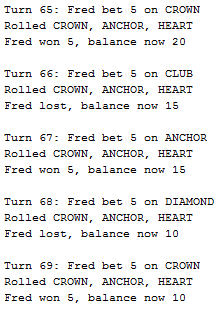
Jason Mortlock

ITC205 Assessment item 4

Debugging Journal

# Bug One

## Replication

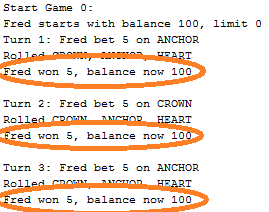


This output exhibits how the balance is not being increased whenever the player wins a match.

## Simplification

1. testBugReplication() method uses mockito to mock Player and Dice into the game
2. src/BugOneTest.java is the test file

## Tracing



Whenever a match is won, the balance does not increase. Assuming this bug is within the Game.java file. Noticed code within PlayRound() that stated if there are more than 0 matches, the player receives their winnings. However the player does not receive any money back. This meant that their bet was not being returned and therefore neither was their bet amount. This then meant that no money was being returned on a win and the balance was not increasing.

# Journal

## Hypothesis

Game is not adding wins to the players balance after winning match

## Predictions

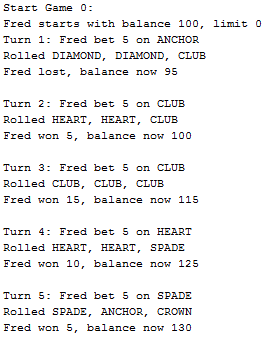
The balance is not being updated correctly after a winning match, or it is being incorrectly displayed

## Test

testHypothesis() is used in order to test that *1 \* the bet* is returned, as per the game rules. Meaning the player will receive their winnings and update the balance.

## Resolution

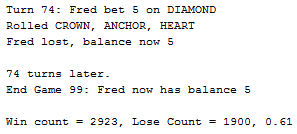
The code is not displaying the new balance after the outcome of the winning match. When a match has been played, in the playRound() method, if matches are greater than 0 and the match is won, return the original bet as well as receive winnings.



*Output above shows the balance now going up with a win*

# Bug Two

## Replication

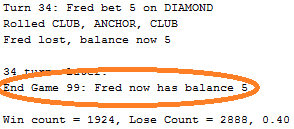


This output shows that the balance is not at 0 when the game ends, rather at 5

## Simplification

1. testBugReplication() uses mockito mocks of Player and Dice in the game to control the game.
2. Src/BugTestTwo is the file for the test.

## Tracing



The game was not ending the balance on zero but rather five. Assumed the bug was within Player.java since it was the players balance. Noticed code within the balaceExceedsLimitBy() method that it was returning when the balance – amount was > than the limit. Meaning that it would always stop before the balance was equal to the limit of 0, therefore stopping at 5. In order to fix this bug, the balance must be stopping when the bug was = to the limit.

# Journal

## Hypothesis

Limit test is not working correctly. Ending game has the balance being greater than the limit. Rather than greater than or equal to.

## Predictions

The test has the balance being greater than limit, not equals to. Therefore having the end balance less than 0.

## Test

testHypothesis() sets the player limit to zero, balance to 15 and bet value to 5.

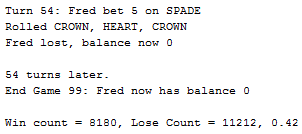
It will loop the makeBet() until balance is at the limit or an exception

Then a final balance will be displayed.

Output:

## Resolution

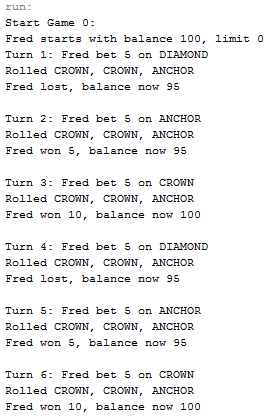
Hypothesis is true. The balanceExceedsLimitBy() method is incorrectly comparing the limit and the balance.



*Output above shows end balance now at zero.*

# Bug Three

## Replication



This output shows that throughout the game, the odds are changed due to the SPADE face never showing. As well as the rolls for each match not changing from CROWN, CROWN, ANCHOR in this case.

## Simplification

1. testBugReplication() uses mockito to mock a player into the game and replicate the result
2. src/BugThreeTest is the file for the test

## Tracing

In order to find the issue, it was assumed that the bug was something to do with the dice value, since they were the deciding factor of the outcome of the game. Searching through Dice.java and DiceValue.java, it was discovered that within DIceValue that the diceValue getrandom method was using an ordinal that was not including the spade. This was the reason why the spade was not showing up.

# Journal

## Hypothesis

The diceValue.getRandom() is not returning a random result that includes all faces.

## Predictions

That DiceValue is set, however the SPADE face is not included, and there is no randomisation occurring.

## Test

testHypothesis() will create a new game with 50 rolls. These 50 games will have random outcomes.

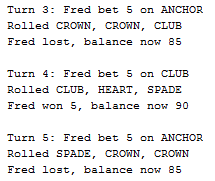
A counter is used to look for spades.

If a spade is found, the system prints out the number of spades found within the 50 games.

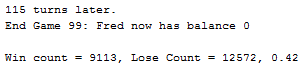
If there are spades found, then assert true.

## Resolution

DiceValuegetRandom() changed RANDOM.nextint(DiceValue.SPADE.ordinal()) to RANDOM.nextint(6) This rectified the spade missing and gave random results. Making the odds around 0.42



*Above output shows SPADE occurring in a match*

**

*Above output shows the odds being correct*

*NOTE:* Noticed a potential bug, didn’t go through the steps of replication and simplification.   
However noticed that the program once run, never actually stops. Went through the Main.java and found a method that called for a user to input ‘q’ and then hit enter in order to stop the program. This could potentially be a bug as once the games are over, the program should stop running and wait for the user. There is no prompt to tell the user to enter ‘q’, which means they will not know how to stop the program from running.