**Bug Fixing Report**

**ITC 205**

**Assessment 4**

**Dylan Qiao**

**Contents：**

**Bug 1：**Game does not pay out at correct level.

User Test 3

Test Output (Failing) 3

Debug Log 4

Before and After Screenshots 4

Test Output (Fixed) 5

User Test (Fixed) 6

**Bug 2:** Player cannot reach betting limit.

User Test 6

Test Output (Failing) 7

Debug Log 7

Before and After Screenshots 8

Test Output (Fixed) 8

User Test (Fixed) 9

**Bug 3:** Odds in the game do not appear to be correct.

User Test 9

Test Output (Failing) 9

Debug Log 9

Before and After Screenshots 11

Test Output (Fixed) 13

User Test (Fixed) 13

**Bug 4:** Additional bug

User Test 13

Test Output (Failing) 14

Debug Log 14

Before and After Screenshots 15

Test Output (Fixed) 16

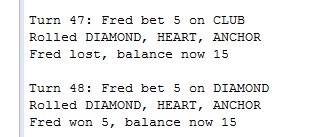
User Test (Fixed)

**Note:**  16

**Bug 1：Game does not pay out at correct level.**

User Test:

When Fred win a game, his balance should be increase, however, his balance did not change in the output.

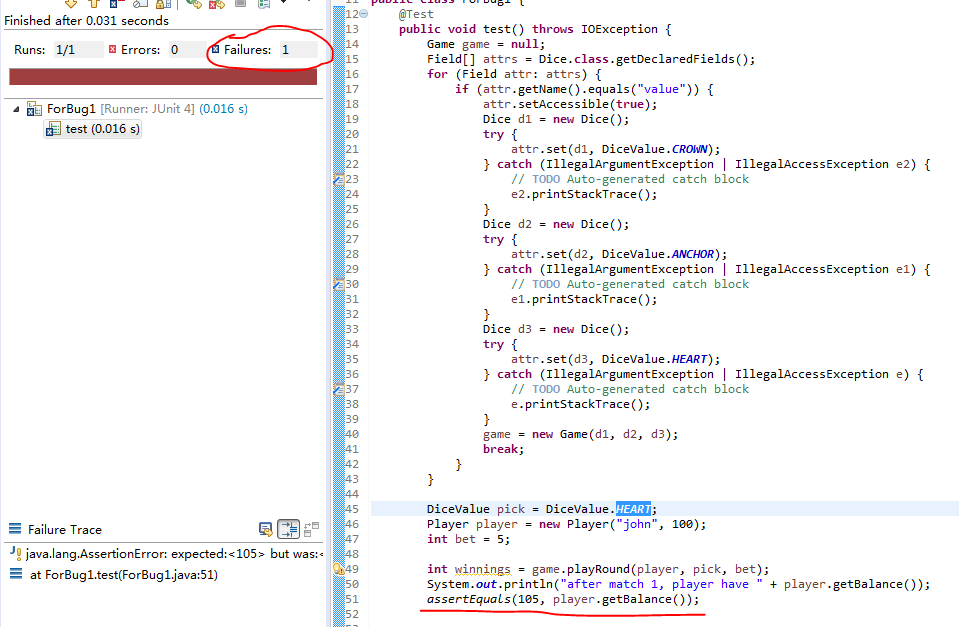


In this step, i assume that the problem might happen in the ***Game*** or ***Player***classes.

For test the bug, i set the all three dice value as ***CROWN, ANCHOR, HEART*** and i also set a player that have ***balance = 100*** and i made 5 dollars bet on ***HEART***.

Base on the game rule, the final balance should become to 105, however the test result is 100, so its obvious that the system did not add ***bet*** into ***balance***.

**Test Output (Failing):**



The test is fail as the expected.

**Debug Log:**

I start to check from the ***Player, Game*** class to see how it can effect to Balance.

Hypothesis: ***Player.getBalance*** is not showing the balance correctly.

Test (Junit): Ensure a player created with $100 returns 100 from ***Player.getBalance***.

Result: Pass.

Conclusion: ***Player.getBalance*** works as intended. Hypothesis is incorrect.

Hypothesis: winnings variable is not being calculated correctly.

Test: Set a breakpoint on line 33 for 10 winnings at line 46 ***(player.receiveWinnings(winnings))*** with 2 matches and a bet of 10, and ensure winnings == 30. If winnings = 20, the hypothesis is correct.

Result: Fail. Winnings = 20.

Conclusion: ***winnings*** is not being calculated correctly.

**Code change**:

***Game***: playRound

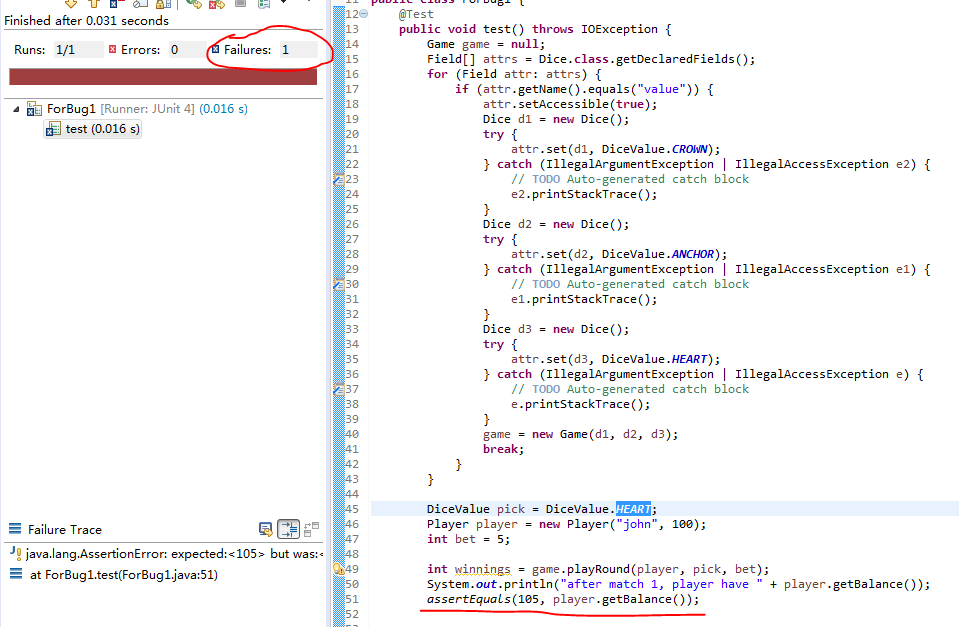
if (matches > 0) {

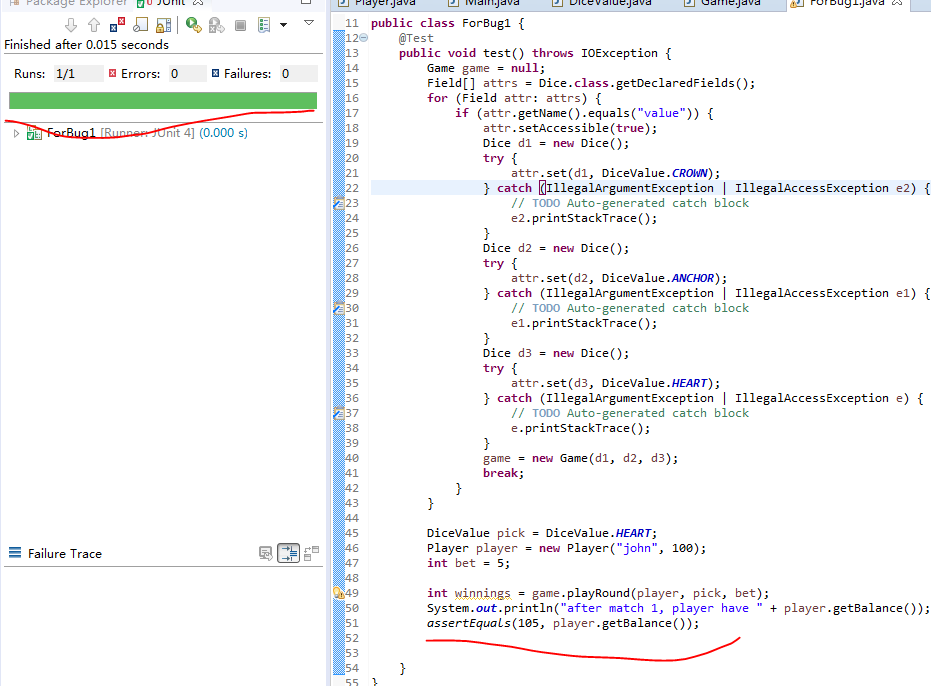
player.receiveWinnings(winnings + bet);

}

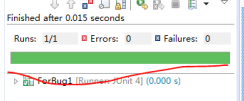
**Before and After Screenshots:**

Before:



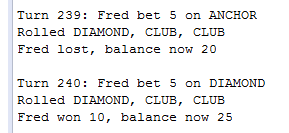
After

**Test Output (Fixed):**



**User Test (Fixed):**

Output for a winning bet is given below:



**Bug 2: Player cannot reach betting limit**

User Test:

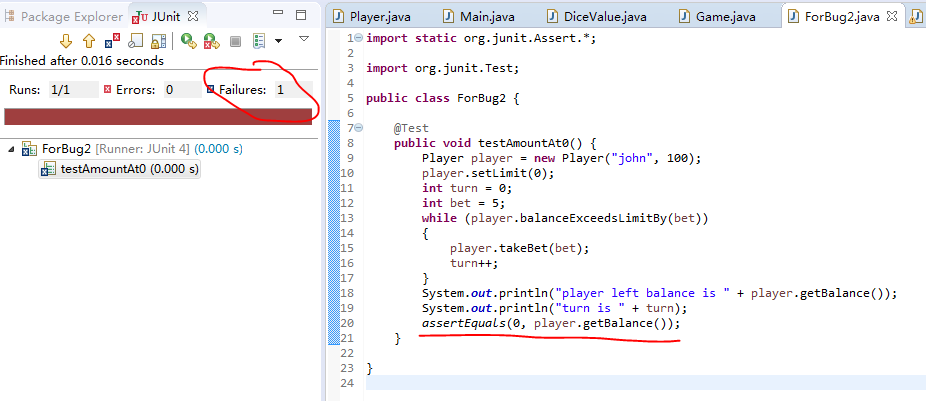
Run the program until the player stop because he has reached his limit. Makesure the balance is 0. however, the test is fail because the balance is always 5 when the player stop.



Set a user (***Balance = 100***), set ***limit = 0*** and make a lot of bet , each bet will be 5 dollars and never wins.

Base on the game rule, the user balance should be 0, however, because of the bug, the result is always 5.

Test Output(Failing):



**Debug log:**

Because the faulty function is only 1 line, so its clear that what is wrong.

Hypothesis: ***balanceExceedsLimitBy***’s calculations are incorrect.

Test (Junit): ***testBalanceExceedsLimitBy\_AllOfBalance***

Result: FAIL. Returns false.

Conclusion: ***balanceExceedsLimitBy*** is not being calculated correctly so the hypothesis is correct.

Code change:

***Player***: balanceExceedsLimitBy

When balance = 5, 5-5>0 is incorrect and program will be stop, actually, when the balance = limit , the program should be work as well.

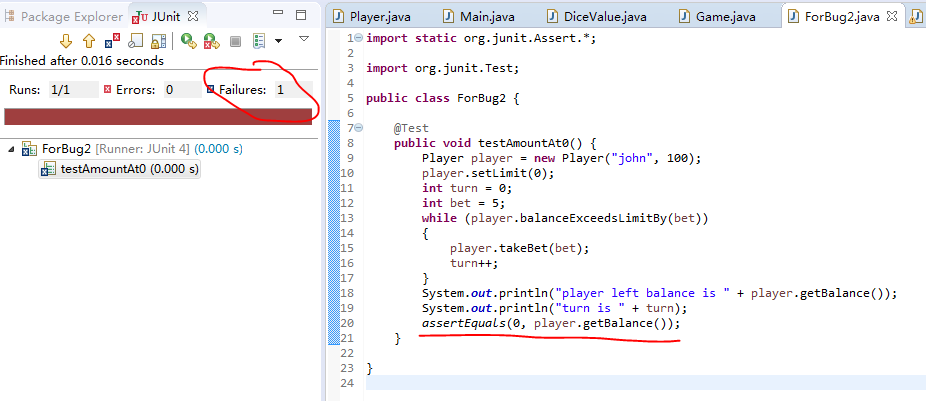
public boolean balanceExceedsLimitBy(int amount) {

return (balance - amount >= limit);

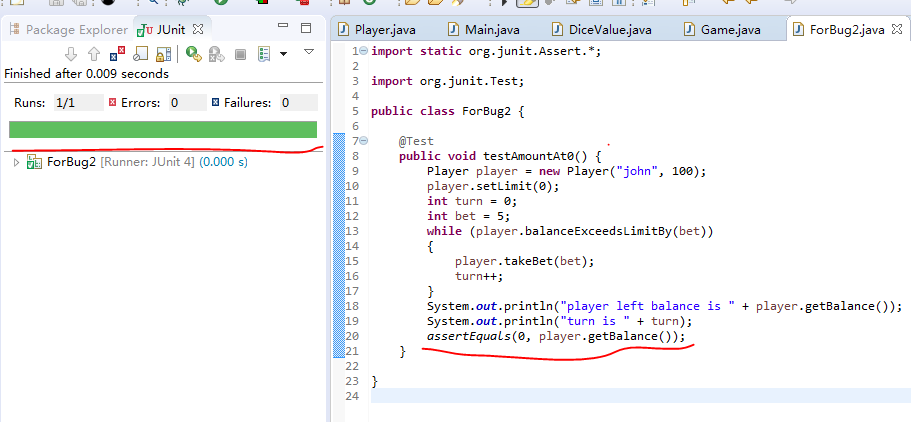
}

**Before and After screenshots:**

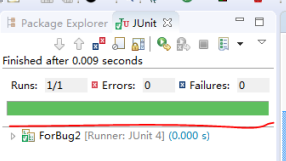
Before:



After:



**Test output (Fixed):**



**User Test Output (Fixed):**



**Bug 3: Odds in the game do not appear to be correct.**

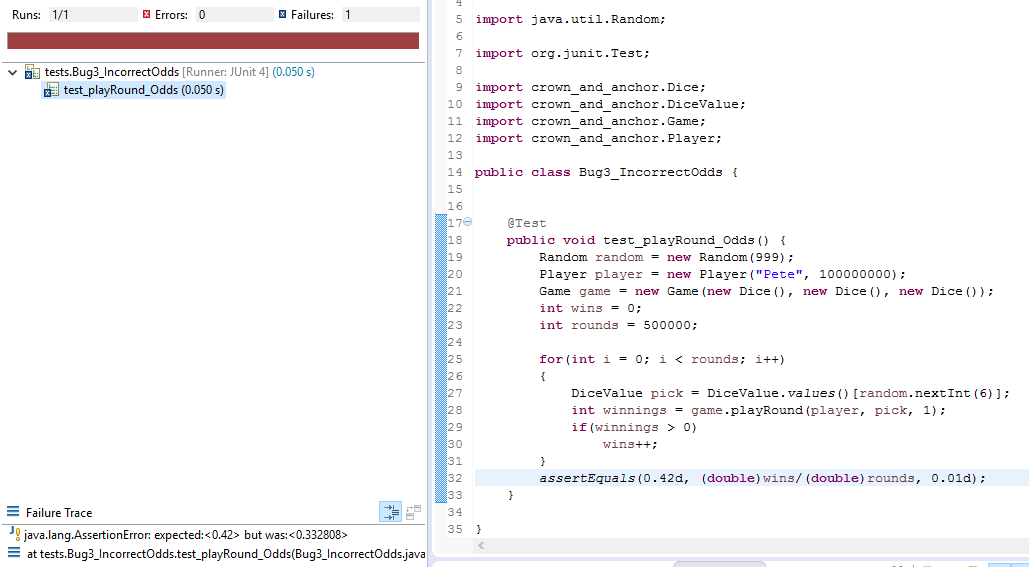
## **User Test:**

Run the program many times and result for win/loss ratio is close to the expected win/loss ratio of 0.42.

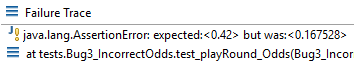
**The test currently fails because the ratio fluctuates wildly, from 0.2 to 0.6.**

****

**Test Output (Failing):**







**Debug Log:**

No matter how many iterations of playRound are run with random picks, certain values come up for the win/loss ratio. 0.16, 0.33, 0.5 are what i found, that makes me feel the result is not actually “random”, for proof that , the first thing going to test will be how is the dices work.

Hypothesis: Dices aren’t random.

Test (Junit): Assert that the chance of each of the 6 faces rolling on the dice is 1/6.

Result: Fail. The odds for all faces except Spade is 0.20. The odds for Spade are 0.

Conclusion: Hypothesis is correct; the bug is somewhere in Dice.roll which is a single line that calls ***DiceValue.getRandom***.

Hypothesis: Changing line 26 to int ***random = RANDOM.nextInt(values().length)*** will result in the game functioning correctly.

Test (Junit): Assert that the chance of each of the 6 faces rolling on the dice is 1/6 with the line changed.

Result: Partial Success. The Dices now work correctly.

Conclusion: While the dices now work correctly, the game still gives the same results.

Looking at how the ***Game.playRound*** interacts, with the Dice, it doesn’t actually use the value returned by roll, but instead gets the value using ***Dice.getValue***. For this reason, the next step is to check that the value returned by ***Dice.getValue*** is the same as that returned by roll.

Hypothesis: ***Dice.getValue*** is not returning the random values.

Test (Junit): Assert that ***Dice.getValue*** returns the value that was rolled.

Result: Fail. ***Dice.getValue*** returns the correct value 1/6 of the time.

Conclusion: The issue is in ***Dice.getValue***. Hypothesis is correct.

The next step is to look at where the value being returned by ***Dice.getValue*** is being set. It is only being set in the constructor, which explains my initial suspicions regarding something happening during initialization that is determining the results in some way.

Hypothesis: Setting value to the random face before returning it from Dice.roll will make the game work properly.

Test (Junit): Assert that the chance of winning the game is 0.42 after making the change.

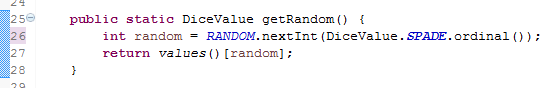
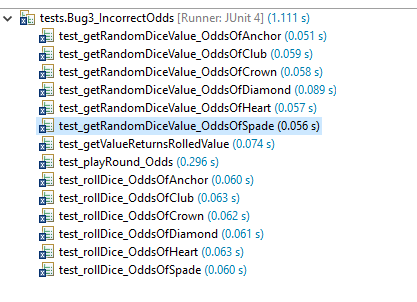
Result: SUCCESS. The Dices now work correctly.

Conclusion: The hypothesis is correct; the game odds are now correct.

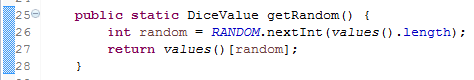
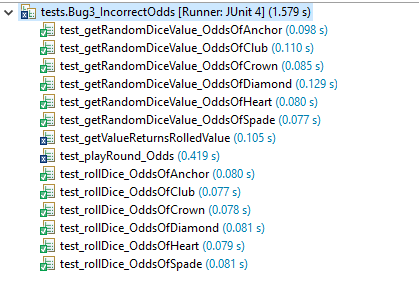
The odds of the game now appear to be correct.

**Before and After Screenshots/Fixed Test Output:**

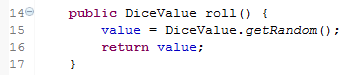
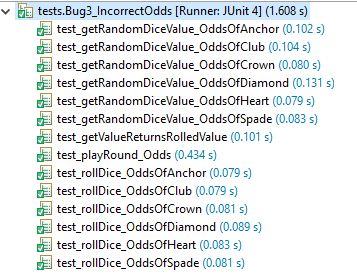
Without fixes:



With first fix:



With both fixes:



**User Test Output (Fixed):**

Note that the win/loss ratio does vary slightly, but is always 0.42 +/- 0.01. This is because the sample size is not large enough to always be exactly 0.42.



**Additional bug**

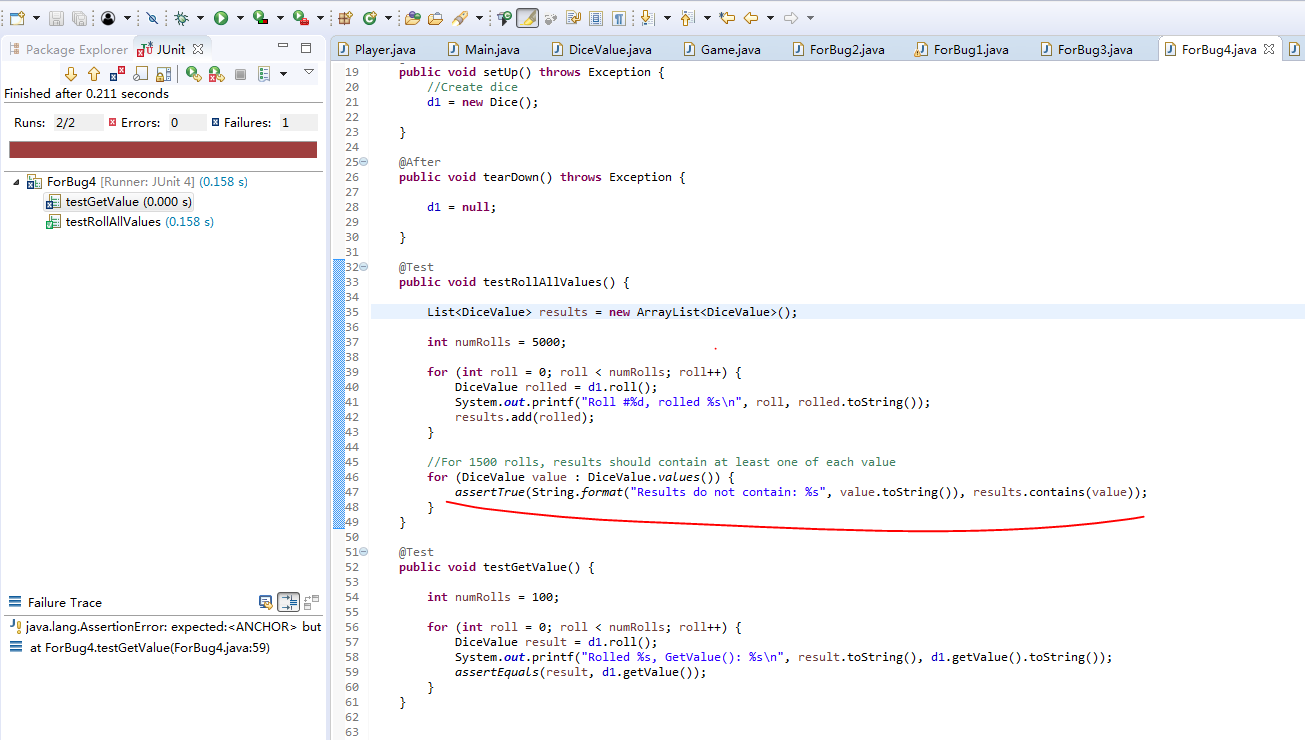
**Bug 4:**

## **User Test:**

The three dices won’t change after first roll. The three dices will be change in every roll.

Assume that make 5000 times dice roll, the value should be cover all the possibilities.(ANCOR, CROWN, HEART...)

**Test Output (Failing):**



It does not cover all the possibilities, it means some thing wrong in the code.

**Debug Log:**

Hypothesis: Dices roll aren’t random.

Test (Junit): Did value contain random roll value.

Result: Fail. The total value does not contain random roll value, the value is only happen in the Dices function, that could explain why only the first roll effect the value number.

Conclusion: The problem happen in the ***DiceValue.getRandom()***

Code change:

public DiceValue roll() {

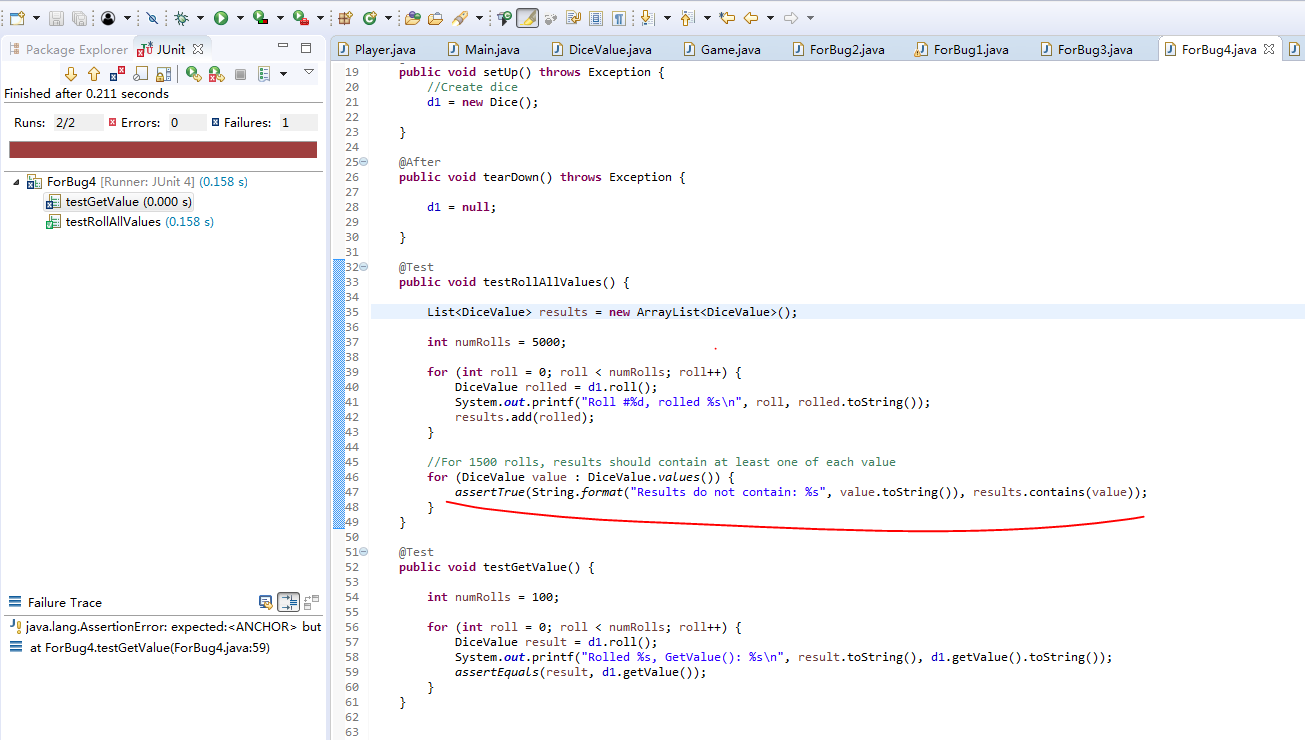
value = DiceValue.getRandom();

return value;

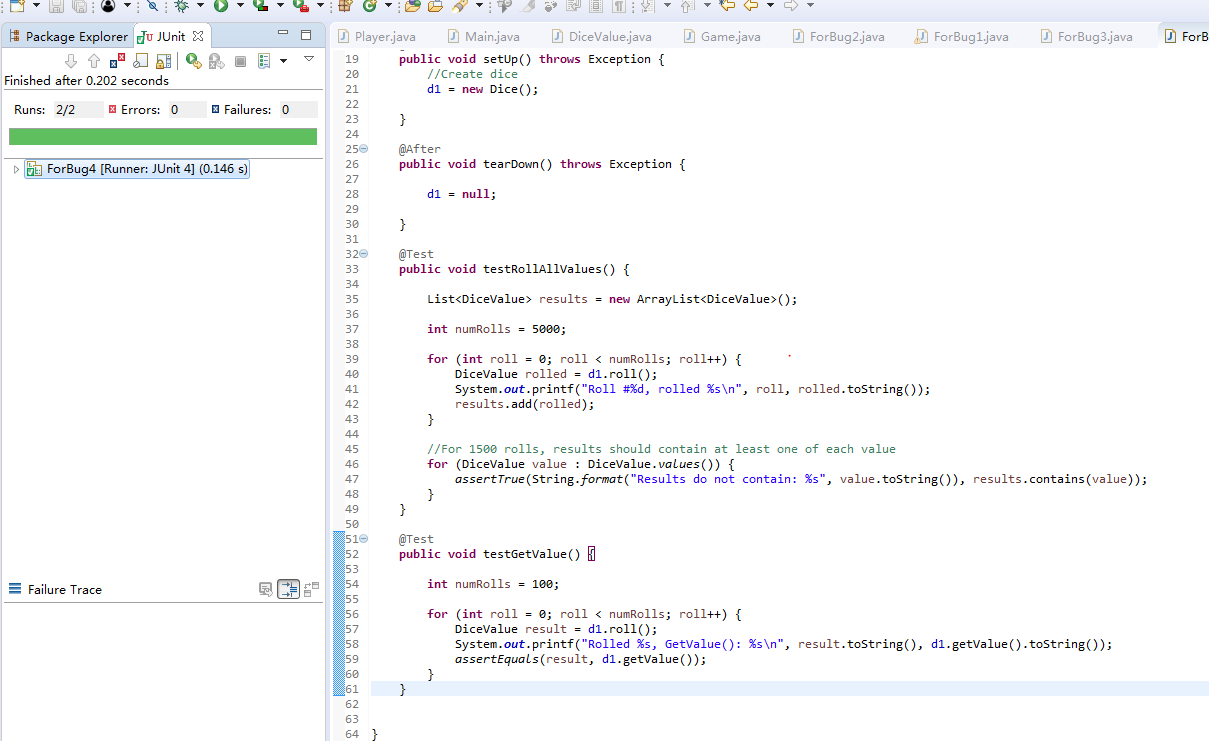
}

**Before and After Screenshots/Fixed Test Output:**

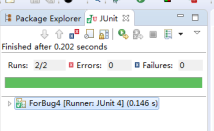
Before:



After:



**User Test Output (Fixed):**



**Note:**

**Those two things is what i found in the program, i don’t think this is bugs, its miss function.**

* The player can bet 0 for some reason. This could get annoying for the dealer.
* The player cannot place multiple bets on a single roll, which seems to be a key mechanic of the game.