Unit Test Specifications

**Assignment 2**

ITC515 – Professional Programming Practice

Justin McKay - Student ID: 11530610

# Reported Bugs

## BUG001 – Game does not pay out at correct level

When player wins on 1 match, balance does not increase.

## UAT001 – Test\_Game\_Does\_Actually\_Pay\_Out\_At\_Correct\_Level

**Purpose:**   
This test will show that the game does actually pay out at the correct level and that no bug is present.

**Pre-conditions:**Player object is valid  
Player object has at least a balance of the bet being taken

**Post-conditions:***Player.Balance* will reflect the balance of the user prior to taking the bet having matched one dice in a game.

**Data required:**Create a new Player object, with the initial balance of double the bet being taken. For instance, a player object with *minimumBalance* of 10;

### Test Results

|  |  |  |
| --- | --- | --- |
| **Date/Time** | **Tester** | **Status** |
| 16/09/2014 7:13am | Justin McKay | Unit test passed confirming no bug exists. |
|  |  |  |

## BUG002 – Player cannot reach betting limit

Limit set to 0, but game ends with player still with 5 (dollars) remaining.

## UAT003 – Bug\_Test\_Player\_Cannot\_Reach\_Betting\_Limit

**Purpose:**   
This test will replicate the conditions required to reproduce the bug that the player cannot reach the betting limit. *This test will pass while the bug exists.* This test will create the Player object with an initial balance of 5. The test will then call the *Player.balanceExceedsLimitBy()* method to prove that the bug exists as this method will return false when it should return true.

**Pre-conditions:**Player object is valid  
Player object has at least a balance of the bet being taken.

**Post-conditions:***Player.balanceExceedsLimitBy()* method will return false.

**Data required:**Create a new Player object, with the initial balance of the bet being taken. For instance, a player object with *minimumBalance* of 5; Attempt to take a bet of 5.

Note: This test will fail once the bug has been corrected.

### Test Results

|  |  |  |
| --- | --- | --- |
| **Date/Time** | **Tester** | **Status** |
| 10/09/2014 7:32am | Justin McKay | Unit test passed confirming existence of bug. |
| 10/09/2014 7:46am | Justin McKay | Unit test failed confirming bug has been resolved. |

## UAT004 – Resolve\_Test\_Player\_Can\_Now\_Reach\_Betting\_Limit

**Purpose:**   
This test will confirm that the reported bug BUG002 has been resolved. This test will create the Player object with an initial balance of 5. The test will then call the *Player.balanceExceedsLimitBy()* method to prove that the bug has been resolved as this method will now return true.

**Pre-conditions:**Player object is valid  
Player object has at least a balance of the bet being taken.

**Post-conditions:***Player.balanceExceedsLimitBy()* method will return false.

**Data required:**Player object  
name: Test  
initialBalance: 5

### Test Results

|  |  |  |
| --- | --- | --- |
| **Date/Time** | **Tester** | **Status** |
| 10/09/2014 7:46am | Justin McKay | Unit test passed confirming bug has been resolved. |
|  |  |  |

## BUG003 – Odds in game do not appear to be correct

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

## UAT005 – Bug\_Test\_Odds\_In\_Game\_Do\_Not\_Appear\_To\_Be\_Correct

**Purpose:**   
This test will replicate the conditions required to reproduce the bug that the odds in the game do not appear to be correct.

**Pre-conditions:**

**Post-conditions:**

**Data required:**

Note: This test will fail once the bug has been corrected.

### Test Results

|  |  |  |
| --- | --- | --- |
| **Date/Time** | **Tester** | **Status** |
|  |  |  |
|  |  |  |

## UAT006 – Resolve\_Test\_Odds\_In\_Game\_Are\_Correct

**Purpose:**   
This test will confirm that the reported bug BUG003 has been resolved.

**Pre-conditions:**

**Post-conditions:**

**Data required:**

### Test Results

|  |  |  |
| --- | --- | --- |
| **Date/Time** | **Tester** | **Status** |
|  |  |  |
|  |  |  |

## BUG004 – Game does not update current dice values rolled.

The game does not update the values as the dice is rolled for each round. This results in the game always having the same three dice values being used for each round.

## UAT007 – Bug\_Test\_Game\_Does\_Not\_Update\_As\_Die\_Are\_Rolled

**Purpose:**   
This test will replicate the conditions required to reproduce the bug that the values of the die as they are rolled are not updated.

**Pre-conditions:**Player object is valid  
Player object has balance of 100 (enough to cover 20 rounds)  
3 die objects  
Game object is valid

**Post-conditions:**20 Rounds of the game have been played.  
The original die1, die2, die3 object values will not have changed at all during these 20 iterations.

**Data required:**Player Object  
name: Test Player  
initialBalance: 100  
Dice Object 1  
No additional data required  
Dice Object 2   
No additional data required  
Dice Object 3   
No additional data required  
Game Object  
die1: DiceObject1  
die2: DiceObject2  
die3: DiceObject3

Note: This test will fail once the bug has been corrected.

### Test Results

|  |  |  |
| --- | --- | --- |
| **Date/Time** | **Tester** | **Status** |
| 30/09/2014 | Justin McKay | The unit test passed successfully. |
|  |  |  |