# Bug 4 Test Investigation/Fix Log

## Description

Each run of the program shows the same dice rolls repeating as the game’s rolls, even though the player’s roll differs each time. Bug 4 is visible in all test results that test the main() method in the game.

## Hypothesis 1

The bug is caused because when the three dice are rolled for the game (i.e. class Dice is instantiated three times), this happens only once at the beginning of the game. When the Game class is instantiated, it takes the three dice objects already created and uses these same dice objects for its methods such as Game.playRound(). When the dice objects are instantiated, they call DiceValue.getRandom() to get the random values, but these values will remain the same in the dice object unless DiceValue.getRandom() is called again. However, when the player’s dice is rolled (the “pick”), the DiceValue.getRandom() is called directly, so a different random value is achieved each turn for the player. This bug would be fixed by instantiating the three dice objects at the start of each turn of the game rather than just once at the beginning.

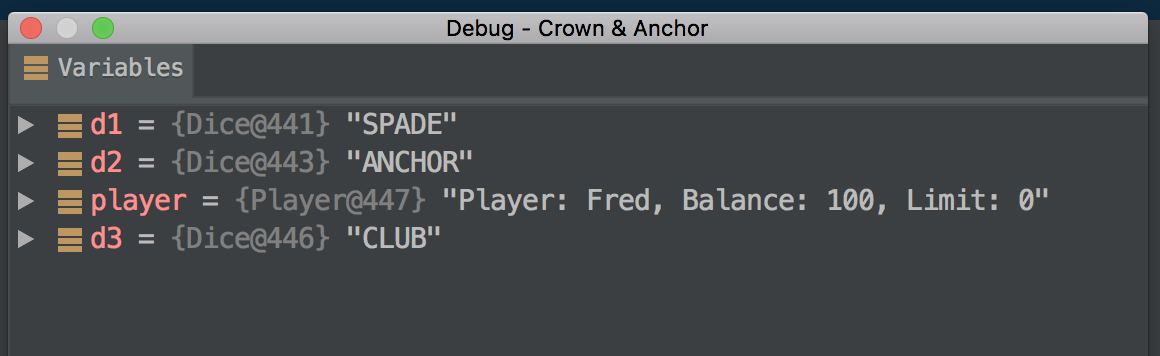
## Action 1: Test Hypothesis

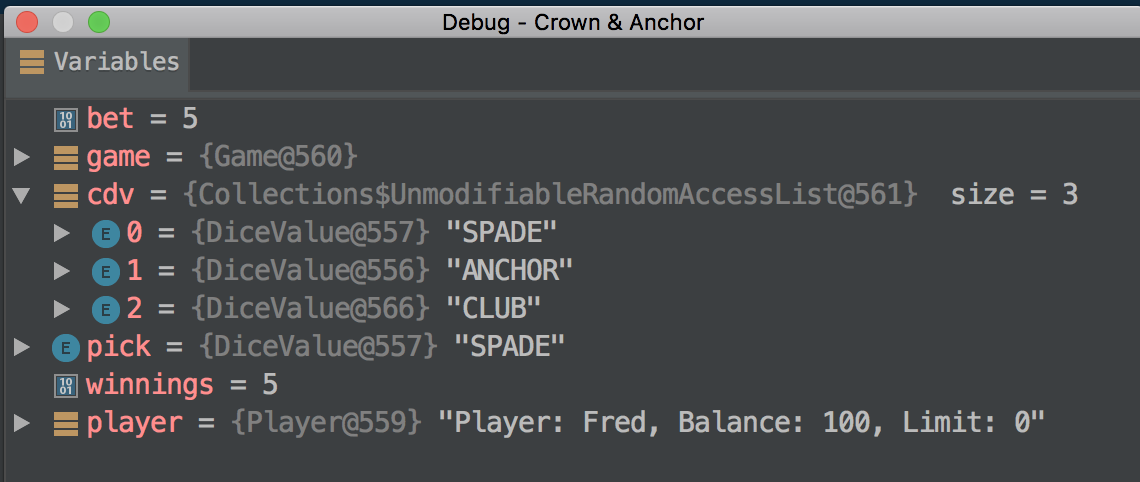
Place breakpoints into the main() method to observe the values of dice rolls at the beginning and during each turn of the game.

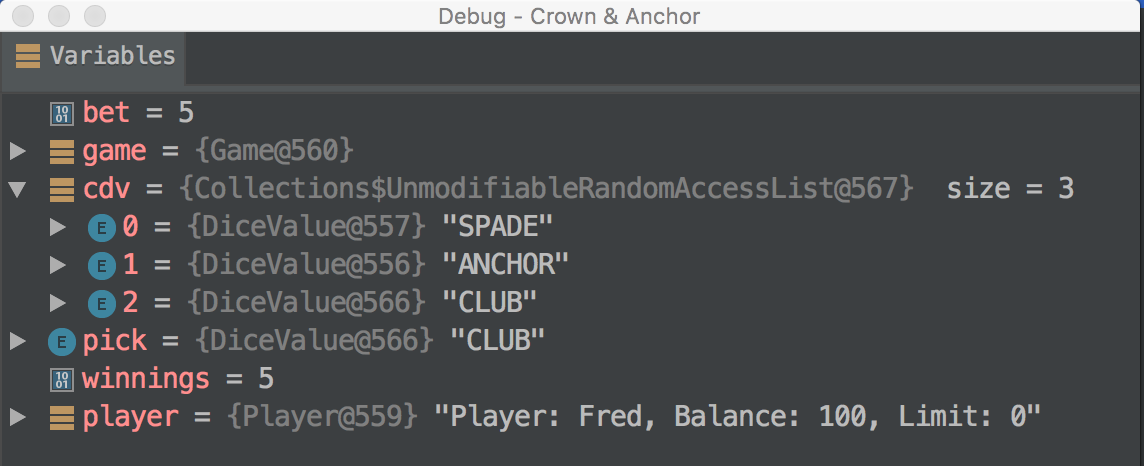
## Results of Test

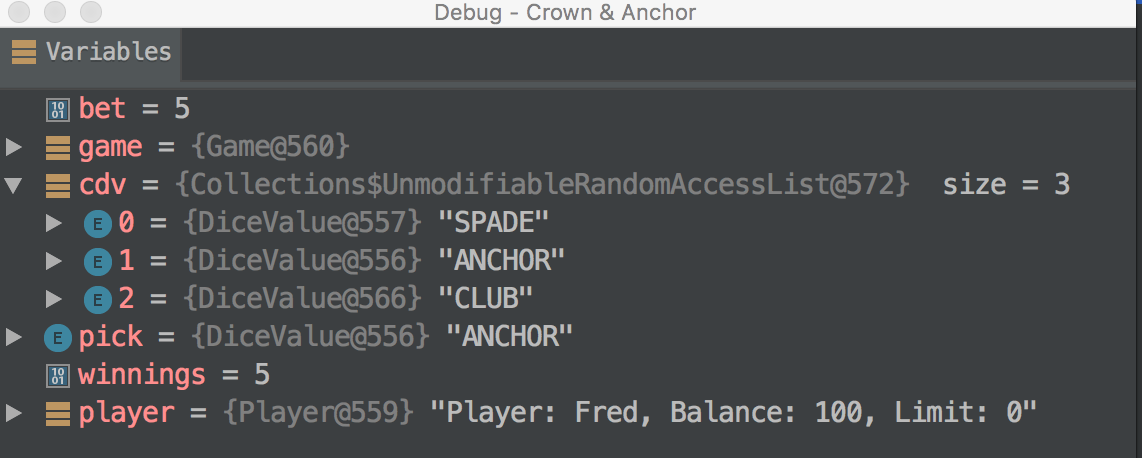
The test of main() proves the hypothesis – variables show that the same dice values for the game are visible in each turn although the player’s roll differs. See the screenshots of variable values below.

Screenshots of variables in main():









## Action 2: Fix main()

The three Dice objects were instantiated and a new Game object also instantiated taking the three Dice objects as arguments at the start of each turn. The game was then run to test the results of the change.

## Results of Test

When the game was run with main() code changes as above, the results were correct, with the game rolls differing each turn. The ratio of player wins is now within the correct range of approximately 0.42 (42%). The hypothesis has been proven. See the screenshots below:

Screenshots of game output:

