Bug 8 Tracing

Legend: **blue** words are classes, **green** words are methods, and **purple** words are instance variables. (If a word is not coloured, that means it is being used in its natural English sense).

Contents

[Description 2](#_Toc463452836)

[Static Review 3](#_Toc463452837)

[Hypothesis testing 6](#_Toc463452838)

[Hypothesis 1 6](#_Toc463452839)

[Hypothesis 2 8](#_Toc463452840)

[Hypothesis 3 8](#_Toc463452841)

[First run 8](#_Toc463452842)

[Second run 9](#_Toc463452843)

[Third run 10](#_Toc463452844)

[Resolution 12](#_Toc463452845)

[Risk assessment 12](#_Toc463452846)

[Testing the solution 13](#_Toc463452847)

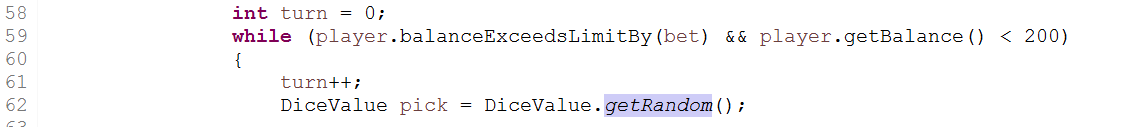
[Conclusion 14](#_Toc463452848)

# Description

Player is unable to reach a zero balance. The player should either finish a game on 200 or on 0, but not 5.

# Static Review

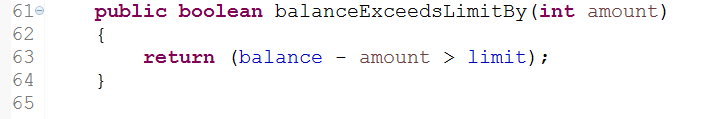
We should look at the part of Main where we are deciding whether to continue with the current Game or not.



We know this is the part where we are deciding whether to continue with the current game, because turn is reset to zero right before it (a Game consists of multiple turns, the first of which is 0).

The second condition is allowing the Game to continue when the player’s balance is less than 200 (so they quit after they have doubled their money). The first condition must be what is keeping them above their limit.

So, let’s have a look at this method.



The method returns true when the difference between the balance and the amount (which is bet in the case where it is being called) is greater than the limit. And the while loop in main will only continue if this method returns true.

Let us consider a few examples.

1. Balance is 10, bet (amount) is 5, limit is 0. Balance – amount = 5 which is greater than limit. So it returns true. That’s good.
2. Balance is 0, amount is 5, limit is 0. Balance – amount = -5 which is less than limit. So it returns false. That’s good (don’t want the player owing money).
3. Balance is 5, amount is 5, limit is 0. Balance – amount = 0 which is equal to limit. So it returns false. This is wrong.

So, we can see why this method infects the while loop condition. When the player is amount to equal their limit, this method returns false when it should return true.

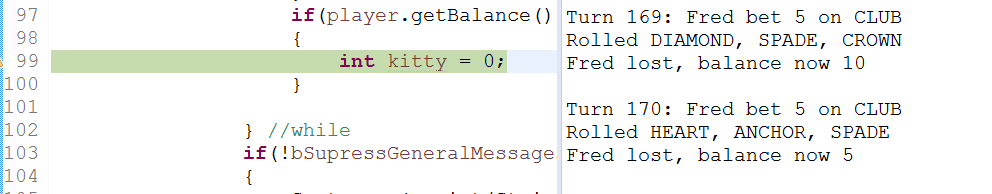
Our hypothesis is:

1. When the amount to be bet would reduce the balance to the limit if lost, the Game is unable to be played (incorrectly).

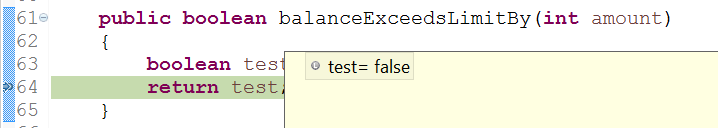
# Hypothesis testing

## Hypothesis 1

To test this one, we place a debugging section to be triggered when the amount to balance minus the amount to bet would equal the limit, and then we see if the balanceExceedsLimitBy method will return true.



Here we are at our breakpoint, and we can see we now have a balance equal to our bet. The next bet would potentially reduce the balance to 0 (the limit). We will step through until we reach balanceExceedsLimitBy and see what it returns.



Here we can see it is about to return false (method edited slightly to allow the return value to be shown in debugging).

So we have verified hypothesis 1.

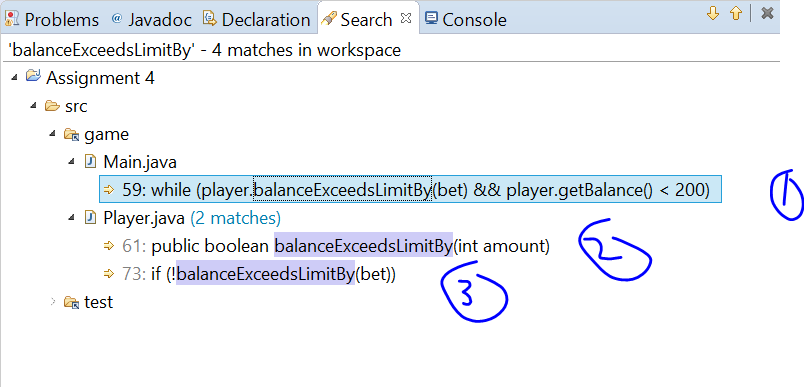
# Resolution

The cause of the bug is that it doesn’t allow the balance – the bet to equal limit. Therefore we can add an equals sign in the method balanceExceedsLimitBy.

## Risk assessment

Could this solution introduce any new defects?

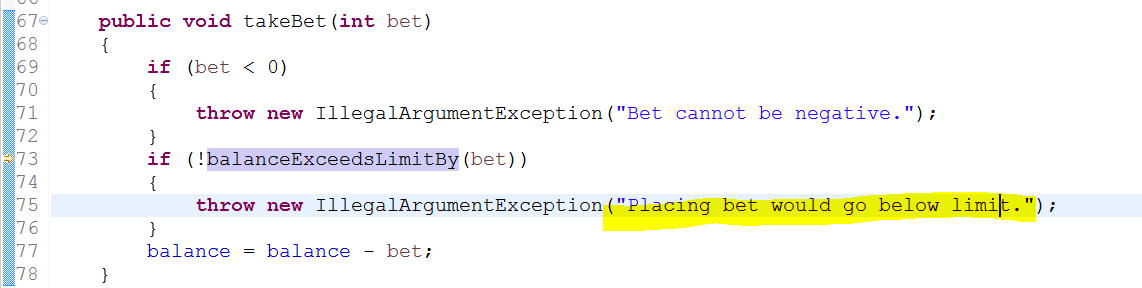
Maybe somewhere in the code, balanceExceedsLimitBy is being called and it is used in a slightly different way than expected, and is using this seemingly incorrect behaviour. We will do a search to see if it is being called anywhere else.



At (1) we have the while loop condition in main that has already been discussed.

At (2) this is just the method declaration.

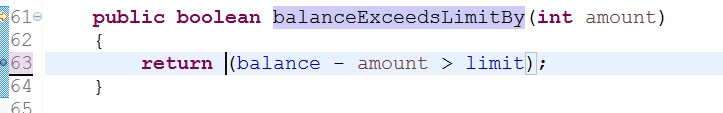
In (3) we need to have a look.



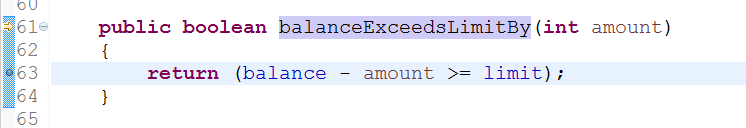
We can see that it is doing a test, which if it fails it throws an Exception with the message “Placing bet would go below limit”. We know that the method not only returns false if it goes below limit, but if it equals limit. So this means the method is exhibiting the wrong behaviour for this call as well, so fixing it will fix this potential problem here as well (maybe the Exception would get thrown when it equals).

### Testing the solution

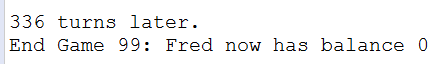
We are going from this:



To this:



So let’s test it, and see if the player can now get to a zero balance.



Yes, looks like he can!

# Conclusion

The bug is now resolved, and no new defects have been introduced by the resolution.