ITC 205

Assignment 4

STUDENT NAME: Ben Loughlin Johnson

STUDENT ID: 11522230

Repository: <https://github.com/bjohnson1986/Ben_ITC205_Asg4>

Table of Contents

[Contents 2](#_Toc432532631)

[Bug 1. Game does not pay out at correct level when player wins on 1 match, balance does not increase. 3](#_Toc432532633)

[Bug 2. Player cannot reach betting limit - Limit set to 0, but game ends with player still with 5 (dollars) remaining. 6](#_Toc432532634)

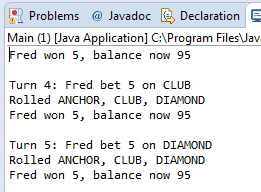
[Bug 3. Odds in the game do not appear to be correct - Crown and Anchor games have an approximate 8% bias to the house. 9](#_Toc432532635)

[Bug 4. The value of the dice never change from turn to turn. 12](#_Toc432532636)

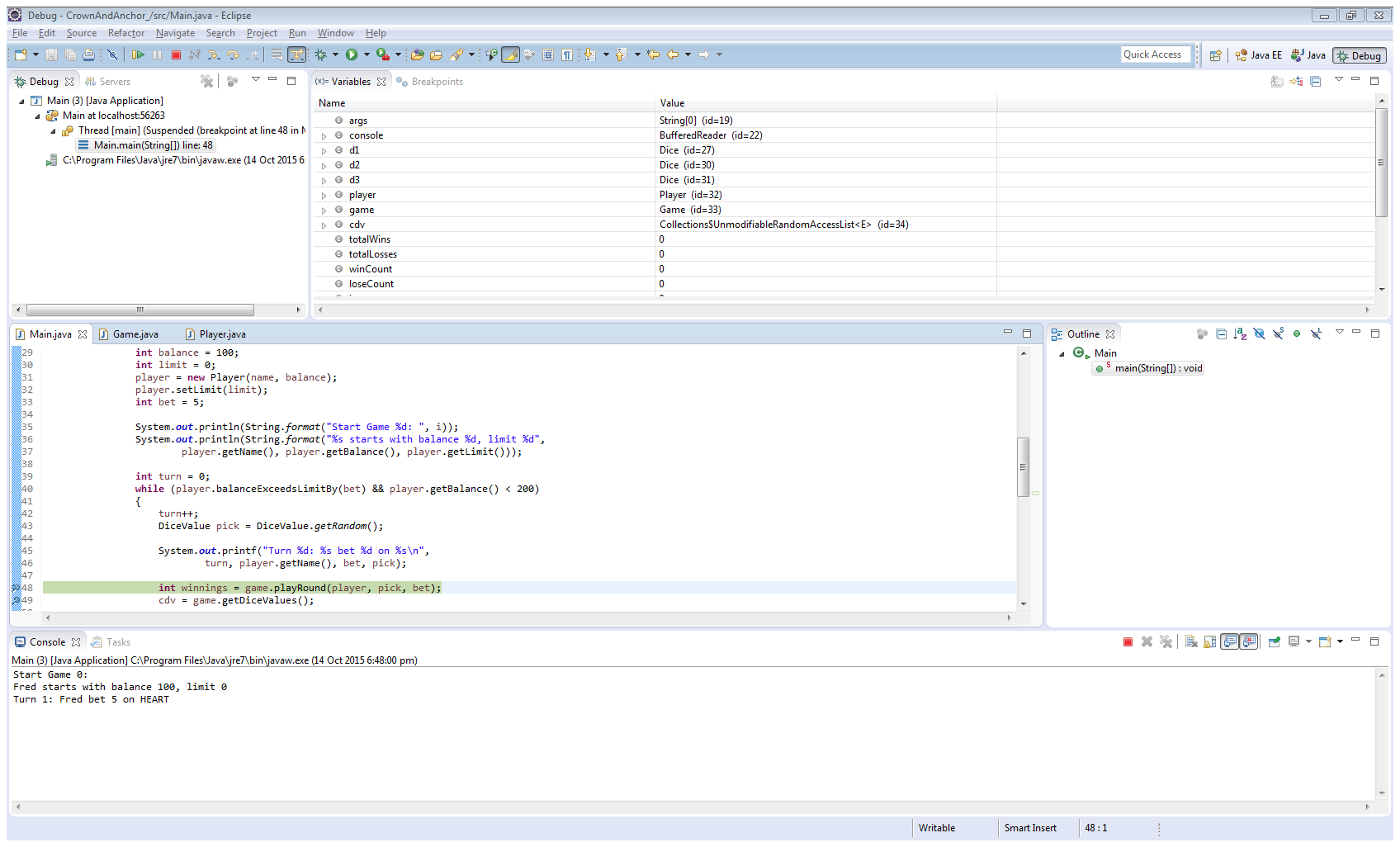
[Bug 5. Console never prints overall win rate.. 38](#_Toc432532636)

### Bug 1. Game does not pay out at correct level when player wins on 1 match, balance does not increase.

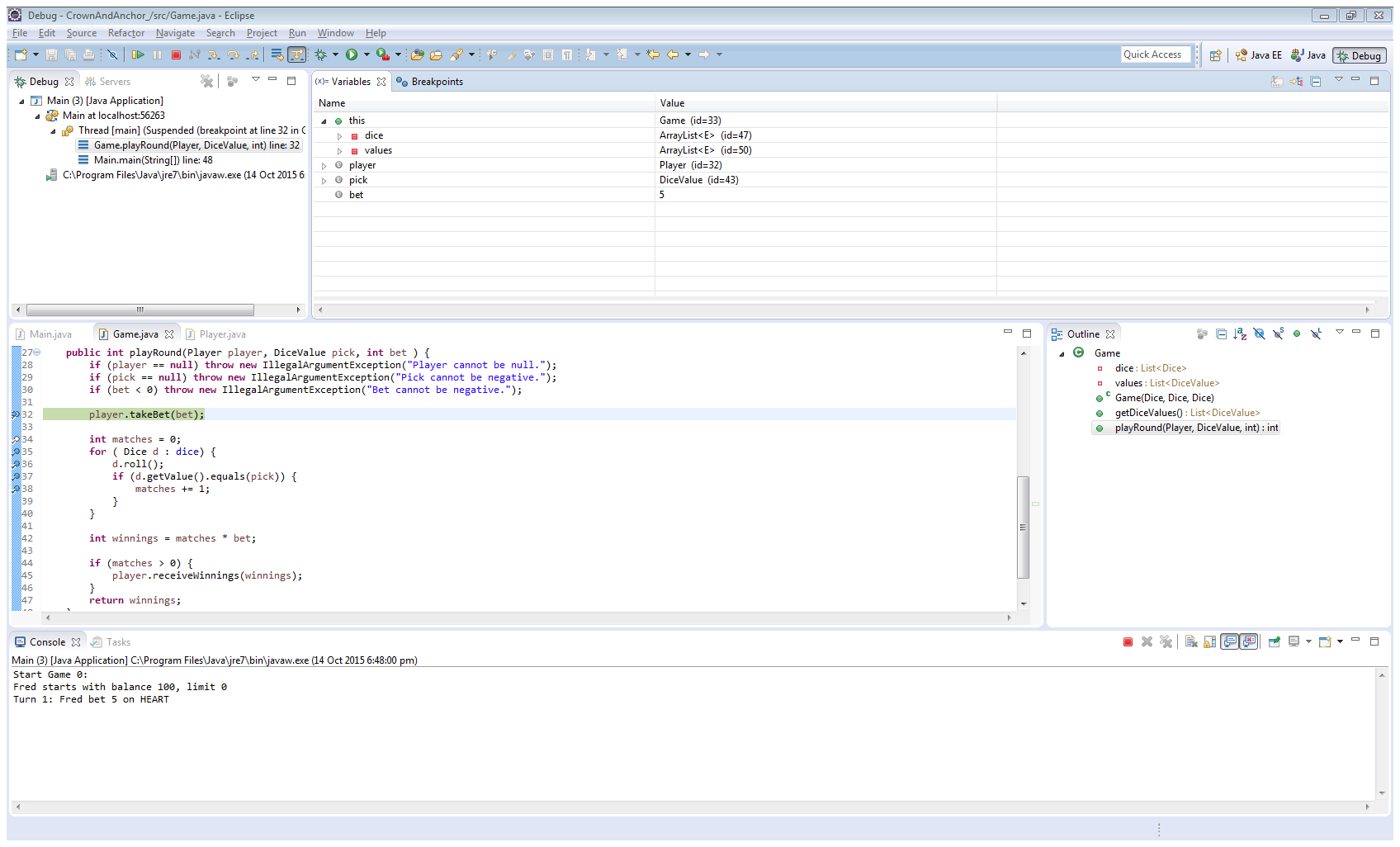
Evidence of bug replicated.



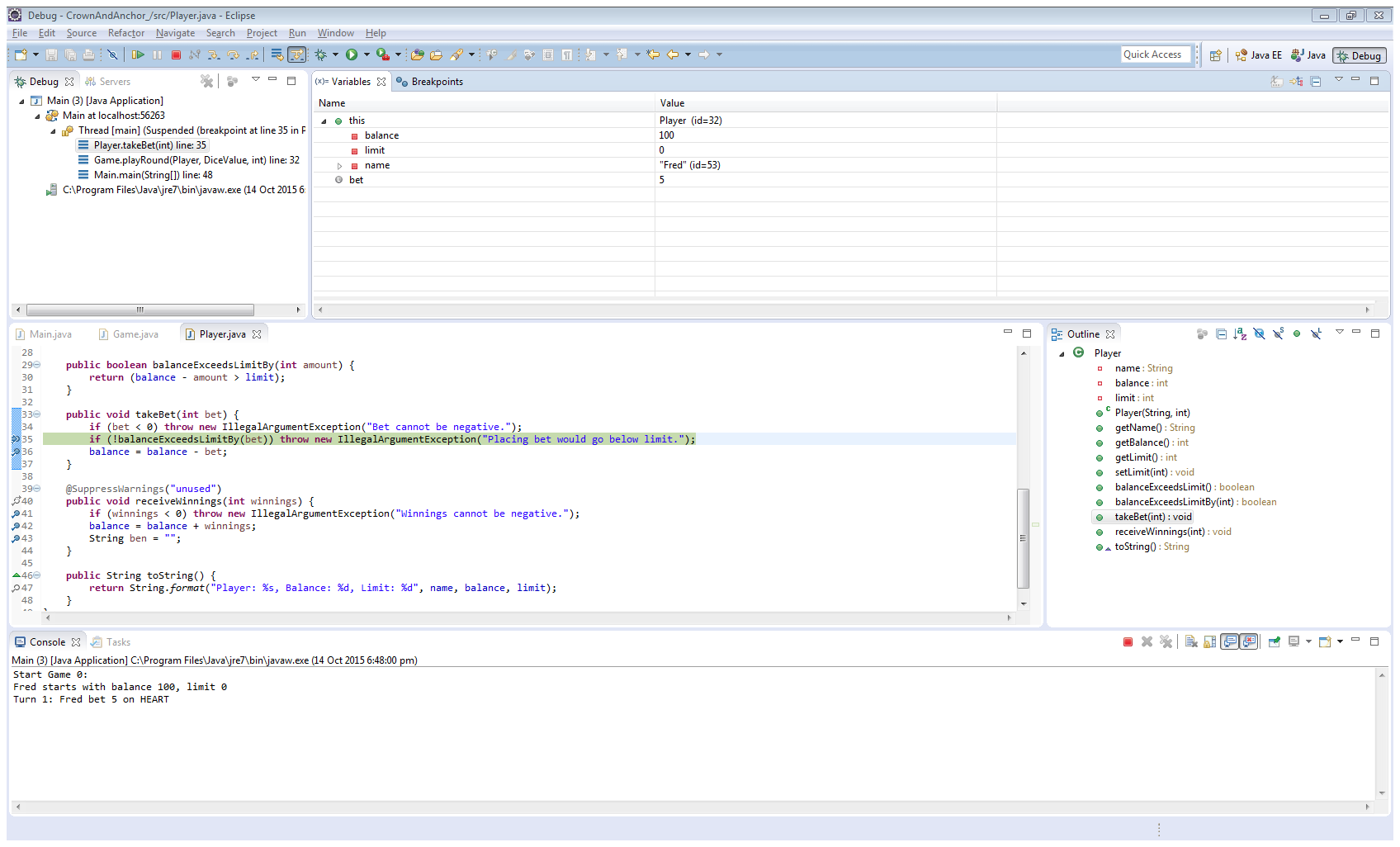
Running in debug mode – Main calls playRound.



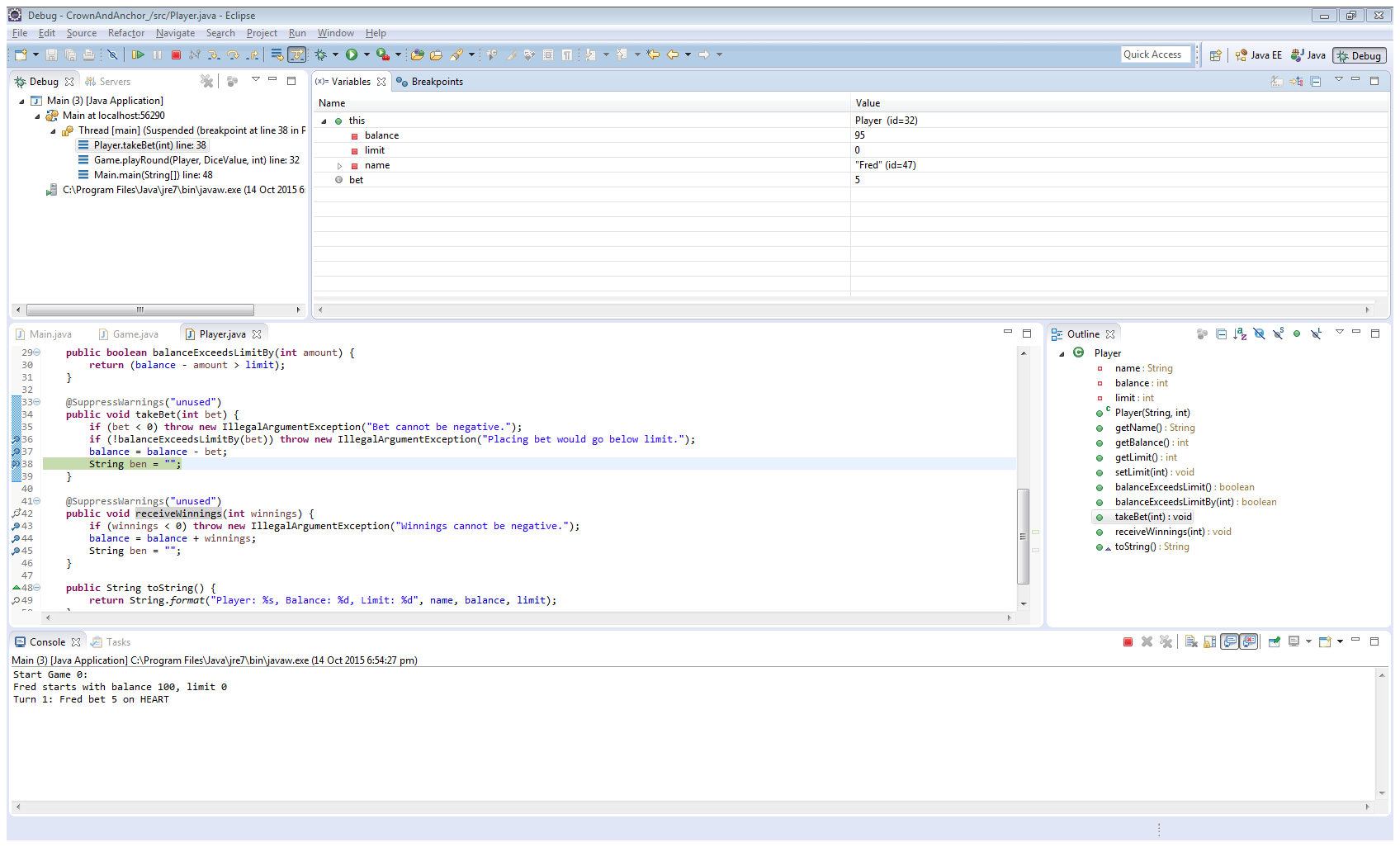
PlayRound calls takeBet.



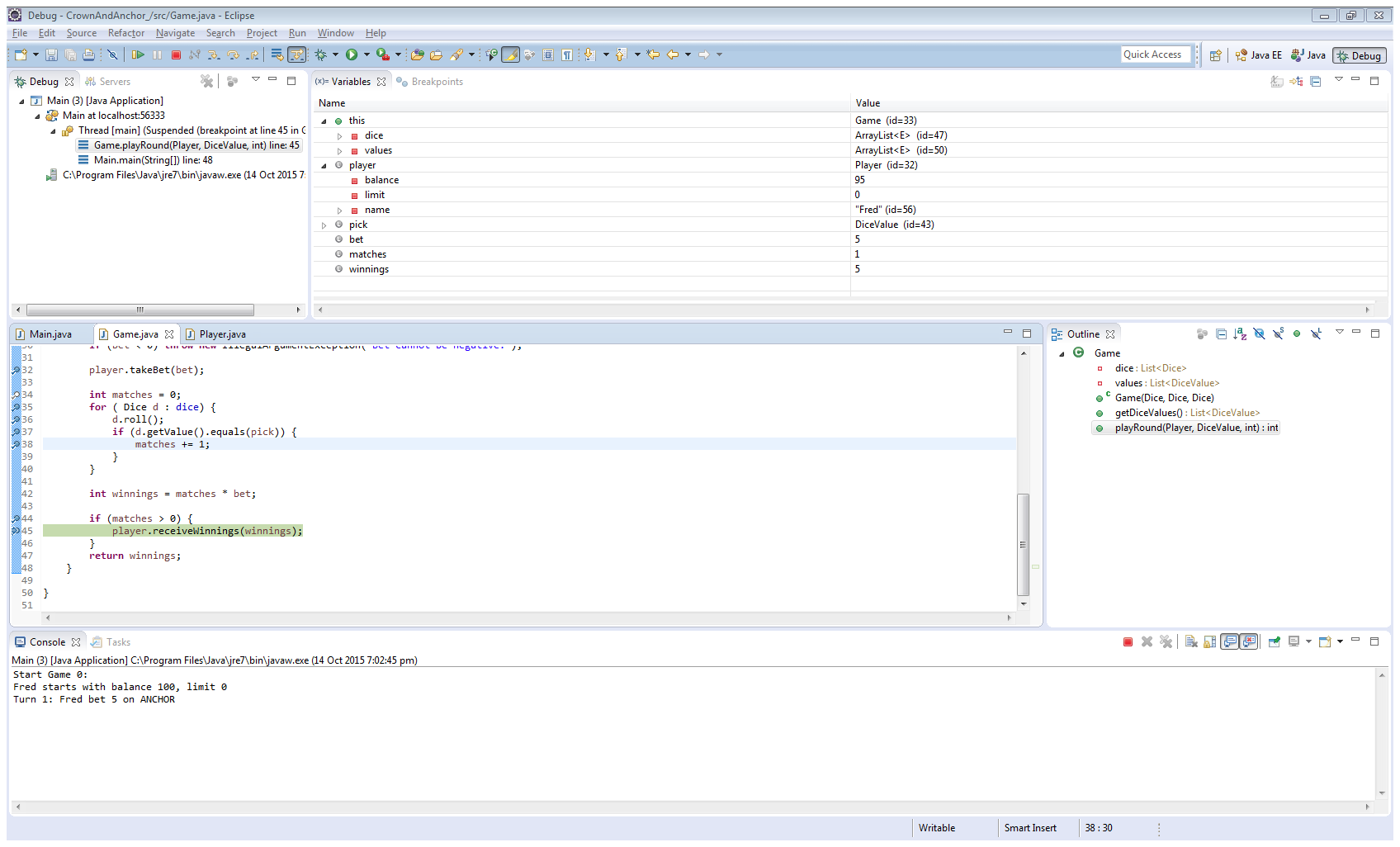
The Player class currently has the variable ‘Balance’ with a value of 100.



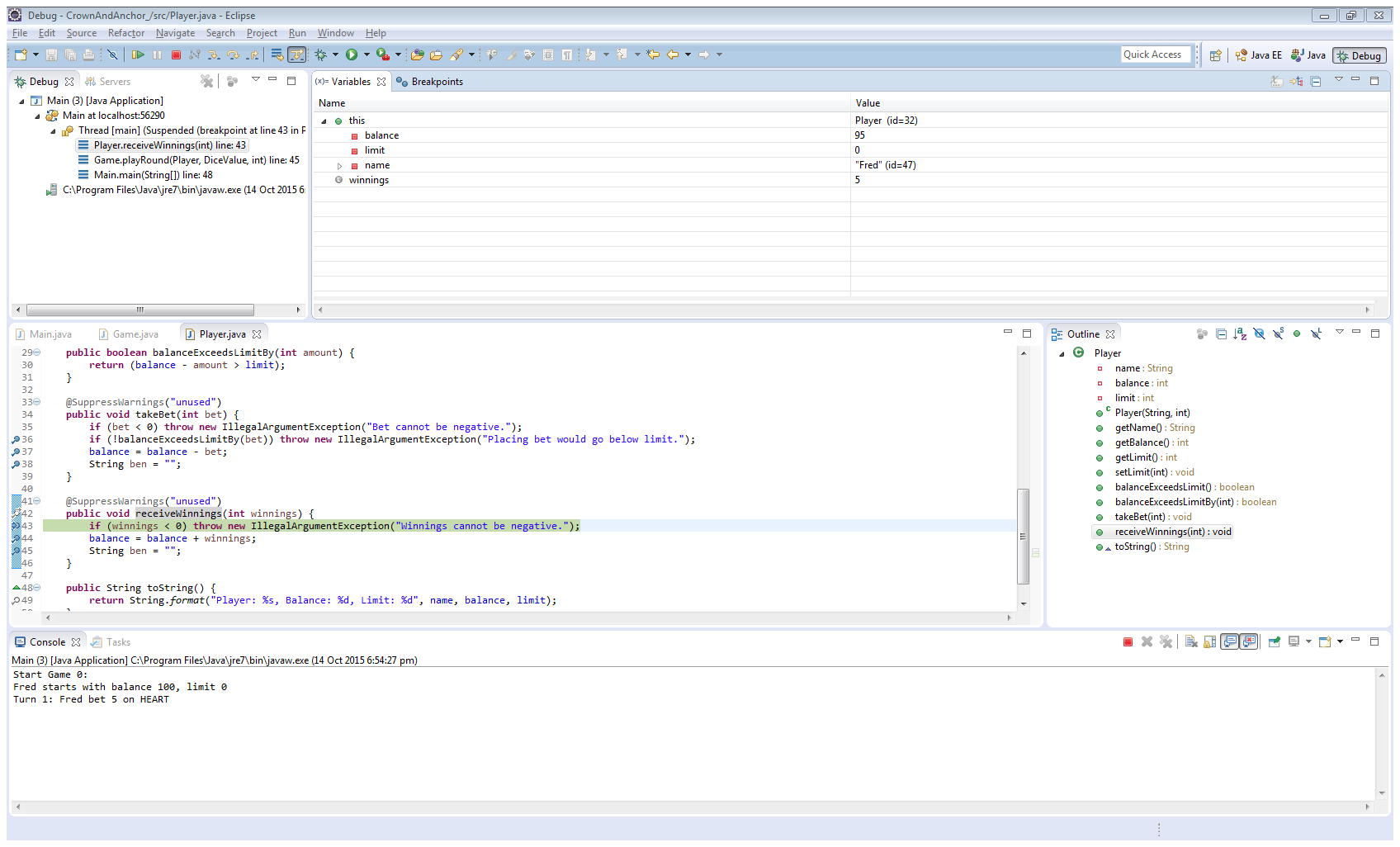
The Player class now has the variable ‘Balance’ with a value of 95.



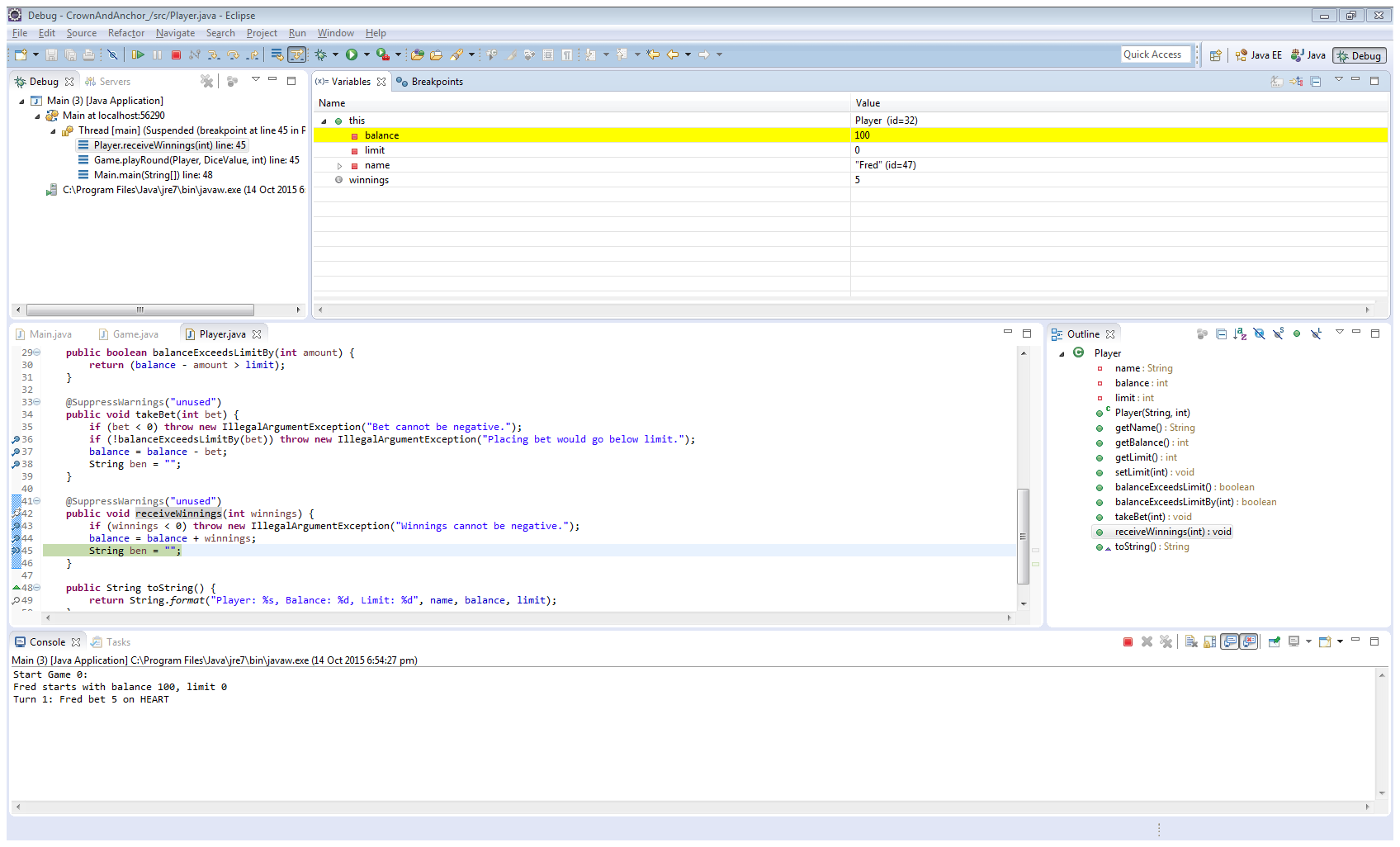
The Game class then calls Player’s receiveWinnings method.



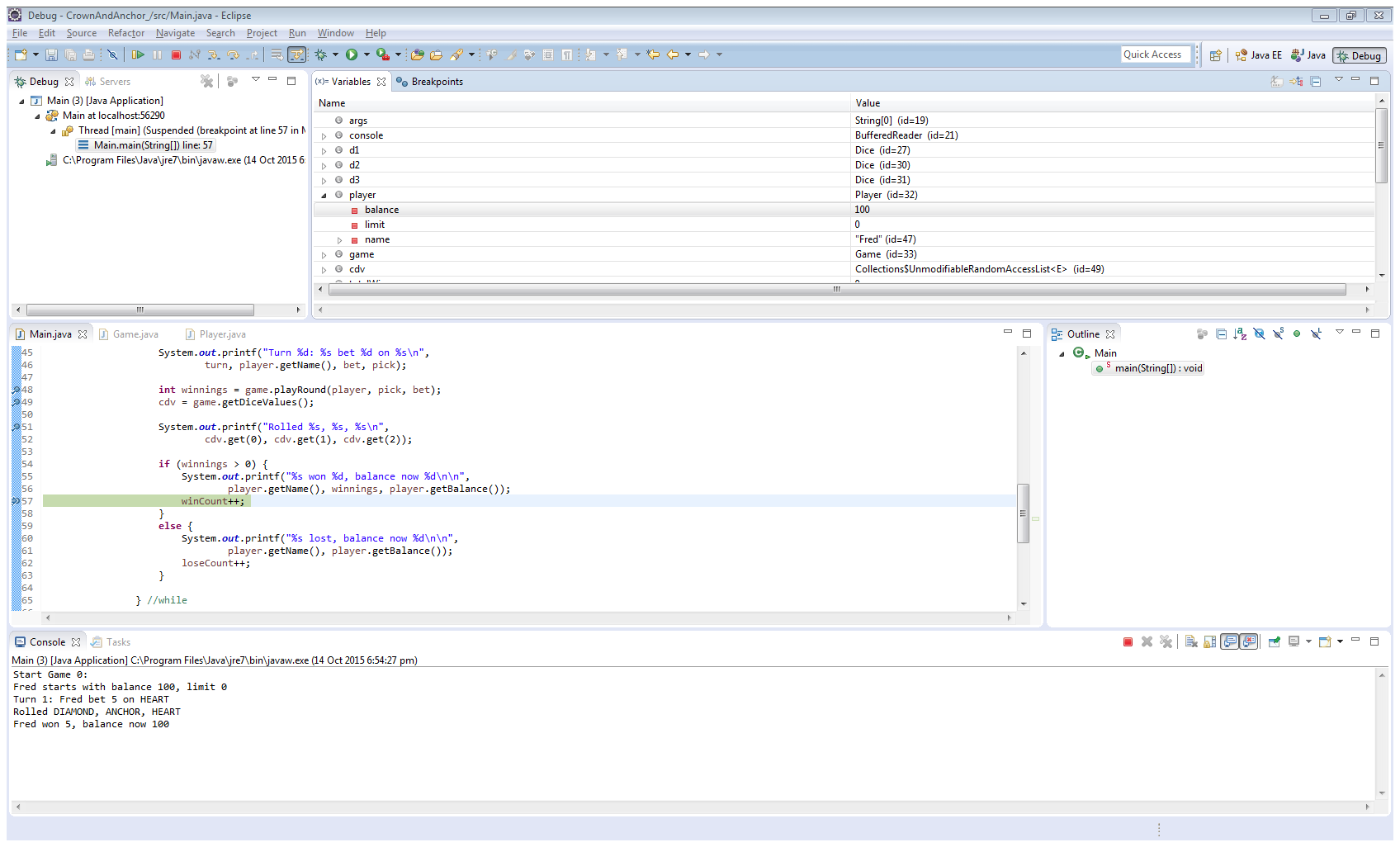
The Player class still has the variable ‘Balance’ with a value of 95.



The Player class now has the variable ‘Balance’ with a value of 100.

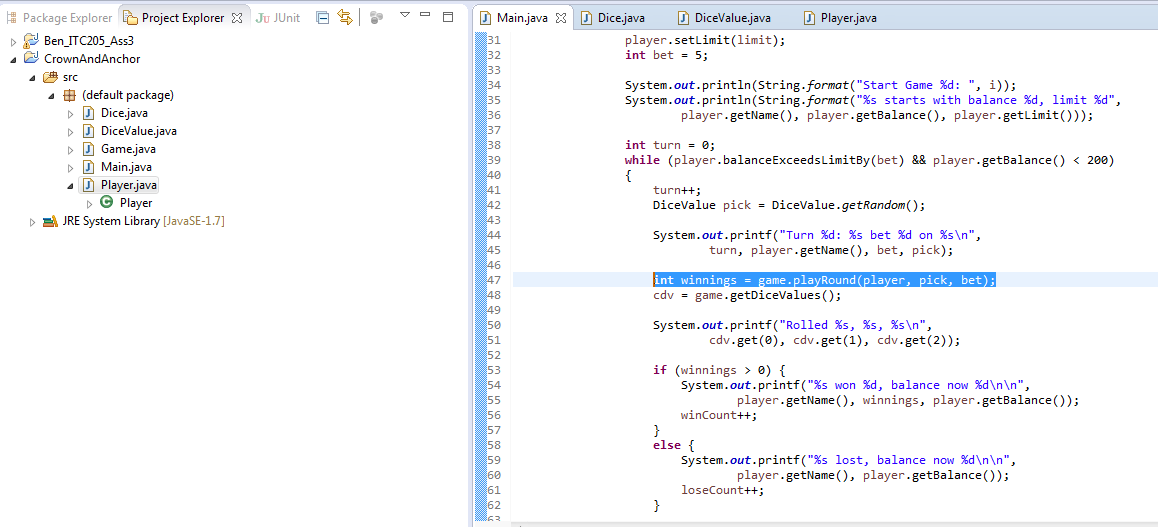


The application returns to Main, and the player has received their winnings, but never had their original wager returned.

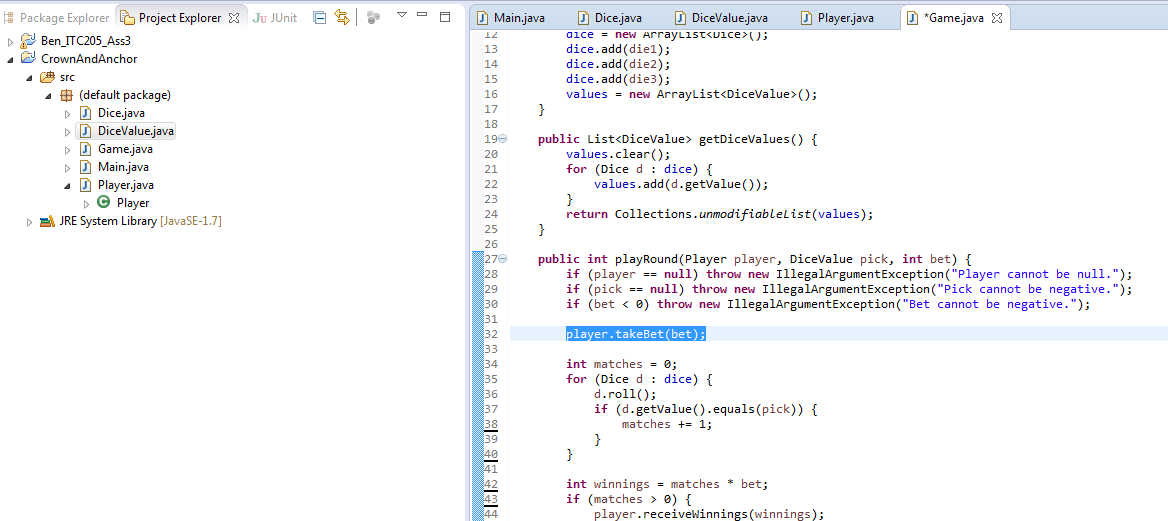


Tracing the code.

Winnings are determined by calling the PlayRound method within the Game class.



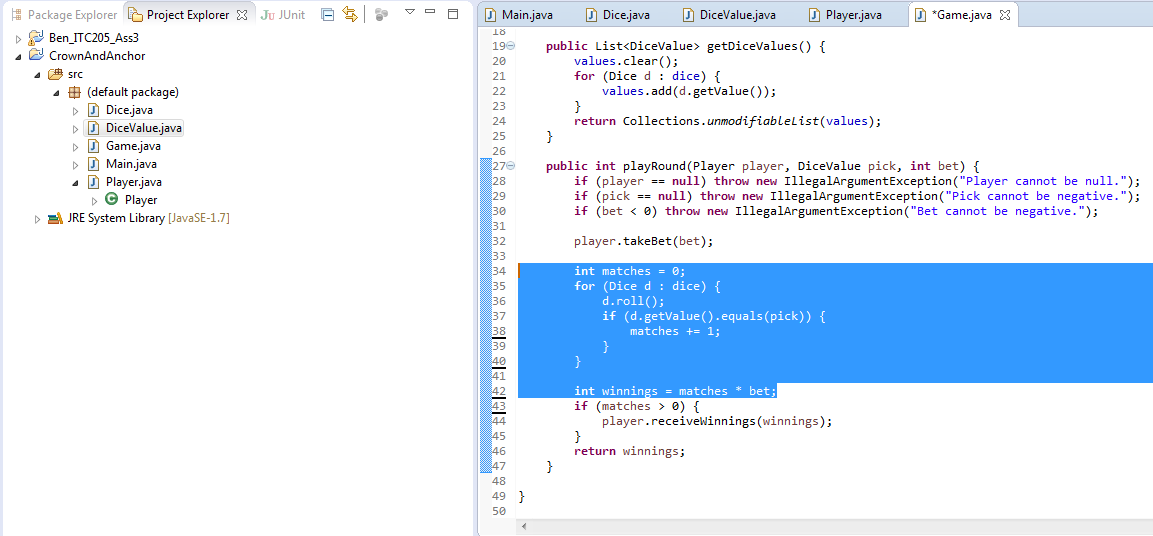
The takeBet method of Player class is called.



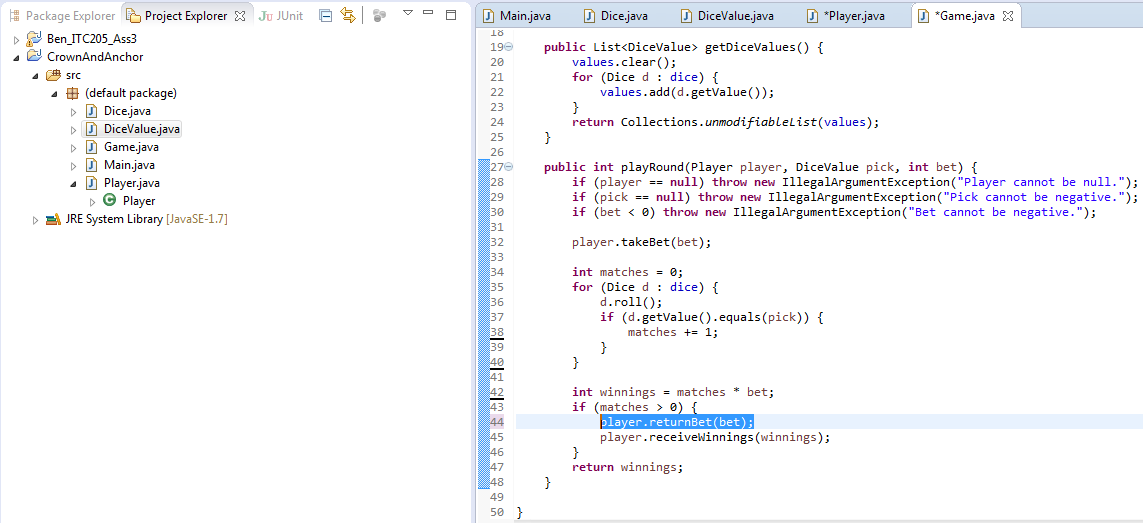
The takeBet method subtracts the bet amount from the player’s balance (note need to ensure that this get added back if won).

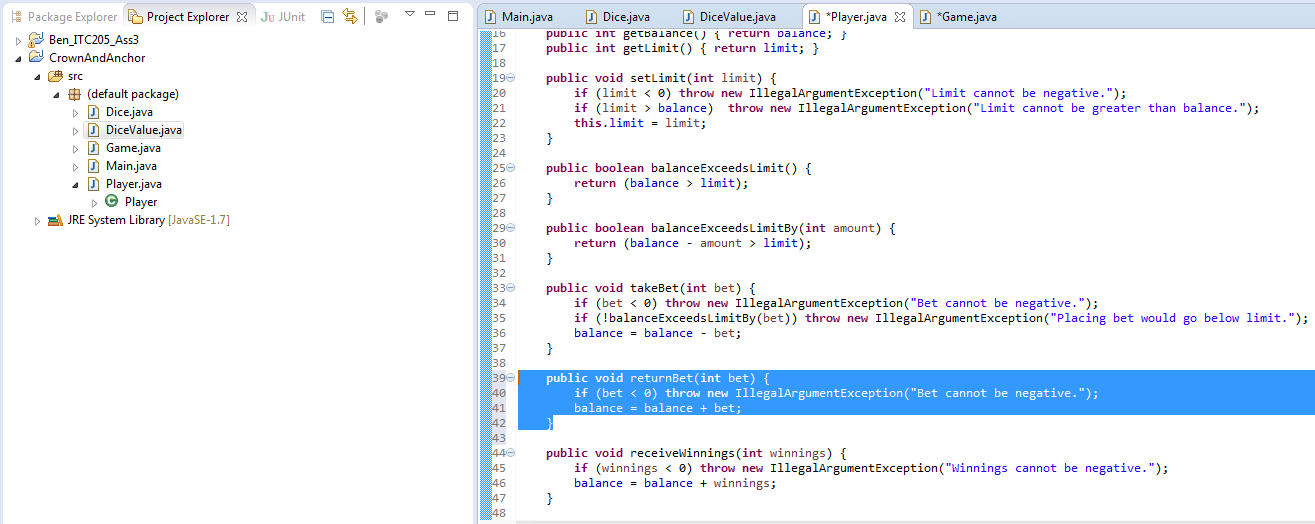


Subsequently, when the playRound method in the Game class continues the calculation does not return the original wager.

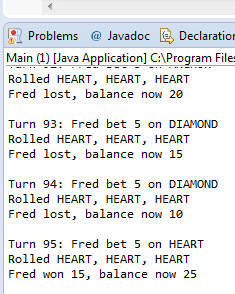


The line highlighted will resolve this issue if there are any matches – it was found that this bug would occur not only for a singular match, but also double or triple.

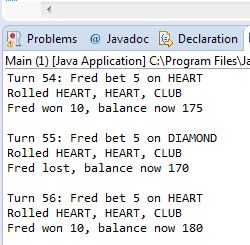




Evidence of resolution.



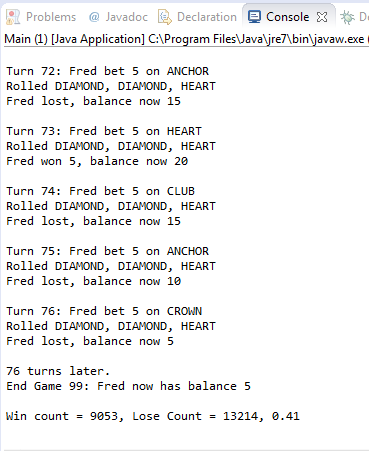
On the left, the scenario of zero and three matches are resolved.



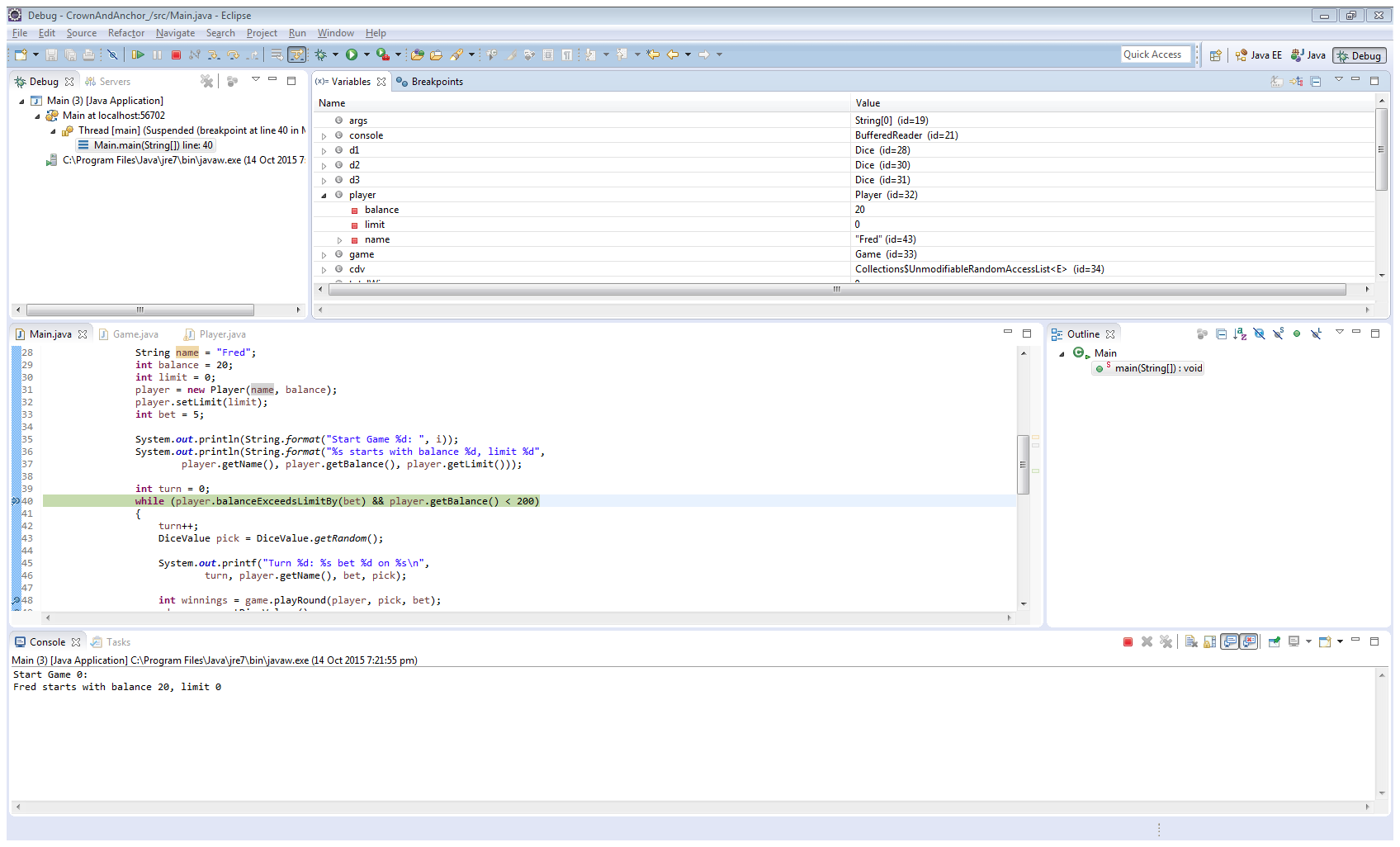
On the left, the scenario of two matches are resolved.

### Bug 2. Player cannot reach betting limit - Limit set to 0, but game ends with player still with 5 (dollars) remaining.

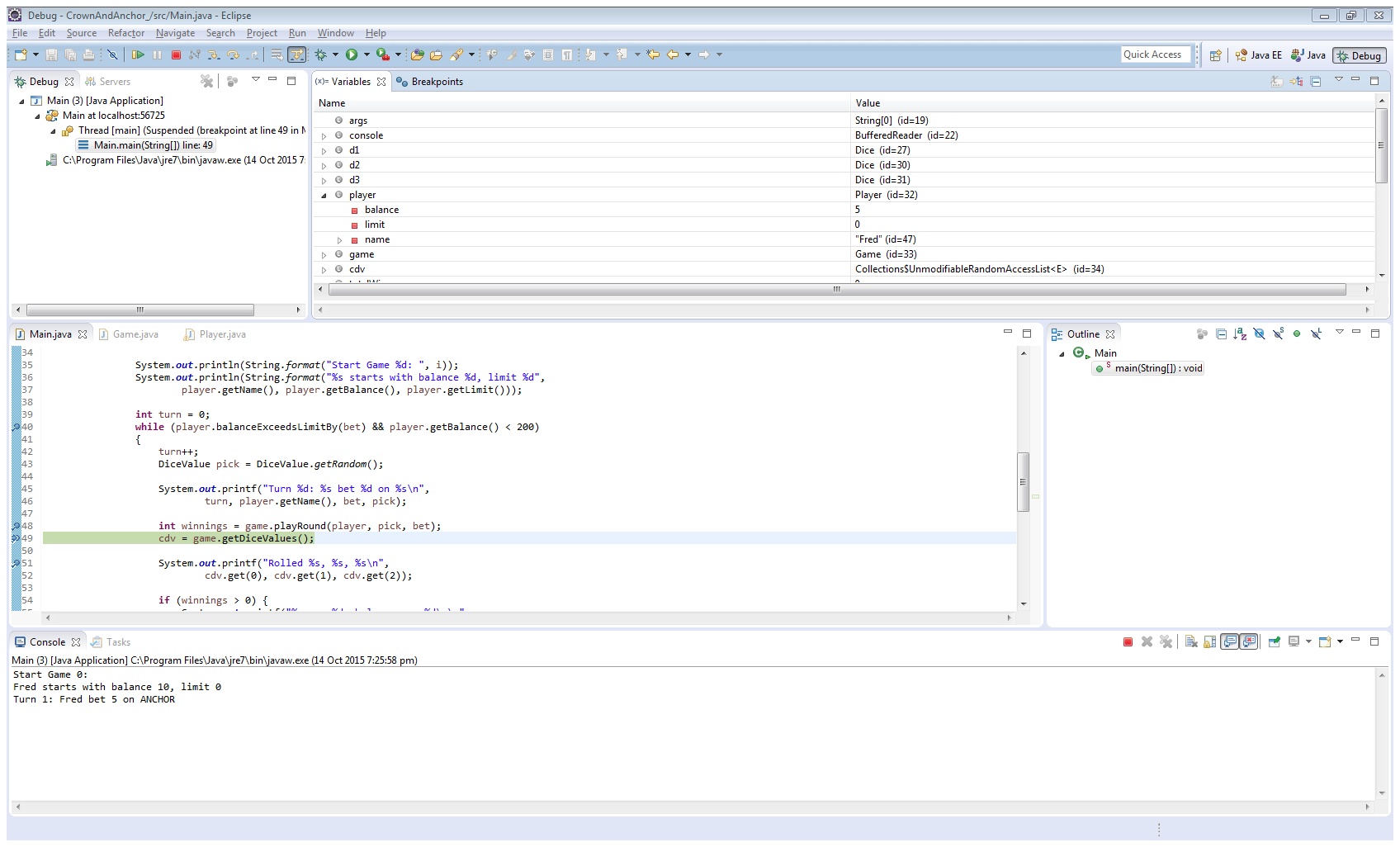
Evidence of bug replicated.



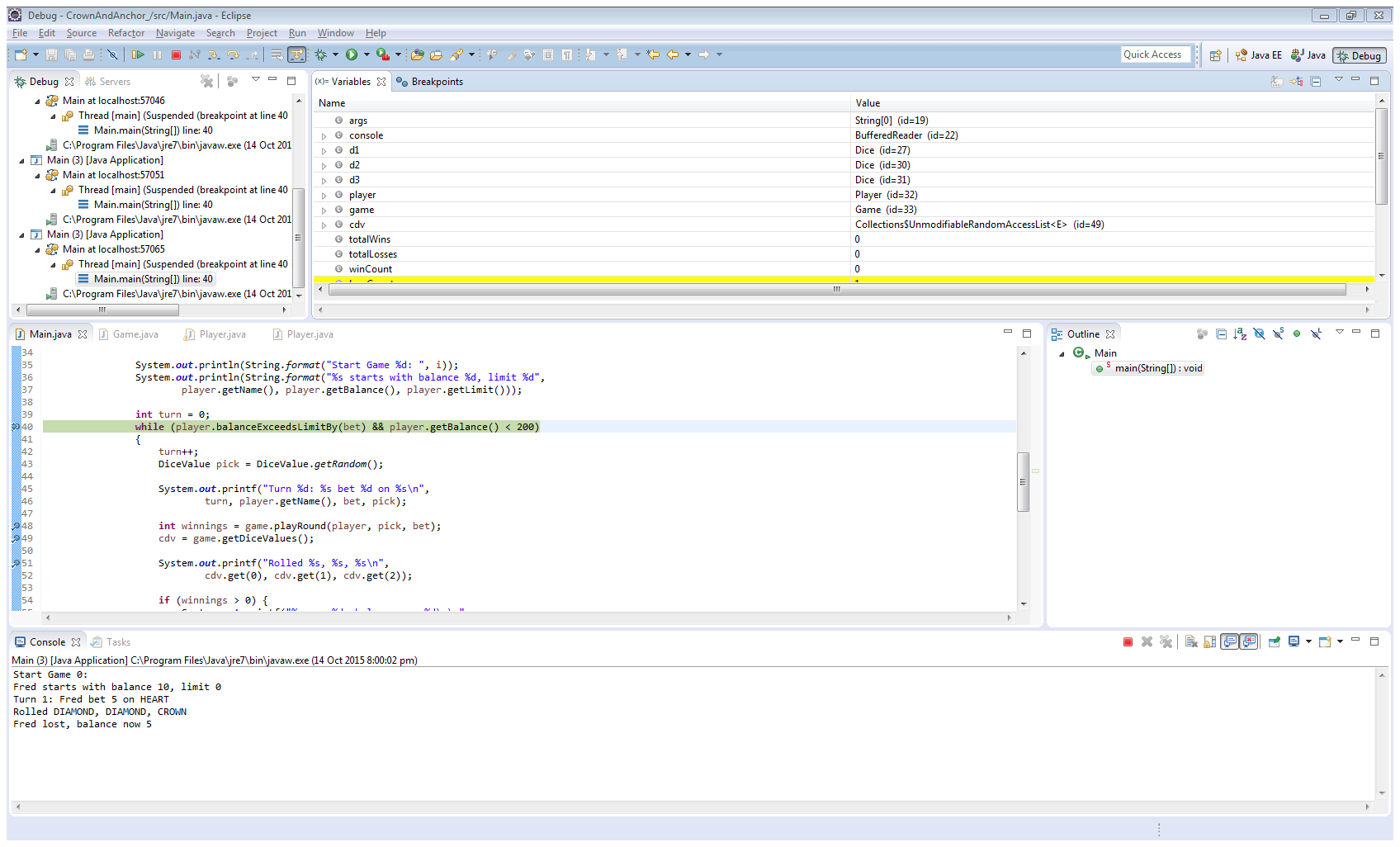
Running in debug mode – Main runs a while loop that continues until the player runs out of money or the player has $200 or more.



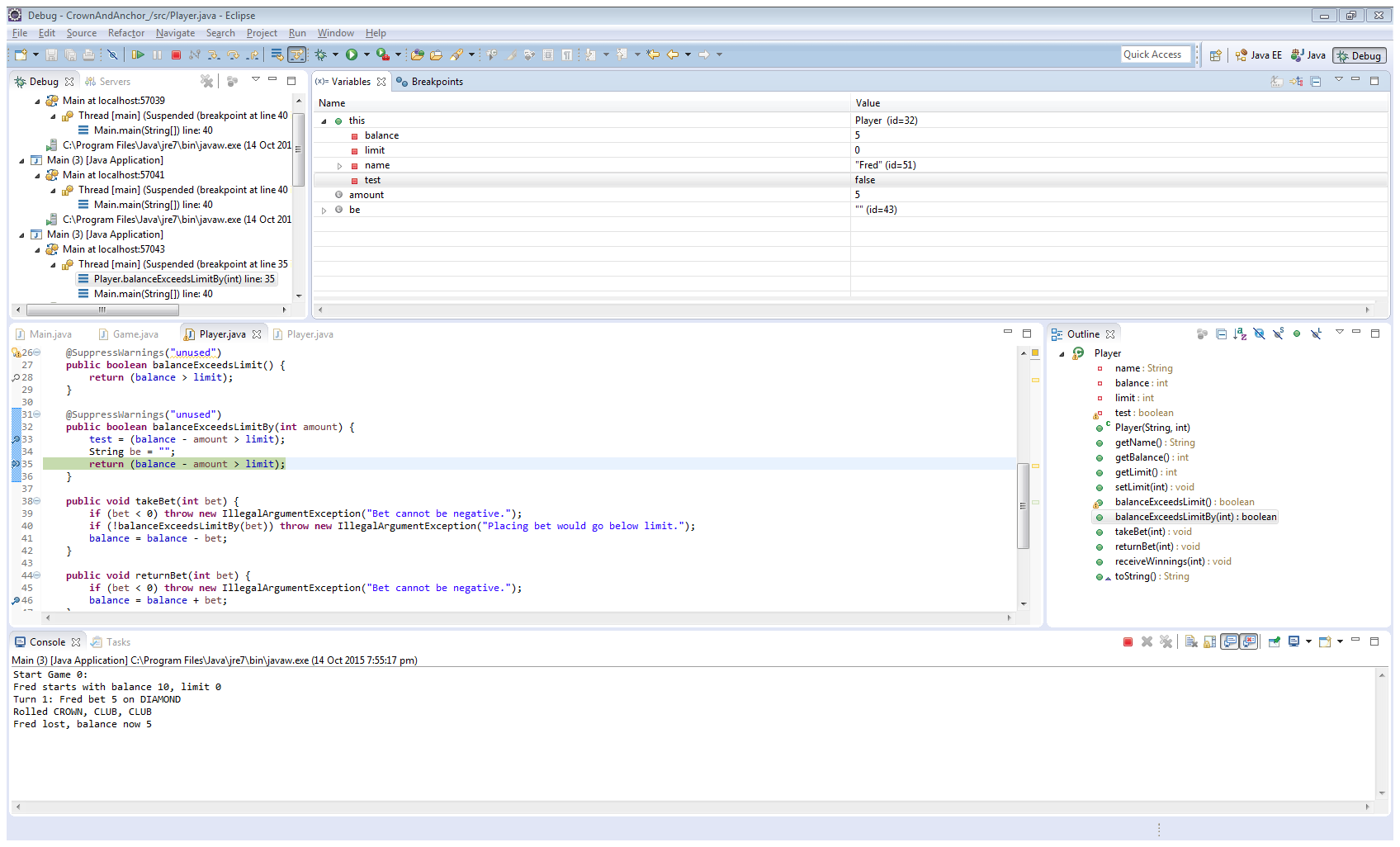
After the game is determined, player has a balance of $5.



The While loop then determines whether the compound statement is still true before rerunning itself.

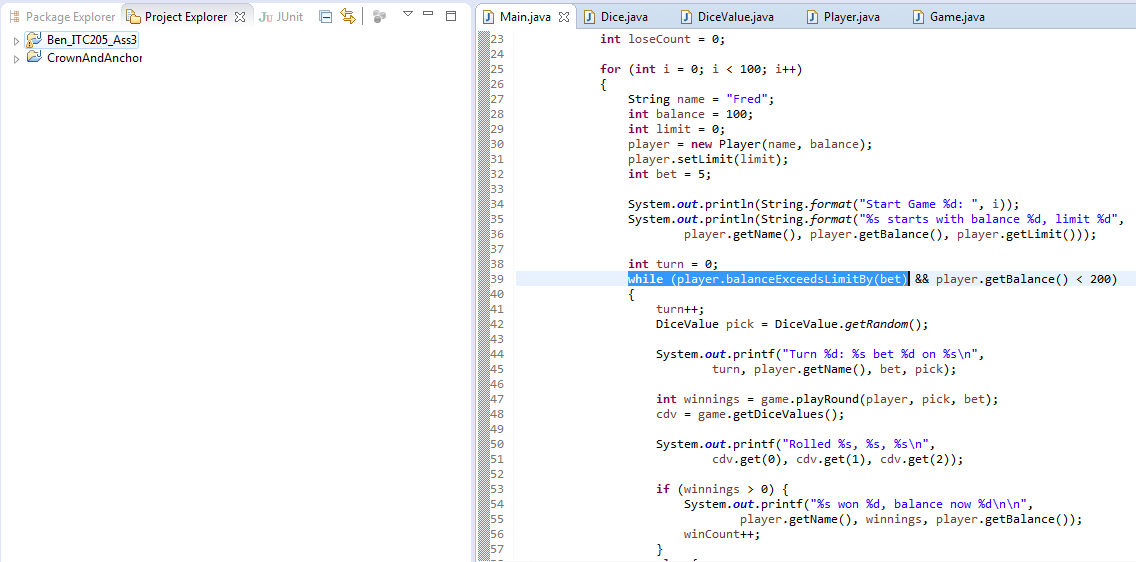


The variable test contains the value of False. Therefore, it can be identified that the relational operator is not logically correct. It should be >=, not >.

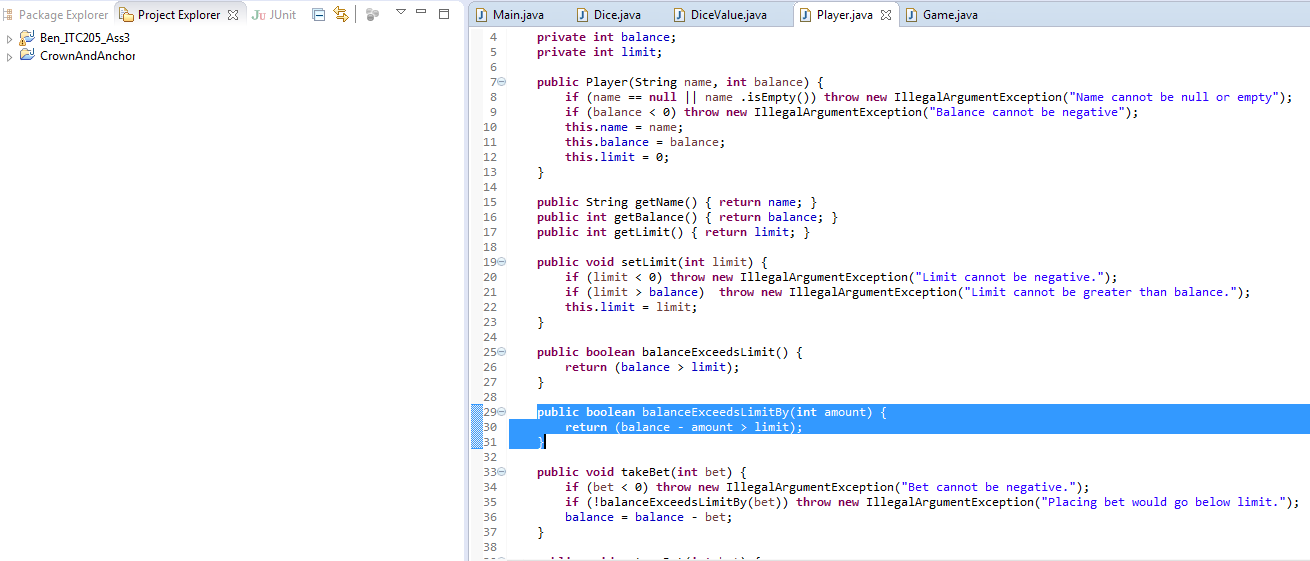


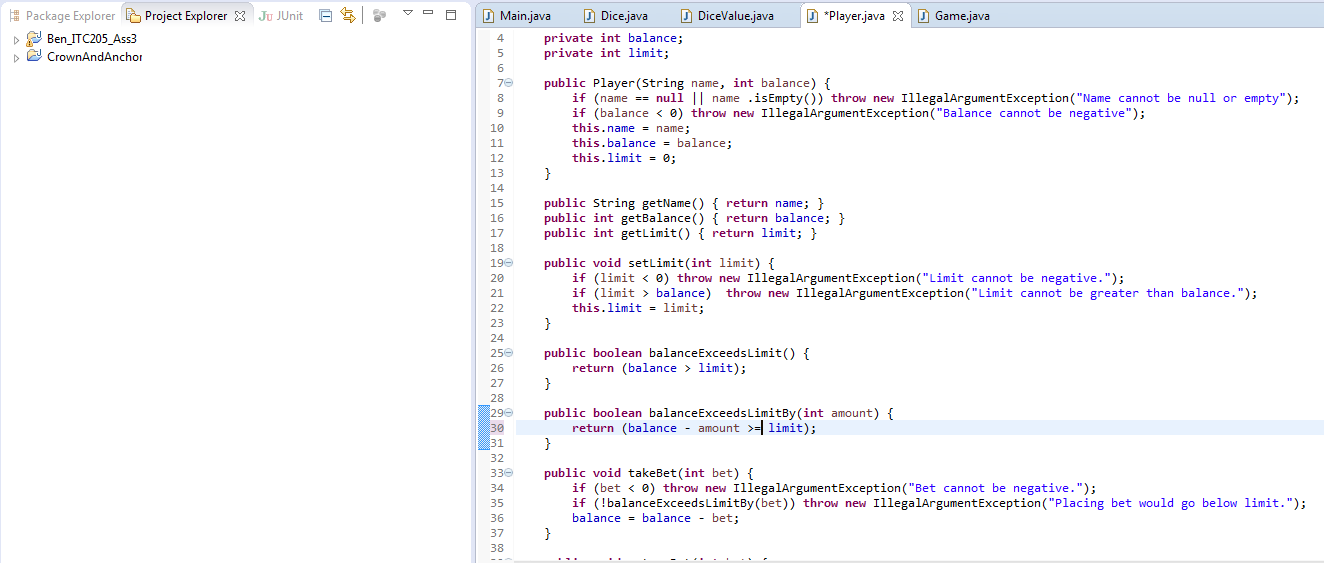
Tracing the code.

The loop runs until the player has $200 or more, or has no money left.

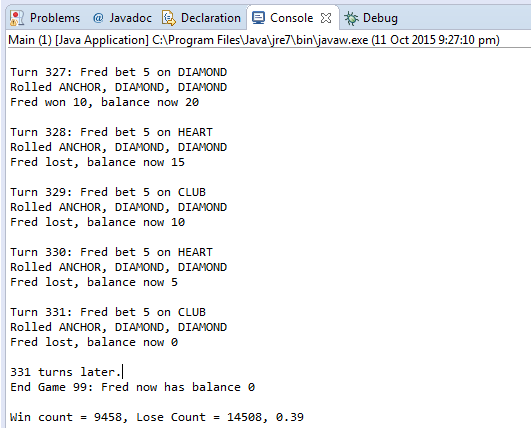


The Player class does not let the player wager with an amount if it places their balance at 0 (the return statement should contain >=, not >0).



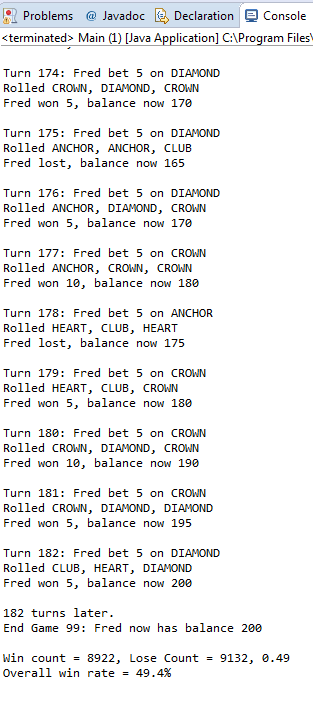


Evidence of resolution.



### Bug 3. Odds in the game do not appear to be correct - Crown and Anchor games have an approximate 8% bias to the house.

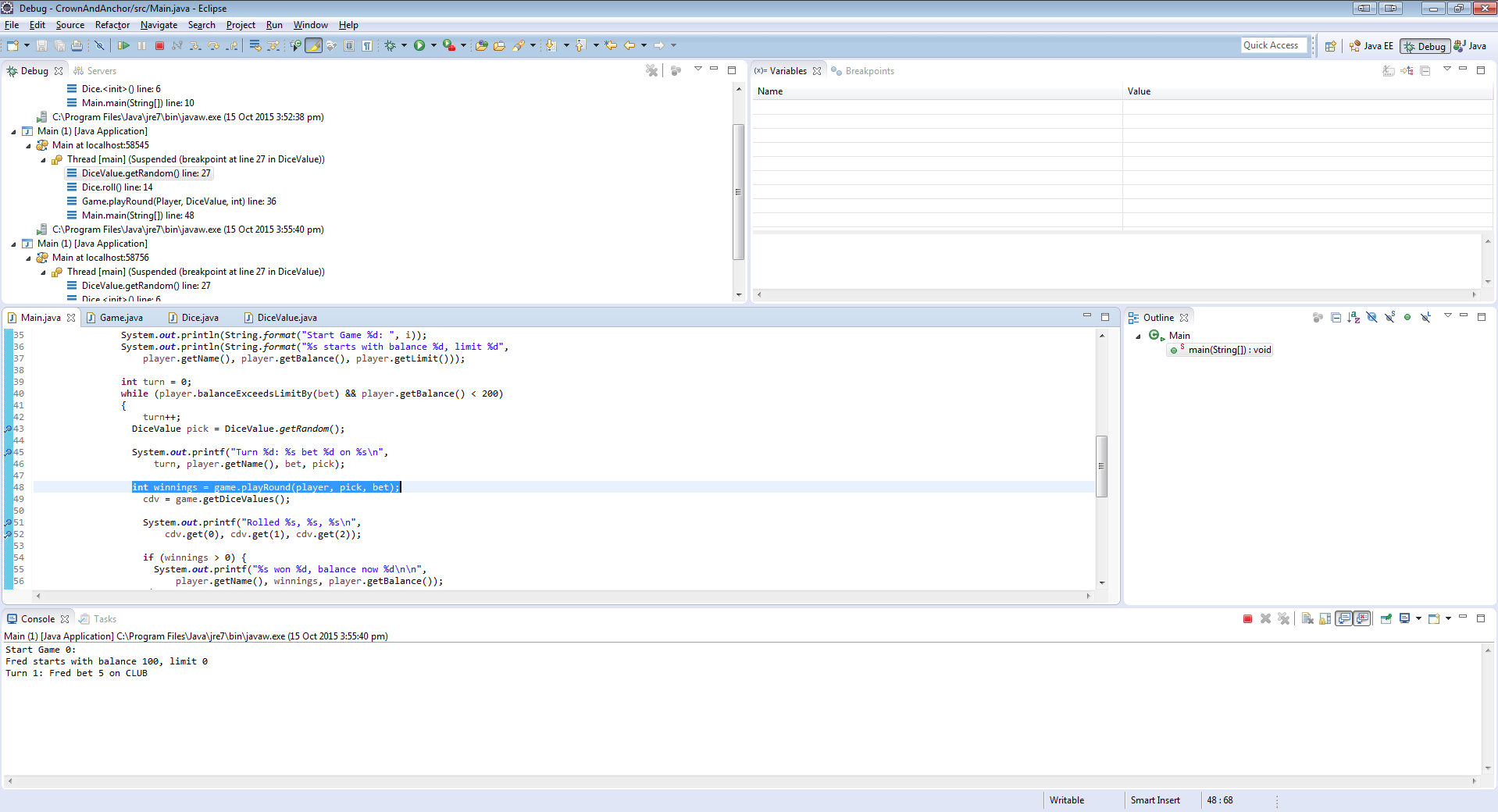
Evidence of bug replicated.



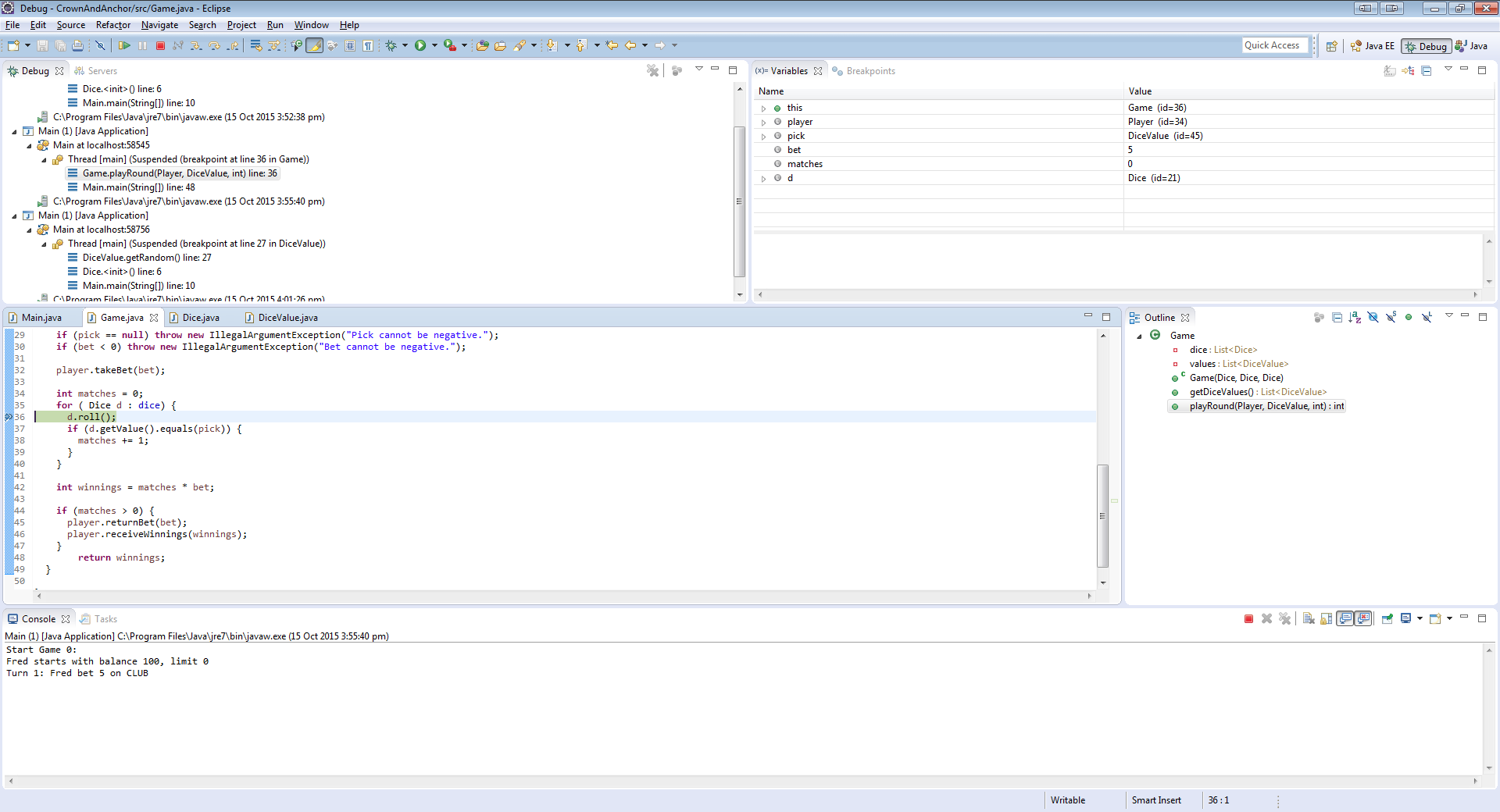
Two things to be noted:

* Firstly, the win to bets placed ratio is 49%; and
* Secondly, SPADE is never chosen amongst the random dice values.

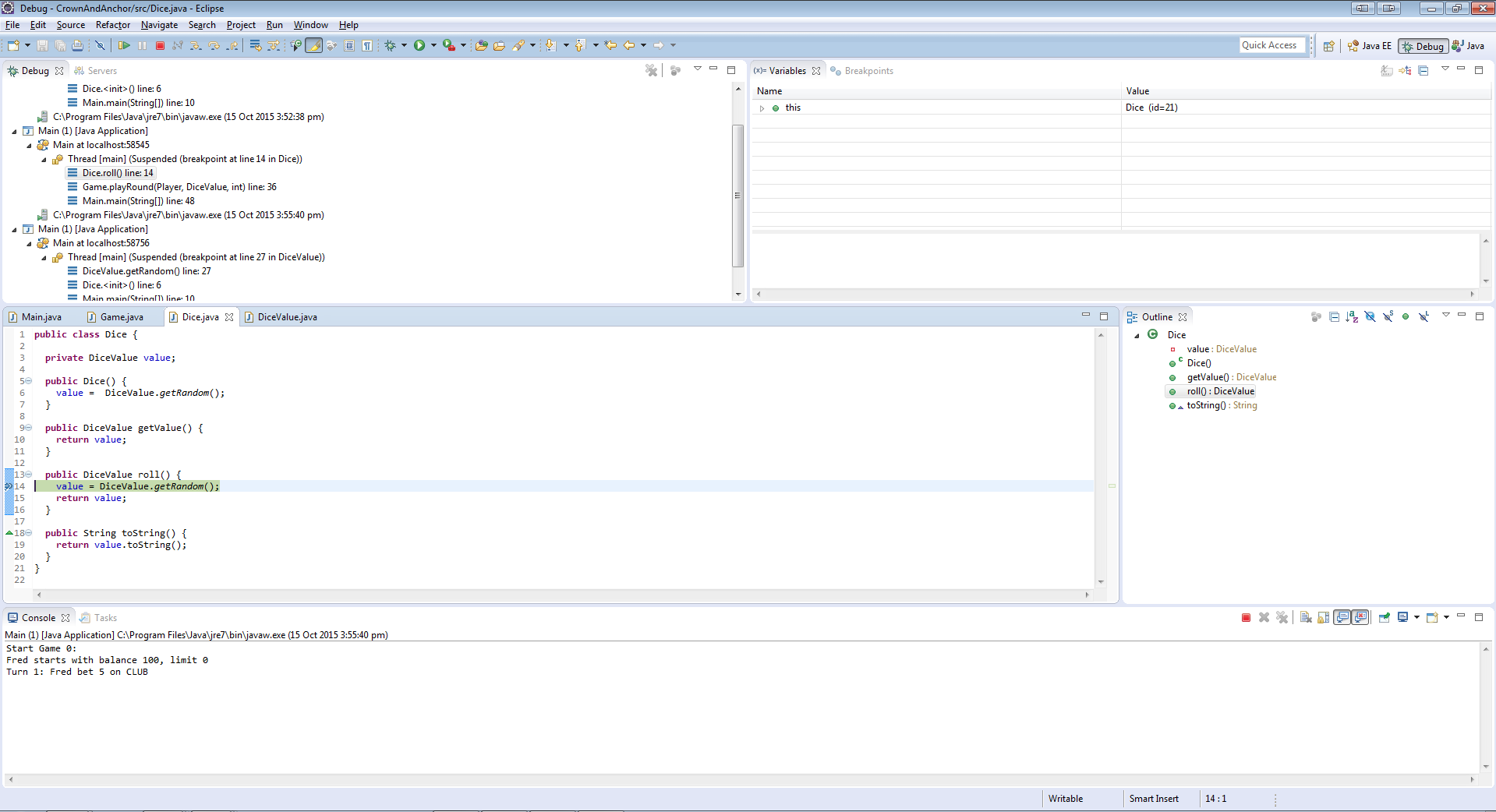
Running in debug mode – Main calls Player’s playGame method.



Game’s playGame method calls Dice’s roll method.



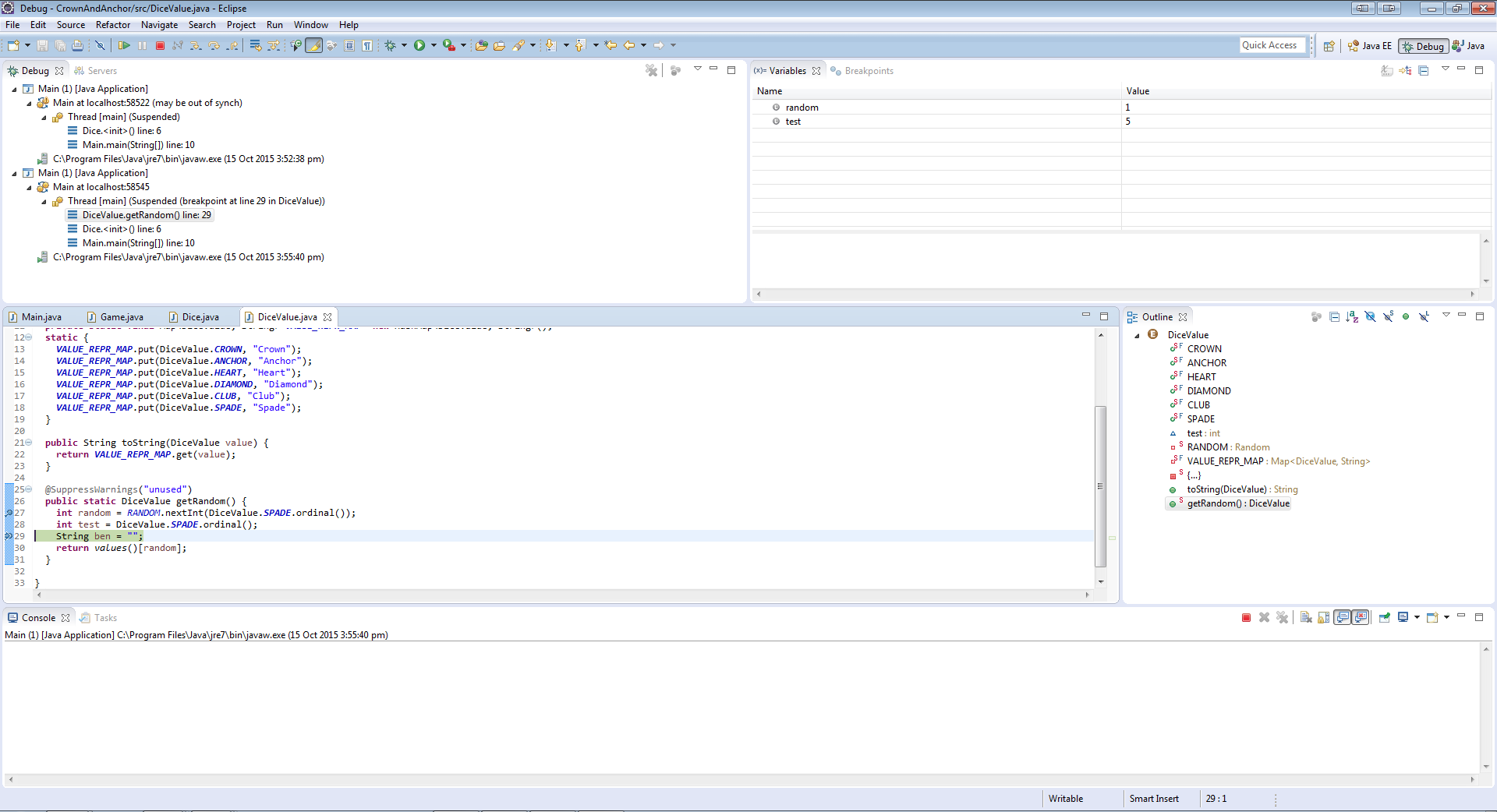
Dice’s roll method then calls DiceValue’s getRandom method.



In the getRandom method of the DiceValue class, DiceValue.SPADE.ordinal() is returning 5 (which is SPADE’s position).

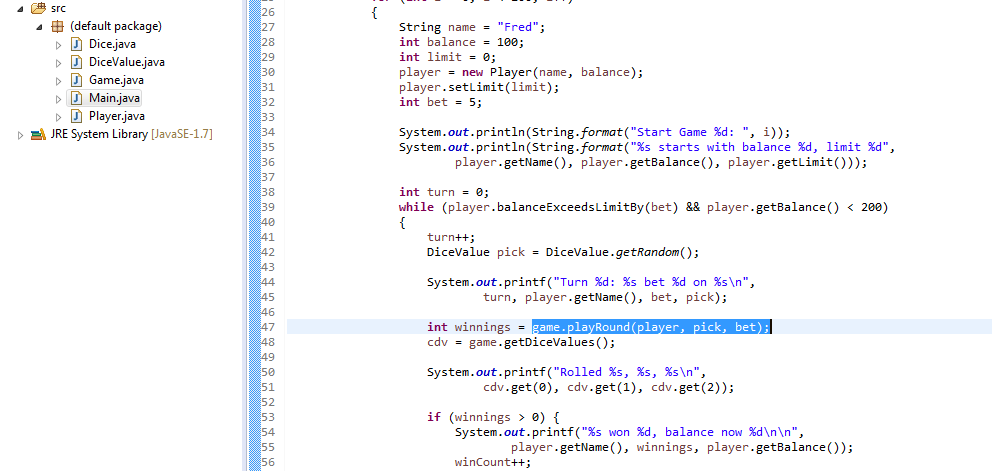
Documentation on the nextInt method shows that the integer argument used will return a random number between 0 (inclusive) and the argument (exclusive – or up to but not including). Documentation: <http://www.tutorialspoint.com/java/util/random_nextint_inc_exc.htm>.

The argument needs to be changed to 6 to allow elements 0-5 to be chosen.

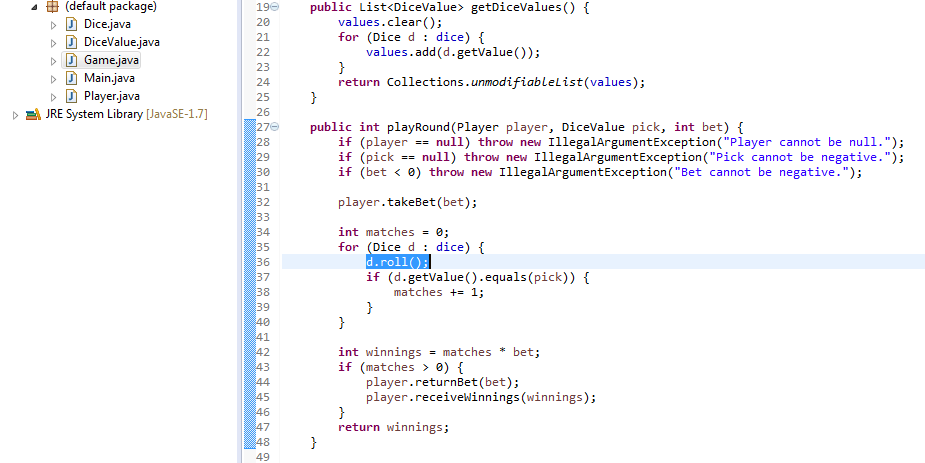


Tracing the code.

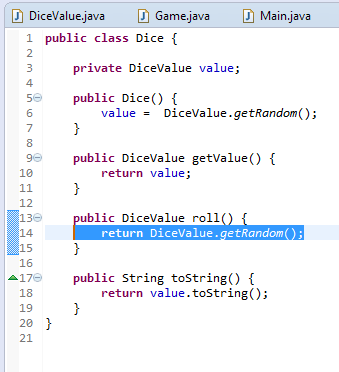
Main calls the playRound method of the Game class.



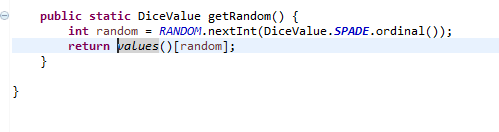
The playRound method of the Game class then calls the Dice class’s roll method.

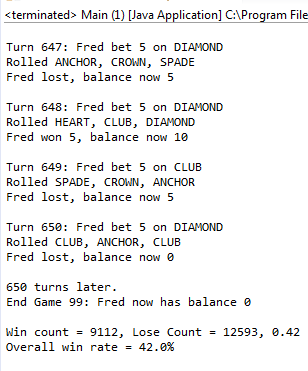


The roll method of the Dice class then calls the DiceValue class’s getRandom method.



The getRandom method of the DiceValue class then determines a random value by calling the Random class’s nextInt method. As mentioned previously, documentation stipulates that the integer argument determines that a random number will be selected between 0 and up to, but not including, the argument. Therefore the argument needs to be changed to 6, however, rather than using a literal, the size of VALUE\_REPR\_MAP will be used (to cover the unlikely chance that the number of values will be changed from 6).

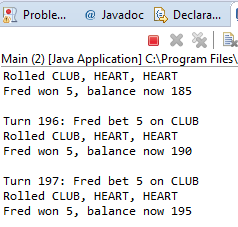
Evidence of resolution.

****

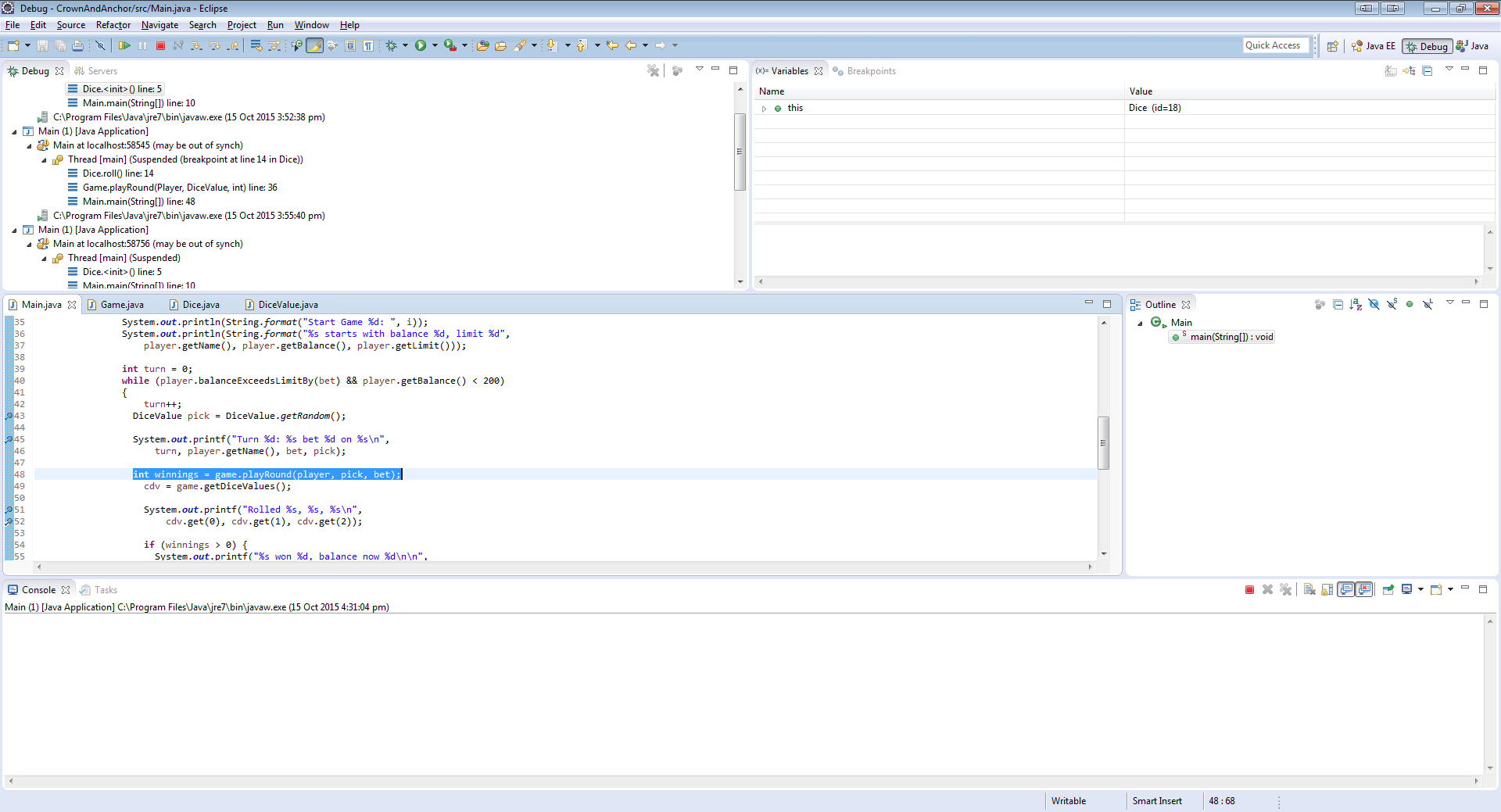
### Bug 4. The value of the dice never change from turn to turn.

Please note: this bug was fixed after bug 1 and 2, but before bug 3.

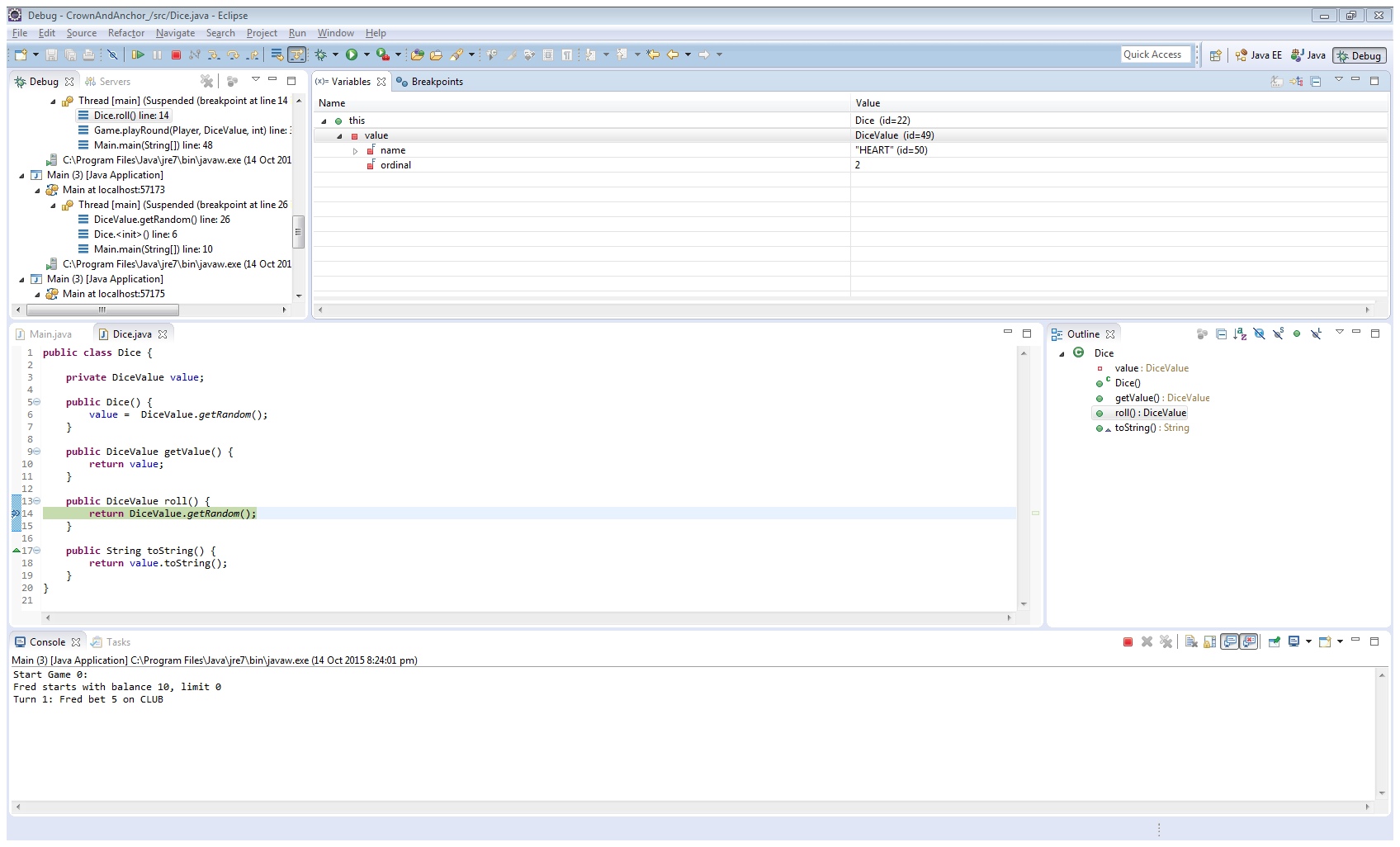
Evidence of bug replicated.

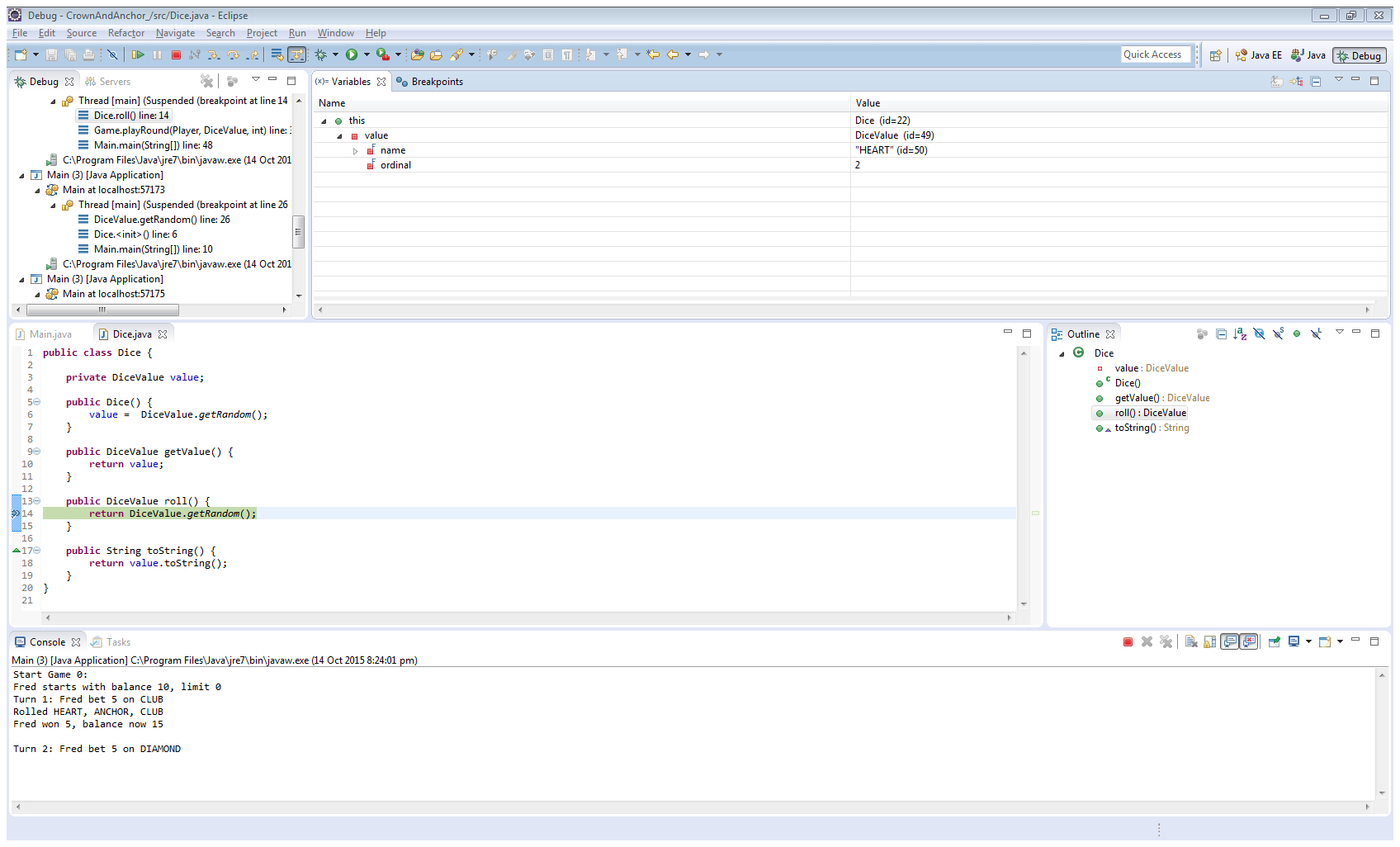


Running in debug mode – Main calls playGame.

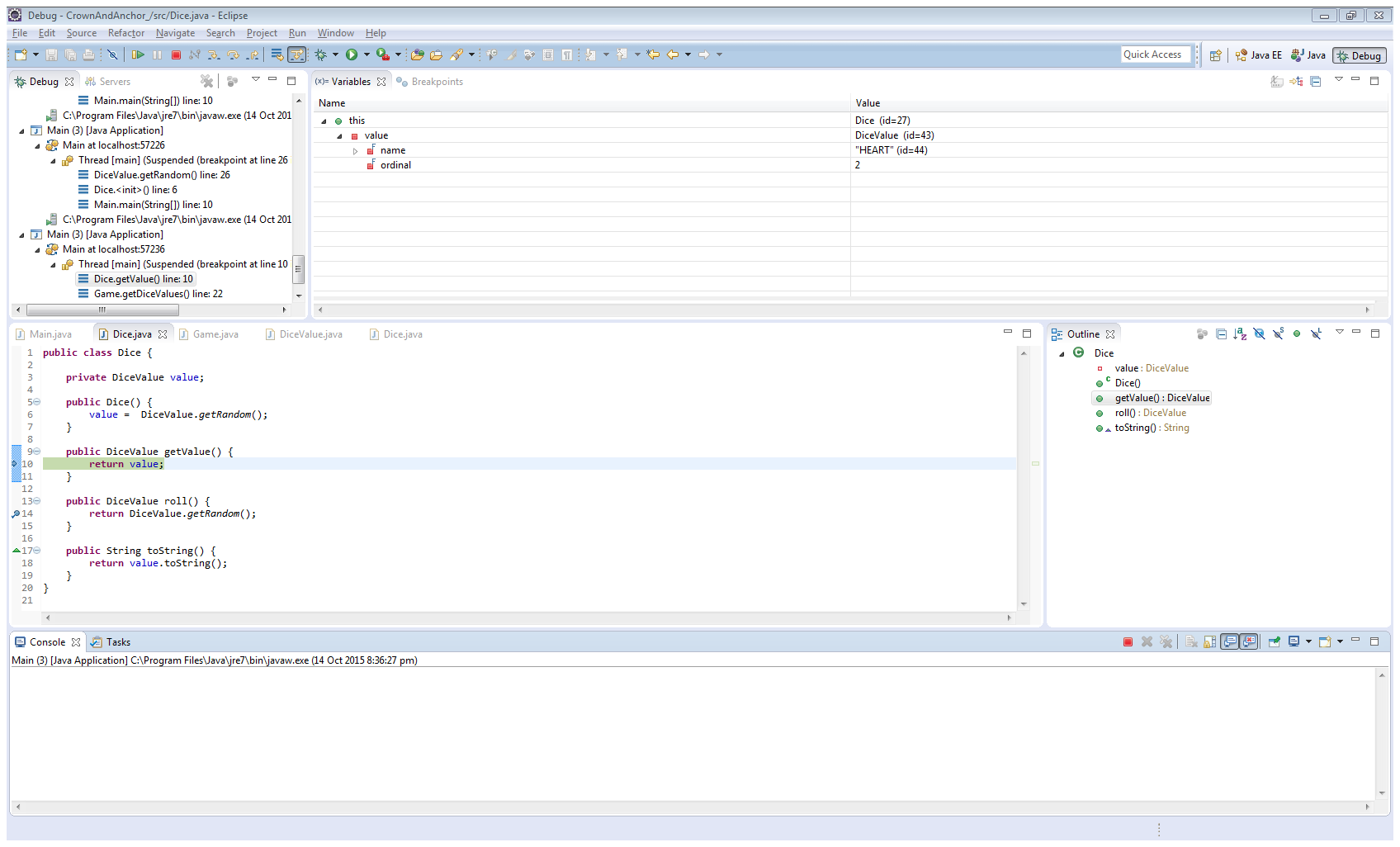


The playGame method in Game class calls Dice’s roll method, currently the first die is a Heart.



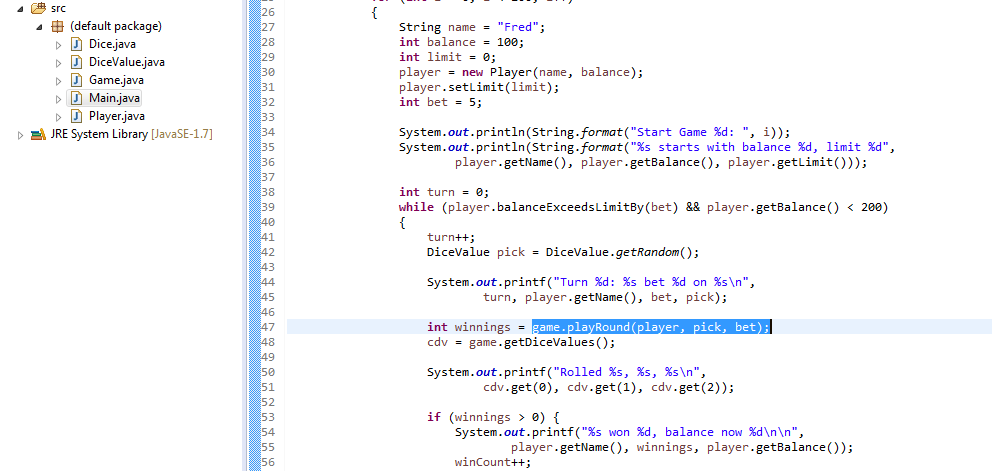
The second turn - the first die is still a Heart. 

The method which returns the value of a die is Dice’s getValue method. Upon inspection, it appears that the reason the value of the die never changes is because when the new value is received in the method roll, it is not stored in the class’s variable, named value.

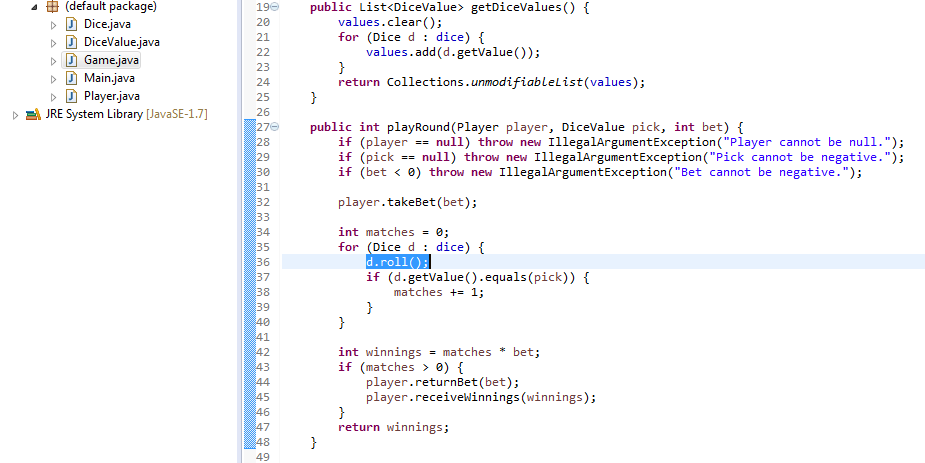


Tracing the code.

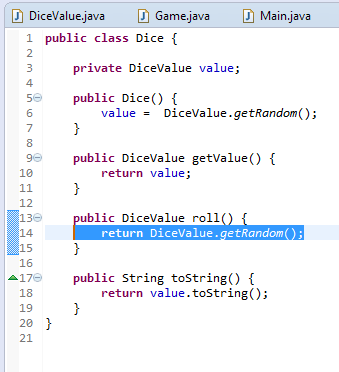
Main calls the playRound method of the Game class.



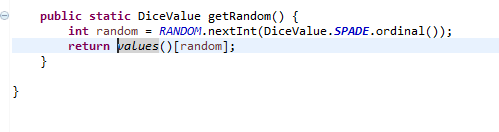
The playRound method of the Game class then calls the Dice class’s roll method.



The roll method of the Dice class then calls the DiceValue class’s getRandom method.

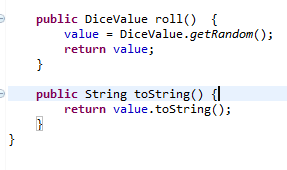


The getRandom method of the DiceValue class then returns a random die value.



However, when the code resumes operation in the roll method of the Dice class, the new value is not stored.

The resolution is call the getRandom method of the DiceValue class, and then store the returned variable in the Dice class’s private variable named ‘value’, as below.

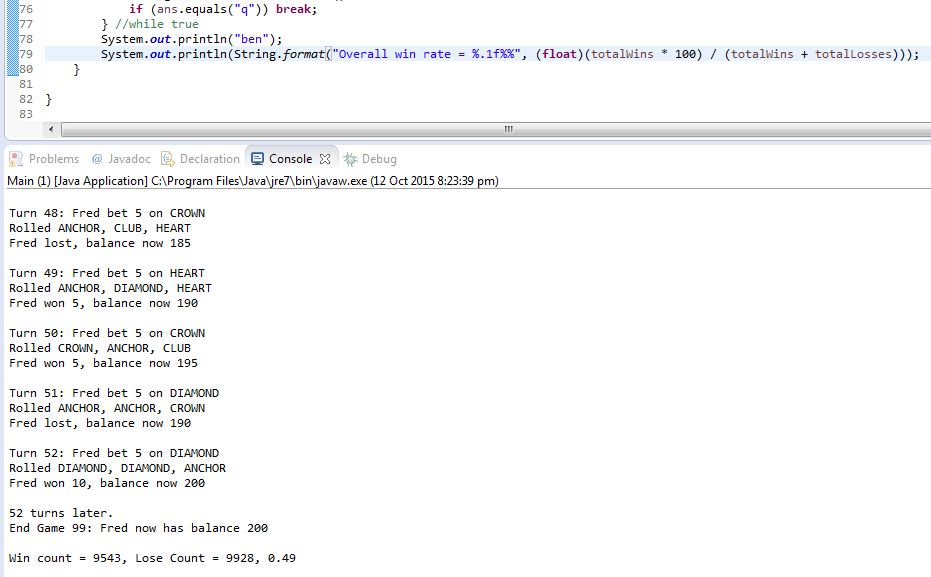


Evidence of resolution.

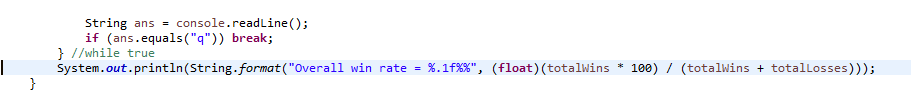


### Bug 5. Console never prints overall win rate.

Evidence of bug replicated.



Furthermore, the console is never used as IO operator.



Very straight forward - removed while loop that would run until false, and also removed two lines above the closing parenthesis of the while loop.

Evidence of resolution.

