**Bug One:**

**UAT Case:**

Description:

* Game does not pay out at correct level.

Pre-conditions:

* Class files exist and are in a runnable state.

Post-conditions:

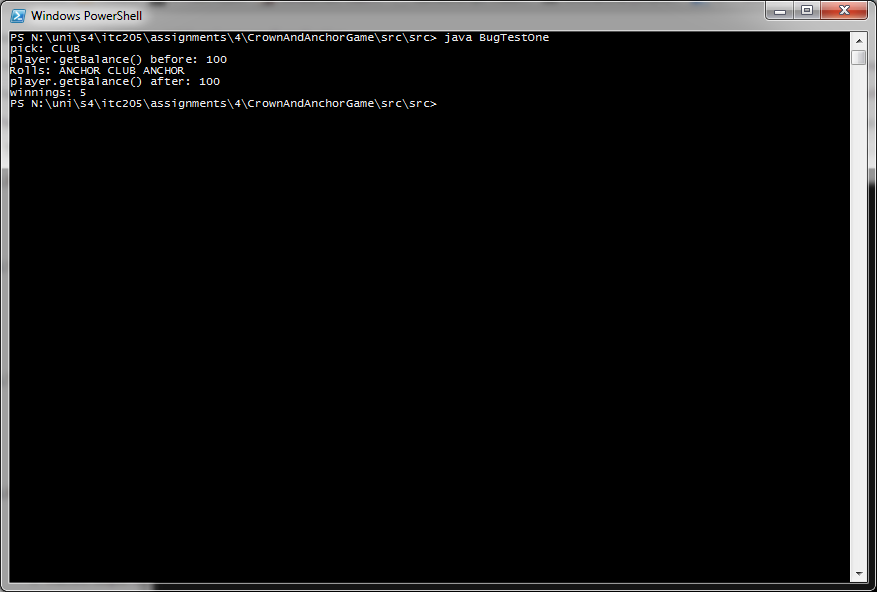
* Players balance does not increase.

**Simplification:**

Reduce code size so that only one round of the game is played, with additional print statements to show the value of variables at different points during execution.

See BugTestOne.java for the simplified code.

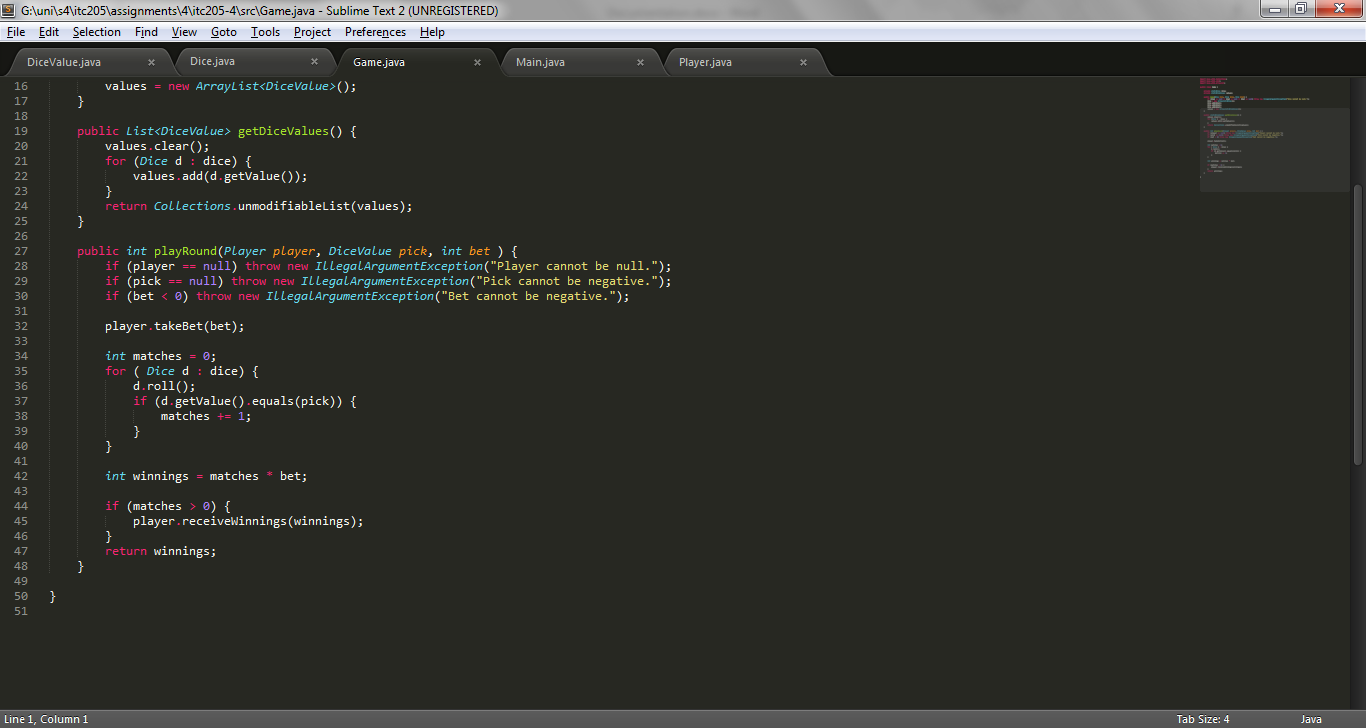
**Test Code Output**:



Note: Test code may have to be run more than once to get a single match.

**Hypothesis:**

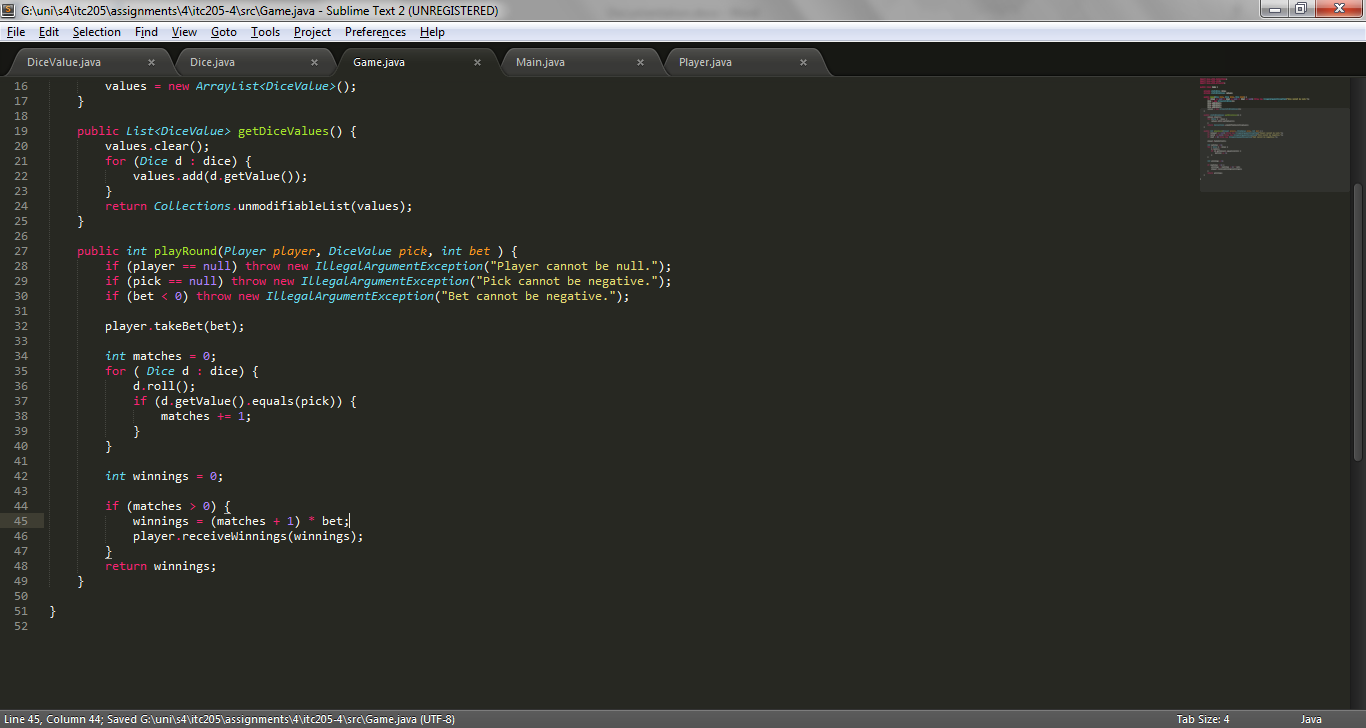
Winnings are not being calculated correctly when the player wins a round with one match. This must be taking place in the playRound method of Game, starting on line 27, as this is where the calculation of winnings is executed.



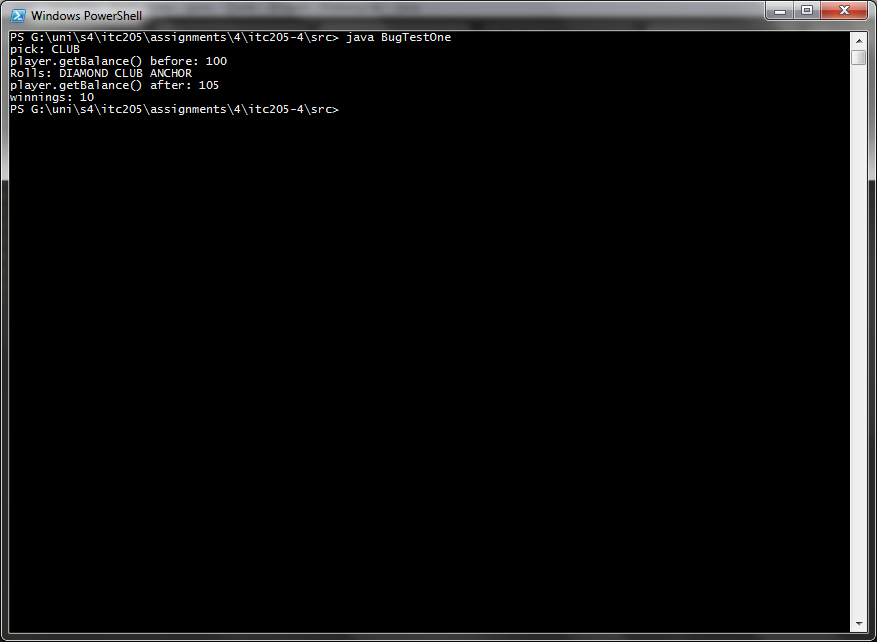
In the above screenshot, it is shown that the variable *matches* is initialised with the value 0. For each match the player gets, *matches* is increased by one. The variable *winnings* is initialised by multiplying the number of matches with the bet amount. So if the player gets no matches, *winnings* equals 0, and if the player gets one match, *winnings* equals 5. This is incorrect, as 1/1 odds gives 1 dollar for each dollar the player bets.

**Correction:**

The correct calculation of *winnings* is (matches + 1) \* bet. This will give the correct result. This can be implemented by initialising *winnings* with a value of 0, as if there are no matches *winnings* will be 0, and doing the correct calculation after checking if there are any matches.



**Output showing correction:**



As shown in the screenshot above, the test code now has the correct balance after the fix.

**Bug Two**:

**UAT Case:**

Description:

* Player cannot reach betting limit.

Pre-conditions:

* Class files exist and are in a runnable state.

Post-conditions:

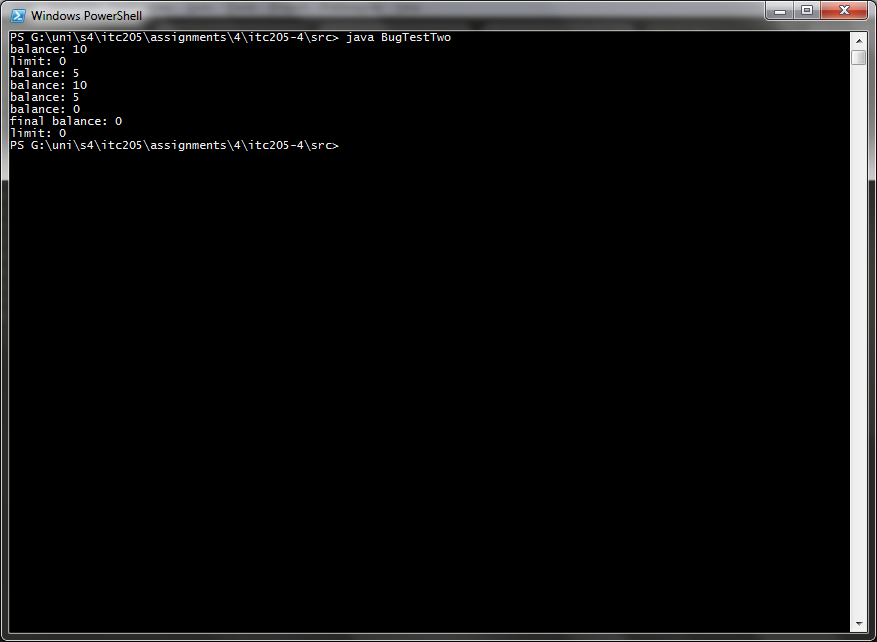
* Players limit is not reached.

**Simplification:**

Simply game loop of *Main*, setting the player balance to 10 to reduce the running time. At the start of the loop, print out the initial balance and limit, and at each step print the current balance. Once the loop is complete, print out the final balance to show that the initial limit was not reached, and the limit again to confirm that it was not modified.

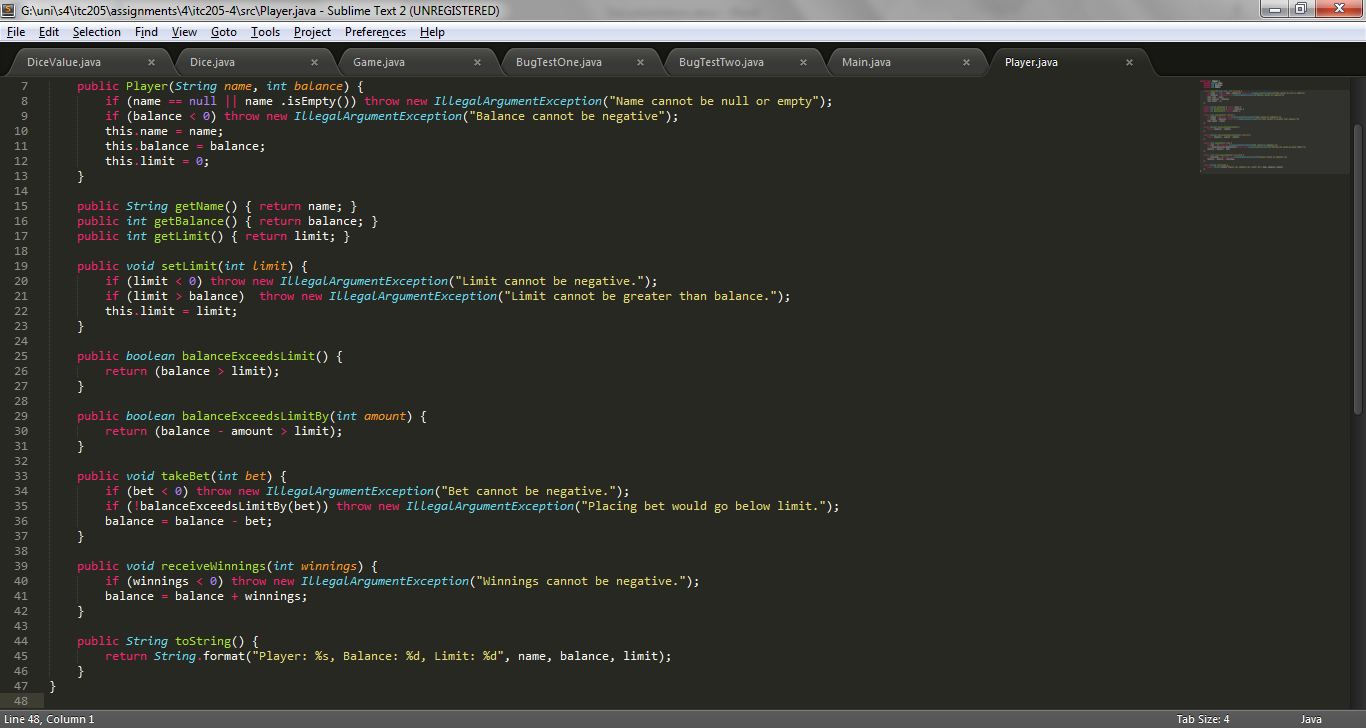
See BugTestTwo.java for the simplified code.

**Test code output:**

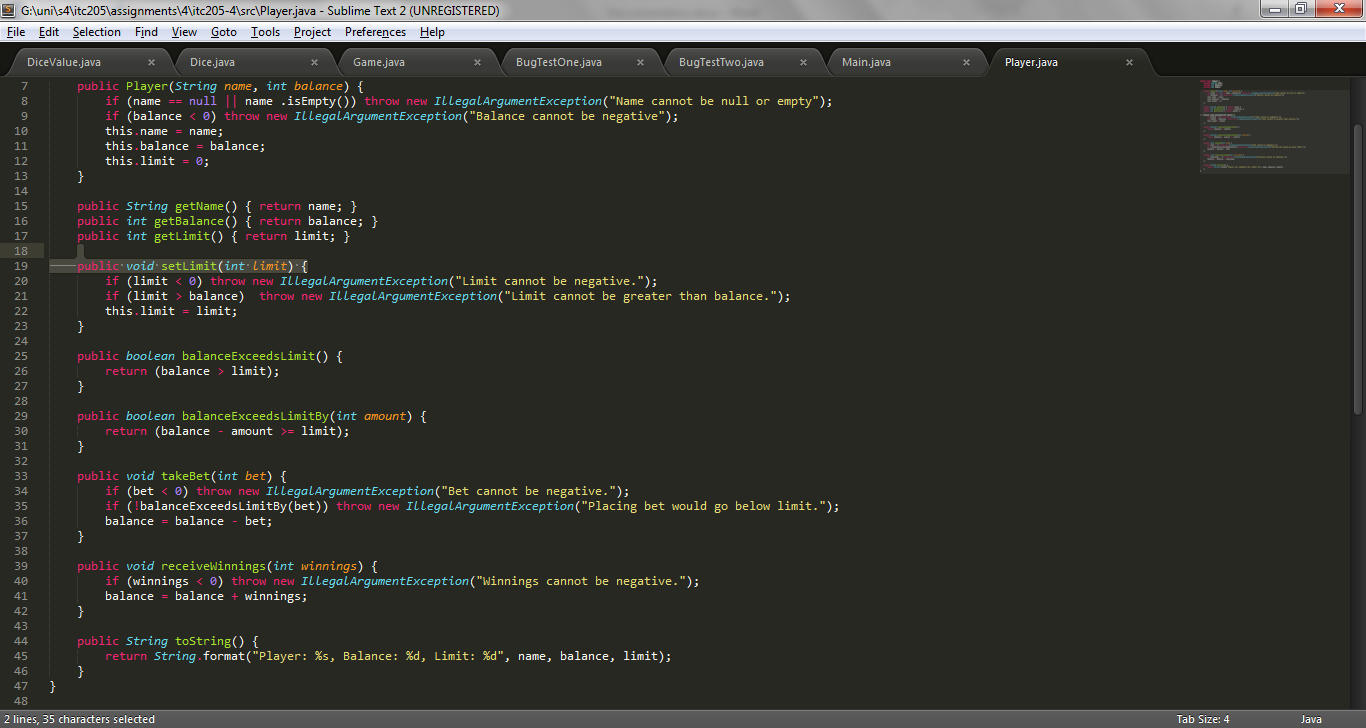


**Hypothesis:**

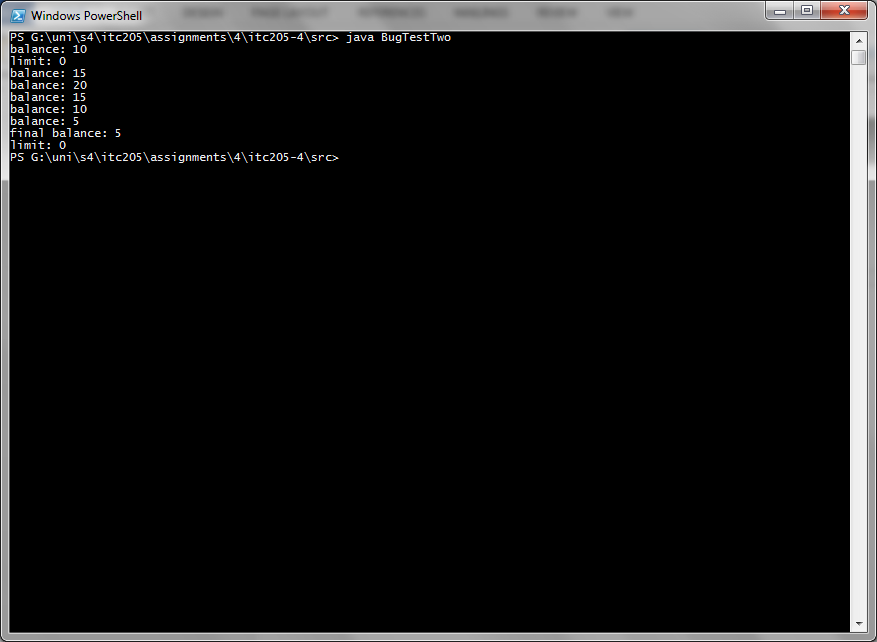
As shown in the test code output above, the loop has exited without the limit being reached and with the limit being the same value. Therefore, there must be a problem with the condition of the while loop, the *balanceExceedsLimitBy* function.



As shown in the screenshot above, *balanceExceedsLimitBy* only returns true when balance minus amount is greater than the limit. For example, when the limit is 0, balance is 5 and amount is 5, then *balanceExceedsLimitBy* returns false. *balanceExceedsLimitBy* should only return false when balance minus amount is less than the limit.



**Output showing correction:**



As shown in the screenshot above, the test code now reaches the limit.

**Bug Three:**

**UAT case:**

Description:

* Odds in the game do not appear to be correct.

Pre-conditions:

* Class files exist and are in a runnable state.

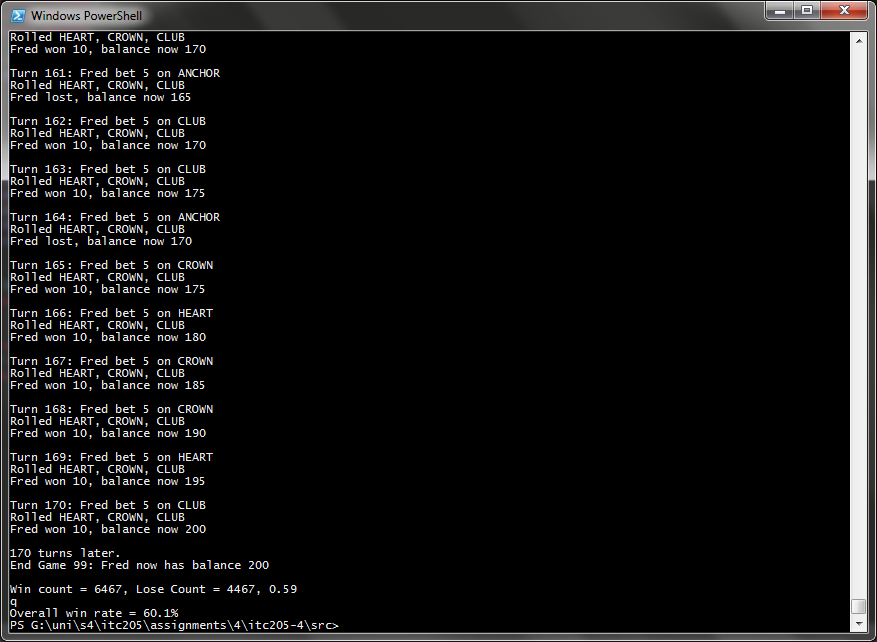
Post-conditions:

* The win/(win+lose) odds at end of game are not 0.42 in the houses favour.

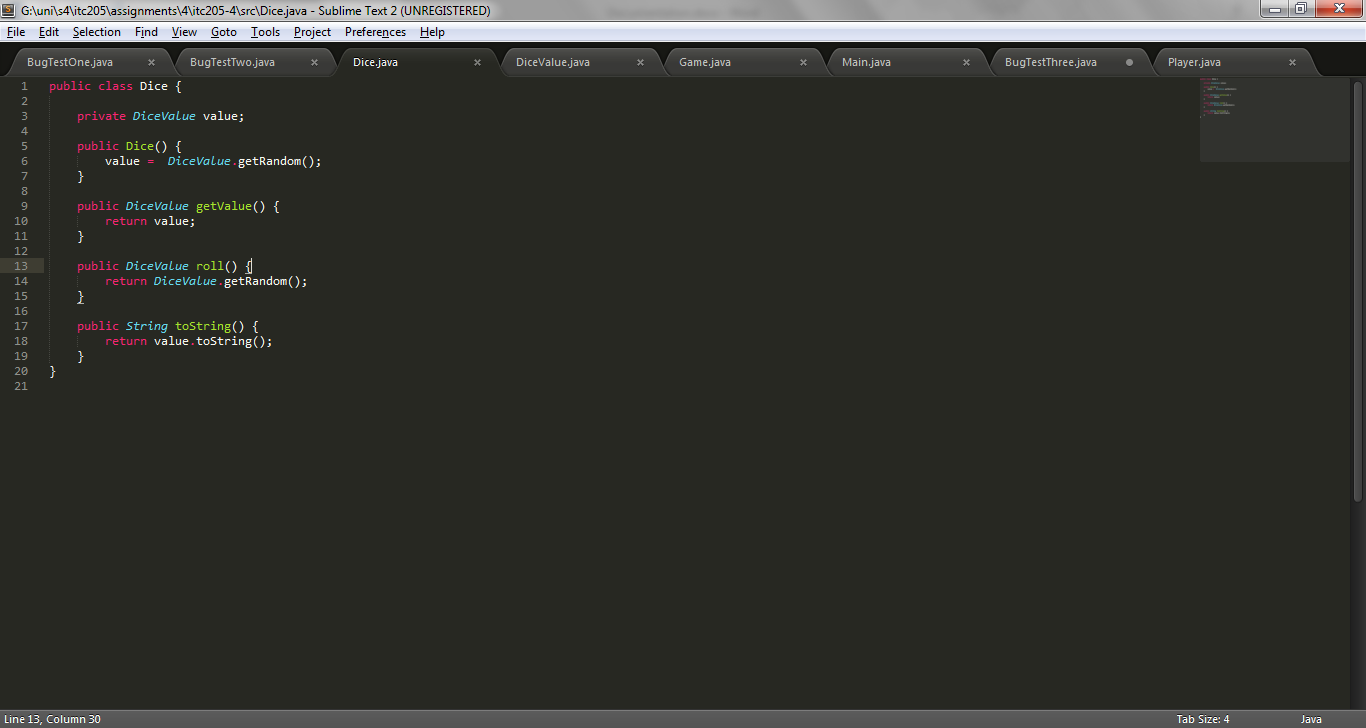
**Simplification:**

There is no simplification needed to test this bug, as *Main* is simple enough to reliably reproduce the bug and gives enough information to debug it properly.

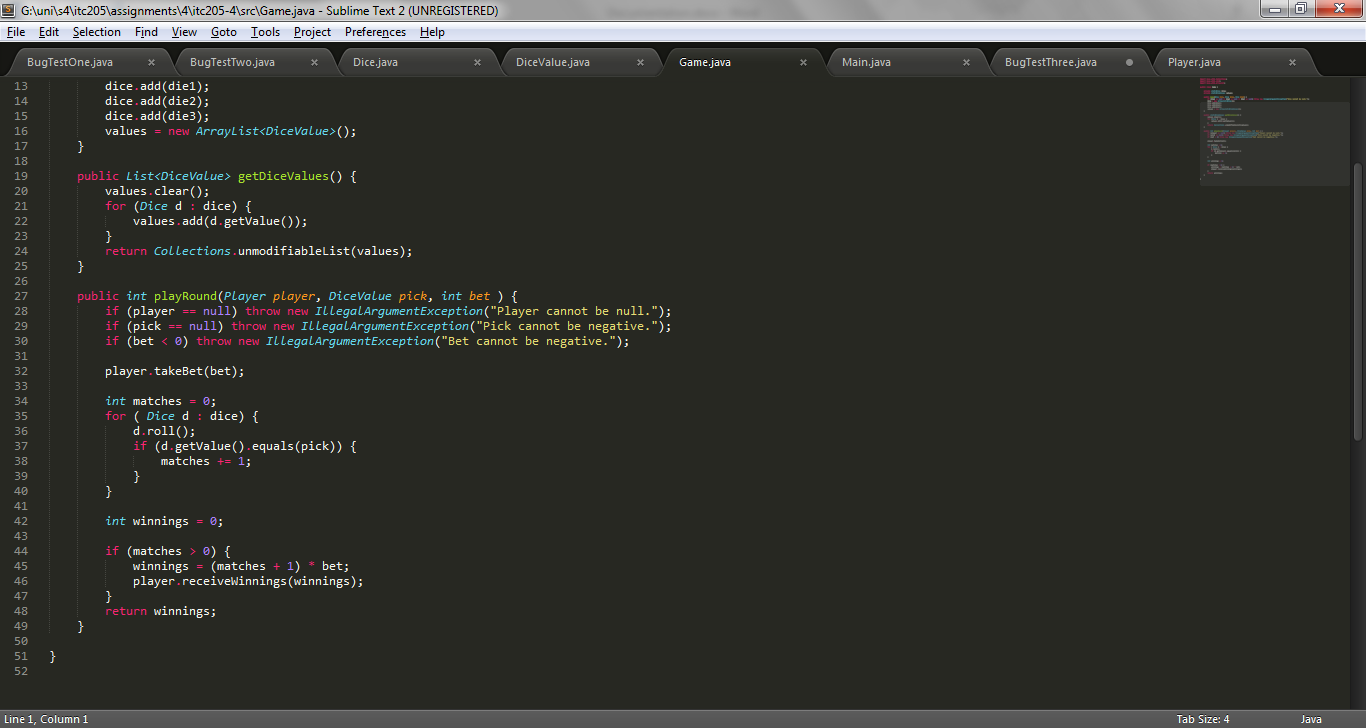
**Hypothesis:**



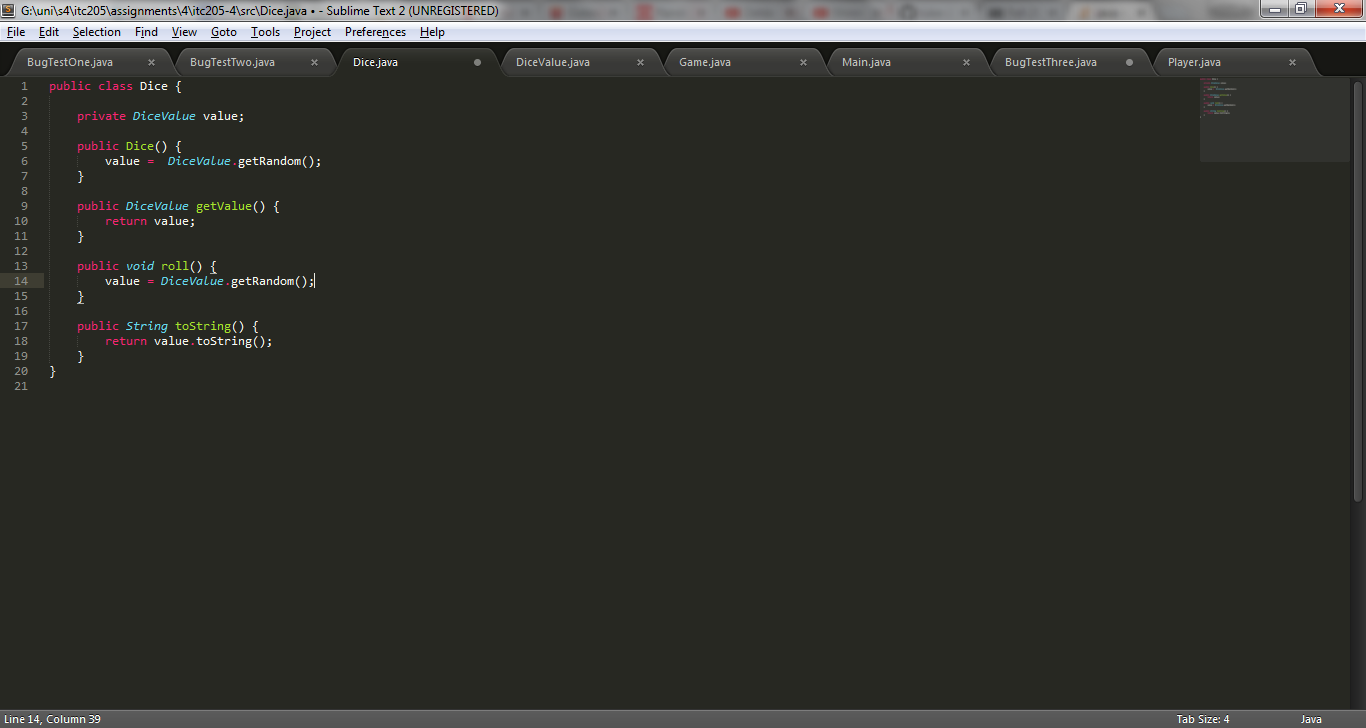
As shown in the screenshot above, each turn produces the same result. Looking at the code for the *roll* function in *Dice*, as shown in the screenshot below, the value of the Dice does not change, and the same value is returned each time the function runs.

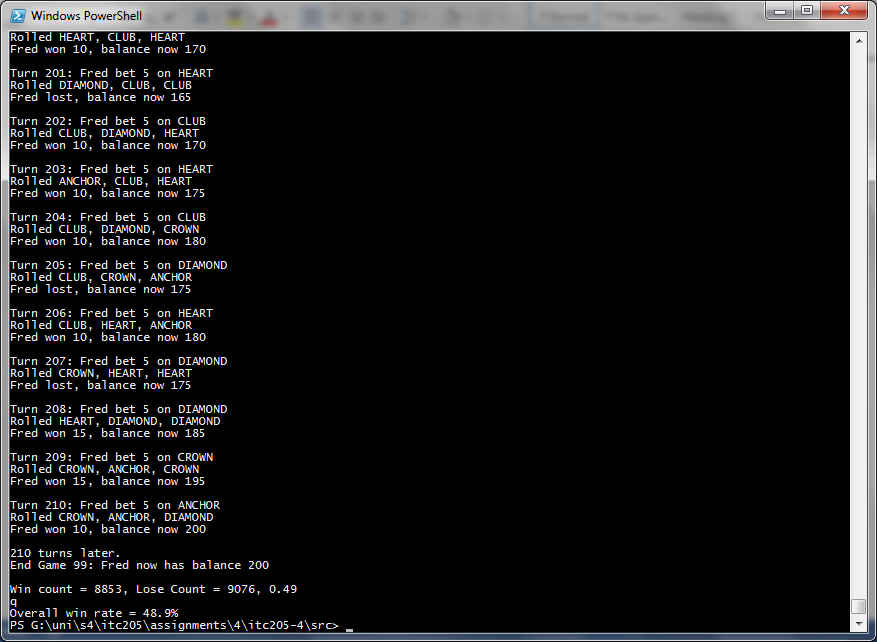


Also looking at the place where *roll* is called, in the *playRound* function of Game, it seems as though that this is not the way the *roll* function is supposed to be implemented. See the screenshot below.



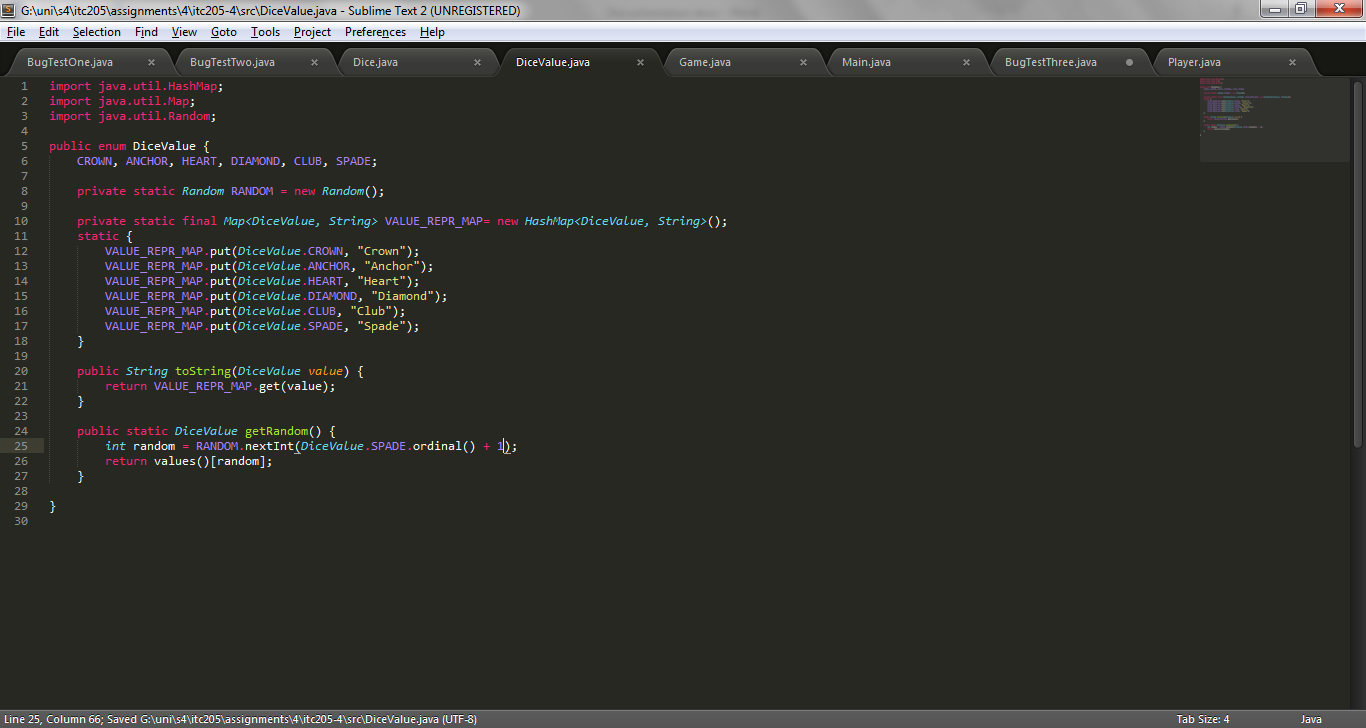
The way that *roll* is run here, it does not seem that there is any need to return the DiceValue in *roll.* The way this is supposed to work is that for each die, the dice is rolled, and then the value of the dice is checked. Shown in the screenshot below is the fix, and below that the output of Main after the fix.



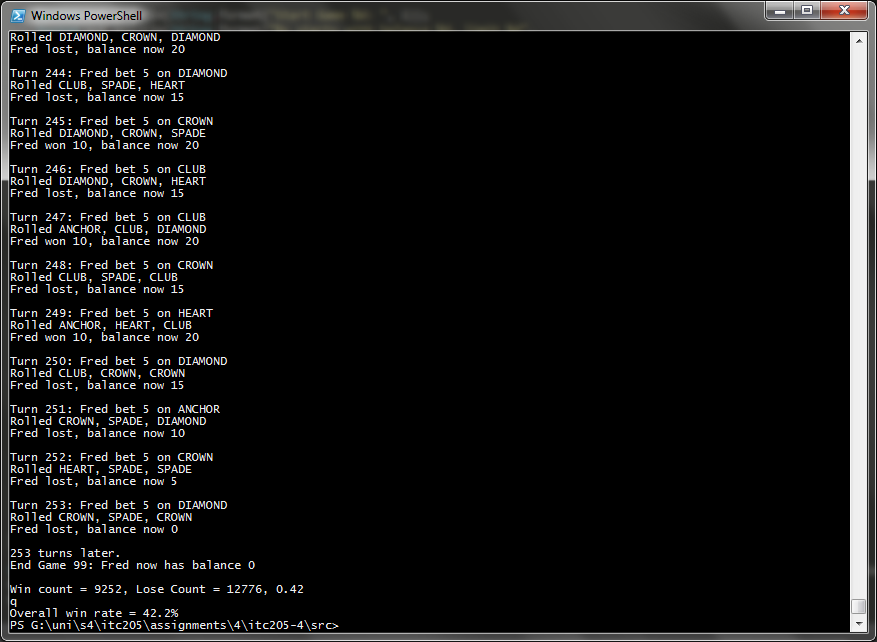


However, this does not fix the bug completely, as now each game as approximately 50% overall win rate.

Looking through the rolls, it appears that SPADE never occurs. Looking at the function responsible for determining the value of the dice, *getRandom* in DiceValue, the upper limit of the random number generating function, *nextInt,* is the ordinal of SPADE, which is 5. The way *nextInt* works is that the parameter is the exclusive upper limit, so just adding one to this number will result in SPADE occurring in rolls, as seen in the screenshots below.



**Output showing correction:**



As shown, the win rate is now approximately 0.42, and SPADE is being rolled.