Debugging document

ITC205\_Assignment Item 4

CSU SYDNEY STUDY CENTRE

Professional Programming Practice

**Table of Contents**

[**Replication** 3](#_Toc464434695)

[1. Produce Random Dice Values 3](#_Toc464434696)

[2. Game Pay Out at Correct Level 4](#_Toc464434697)

[3. Game Ends When Reached Betting Limit 5](#_Toc464434698)

[4. Odds in The Game Appears To Be Correct 6](#_Toc464434699)

[**Simplification** 7](#_Toc464434700)

[1. Produce Random Dice Values 7](#_Toc464434701)

[Testing Target Code 7](#_Toc464434702)

[Current Result 7](#_Toc464434703)

[2. Game Pay Out at Correct Level 8](#_Toc464434704)

[Testing Target Code 8](#_Toc464434705)

[Current Result 8](#_Toc464434706)

[3. Game Ends When Reached Betting Limit 9](#_Toc464434707)

[Testing Target Code 9](#_Toc464434708)

[Current Result 9](#_Toc464434709)

[4. Odds in The Game Appears To Be Correct 10](#_Toc464434710)

[Testing Target Code 10](#_Toc464434711)

[Current Result 10](#_Toc464434712)

[**Tracing** 11](#_Toc464434713)

[1. Produce Random Dice Values 11](#_Toc464434714)

[Automated Testing Class and method 11](#_Toc464434715)

[Testing Result 11](#_Toc464434716)

[Origin of bug found 12](#_Toc464434717)

[2. Game Pay Out at Correct Level 13](#_Toc464434718)

[Automated Testing Class and method 13](#_Toc464434719)

[Testing Result 13](#_Toc464434720)

[Origin of bug found 14](#_Toc464434721)

[3. Game Ends When Reached Betting Limit 15](#_Toc464434722)

[Automated Testing Class and method 15](#_Toc464434723)

[Testing Result 15](#_Toc464434724)

[Origin of bug found 16](#_Toc464434725)

[4. Odds in The Game Appears To Be Correct 17](#_Toc464434726)

[Automated Testing Class and method 17](#_Toc464434727)

[Testing Result 17](#_Toc464434728)

[Origin of bug found 18](#_Toc464434729)

[**Resolution** 19](#_Toc464434730)

[1. Produce Random Dice Values 19](#_Toc464434731)

[Fixed Code 19](#_Toc464434732)

[Fixed Output 19](#_Toc464434733)

[2. Game Pay Out at Correct Level 20](#_Toc464434734)

[Fixed Code 20](#_Toc464434735)

[Fixed Output 20](#_Toc464434736)

[3. Game Ends When Reached Betting Limit 21](#_Toc464434737)

[Fixed Code 21](#_Toc464434738)

[Fixed Output 21](#_Toc464434739)

[4. Odds in The Game Appears To Be Correct 22](#_Toc464434740)

[Fixed Code 22](#_Toc464434741)

[Fixed Output 22](#_Toc464434742)

# **Replication**

## 1. Produce Random Dice Values

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Name** | | Produce random dice values | | | |
| **Use Case Tested:** | |  | | | |
| **Test Description:** | | Check if application produce random Dice Value properly | | | |
| **Pre-conditions** | | DiceValue Class is ready and set for producing randomized dice values | | | |
| **Post-conditions** | | In each game, random set of 3 dice values has to be produced | | | |
| **Notes:** | | *Test step would be just running Application as game starts as soon as application run* | | | |
| **Result**  **(Pass/Fail/Warning/Incomplete)** | | Fail | | | |
|  | **Test Step** | **Expected Test Results** | **Actual Results** | **P** | **F** |
| 1. | Run Application | Produce random dice values for each turn of a game | Produced same values for every turn of games |  | **X** |
| 2. |  |  |  |  |  |
| 3. |  |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Data Table** | | | | | |
|  | **1** | **2** | **3** | **4** | **5** |
| **Player (Domain)** | Player.name = “Fred”; (String) | Player.limit = 0;  (int) | Player.balance = 100; (int) |  |  |
| **bet** | bet = 5; |  |  |  |  |
| **DiceValue** | *CROWN*, *ANCHOR*, *HEART*, *DIAMOND*, *CLUB*, *SPADE*; |  |  |  |  |
| **limit** | limit = 0; (int) |  |  |  |  |

## 2. Game Pay Out at Correct Level

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Name** | | Game pay out at correct level | | | |
| **Use Case Tested:** | |  | | | |
| **Test Description:** | | Check if application pay winning prize at correct level | | | |
| **Pre-conditions** | | Game Class is ready and set for producing winning prize value | | | |
| **Post-conditions** | | Game pays Player at correct level based on amount of bet | | | |
| **Notes:** | | *Test step would be just running Application as game starts as soon as application run* | | | |
| **Result**  **(Pass/Fail/Warning/Incomplete)** | | Fail | | | |
|  | **Test Step** | **Expected Test Results** | **Actual Results** | **P** | **F** |
| 1. | Run Application | Pay Player when player win | Paid Player when player won | **X** |  |
| 2. | Run Application | Winning prize is correct | Winning prize does not include the money that Player bet |  | **X** |
| 3. |  |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Data Table** | | | | | |
|  | **1** | **2** | **3** | **4** | **5** |
| **Player (Domain)** | Player.name = “Fred”; (String) | Player.limit = 0;  (int) | Player.balance = 100; (int) |  |  |
| **bet** | bet = 5; (int) |  |  |  |  |
| **DiceValue** | *CROWN*, *ANCHOR*, *HEART*, *DIAMOND*, *CLUB*, *SPADE*; |  |  |  |  |
| **limit** | limit = 0; (int) |  |  |  |  |

## 3. Game Ends When Reached Betting Limit

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Name** | | Game ends when reached betting limit | | | |
| **Use Case Tested:** | |  | | | |
| **Test Description:** | | Check if game ends when reached betting limit | | | |
| **Pre-conditions** | | Limit to be set | | | |
| **Post-conditions** | | Game ends when Player reached limit | | | |
| **Notes:** | | *Test step would be just running Application as game starts as soon as application run* | | | |
| **Result**  **(Pass/Fail/Warning/Incomplete)** | | Fail | | | |
|  | **Test Step** | **Expected Test Results** | **Actual Results** | **P** | **F** |
| 1. | Run Application | Game ends when reached Player’s limit | Game ended when Player still could bet one more time |  | **X** |
| 2. |  |  |  |  |  |
| 3. |  |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Data Table** | | | | | |
|  | **1** | **2** | **3** | **4** | **5** |
| **Player (Domain)** | Player.name = “Fred”; (String) | Player.limit = 0;  (int) | Player.balance = 100; (int) |  |  |
| **bet** | bet = 5; |  |  |  |  |
| **DiceValue** | *CROWN*, *ANCHOR*, *HEART*, *DIAMOND*, *CLUB*, *SPADE*; |  |  |  |  |
| **limit** | limit = 0; (int) |  |  |  |  |

## 4. Odds in The Game Appears To Be Correct

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Name** | | Odds in the game appears to be correct | | | |
| **Use Case Tested:** | |  | | | |
| **Test Description:** | | Check if the application produce correct winning ratio | | | |
| **Pre-conditions** | | Successfully get random dice values | | | |
| **Post-conditions** | | Winning ratio should approximately equal to 42% | | | |
| **Notes:** | | *Test step would be just running Application as game starts as soon as application run* | | | |
| **Result**  **(Pass/Fail/Warning/Incomplete)** | | Fail | | | |
|  | **Test Step** | **Expected Test Results** | **Actual Results** | **P** | **F** |
| 1. | Run Application | Winning ratio is approximately about 42% | Winning ratio seems to be 50% |  | **X** |
| 2. |  |  |  |  |  |
| 3. |  |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Data Table** | | | | | |
|  | **1** | **2** | **3** | **4** | **5** |
| **Player (Domain)** | Player.name = “Fred”; (String) | Player.limit = 0;  (int) | Player.balance = 100; (int) |  |  |
| **bet** | bet = 5; |  |  |  |  |
| **DiceValue** | *CROWN*, *ANCHOR*, *HEART*, *DIAMOND*, *CLUB*, *SPADE*; |  |  |  |  |
| **limit** | limit = 0; (int) |  |  |  |  |

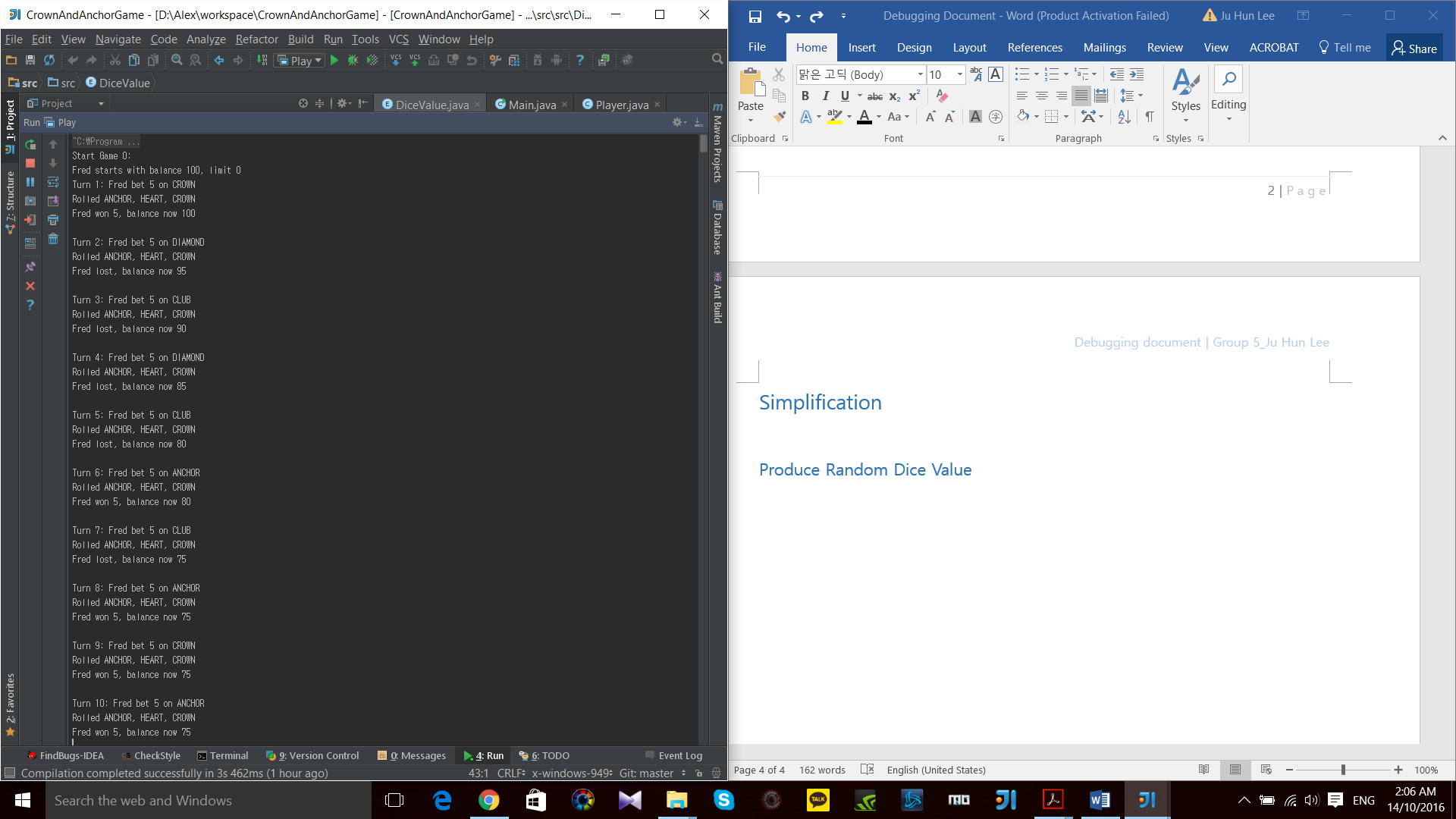
# **Simplification**

## 1. Produce Random Dice Values

### Testing Target Code

while (player.balanceExceedsLimitBy(bet) && player.getBalance() < 200)  
{  
 cdv = game.getDiceValues();  
   
 System.*out*.printf("Rolled %s, %s, %s\n",  
 cdv.get(0), cdv.get(1), cdv.get(2));  
 }

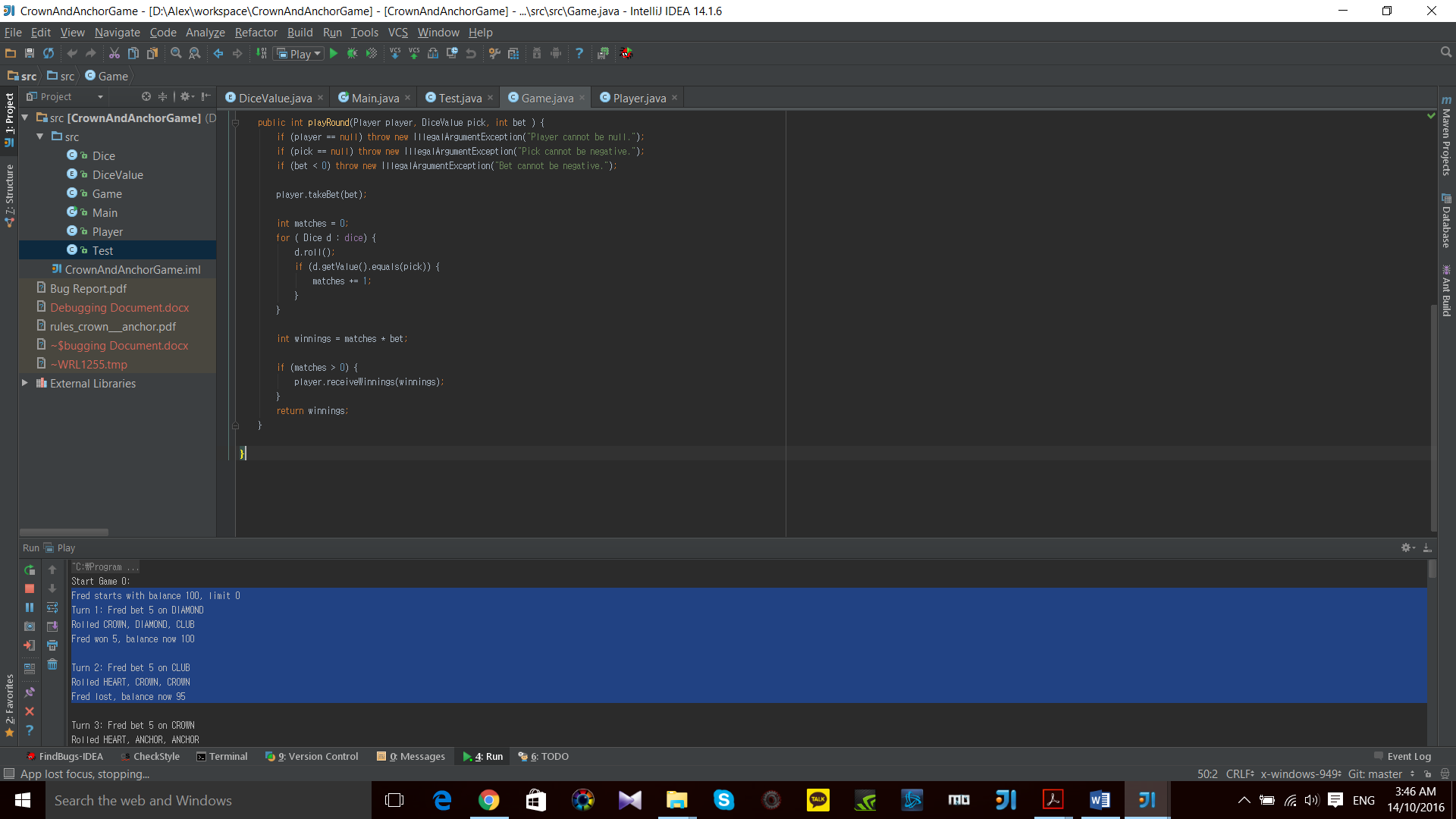
### Current Result



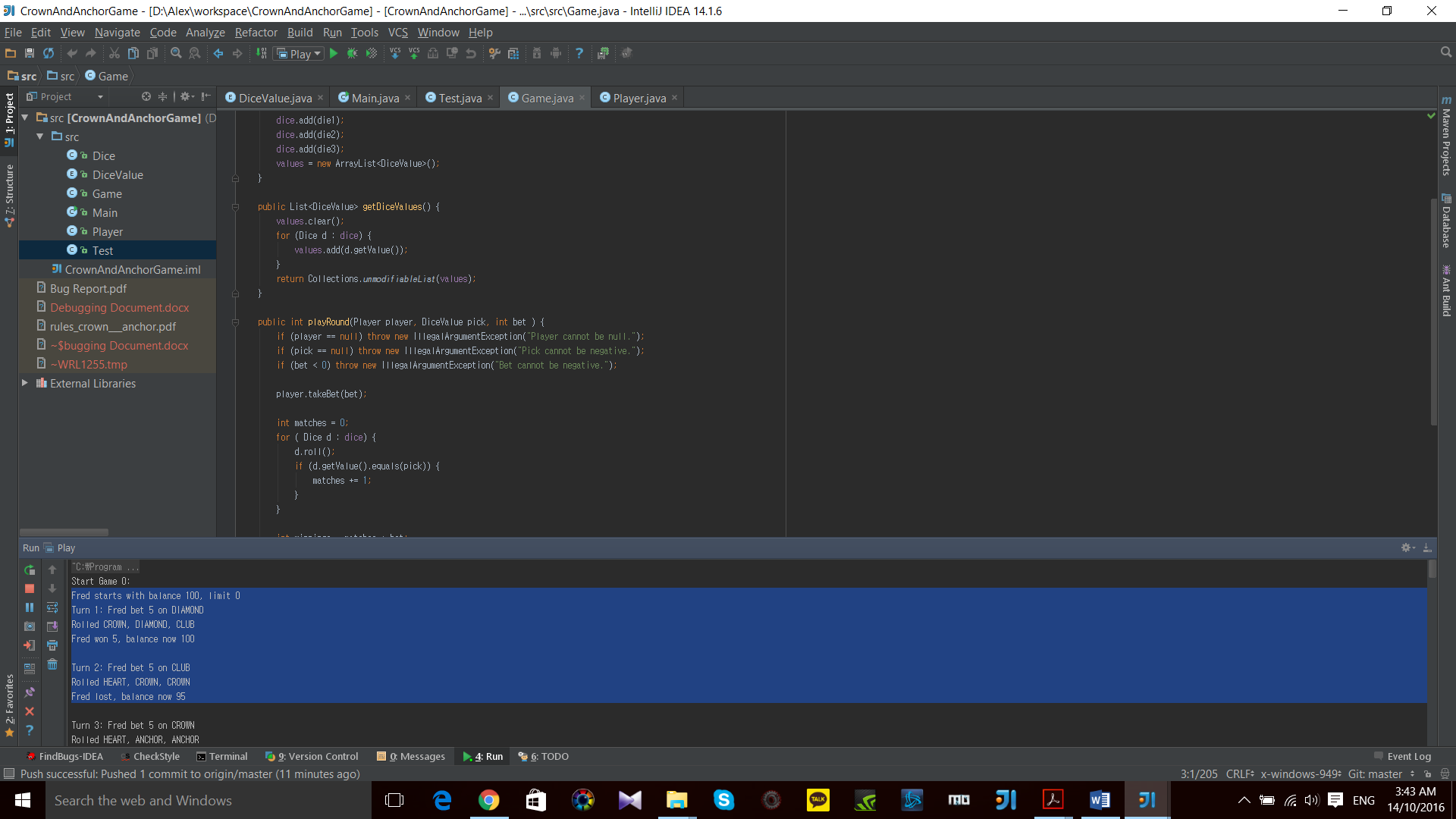
**\* Buggy Behavior: Producing a same set of 3 dices’ value for each turn and each game.**

## 2. Game Pay Out at Correct Level

### Testing Target Code



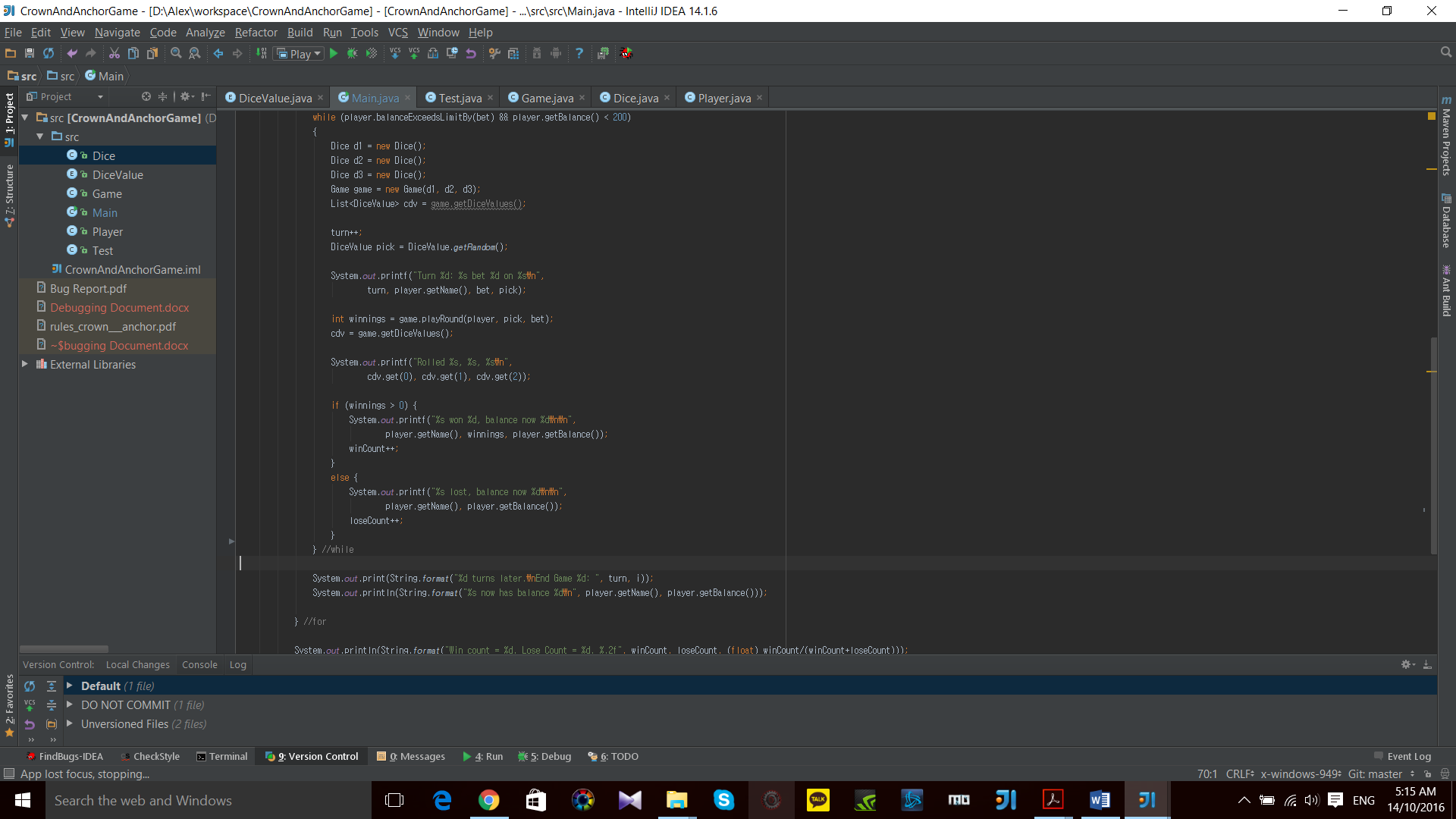
### Current Result



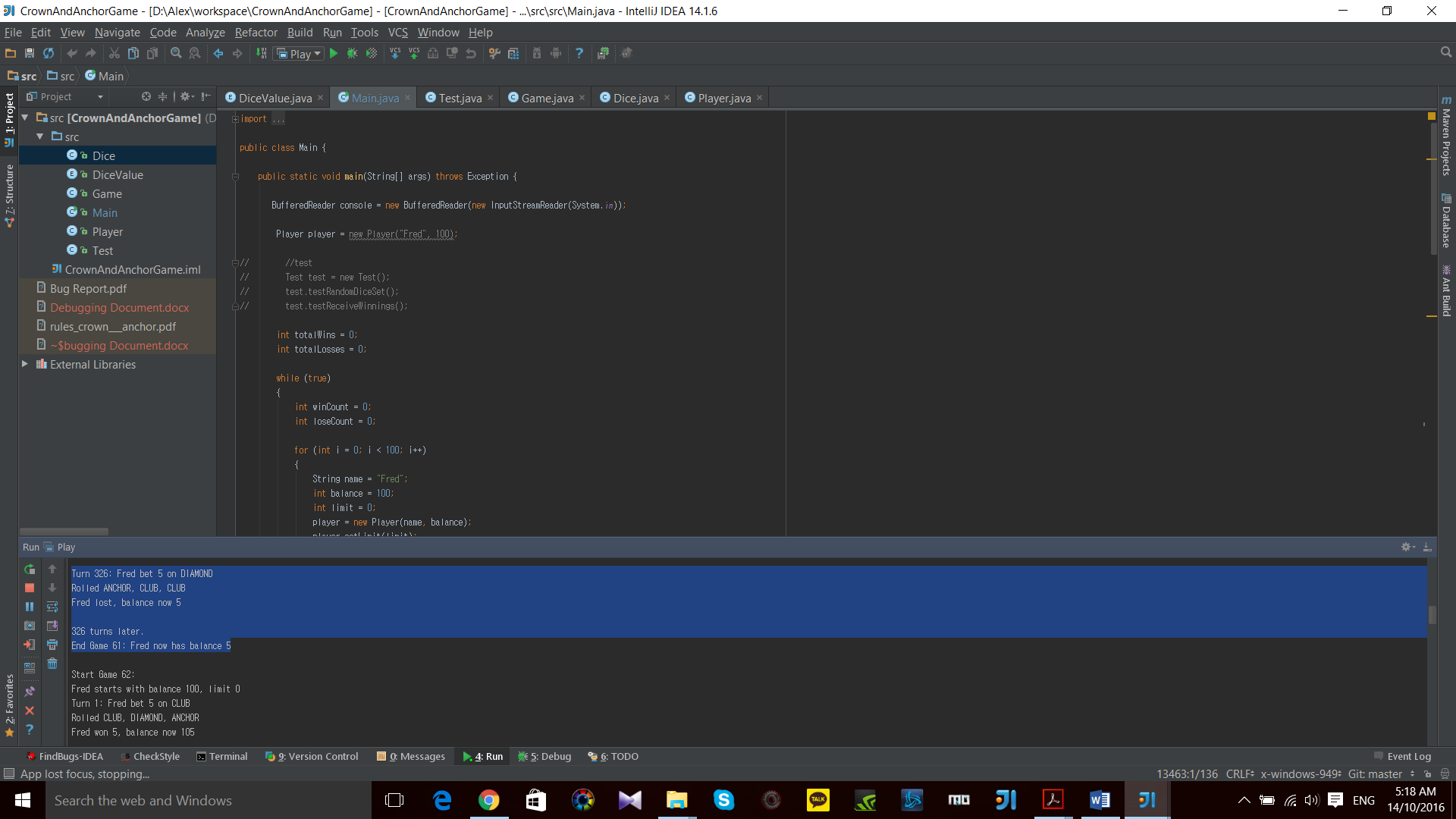
**\* Buggy Behavior: Winning prize does not pay back the bet of turns.**

## 3. Game Ends When Reached Betting Limit

### Testing Target Code



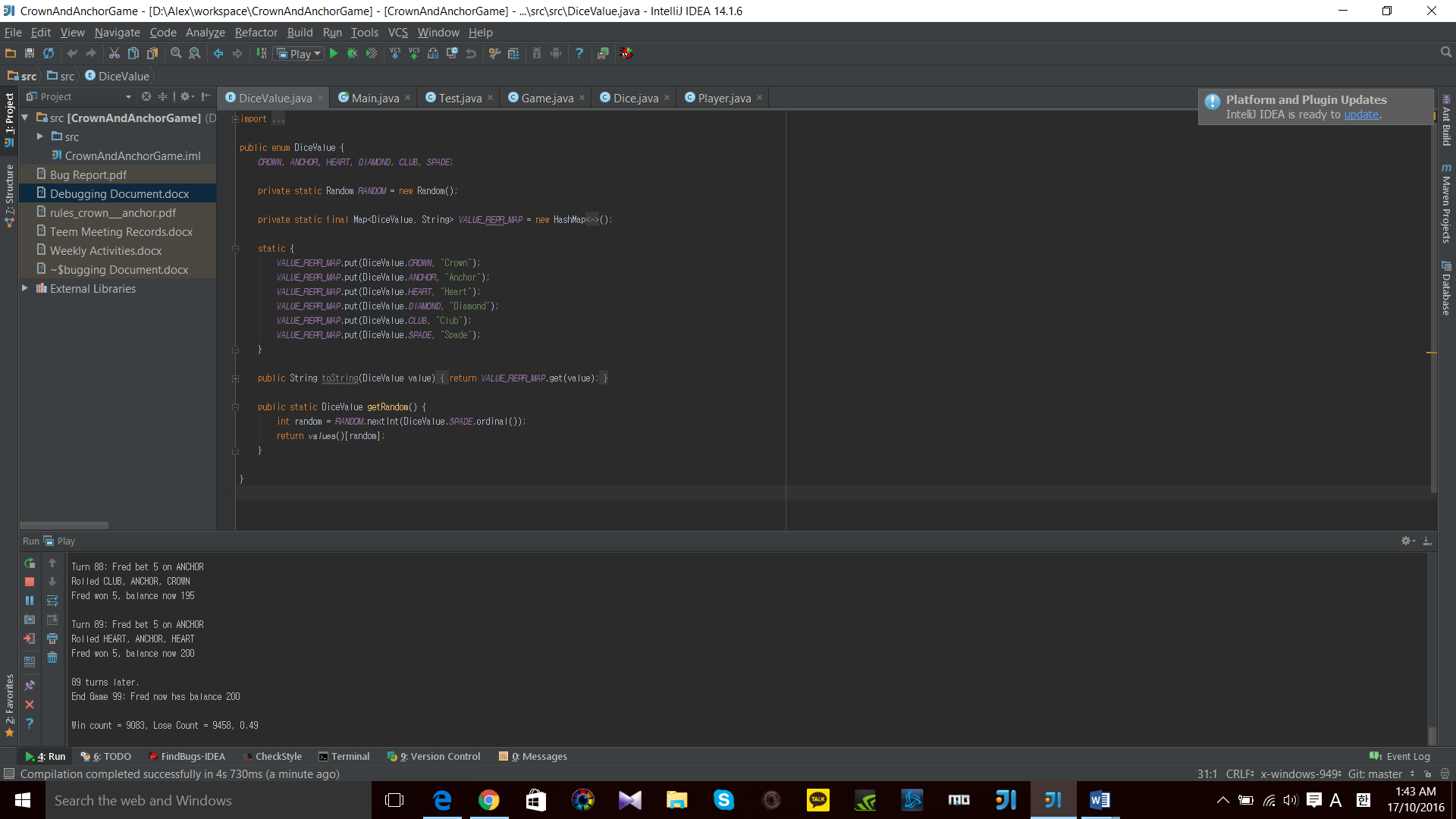
### Current Result



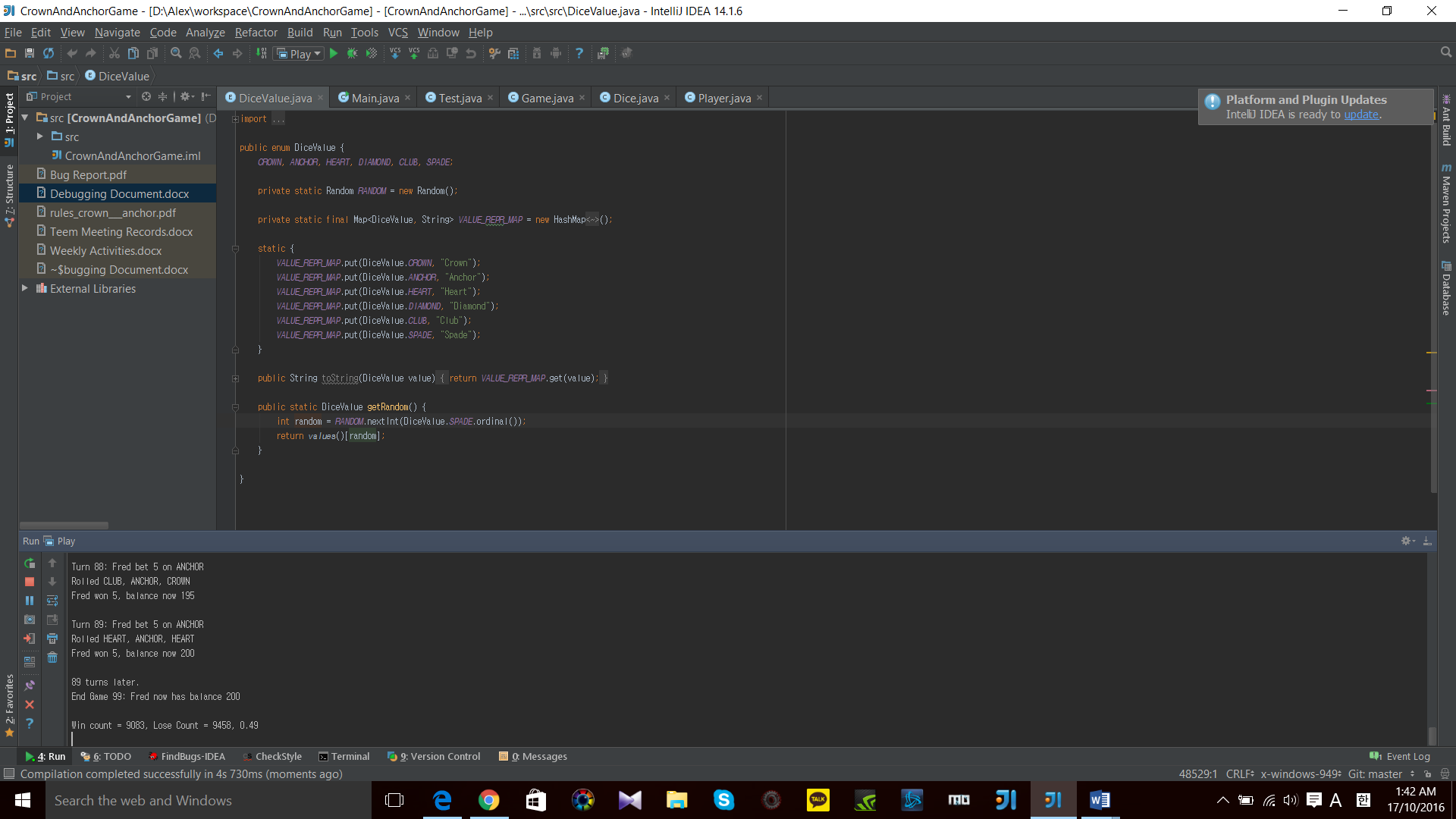
**\* Buggy Behavior: Game ends when Player still has enough balance for another turn.**

## 4. Odds in The Game Appears To Be Correct

### Testing Target Code



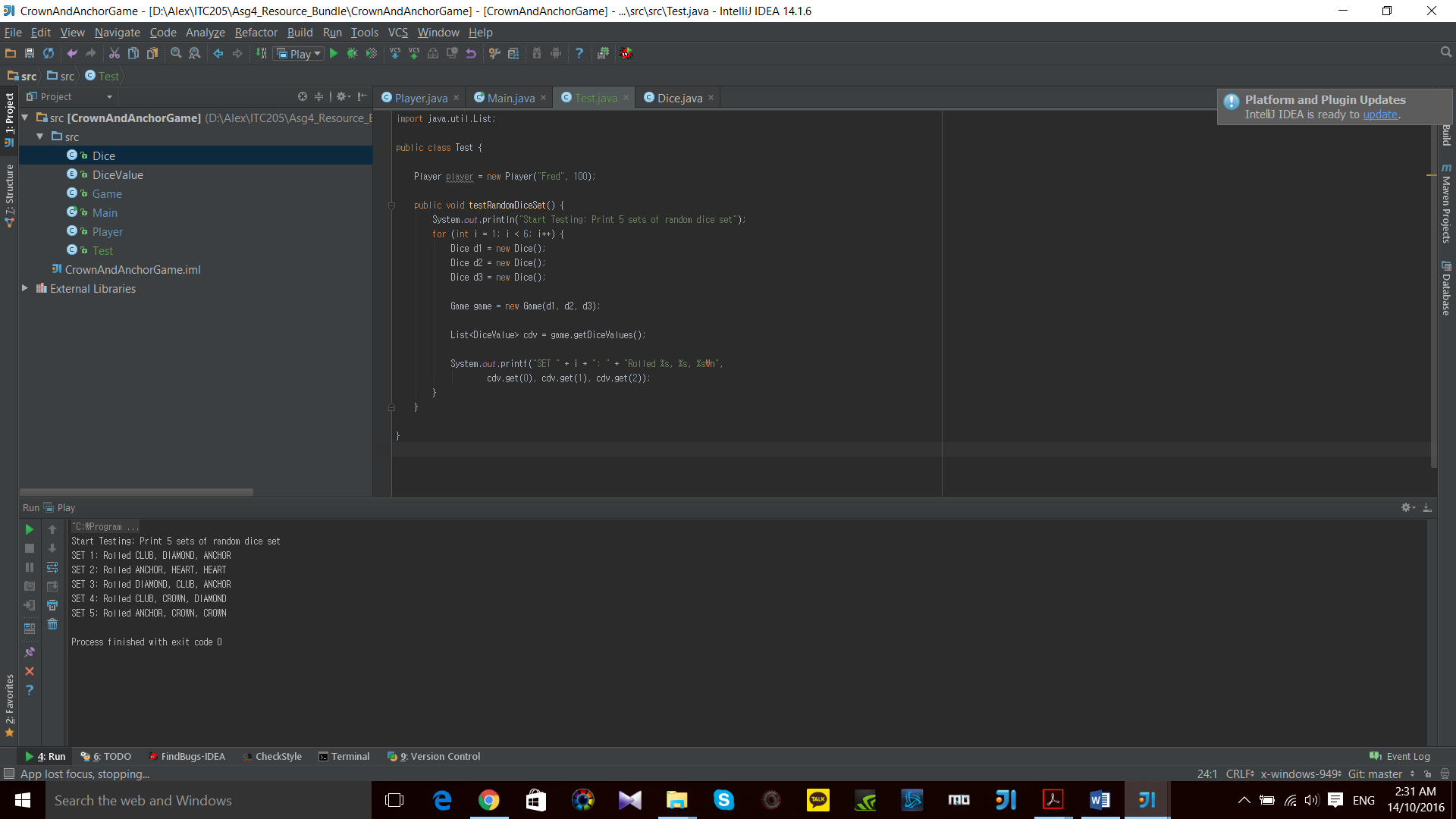
### Current Result



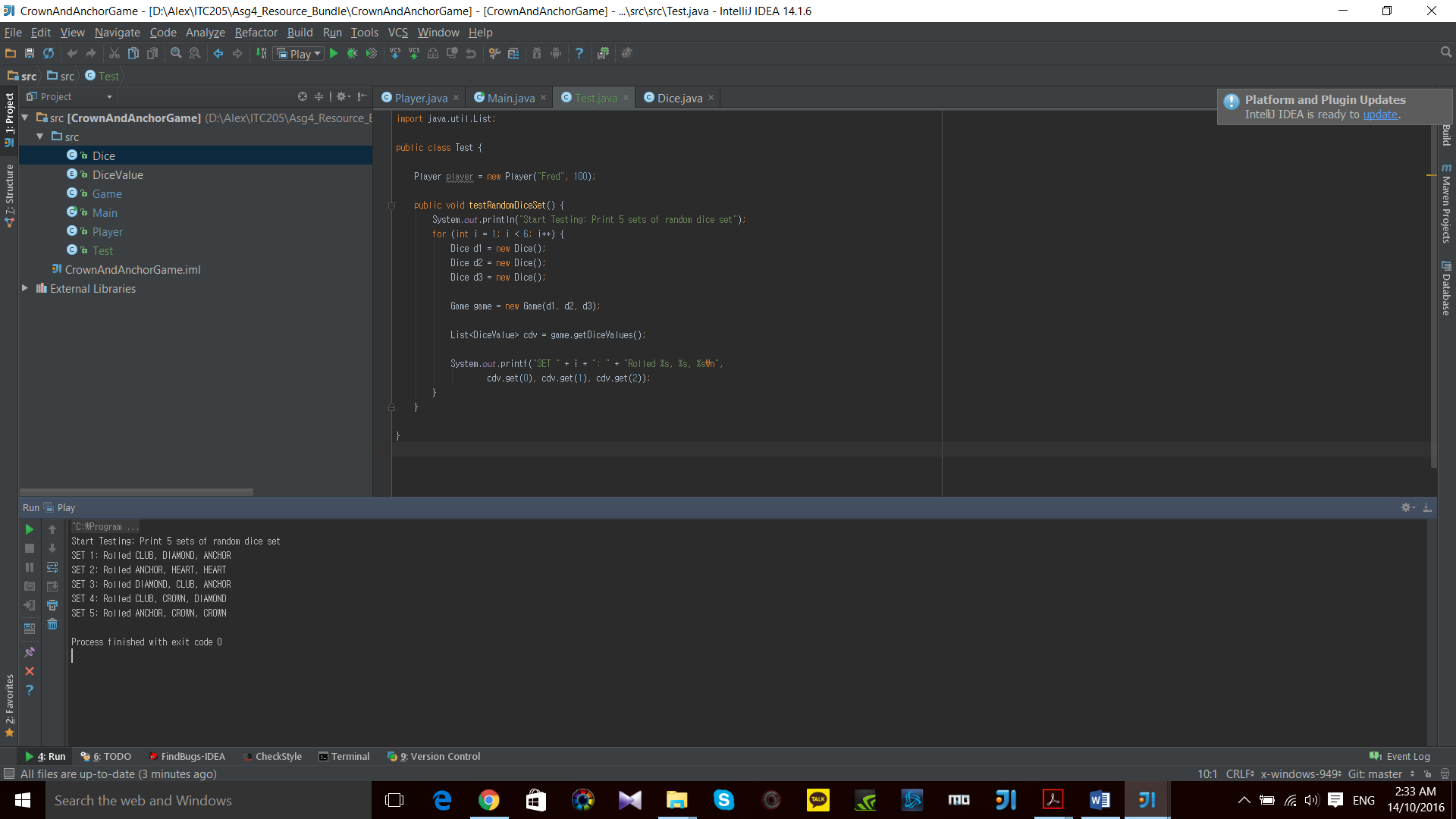
# **Tracing**

## 1. Produce Random Dice Values

### Automated Testing Class and method

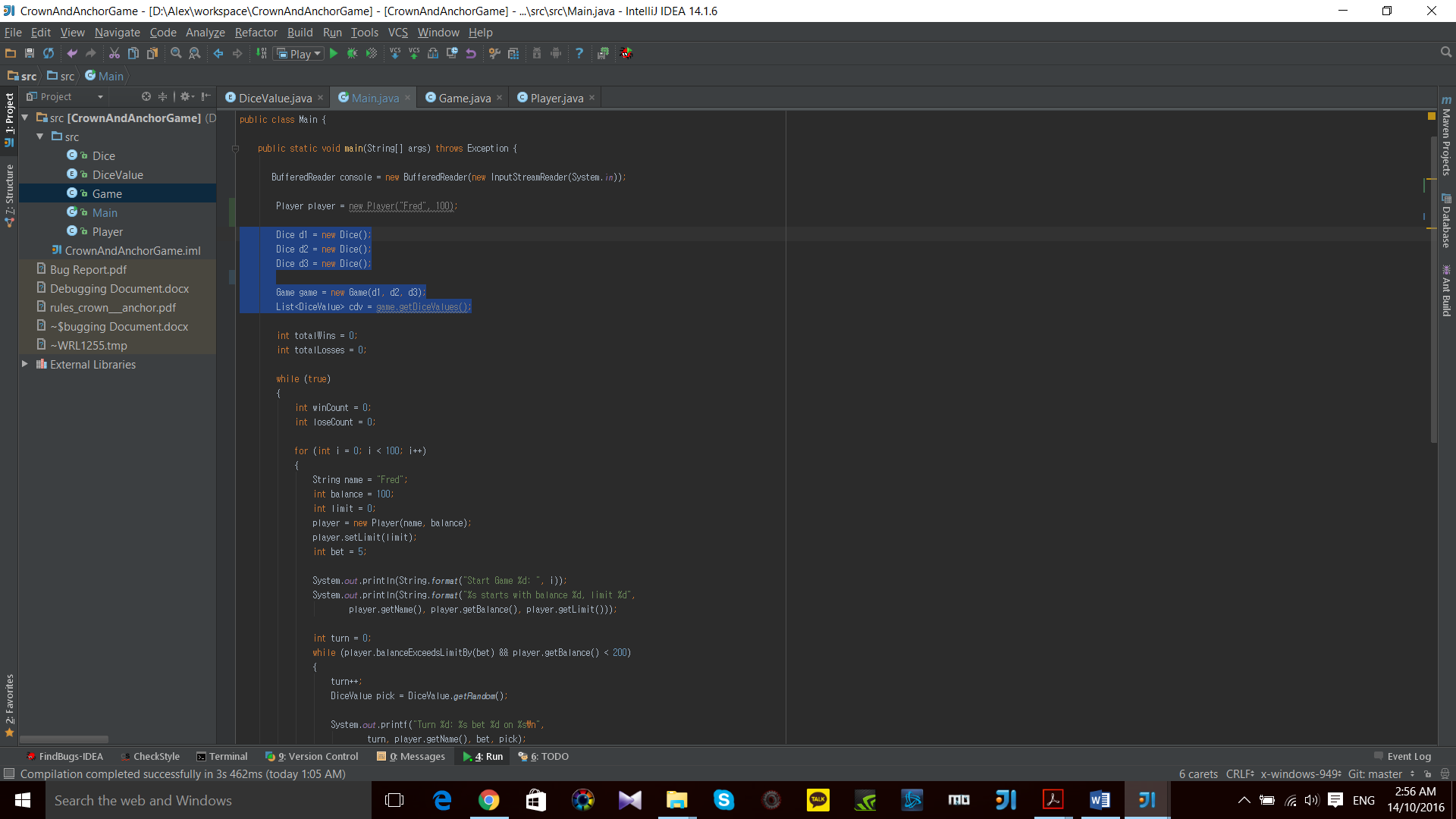


### Testing Result



**\* Producing a set of random dice values for each print. Current buggy behavior from Main class should be caused by logic error in coding.**

### Origin of bug found



1. **Creating new Dice() outside of the loop for ‘game turn’**

Dice d1 = new Dice();  
Dice d2 = new Dice();  
Dice d3 = new Dice();

**\* New Dice(); should be in while loop in order to produce different value for each game.**

1. **Creating a new Game(d1, d2, d3) outside of the loop**

Game game = new Game(d1, d2, d3);

**\* New Game has to be declared in while loop in order to take new Dices’ Value.**

1. **Creating a List of Dice Value outside of the loop**

List<DiceValue> cdv = game.getDiceValues();

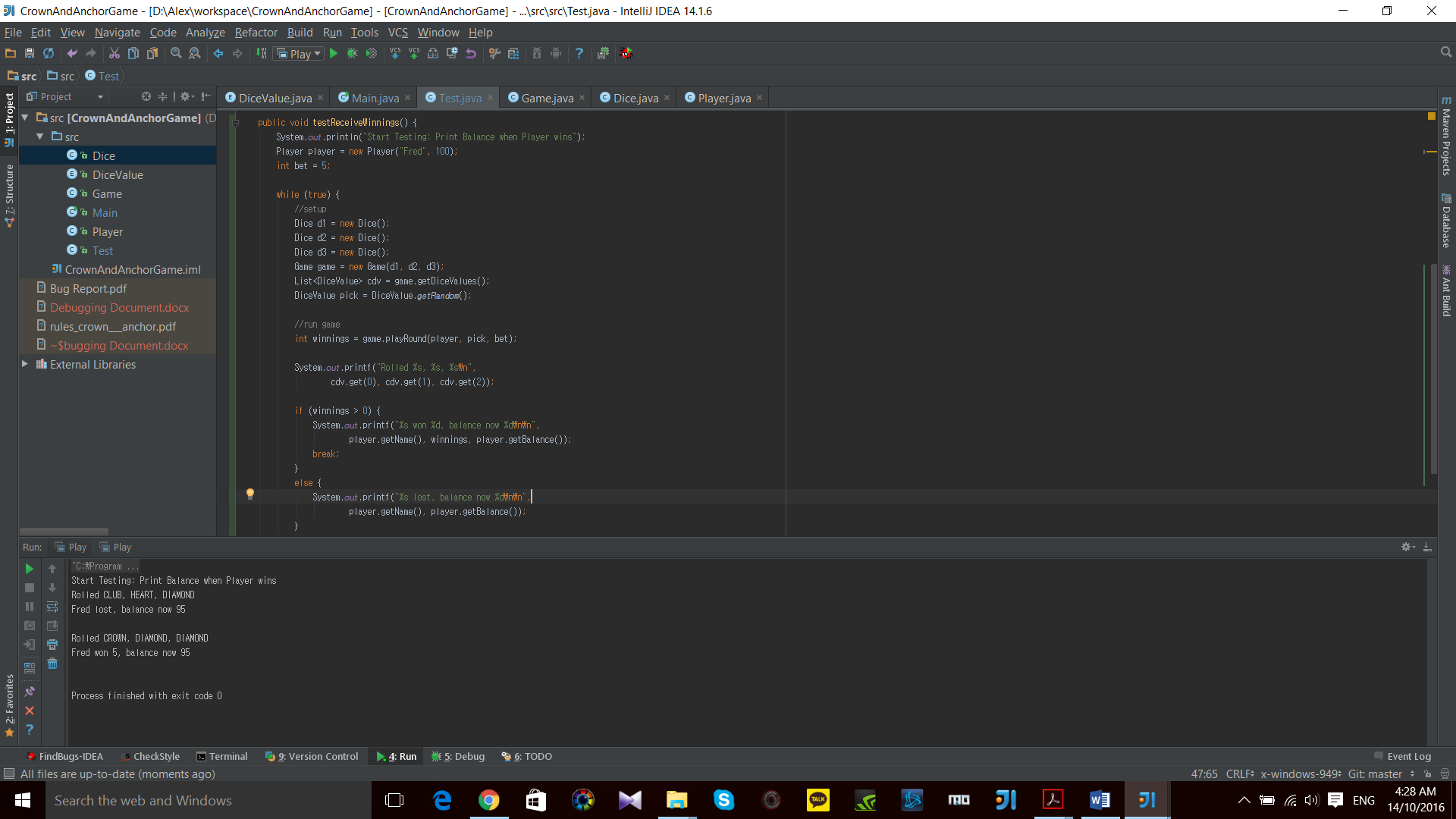
**\* Every Game has a different list of dice values. It should be declared in while loop.**

## 2. Game Pay Out at Correct Level

### Automated Testing Class and method

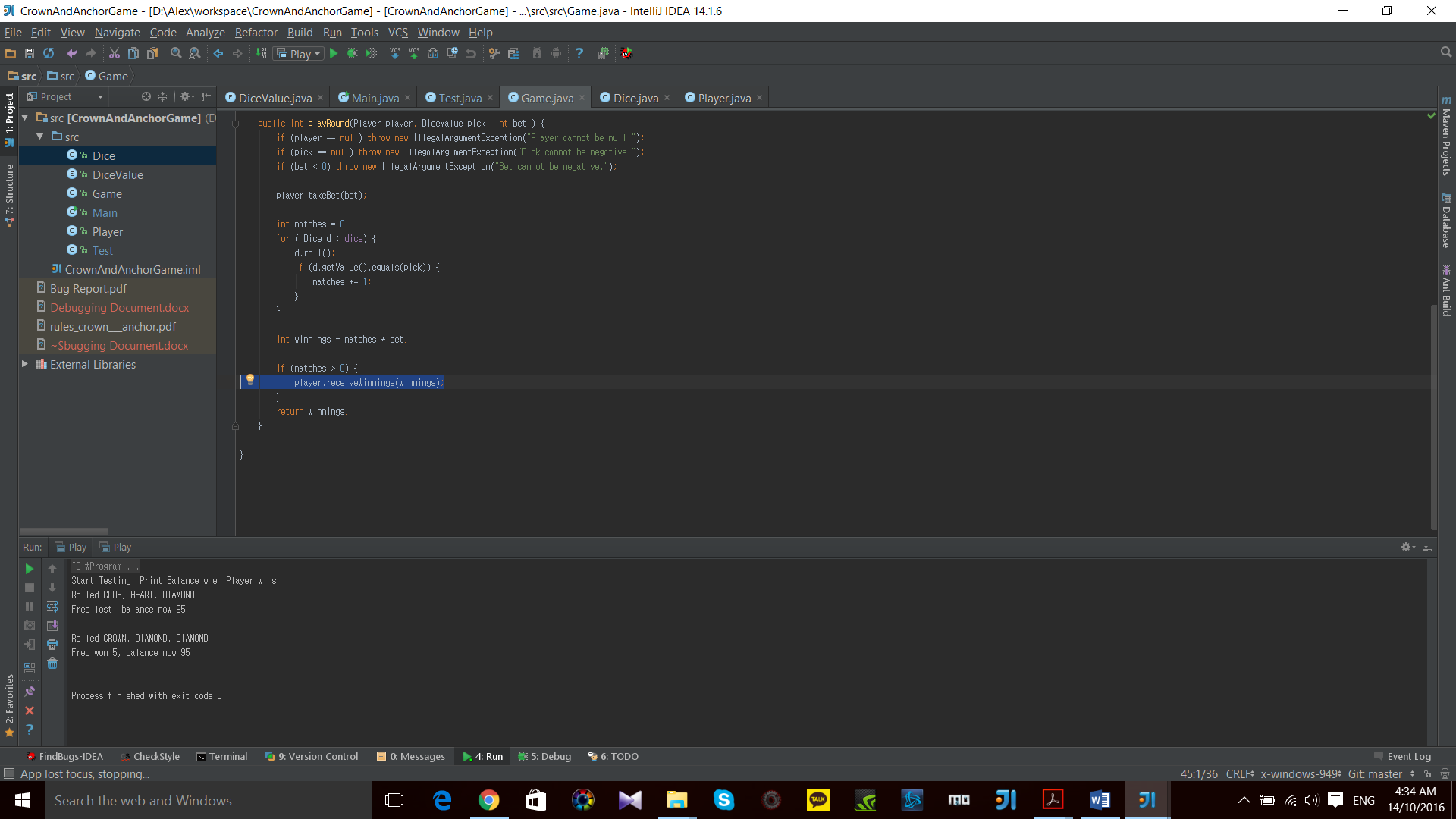


### Testing Result



**\* When player wins, player does not get the betting money back but only winnings.**

### Origin of bug found



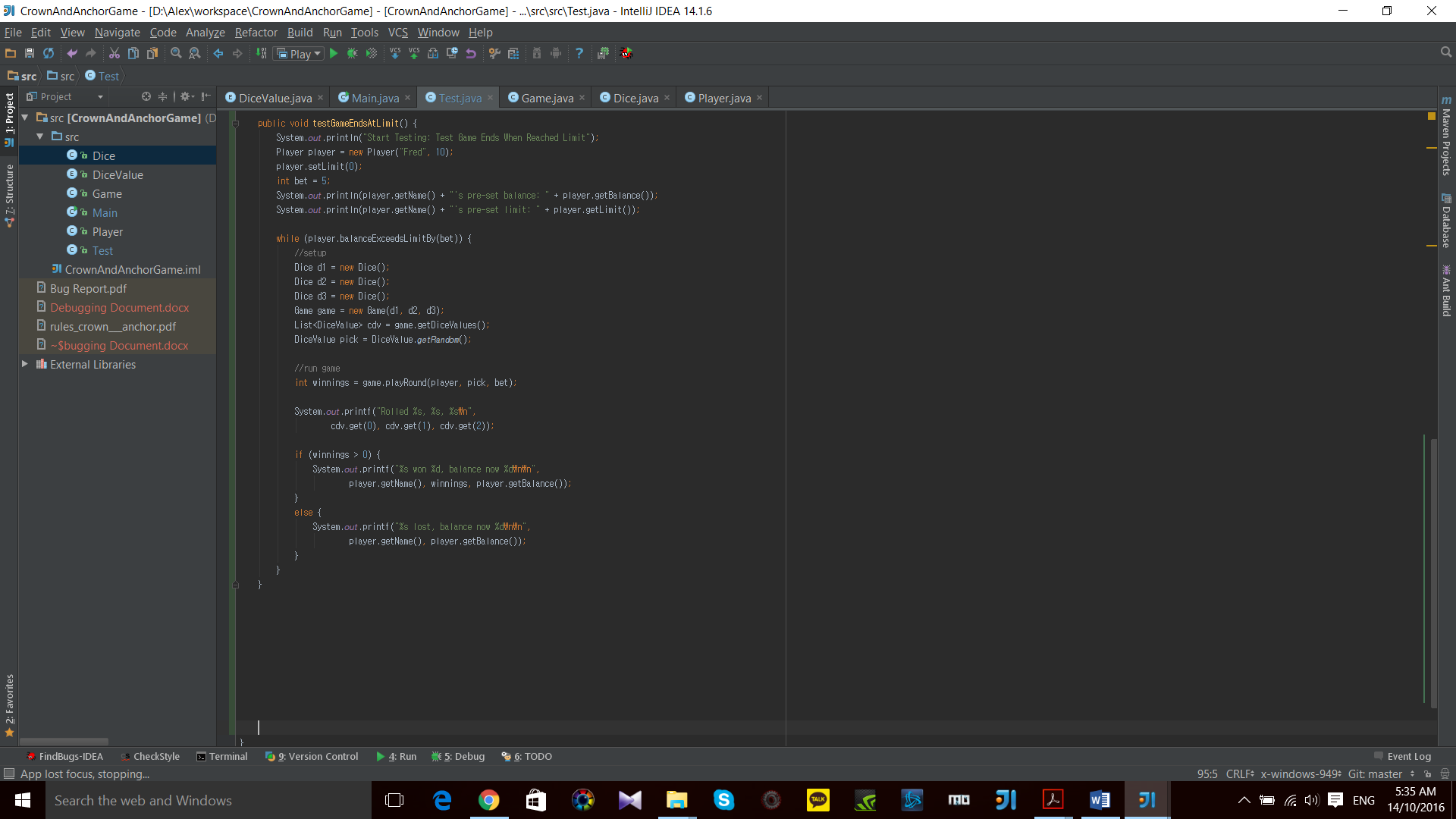
1. **Player only gets to receive winnings without betting cost**

player.receiveWinnings(winnings);

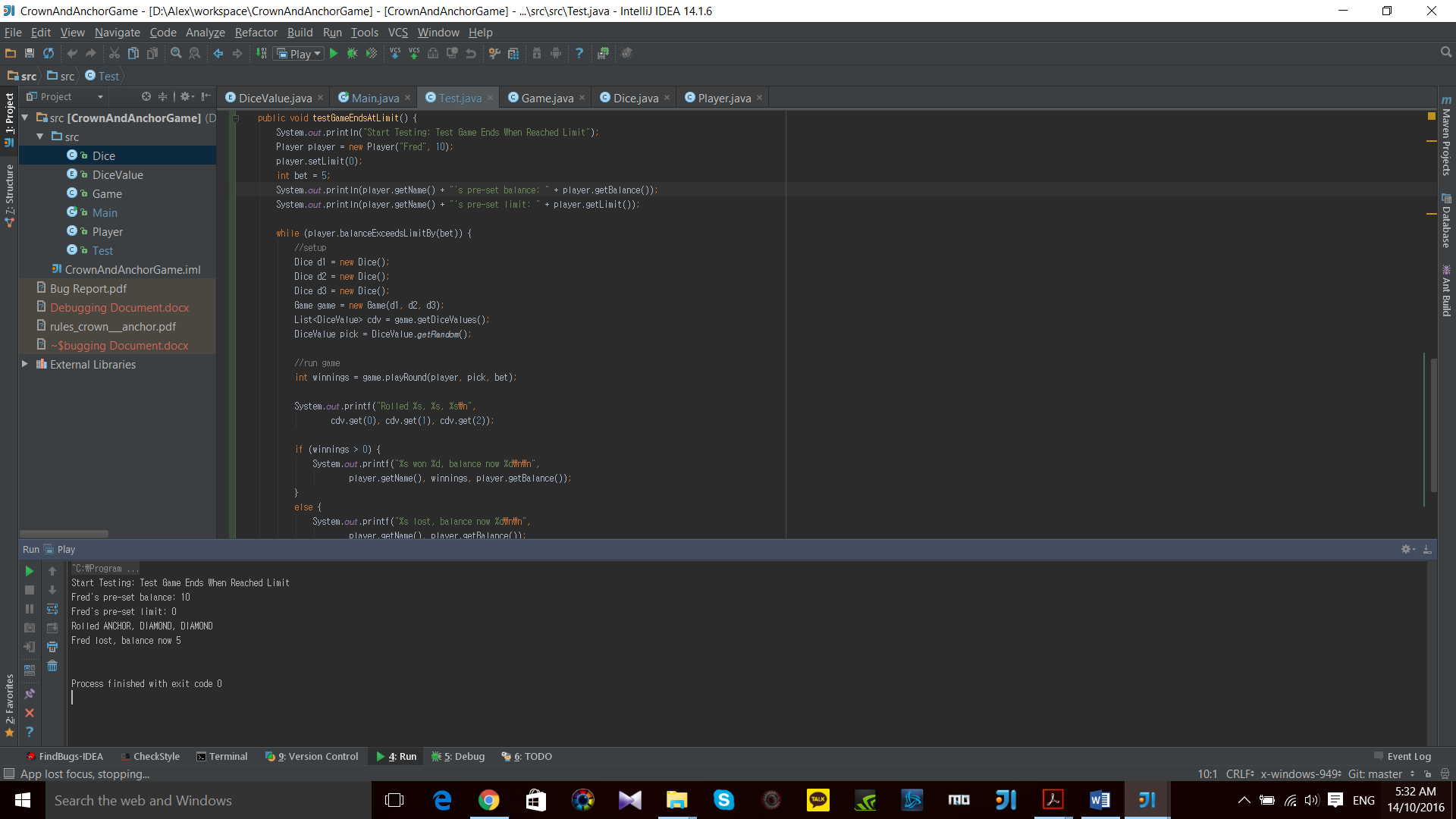
**\* int winnings = matches \* bet; However, player.receiveWinnings(winnings); which excludes money that Player bet.**

## 3. Game Ends When Reached Betting Limit

### Automated Testing Class and method

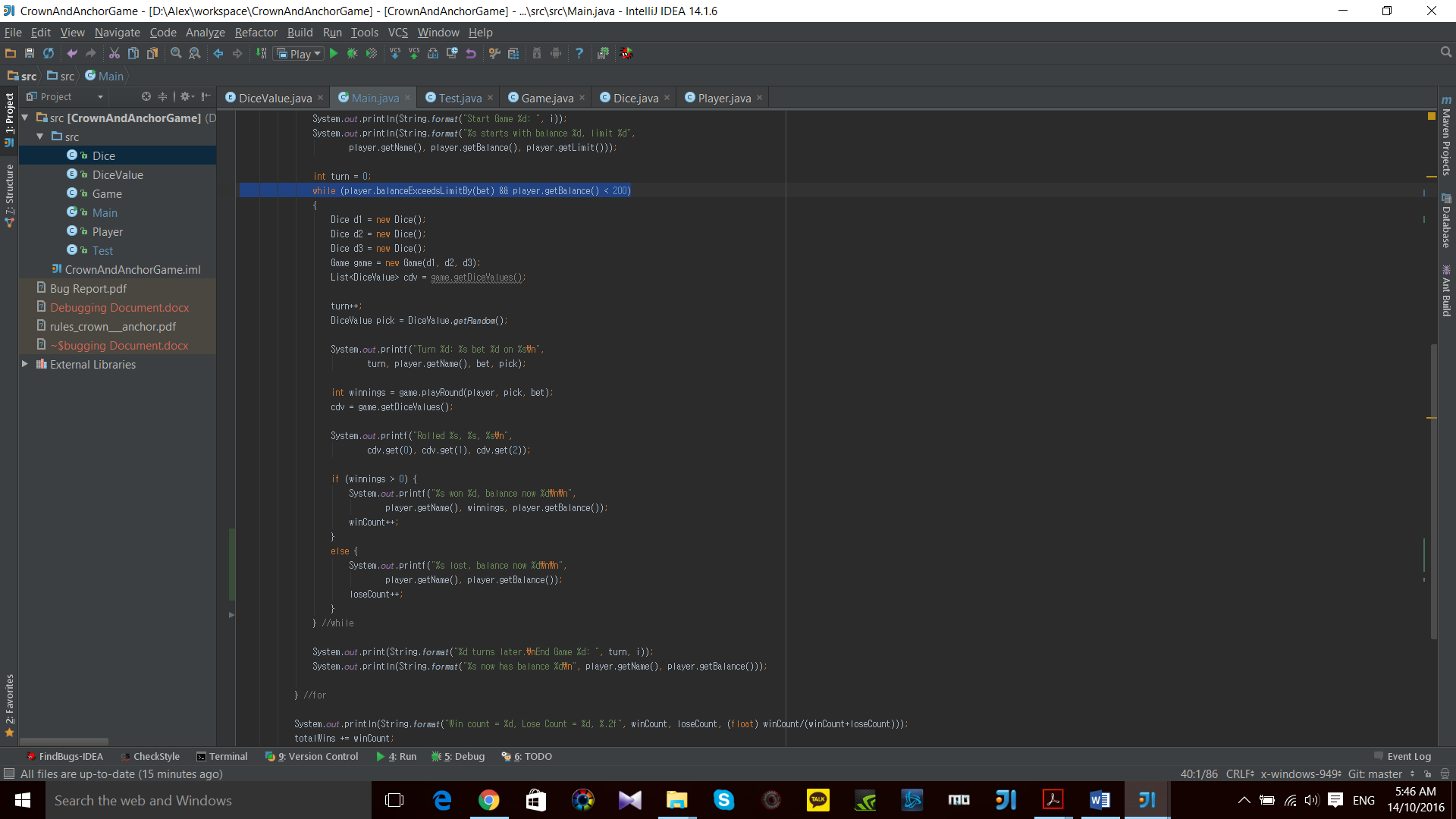


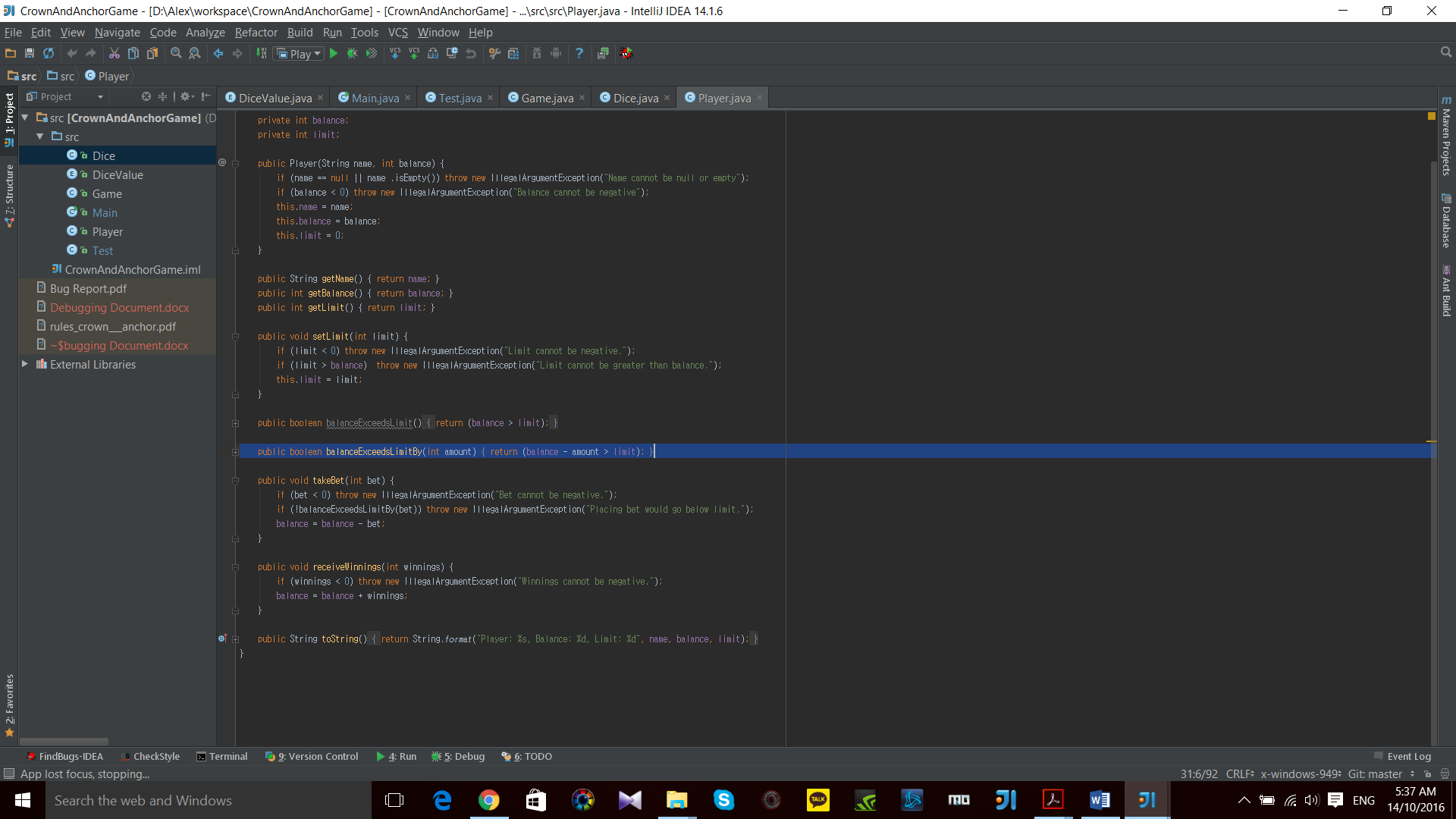
### Testing Result



**\* Game ends when Player still has enough balance for another round.**

### Origin of bug found



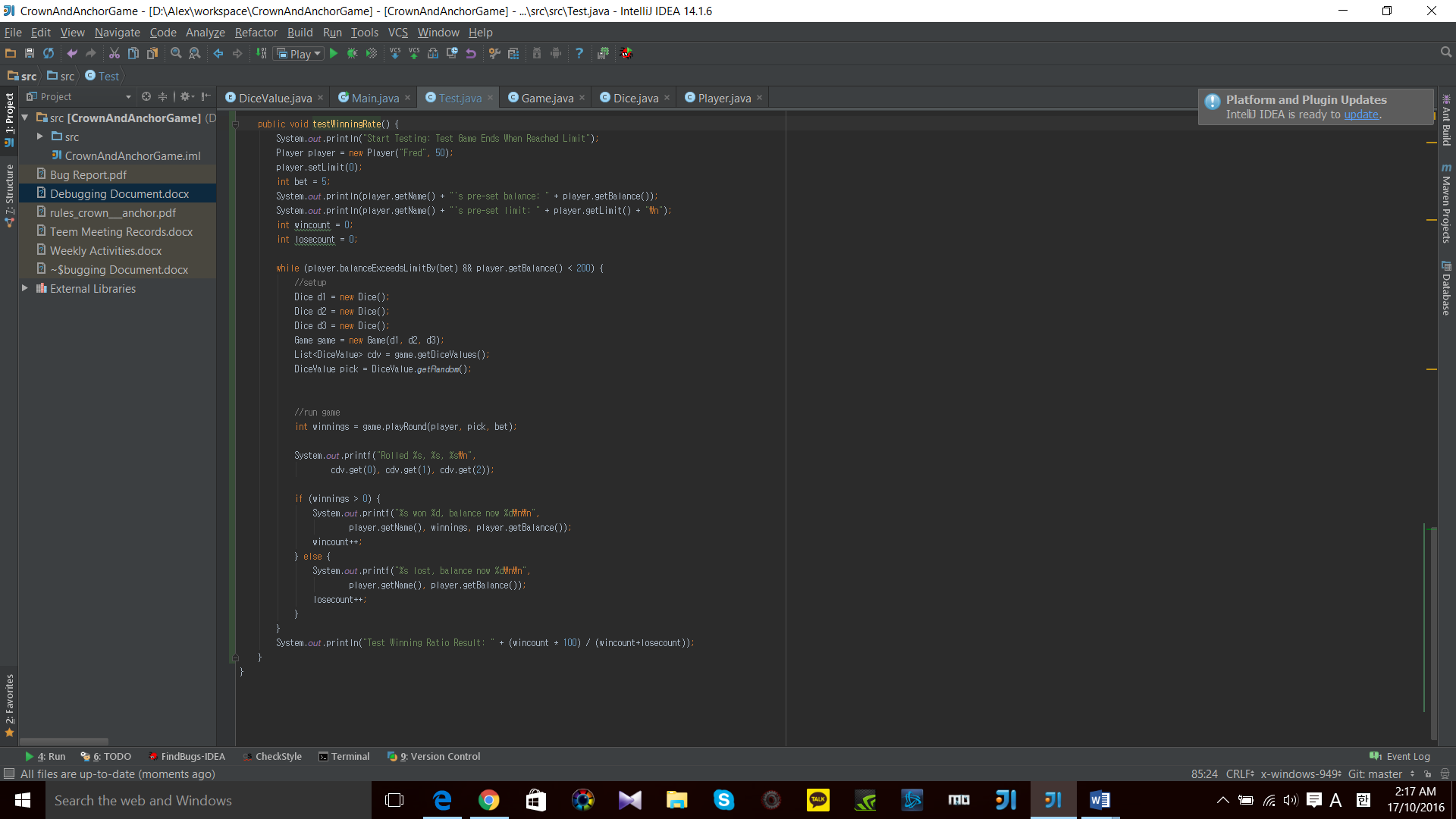


1. **Player can playRound() only while ‘balance – bet > limit’**

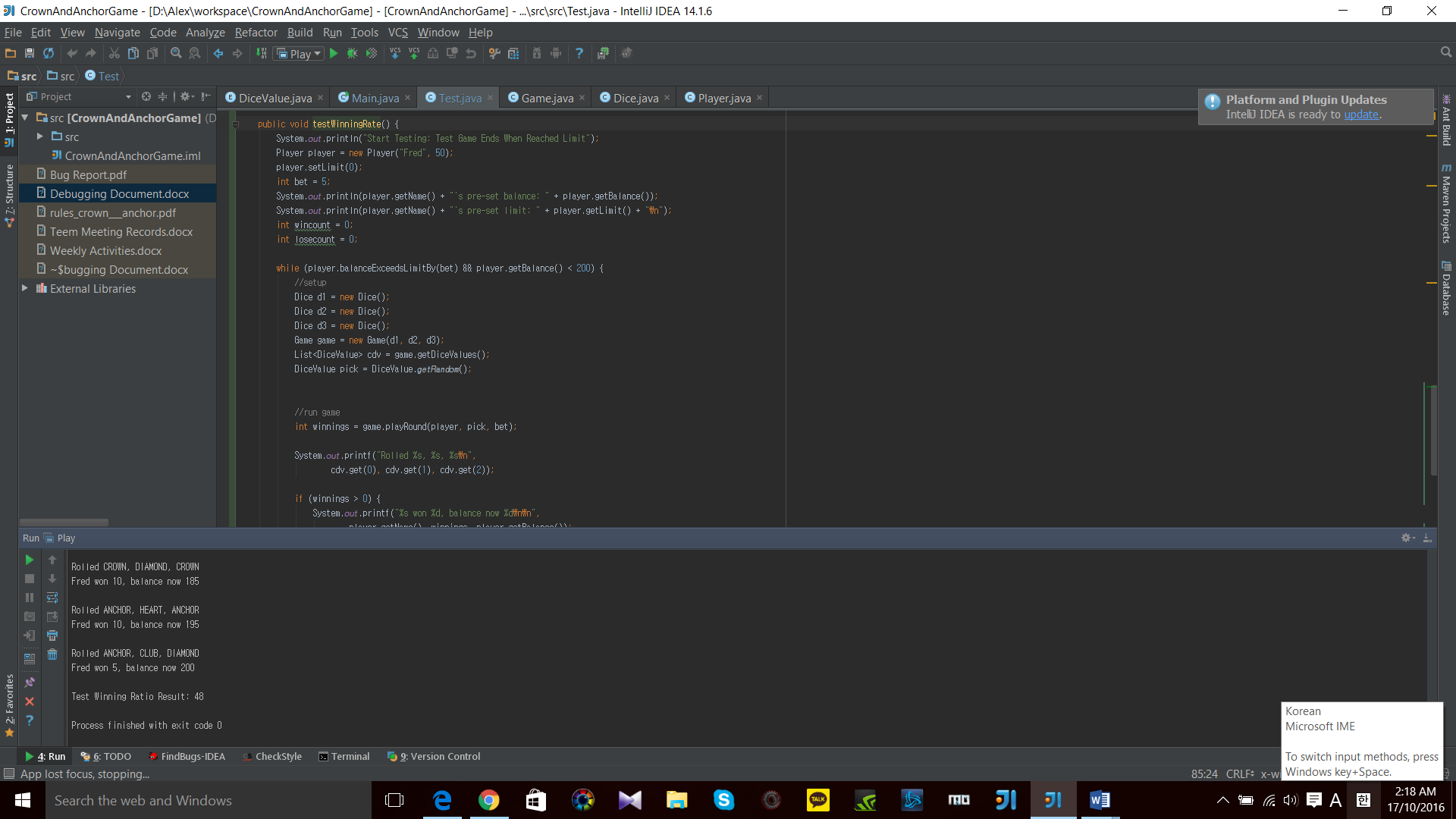
**\* When Player’s ‘balance – bet == limit’, this method will break; while loop**

## 4. Odds in The Game Appears To Be Correct

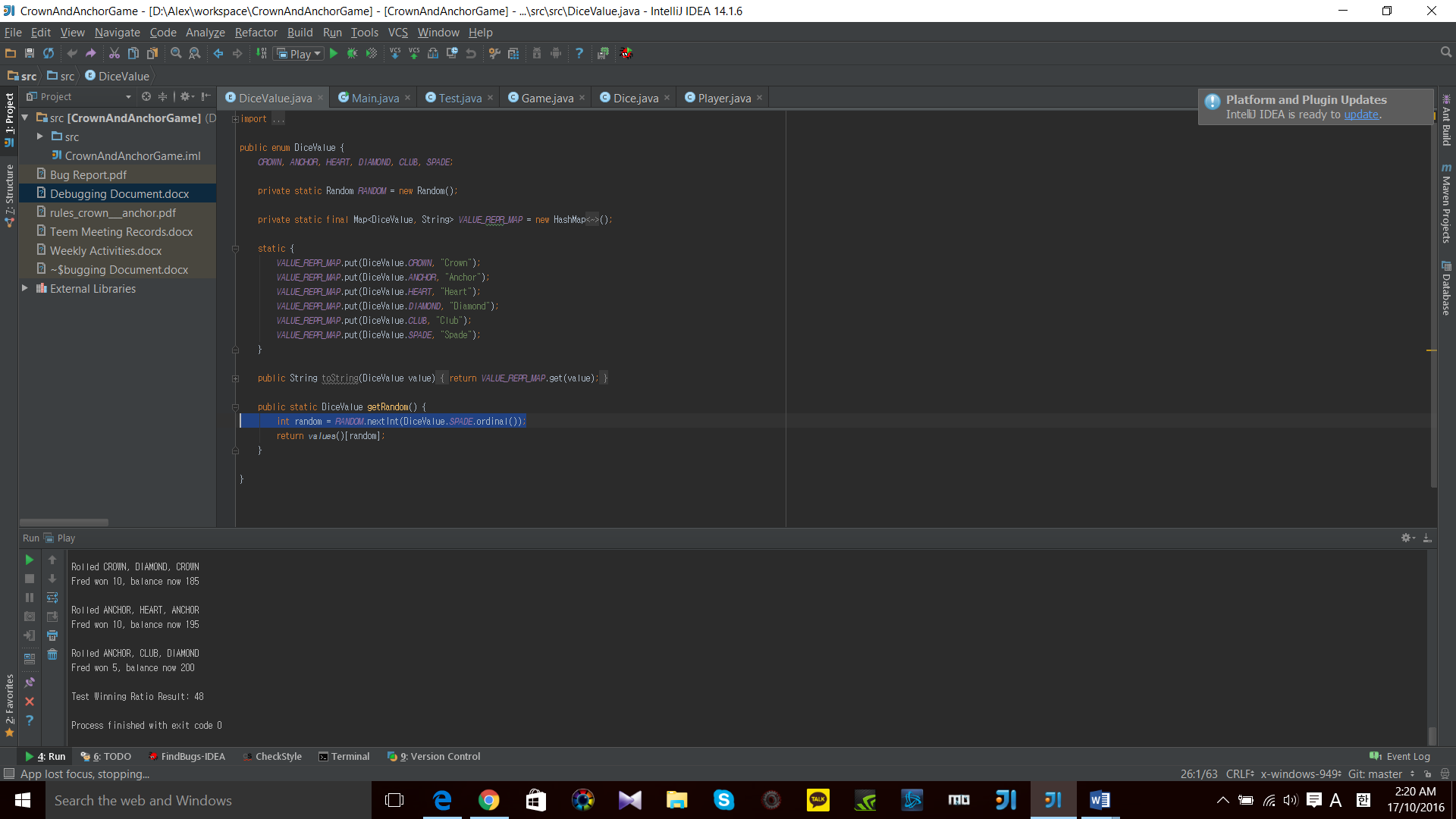
### Automated Testing Class and method



### Testing Result



### Origin of bug found



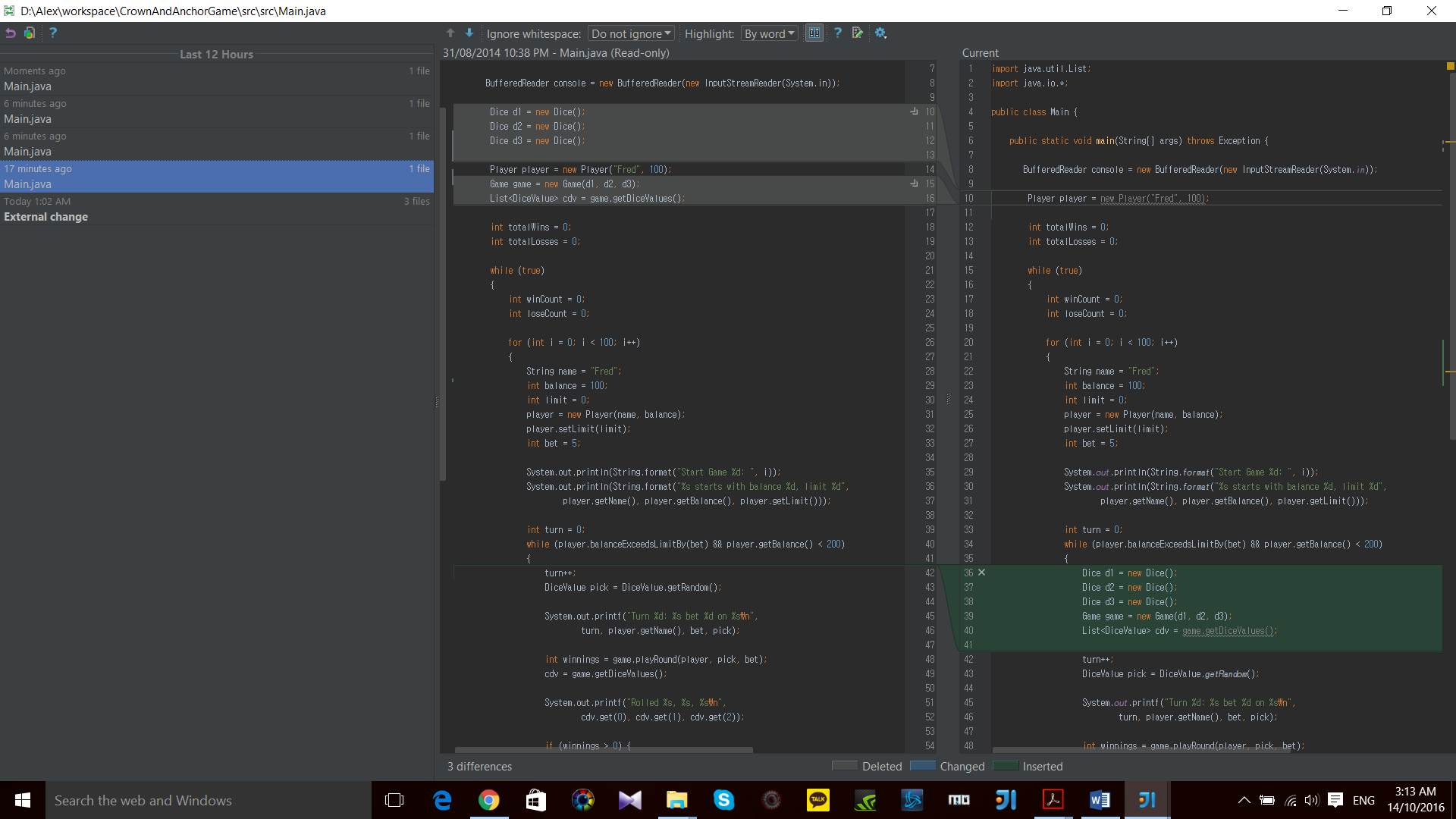
1. **int random do not provide dice value “SPADE”**

**\* RANDOM.nextInt(DiceValue.SPADE.ordinal()) does not include 6th Dice Value (“SPADE”)**

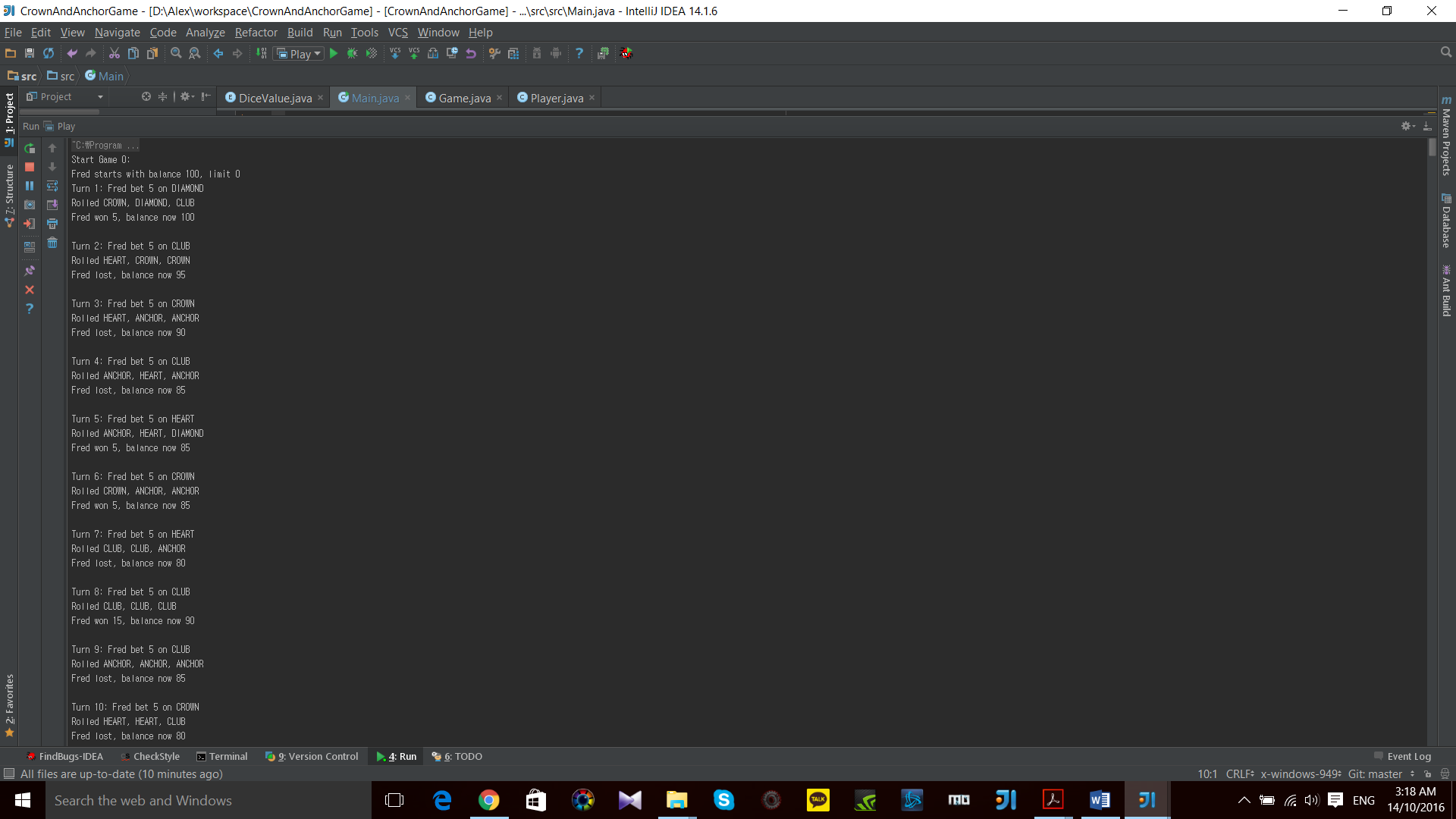
# **Resolution**

## 1. Produce Random Dice Values

### Fixed Code

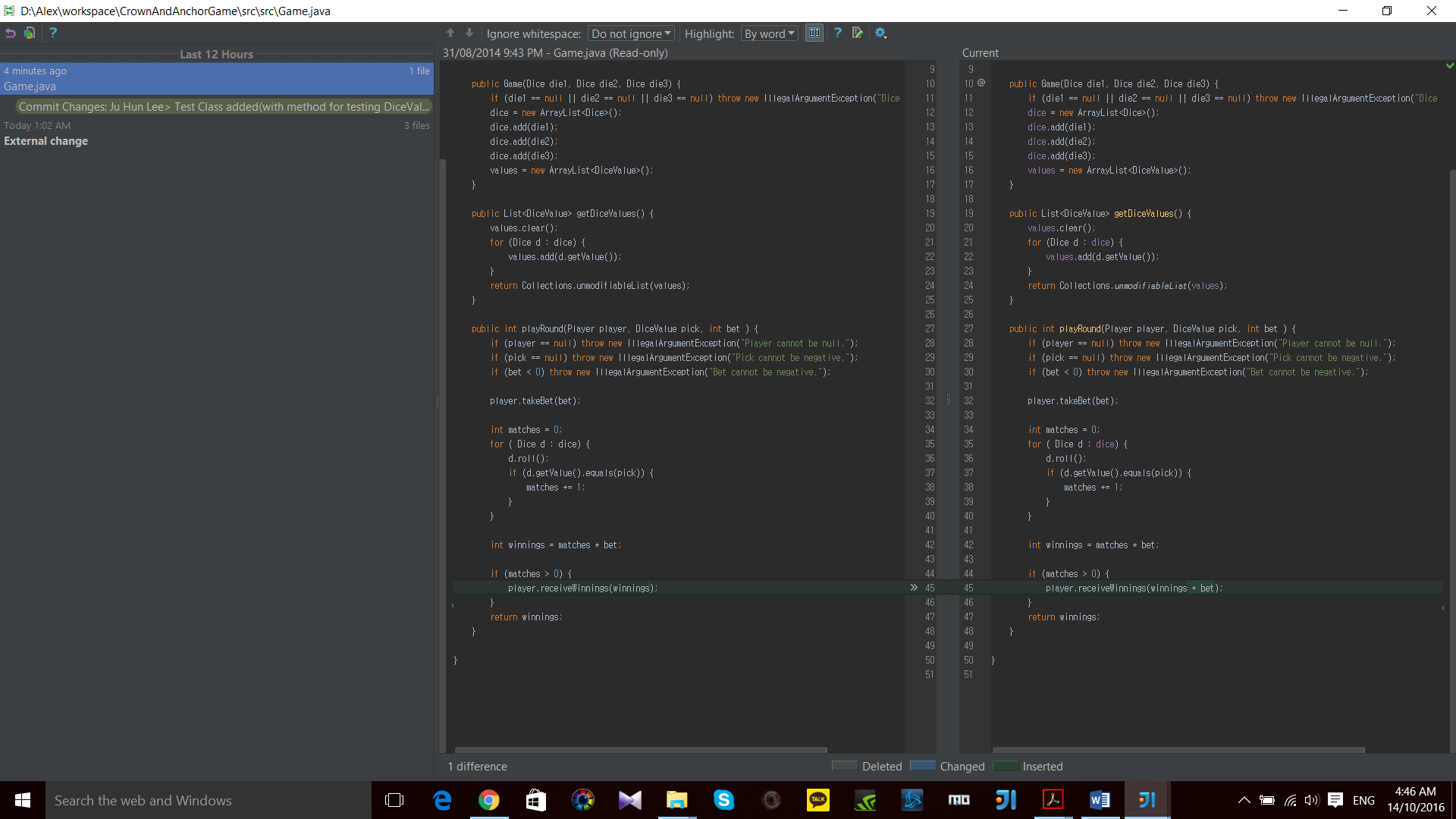


### Fixed Output

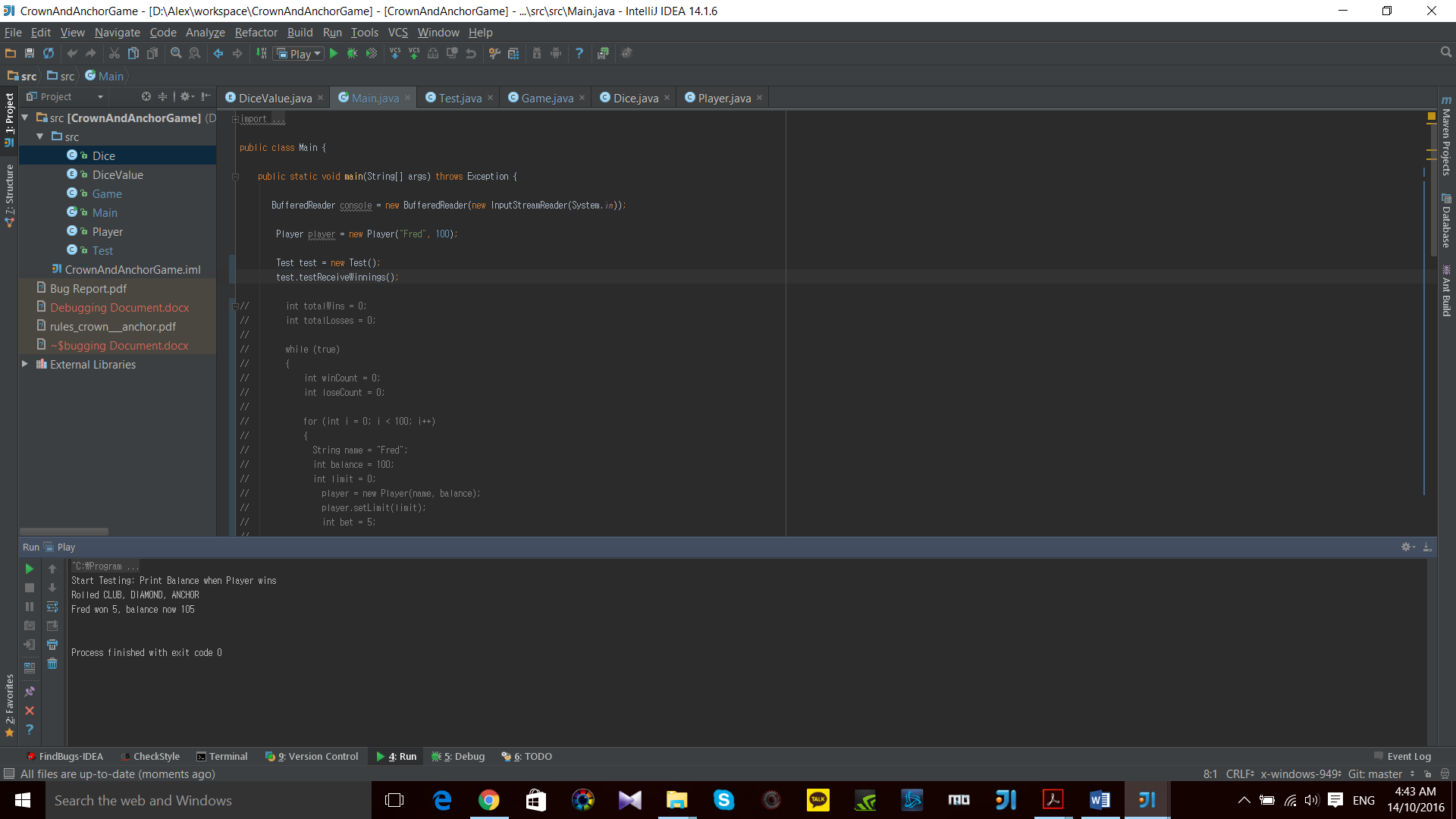


## 2. Game Pay Out at Correct Level

### Fixed Code

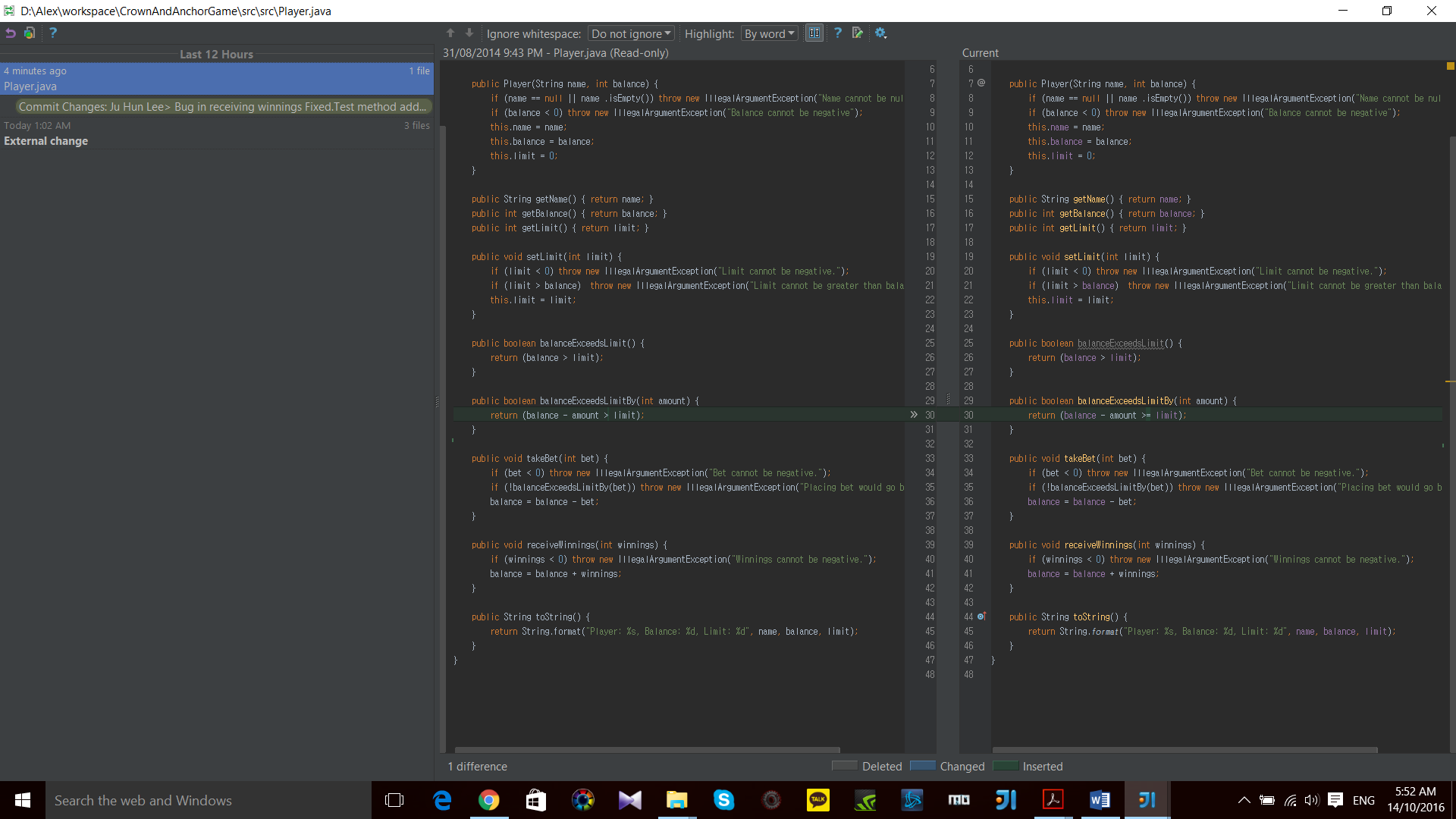


### Fixed Output

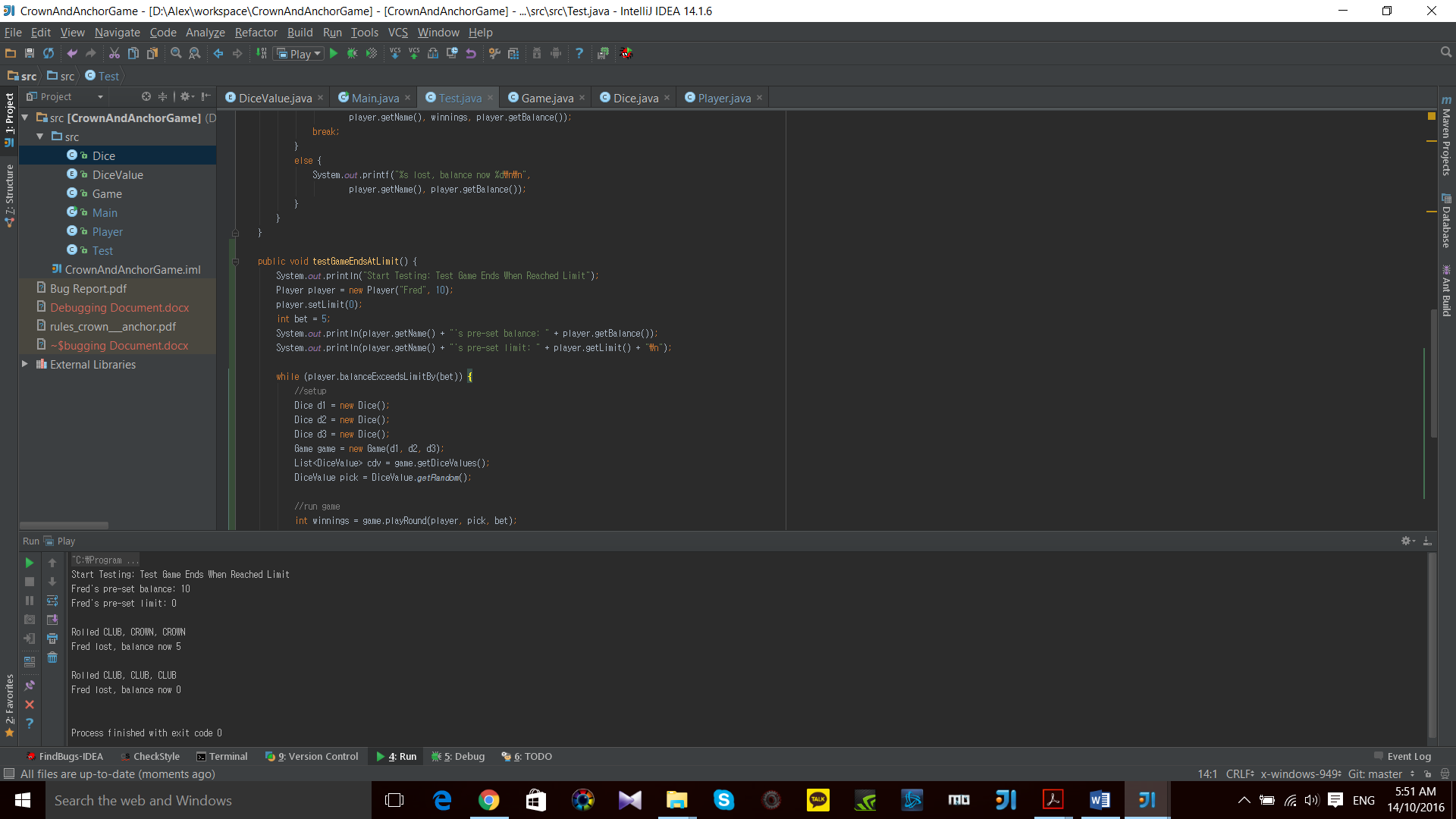


## 3. Game Ends When Reached Betting Limit

### Fixed Code

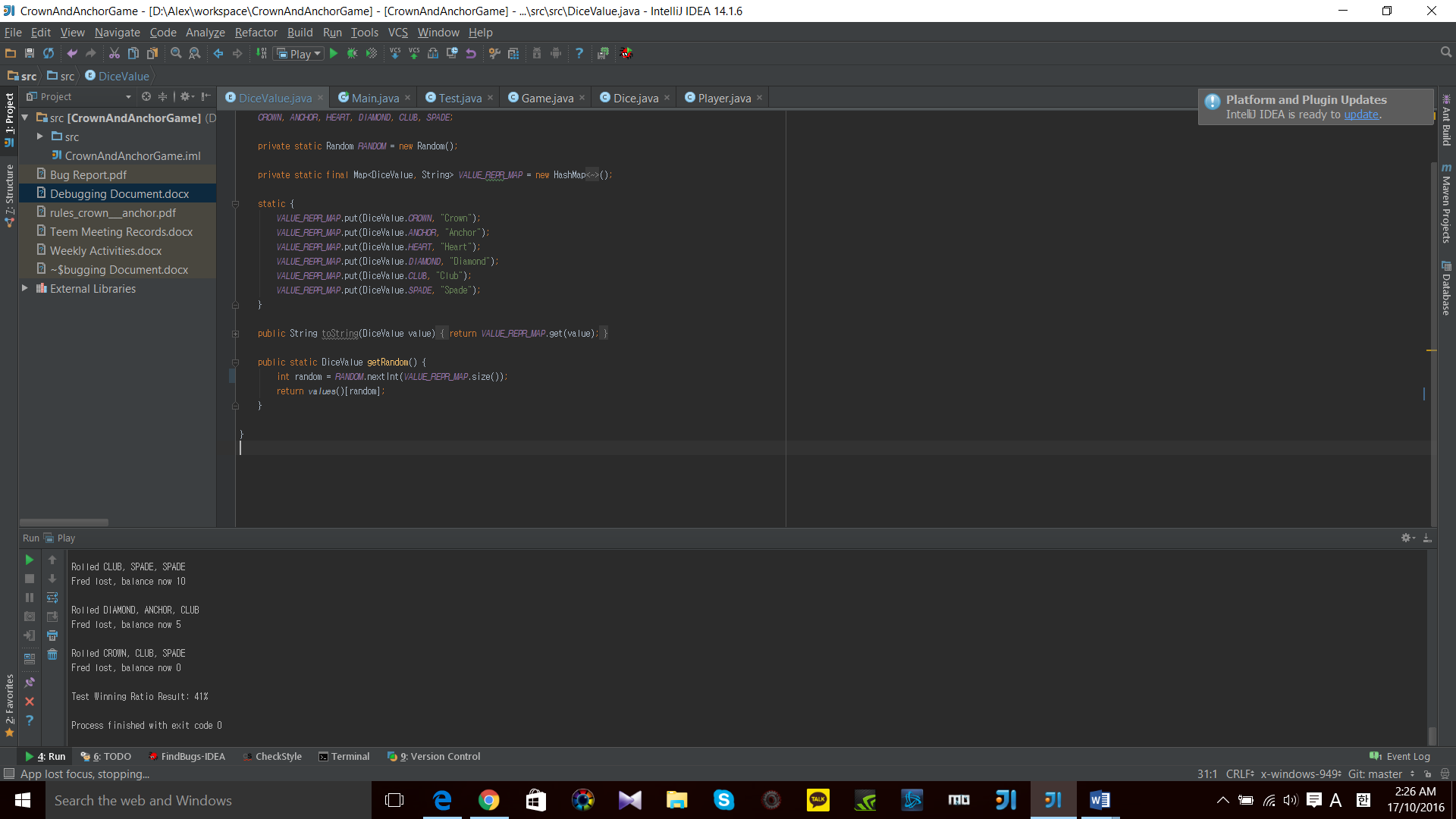


### Fixed Output



## 4. Odds in The Game Appears To Be Correct

### Fixed Code



### Fixed Output

