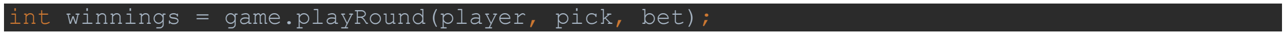
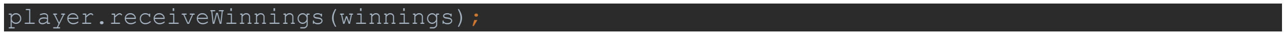
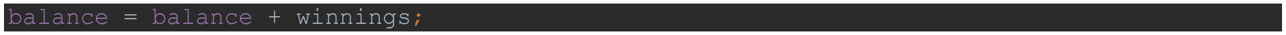
Test Script: TestItem01	
UAT: One Match Win	Date: 04/10/2017

Test Name	Winnings is ‘winnings alone’
Use Case Tested:	One Match Win
Test Description:	A player sits at the table, the player’s bet is taken and the dice thrown on a round by round basis until the player wins with a single matching symbol. The winnings are credited to the player’s account.
Pre-conditions	Player, ‘game’, three six-faced die.
Post-conditions	Player’s account is credited with winnings.

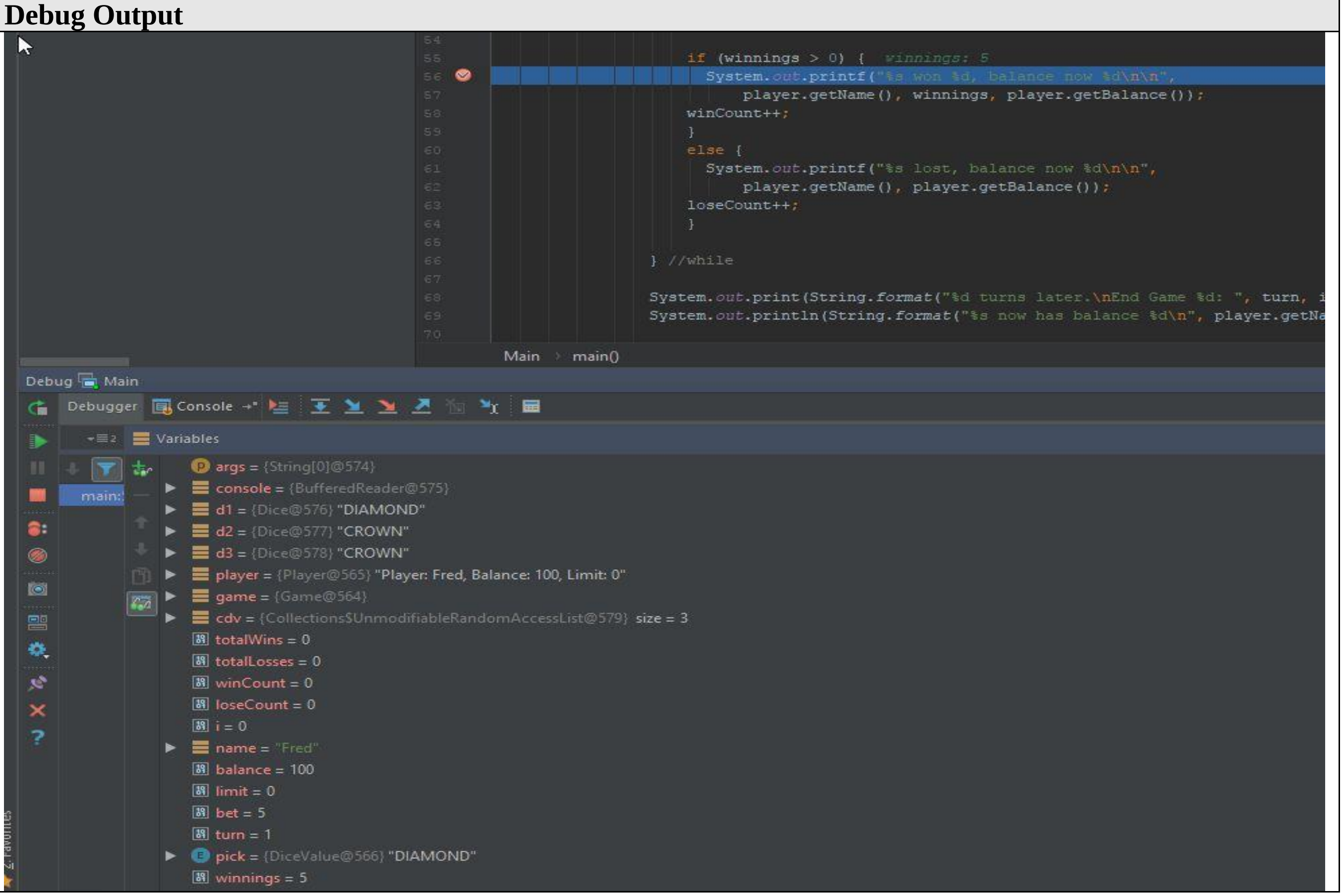
Notes:	**Hypothesis – ‘winnings’ is winnings alone (i.e. ‘stake’ is not added to ‘winnings’ at any point in the program).**
	No direct user input required.
	The flow of execution is observed by running the script once and observing values at following locations:
	- **a breakpoint is set at line 53 of TestItem01.java - StepA**
	
	`int winnings = game.playRound(player, pick, bet);`
	- **a breakpoint is set at line 44 of Game.java in ‘playRound’ -StepB**
	
	`player.receiveWinnings(winnings);`
	- **a conditional breakpoint is set at line 41 (‘winnings==5’) of Player.java. - StepC**
	
	`balance = balance + winnings;`

Result (Pass/Fail/Warning/Incomplete)	**P**

	TEST RUN	EXPECTED TEST RESULTS	P	F
1.	Run *TestItem01:*	Winnings is winnings alone.	P	

Test Script: TestItem01	
UAT: One Match Win	Date: 04/10/2017

Test Run	
Player	"Fred" (limit = 0, balance = 100)

StepA	**Debug Output**
Result	

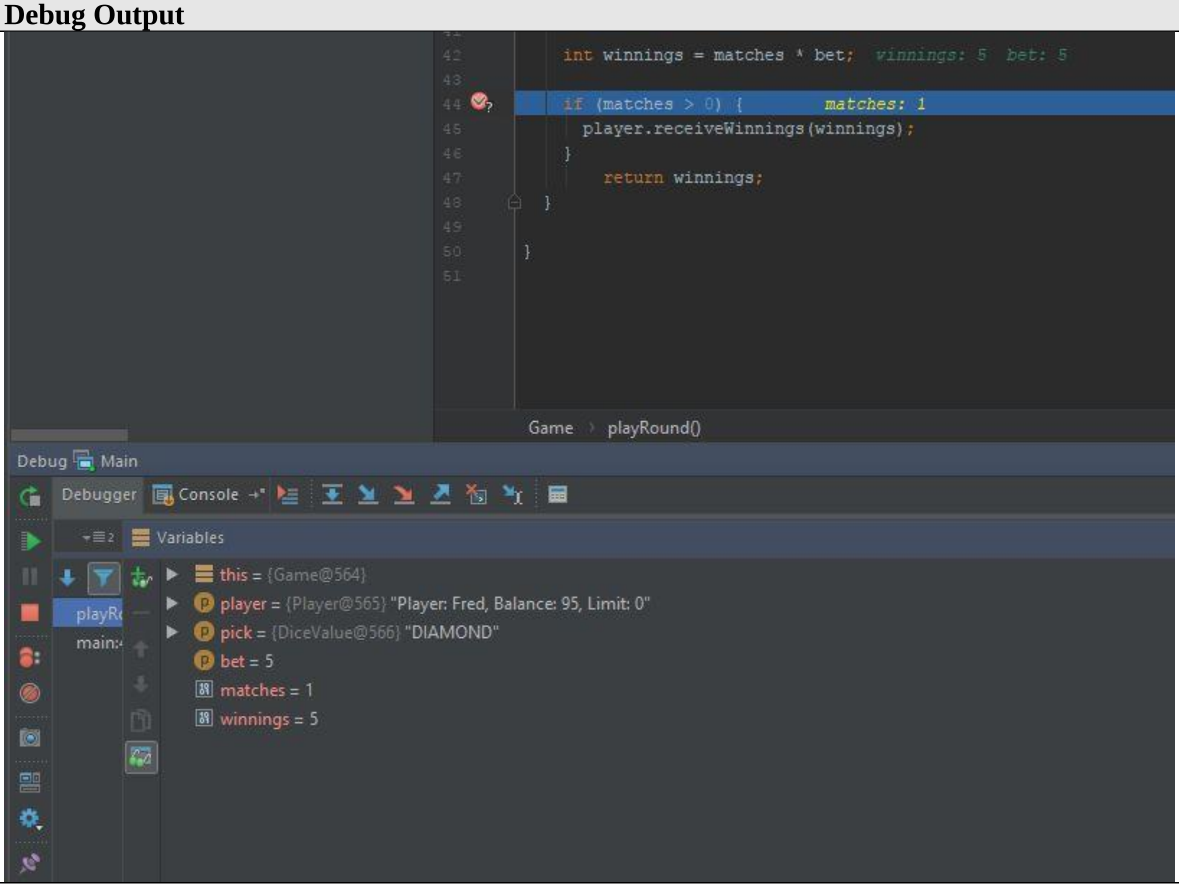
```
            if (winnings > 0) {  winnings: 5
              System.out.printf("%s won %d, balance now %d\n\n",
                  player.getName(), winnings, player.getBalance());
            winCount++;
            }
            else {
              System.out.printf("%s lost, balance now %d\n\n",
                  player.getName(), player.getBalance());
            loseCount++;
            }

        } //while

        System.out.print(String.format("%d turns later.\nEnd Game %d: ", turn, i
        System.out.println(String.format("%s now has balance %d\n", player.getNa
```

```
args = {String[0]@574}
▶ console = {BufferedReader@575}
▶ d1 = {Dice@576} "DIAMOND"
▶ d2 = {Dice@577} "CROWN"
▶ d3 = {Dice@578} "CROWN"
▶ player = {Player@565} "Player: Fred, Balance: 100, Limit: 0"
▶ game = {Game@564}
▶ cdv = {Collections$UnmodifiableRandomAccessList@579} size = 3
totalWins = 0
totalLosses = 0
winCount = 0
loseCount = 0
i = 0
▶ name = "Fred"
balance = 100
limit = 0
bet = 5
turn = 1
▶ pick = {DiceValue@566} "DIAMOND"
winnings = 5
```

Test Script: TestItem01	
UAT: One Match Win	Date: 04/10/2017

StepB	**Debug Output**
Result	

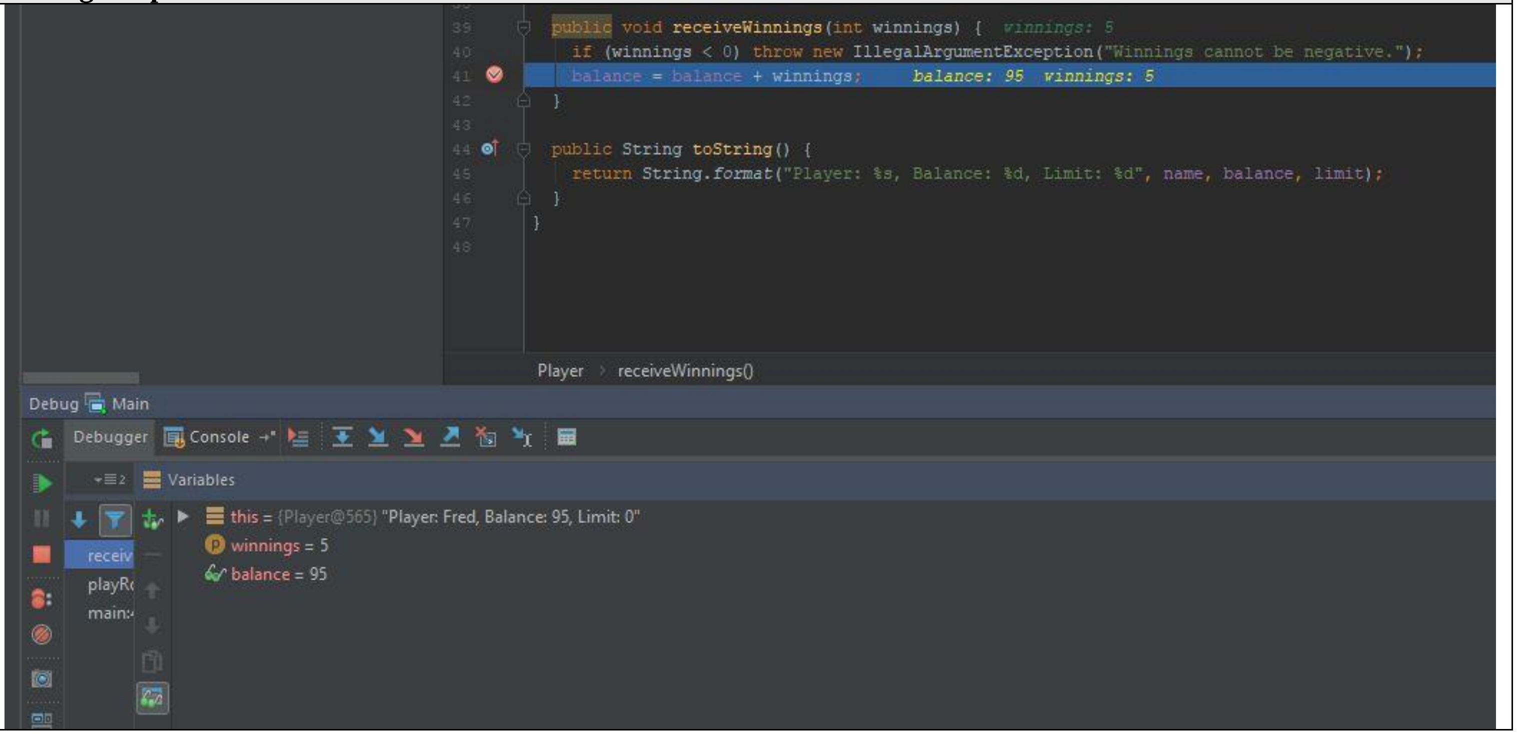
```
        int winnings = matches * bet;  winnings: 5  bet: 5

        if (matches > 0) {          matches: 1
          player.receiveWinnings(winnings);
        }
            return winnings;
    }

}
```

```
this = {Game@564}
player = {Player@565} "Player: Fred, Balance: 95, Limit: 0"
pick = {DiceValue@566} "DIAMOND"
bet = 5
matches = 1
winnings = 5
```

Test Script: TestItem01	
UAT: One Match Win	Date: 04/10/2017

StepC	**Debug Output**
Result	

```
  public void receiveWinnings(int winnings) {  winnings: 5
    if (winnings < 0) throw new IllegalArgumentException("Winnings cannot be negative.");
    balance = balance + winnings;      balance: 95  winnings: 5
  }

  public String toString() {
    return String.format("Player: %s, Balance: %d, Limit: %d", name, balance, limit);
  }
}
```

Player › receiveWinnings()

Debug Main

Debugger Console

Variables

```
▶ this = {Player@565} "Player: Fred, Balance: 95, Limit: 0"
  winnings = 5
  balance = 95
```

receiv
playR
main: