**Step 1: Replication**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bug Number 4** | | Simulation turns are not random. | | |
| **Bug Description** | | Crown and Anchor games consist of multiple rounds in which 3 dice are rolled in each round. In the simulation, this is not the case – each game consists of rounds with the same dice values.  I worry some smart player might notice this and clean up! | | |
| **Reproduction Test Description:** | | Run simulation, find instance where player wins on one match and compare balance to previous balance. | | |
| **Pre-conditions** | | N/A | | |
| **Post-conditions** | | N/A | | |
|  | **TEST STEP** | | **EXPECTED TEST RESULTS** | |
|  | Run the simulation (execute Main.main()). | | Simulation starts & prints output of simulation (100 games and stats) to console. |  |
|  | Scroll to end (end of simulation) of output in console. | | Output ends with stats on the overall simulation – with the last line showing the win count, lose count and win+lose ratio. |  |
|  | Scroll up until the start of the last game begins. | |  |  |
|  | Review the values rolled in each turn. | | The values rolled in each turn should be the same. |  |

**Step 2: Simplification**

|  |  |
| --- | --- |
| **Automated Test** | TestBug4 |
| **Test Description** | Given a game of ten rounds, the dice values rolled should not be the same in all rounds. |
| **Test Output** | Test fails – all ten rounds have same values. |

**Step 3: Tracing**

|  |  |
| --- | --- |
| **Debugging Log** | For Bug4: using TestBug4 |

**Debugging Preparation:** Observation of TestBug4

|  |  |
| --- | --- |
| **Null Hypothesis** | The test works! |
| **Prediction** | Given a game of ten rounds, the dice values rolled should not be the same in all rounds. |
| **Observation** | Test fails – all ten rounds have same values. |
| **Conclusion** | The hypothesis is rejected. |

**Hypothesis 1**

|  |  |
| --- | --- |
| **Null Hypothesis** | The invocation of cdv = game.getDiceValues(); produces different dice values each turn |
| **Prediction** | game.getDiceValues(); produces different dice values each turn |
| **Observation** | During each iteration of the loop, game.getDiceValues() produces the same dice value – see screenshots of first two iterations below. |
| **Conclusion** | The hypothesis is rejected. |

**Hypothesis 2**

|  |  |
| --- | --- |
| **Null Hypothesis** | From the previous hypothesis, we know that game.getDiceValues() produces the same values each invocation – looking within the method we see that the method returns the values of the dice  values.add(d.getValue());  Thus, the next null hypothesis is that the dice value produced by the invocation of d.getValue() for each die changes in every iteration. |
| **Prediction** | The invocation of d.getValue() produces a different (random) value for each die in each iteration. |
| **Observation** | In every iteration, the invocation of d.getValue() produces the same value for each die. |
| **Conclusion** | The hypothesis is rejected. |

**Hypothesis 3**

|  |  |
| --- | --- |
| **Null Hypothesis** | From the previous hypothesis, we know that d.getValue() returns the same value (for a given die) in each iteration. Looking at the method, it simply returns the current value of the die, whereas the roll() method generates a random value.    Preceeding the earlier statement that was investigated cdv = game.getDiceValues(); is the statement:  int winnings = game.playRound(player, pick, bet);  In this method both d.roll() and d.getValue() are invoked. However, we know that d.getValue() simply returns the private value variable    Thus, the next null hypothesis is that d.roll() sets the die’s private value to a new, random value |
| **Prediction** | d.roll() sets the die’s private value to a new, random value |
| **Observation** | d.roll() generates a random value – but does NOT set the die’s private value to that new value.  Before d.roll() is invoked, the die value is ‘CROWN’    After d.roll() is invoked, the die value remains ‘CROWN’ |
| **Conclusion** | The hypothesis is rejected.  Looking at the d.roll() method we can see that it simply generates a random number, but does not update the value of the die. |

**Step 4: Resolution**

|  |  |
| --- | --- |
| **Design** | From the debugging step above, we know that d.roll() method should update the value of the die, so that it can be returned by the d.getValue() method used in the game.playRound() method.  To start with at least, we would not change the method signature – so it still returns the value, but now sets the private value of the die first (later we could see if this can become a void method, but that should be conducted separately). |
| **Confirm automated test shows resolution of bug** | BugTest4 now passes after making that changes – see first screenshot below – and trace showing sane values in second screenshot. |
| **Confirm user test shows resolution of bug** | Carrying out the user-reproduction test (running the simulation by executing Main.main()) now shows that the values rolled in each turn are different – and the win ratio is now correct (from Bug3). |