Debugging Log

**Assignment 2**

ITC515 – Professional Programming Practice

Justin McKay - Student ID: 11530610

# Debugging Log

## BUG001 - Game does not pay out at correct level

### Steps taken to trace error

* Check *Player.receiveWinnings* method. Method appears to add winnings to balance correctly.
* Check *Game.playRound* method. Note: playRound method calls the takeBet method prior to testing the rolled dice against the value selected.
* This does not appear to be a bug. There could be confusion with the game only the winnings, so you don’t actually see the bet being taken out, and then the winnings put back onto the balance.
* The following 4 screenshots show the progression from taking the bet (balance before and after) and receiving the winnings (balance before and after)

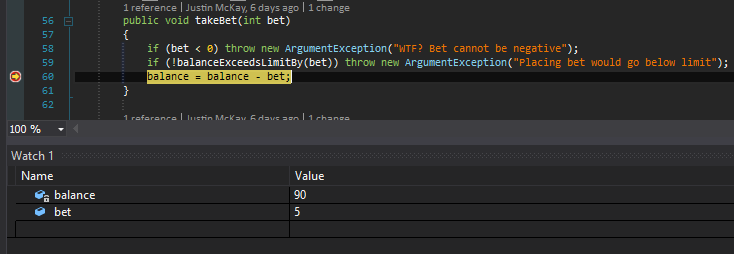


Figure 1 - Showing balance before the bet has been taken

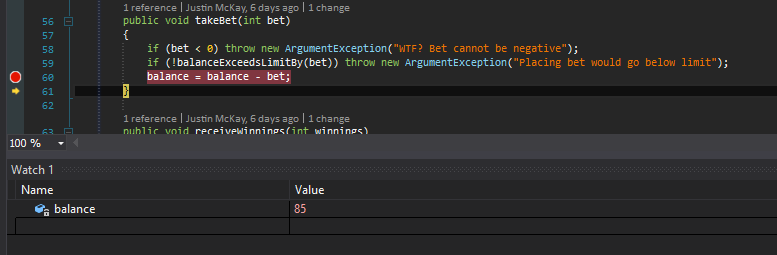


Figure 2 - Showing the balance after the bet has been taken

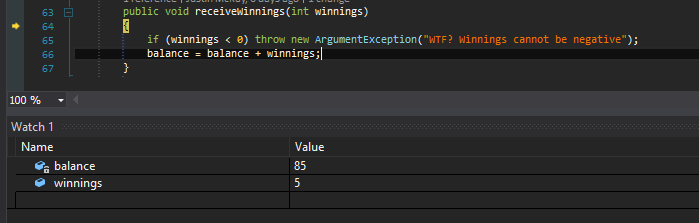


Figure 3 - Showing the balance before the winnings have been received

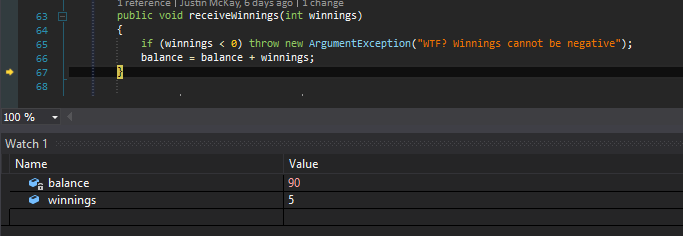


Figure 4 - Showing the balance after the winnings have been added back to the balance

* Will write a test that confirms the bet has been placed (bet deducted from balance) and when a single dice wins, shows the winnings added to the balance.

### Bug elimination steps

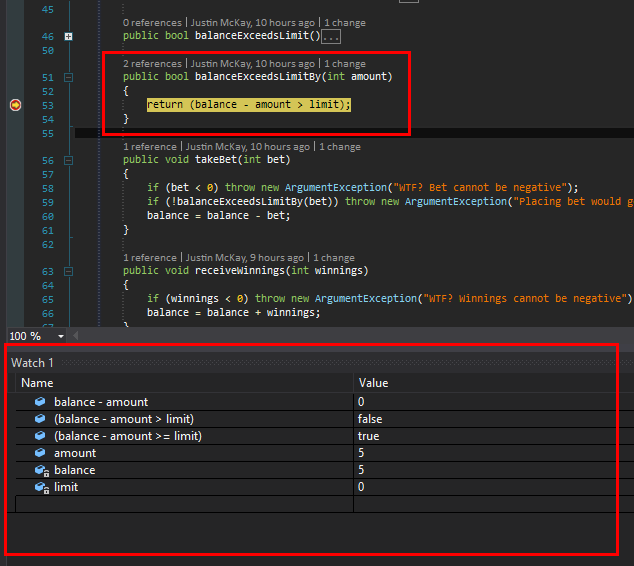
As this is not actually a bug, no steps are required for corrective action.

### Possible Improvements

Change the UI to reflect both the bet being taken off the balance and the winnings being added to the balance. This should correct the issue.

## BUG002 - Player cannot reach betting limit

### Steps taken to trace error

* Check *Player.takeBet* method. Noticed *Player.balanceExceedsLimitBy method* will prevent the bet from being taken if false is returned.
* Step into *Player.balanceExceedsLimitBy* method. Notices that the method will only return true (and allow the player to take a bet) if the balance – bet is greater than 0 (so that you can’t end up with a negative limit). However, as the limit is 0, this method should be checking for balance – bet is greater than *or equal to* the betting limit.
* This behaviour is confirmed with the following debugger screenshot:  
  

### Bug Elimination Steps

To fix the bug, the following code needs to be updated:

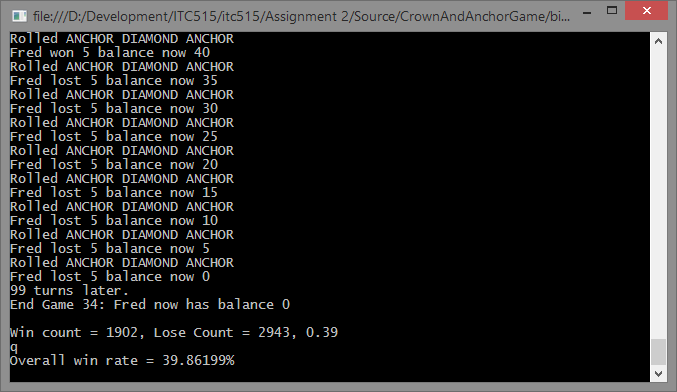
**Player.cs:**Change line 53 to be amount >= limit

### Possible Code Refactoring Improvements

Consider renaming method from *balanceExceedsLimit* to *balanceReachedLimit* so that it’s slightly clearer to the developer that the balance can reach (equal to) or exceed (greater than) the limit.

## BUG003 - Odds in game do not appear to be correct

### Steps taken to trace error

* Initially unable to reproduce this error, as running the program will show a valid score of result of 0.39 (which is much closer to the 0.42) ratio than the reported 0.60 values: See screenshot below:  
  
* Continuing on with the bug report none-the-less, concentrating on the Dice.roll() method as this seems to use a random number to generate the value of the dice rolled.
* Looking at the screenshot above, the last 9 or rolls are the same, so I don’t think the random number stuff is working correctly and incrementing the seed properly (perhaps using milliseconds which is much too slow for these types of iterations).
* This actually makes sense, as the game will roll 3 different dice objects, which will each have its own RANDOM property which is not getting updated, as evidenced by the repeating pattern. I’ll write a unit test for this.
* Created unit test *Dice\_Roll\_Will\_Always\_Return\_A\_High\_Repetition\_Rate* which will detect any repeat rolls where the number of repeats are > 5. This is not the cause the bug. Back to the drawing board.

## BUG004: Game does not update current dice values rolled.

### Steps taken to trace error:

* Noticed that the game is not updating the values of the dice outputted to the console when the game is iterating through the rounds.
* To confirm this, I’ll write a unit test that shows that the each time a round is played, the dice values remain unchanged. 20 rounds without change should be sufficient for this test.