step reasoning** | Make the method assign the random DiceValue to value, and return value to change as less code as possible. |
| **Hypothesis 9** | Changing Dice.roll() method to set the random value calculated to ‘value’, then returning that value will resolve the issue |
| **Test** | Change the code, rerun automated test |
| **Prediction** | This will resolve the issue |
| **Result** | The automated test passed.  This also resolved another bug – that the rolled dice values not change during the game. Now they change from turn to turn :D |

## ‘Before’ and ‘after’ screen shots of bug origin

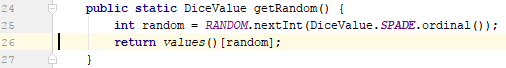
Initial bug, /src/Dice.java, line 14 – returns value before setting it to internal member



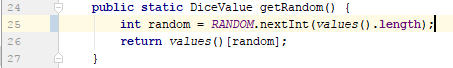
Resolved bug, /src/Dice.java, line 14



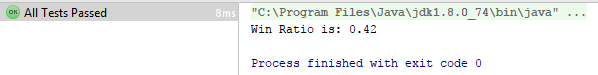
Initial bug, /src/DiceValue.java, line 25 – getting random number 0->5, not 0->6



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

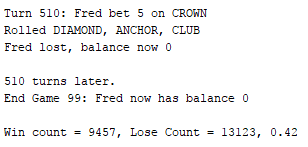


## Automated test demonstration after resolution



## Test results from UAT

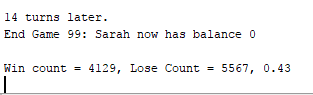
| **Test Name** | | | Incorrect Winning Ratio | | | |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case Tested:** | | | Play Game | | | |
| **Test Description:** | | | User plays a game and the game win : lose ratio is 0.42 | | | |
| **Pre-conditions** | | | User must have a balance | | | |
| **Post-conditions** | | | The win : lose ratio should be 0.42 | | | |
| **Notes:** | |  | | | | |
| **Result (Pass/Fail/Warning/Incomplete)** | | **Pass** | | | | |
|  | **TEST STEP** | | | **EXPECTED TEST RESULTS** | P | F |
|  | Enter balance and betting amount in code and run the game | | | When game ends, the win : lose ratio is 0.42 | X |  |



# ‘Bug 5’ – No ending of application

## UAT demonstrating the bug

| **Test Name** | | | App does not end | | | |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case Tested:** | | | Play Game | | | |
| **Test Description:** | | | User plays a game and when the game ends, the application continues to run | | | |
| **Pre-conditions** | | | User must have a balance and started a game | | | |
| **Post-conditions** | | | The application should still run | | | |
| **Notes:** | |  | | | | |
| **Result (Pass/Fail/Warning/Incomplete)** | | **Pass** | | | | |
|  | **TEST STEP** | | | **EXPECTED TEST RESULTS** | P | F |
|  | Enter name, balance and betting amount and run the game | | | When game ends, application still runs | X |  |



## Debugging Log

|  |  |
| --- | --- |
| **Step reasoning** | Issue should be at the UI as the user does not get information on what to do next |
| **Hypothesis 1** | UI does not give instructions on how to close the app |
| **Test** |  |
| **Prediction** | App is running in definite |
| **Result** | Correct, app is running until manually stopped |

|  |  |
| --- | --- |
| **Step reasoning** | Stepping through the code to see if there are hidden instructions |
| **Hypothesis 1** | Instructions on how to close the app are hidden inside the code |
| **Test** | Debug mode |
| **Prediction** | The is a line of code which explains how to exit the app |
| **Result** | Correct, at the end of the Main.java class in line 84 the application is waiting for user input and if it is the correct input, the app closes |

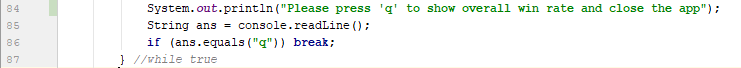
|  |  |
| --- | --- |
| **Step reasoning** | To make it easy for the user to identify those instruction, should add a system output line |
| **Hypothesis 1** | User are not aware of this instruction |
| **Test** | Run app after inserting code |
| **Prediction** | User instructions on how to close app are shown and user is able to close the app |
| **Result** | Correct, instructions are shown and user is able to close the app |

## ‘Before’ and ‘after’ screen shots of bug origin

Initial bug, /src/Main.java, line 84

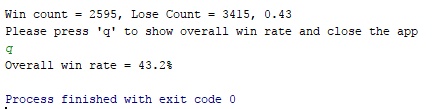


Resolved bug, /src/Main.java, line 84



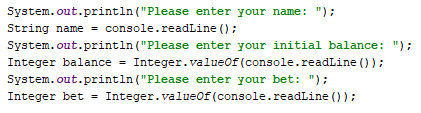
## Test results from UAT

| **Test Name** | | | App does not end | | | |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case Tested:** | | | Play Game | | | |
| **Test Description:** | | | User plays a game and when the game ends, the application give instruction on how to end it | | | |
| **Pre-conditions** | | | User must have a balance and started a game | | | |
| **Post-conditions** | | | The application should give instruction on how end it | | | |
| **Notes:** | |  | | | | |
| **Result (Pass/Fail/Warning/Incomplete)** | | **Pass** | | | | |
|  | **TEST STEP** | | | **EXPECTED TEST RESULTS** | P | F |
|  | Enter name, balance and betting amount and run the game | | | When game ends, application gives closure instructions | X |  |



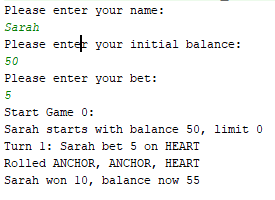
# User input implementation

To change the users name, starting balance or betting amount, the code needs to be changed. To make it easy for users to assign these values, I implemented user input.



This is in the Main.java class before the while loop of the game.

The output is the following:



I came across this idea when I saw this line in the code:



Which indicates there will be a user input somewhere in the code but I did not find one hence I created the above.