**Step 1: Replication**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bug Number 2** | | Player cannot reach betting limit. | | |
| **Bug Description** | | Limit set to 0, but game ends with player still with 5 (dollars) remaining. | | |
| **Reproduction Test Description:** | | Run simulation, find instance where player wins on one match and compare balance to previous balance. | | |
| **Pre-conditions** | | Use ‘baseline’ version of Main class, or version after resolution of Bug 1 with steps noted below. | | |
| **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 the player’s balance after the last game (and then stats on the overall simulation) |  |
|  | If the last game ends with the player having a balance of 200, rerun the simulation until the game ends with the player having a balance of 5 .  Note: if using the version after resolution of bug 1 this means re-running the simulation until at least 2 dice are rolled with the same value. | | See above |  |
|  | Scroll up until the start of the last game is found | | The output will describe the starting conditions for game number 99 |  |
|  | Compare the limit (declared at the start of the game) to the final balance (at the end) in the console output. | | While the limit is declared as 0, the game ends while the payer’s balance is still 5:  Start CrownAndAnchorGame.Game 99:  Fred starts with balance 100, limit **0**  Turn 1: Fred bet 5 on CROWN  Rolled DIAMOND, ANCHOR, ANCHOR  Fred lost, balance now 95  …  Turn 78: Fred bet 5 on CLUB  Rolled DIAMOND, ANCHOR, ANCHOR  Fred lost, balance now **5**  78 turns later.  End CrownAndAnchorGame.Game 99: Fred now has balance 5  Win count = 9016, Lose Count = 13587, 0.40 |  |

**Step 2: Simplification**

|  |  |
| --- | --- |
| **Automated Test** | TestBug2 |
| **Test Description** | Given a player with a limit of zero, when he plays a round with three dice with a different value to his pick, then the player should end with a balance of zero (equal to his limit). |
| **Test Output** | An exception is thrown: “Placing bet would go below limit.” |

**Step 3: Tracing**

|  |  |
| --- | --- |
| **Debugging Log** | For Bug2: using TestBug2 |

**Debugging Preparation:** Observation of TestBug2

|  |  |
| --- | --- |
| **Null Hypothesis** | The test works! |
| **Prediction** | At the conclusion of a losing game, the player’s final balance should be equal to the player’s limit. |
| **Observation** | The game ended before the player’s final balance reached the limit (see screenshot above from TestBug2 output). |
| **Conclusion** | The hypothesis is rejected. |

**Hypothesis 1**

|  |  |
| --- | --- |
| **Null Hypothesis** | The limit setter in line 25 does not actually set the player’s limit to zero  24: int limit = 0;  25: player.setLimit(limit); |
| **Prediction** | player.limit will not be set to zero after execution of line 25 |
| **Observation** | player.limit is in fact set to 0 (the specified limit)  See screenshots below:  Before origin limit variable (line 24) is sane (== 0)  After origin player’s limit (line 25) is also sane (== 0) |
| **Conclusion** | The hypothesis is rejected. |

**Hypothesis 2**

|  |  |
| --- | --- |
| **Null Hypothesis** | When game.playRound() is invoked (line 34) it is passed incorrect parameters |
| **Prediction** | game.playRound() will not be called with the correct parameters (player, pick, bet) |
| **Observation** | game.playRound() is in fact called with the correct parameters (player, pick, bet)  See screenshots below:  Before invocation, parameters are sane and correct (first screenshot)  During invocation, parameters remain sane (second and third screenshots) |
| **Conclusion** | The hypothesis is rejected. |

**Hypothesis 3**

|  |  |
| --- | --- |
| **Null Hypothesis** | During invocation of game.playRound(),player.takeBet(bet) is called with invalid / incorrect parameter |
| **Prediction** | player.takeBet()will be called with an incorrect argument |
| **Observation** | player.takeBet()is in fact called with the correct parameters (bet is correct value)  See screenshots below:  Before invocation, parameter is sane and correct (first screenshot)    However, within player.takeBet()there is a call to player.balanceExceedsLimitBy()that causes the exception to be thrown |
| **Conclusion** | The hypothesis is rejected – but we have identified the ultimate source of the bug. |

**Hypothesis 4**

|  |  |
| --- | --- |
| **Null Hypothesis** | Invocation of player.balanceExceedsLimitBy() with argument equal to the value of bet should return true. |
| **Prediction** | player.takeBet(bet)will return true |
| **Observation** | player.takeBet()is in fact called with the correct parameters (amount is the correct value, equal to 5 – the value of bet and balance is also 5)  See screenshots below:  Before invocation, parameter is sane and correct (first screenshot) as is the balance (both equal to 5).    However, invocation of player.balanceExceedsLimitBy()with argument of 5 (equal to both bet and balance) returns false – not sane – see screenshot below. |
| **Conclusion** | The hypothesis is rejected – and player.balanceExceedsLimitBy() should return true if balance exceeds or equals the player’s limit after the bet is taken i.e. the name of the method is accurate: the balance should equal (or be greater than) the limit plus the argument. |

**Step 4: Resolution**

|  |  |
| --- | --- |
| **Design** | From the debugging step above, we know that player.balanceExceedsLimitBy() should return true if the balance is equal to (or greater than) the limit plus the argument.  Thus the change (to a common ‘off-by-one type error’) is simply to test for ‘greater-than**-or-equal-to’.** |
| **Confirm automated test shows resolution of bug** | BugTest2 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 when the player loses a game, his balance equals zero. |