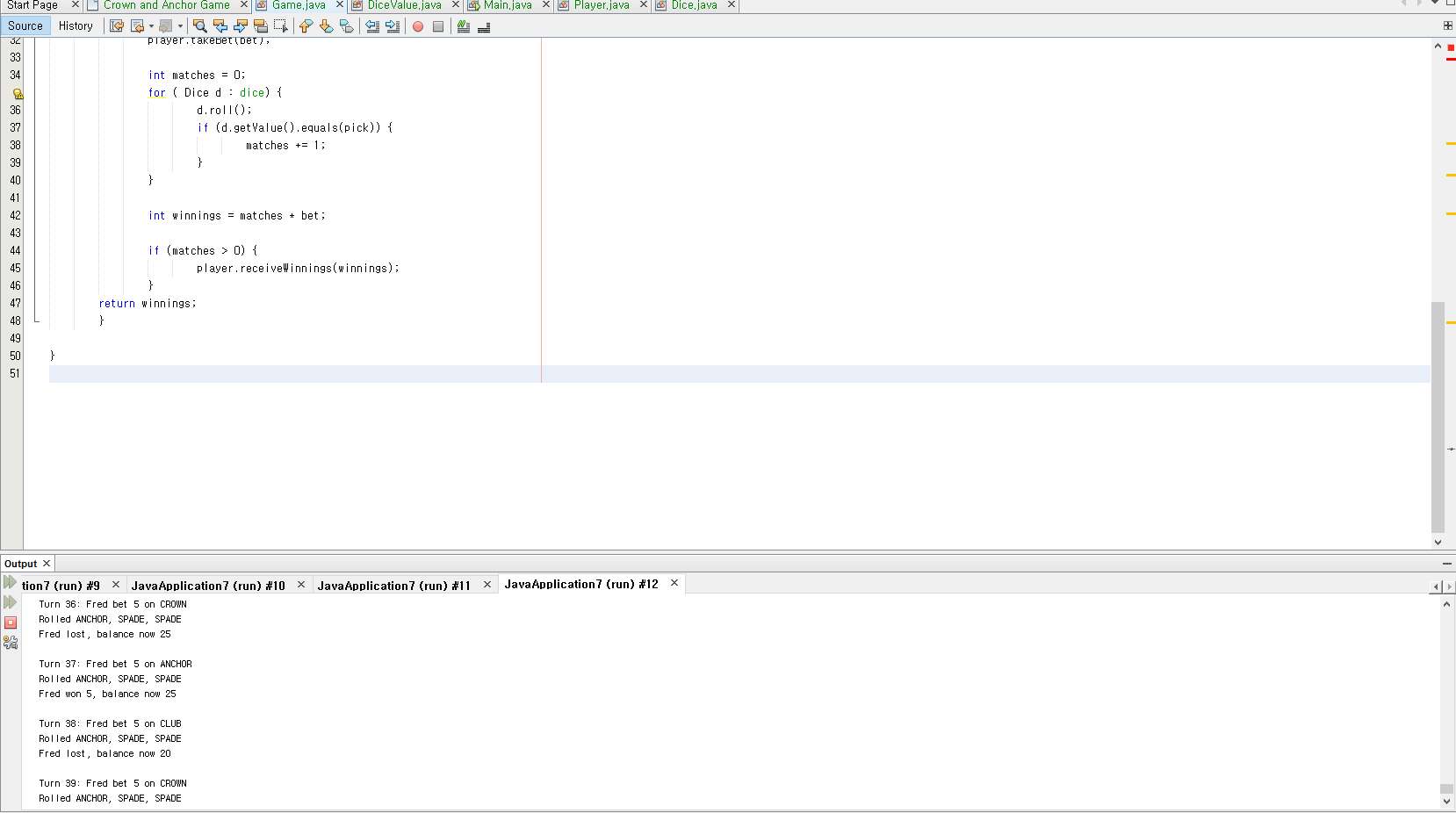
**Demonstration / hypothesis and J Unit Testing for each Bug**

# Bug 1

If the player wins the game the balance won’t increase



As you can see from bug 1, even though player wins the game the player point won’t increase for some reason that is one of the major bug of the system.

Idea 1:

Just simply putting + bet in line 45 where it says

If {matches >0 { player.receiveWinnings(winnings + bet); }

It will fix the problem as you can see the result.

Idea 2:

You can add the new method like this

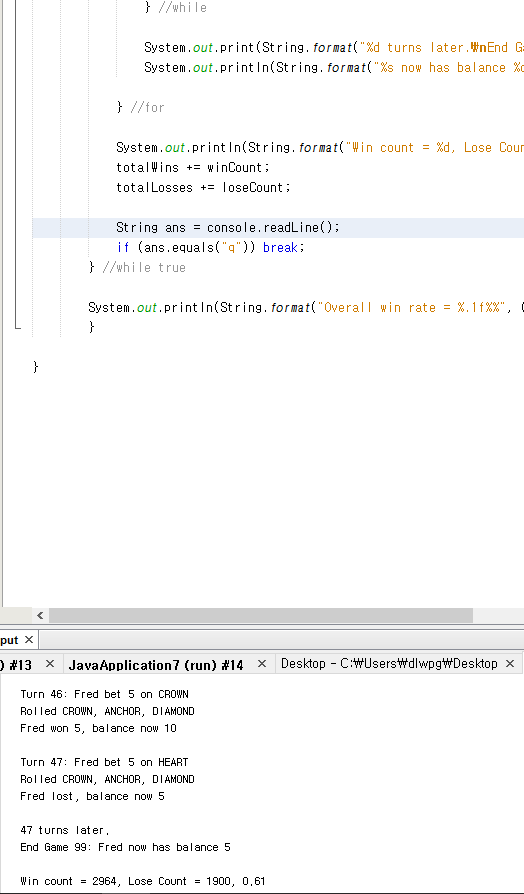
You can add new class for player point

Int winnings = matches + player point



# Bug 2

As you can see from bug 2



Player still has 5 points in the output and game ends, this is an error because game should end with player point 0 not 5.

Hypothesis for this bug 1, there are two ways of fixing this bug.

View following:

Idea 1:

Just simply putting + bet in line 30 where it says

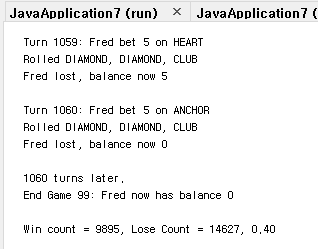
Return (balance – amout >= limit);

.=

It will fix the problem as you can see the result.

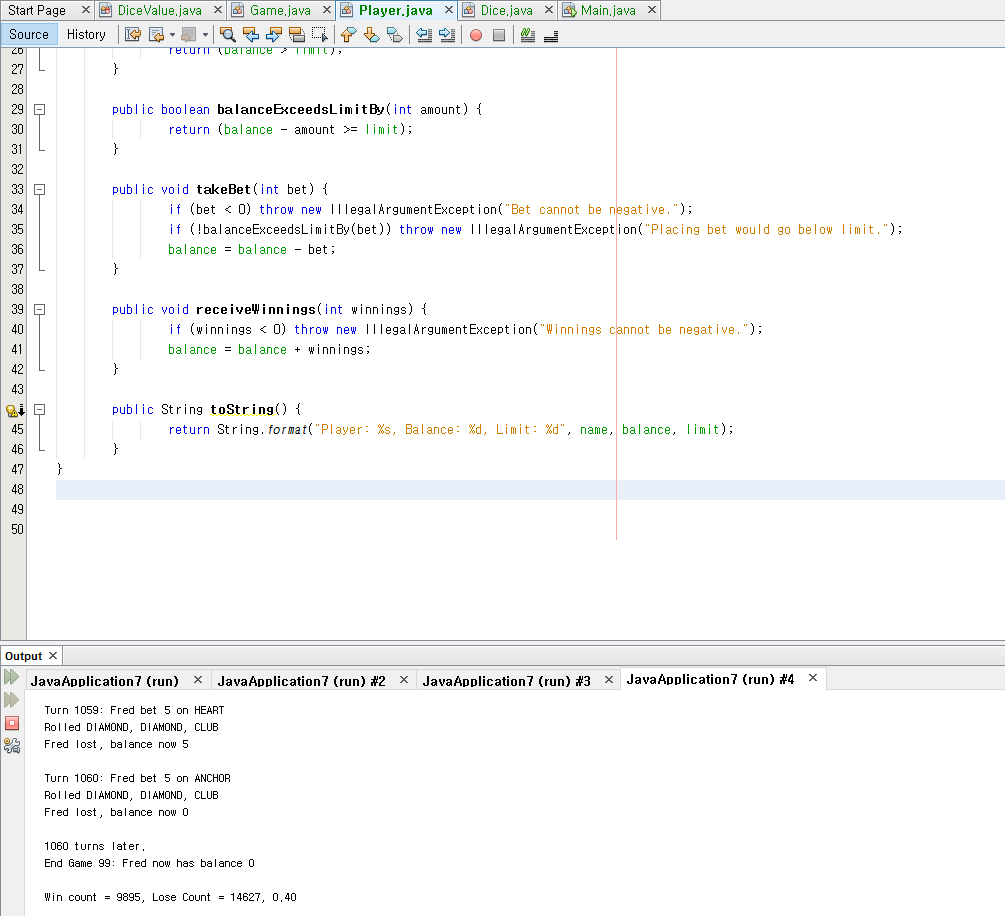
Idea 2:

The other way is to set the player point to null from the start and if the game starts automatically gives player 5 points but this didn’t work because for that method I had to make another method for player{ and game java class that was too complicated.



# Bug3

Crown and Anchor games have an approximate 8% bias to the house. So he wins: (win+lose) ratio should approximately equal 0.42. This does not appear to be the case.



If you see the program it wont appear to match with 0.42 value. Because win ratio wasn’t set in the method under dice value.java

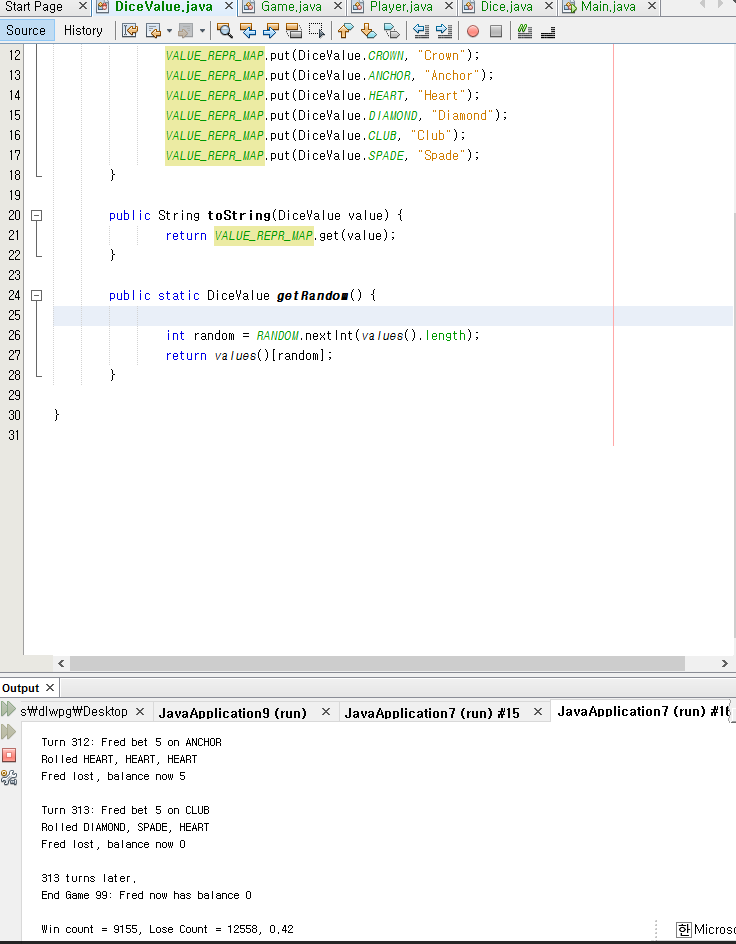
Here an idea to fix the issue.

Idea 1:

Just simply creating an int random = RANDOM.nextInt(values().length); and return its values as a random.

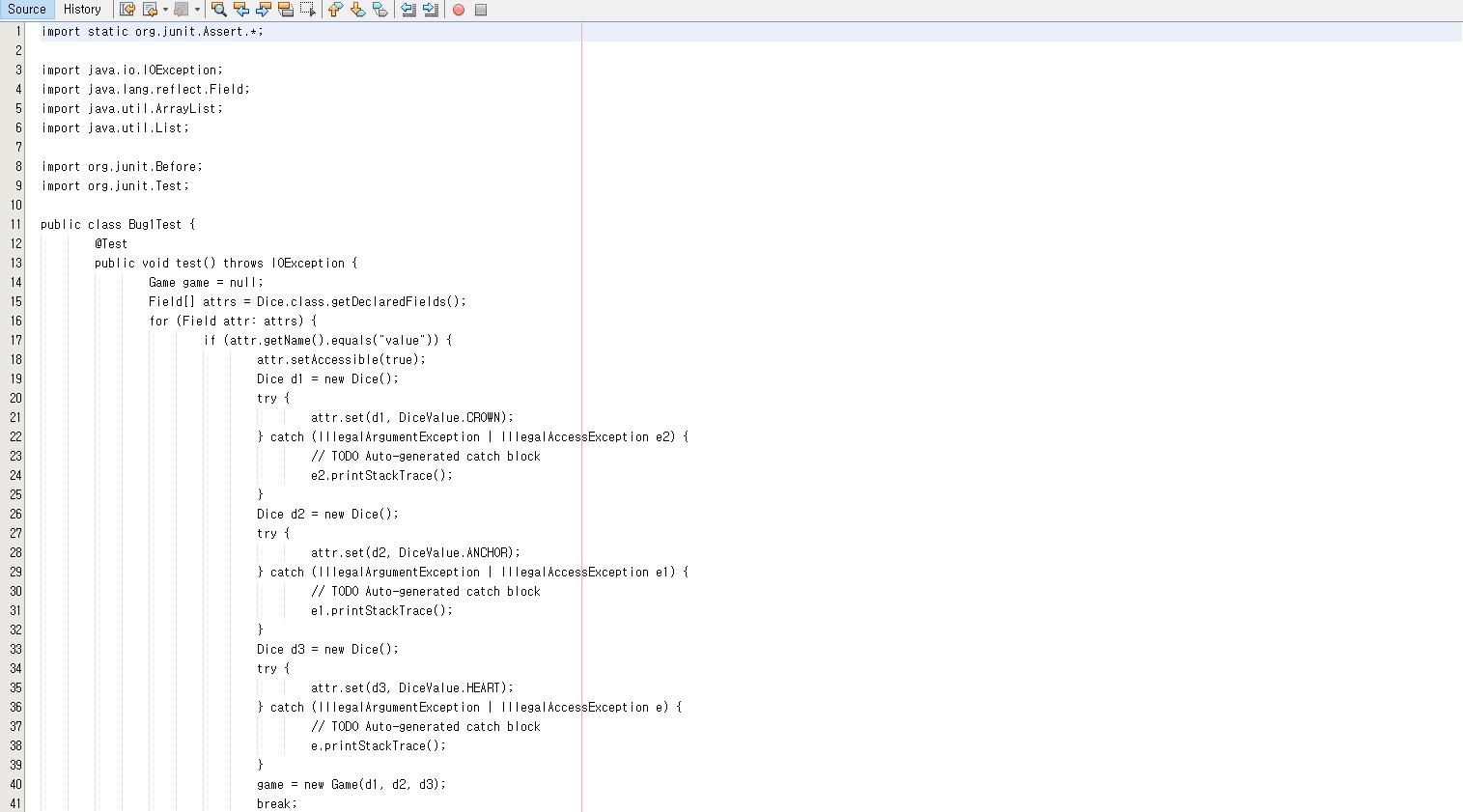
.=

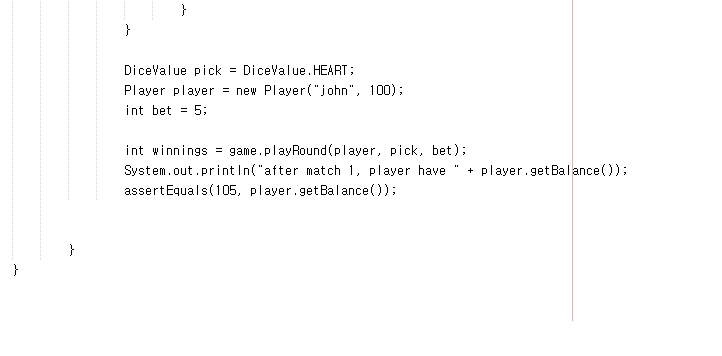
It will fix the problem as you can see the result.



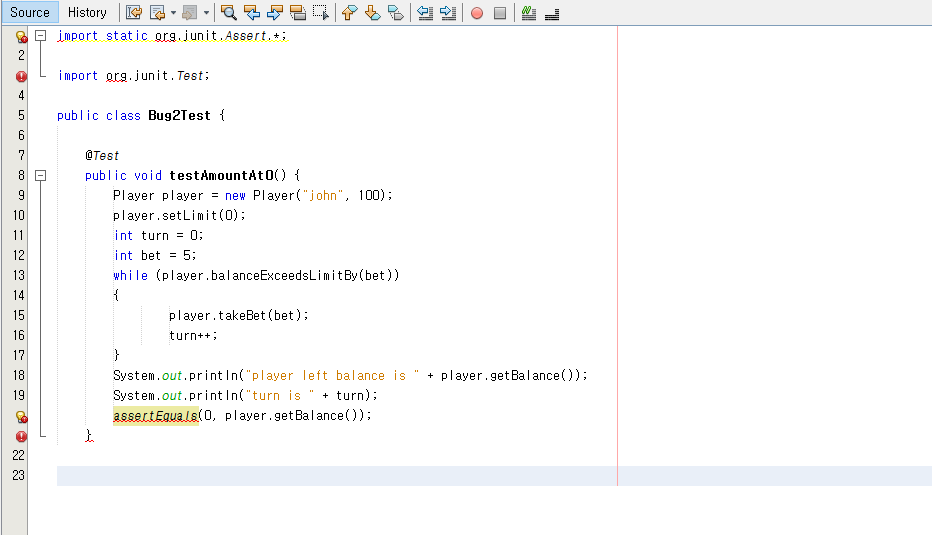
# J unit Test for 3 bugs

**Bug 1**





**Bug 2**



**Bug 3** 