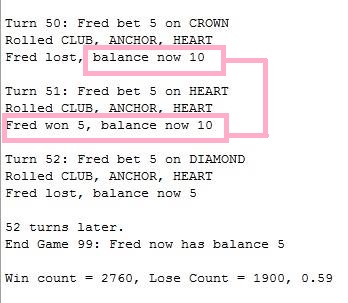
Hypothesis 1 – Bug 1

* Game does not pay out at correct level.
* When player wins on 1 match, balance does not increase.

The game must not be calculating the amount the player gives – i.e. the player bets 5 and he wins (so the game must be ‘returning the bet he placed and not adding the further winnings i.e. another 5).

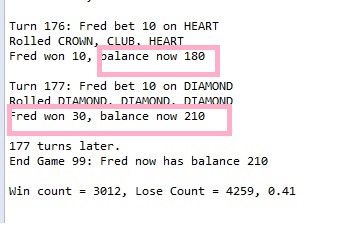
Balance was not increase (Before)

Tracking –

Going over methods to see which one is not returning or ‘setting the bet.

player.takeBet(bet); was missing from the if statement. It was in the incorrect spot.

By changing this this and testing with the junit it fixed the bug.

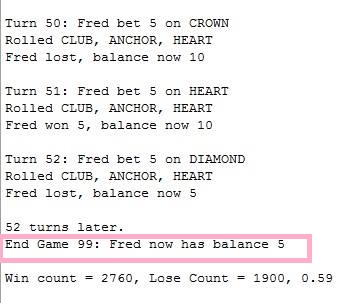
 (FIXED) Fred won 30 – and it added 30 rather than adding 25.

Hypothesis was correct. Taking the bet but not returning it.

Hypothesis 2 – Bug 2

* Player cannot reach betting limit:
* Limit set to 0, but game ends with player still with 5 (dollars) remaining.

There must be an error in the code at a particular spot not accounting for equal to limit.

Limit was set to 0 and game ended at 5 (Before)

Tracking –

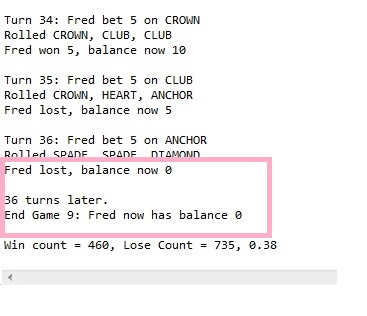
Looking for errors in the code. The code is hypothesised to be missing a part- simple error

Class player.class (method) playerExceedsLimitBy

balance - amount > limit (was the code) is saying if the balance – amount is greater than the limit the game will stop.

Whereas the game would stop at 5 because it’s greater – but wouldn’t allow it to go below because the limit is ‘0’ but cannot be equal.

The line was changed to balance - amount >= limit which allows the limit to be reached and there for the bug is fixed. An all junit tests passed. (No other code was effected.

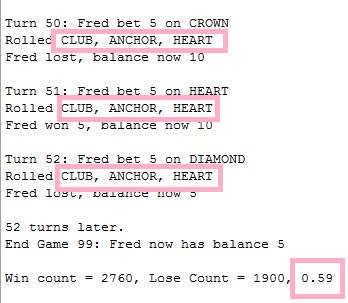


Hypothesis was correct. Tests passed. Not allowing to be equal to limit.

Hypothesis 3 – bug 3

* Odds in the game do not appear to be correct.
* Crown and Anchor games have an approximate 8% bias to the house.
* The win: (win + lose) ratio should approximately equal 0.42.

The game must be selecting at random the first draw of cards but for the next games played within the session it will be using the ones drawn from the first draw.

Continuously choosing the same card for each game (before)

Tracking –

Dice value class – was returning the same thing rather than changing it for each roll.

**public** **void** roll() {

value = DiceValue.*getRandom*();

The method was changed to (above) so each ‘roll’ would have a new set of dice.

Hypothesis was correct but another issue was found (as the tests didn’t return true).

Tracking for ‘bug 4’ which in turn was rather bug 3 also – because test were returning false.

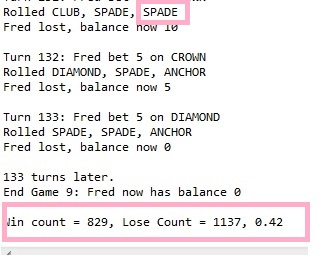
Within the DiceValue class the symbol ‘spade’ was not being picked. Is was changed to:

**public** **static** DiceValue getRandom() {

**int** random = *RANDOM*.nextInt(6);

which is allowing for all (6) of the cards to be picked because of the index.

Bug 3 – hypothesis was correct and all tests returned true.

(After)

Note: all bugs have the same junit test asserting the false and were found to be bugs because it would assert true for the incorrect values. (the asserts have been commented about because bug have been fixed but can be seen for tracking.