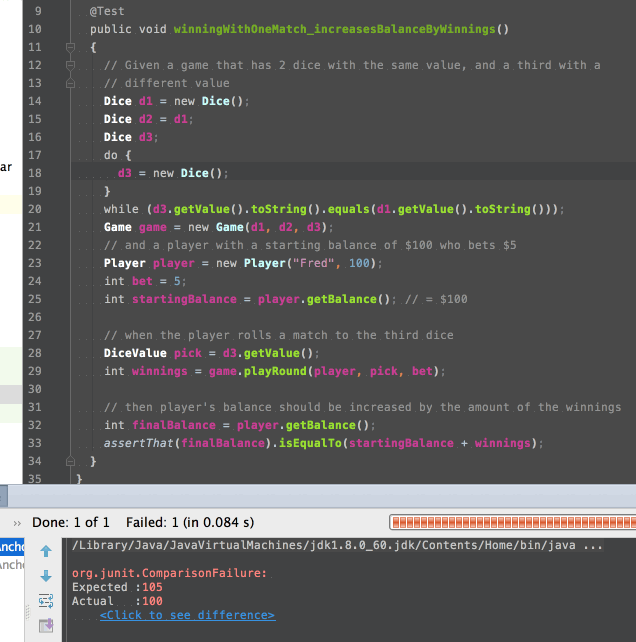
**Step 1: Replication**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bug Number 1** | | Game does not pay out at correct level. | | |
| **Bug Description** | | When player wins on 1 match, balance does not increase. | | |
| **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. | | |
| **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 top (start of simulation) of output in console. | | Output starts with:  Start CrownAndAnchorGame.Game 0:  Fred starts with balance 100, limit 0  … |  |
|  | Read down through the output until you reach the first (or any) instance where Fred wins with one match | | Identify instance by finding a case where the suite bet on is rolled ONLY once (but in no particular order):  Turn 3: Fred bet 5 on **ANCHOR**  Rolled DIAMOND, **ANCHOR**, HEART |  |
|  | Compare the balance in the previous turn to the balance in the turn where Fred wins with one match \*  \* if the first instance is the first round of the game, you need to compare the balance to the balance in the preceding statement ‘Fred starts with balance 100, limit 0.’ | | The balances match (although the new balance should be greater due to the win) e.g.:  Turn 2: Fred bet 5 on ANCHOR  Rolled CROWN, HEART, HEART  Fred lost, balance now **90**  Turn 3: Fred bet 5 on CROWN  Rolled CROWN, HEART, HEART  Fred won 5, balance now **90** |  |

**Step 2: Simplification**

|  |  |
| --- | --- |
| **Automated Test** | TestBug1 |
| **Bug Description** | Given game with 1 die that matches player’s roll, final balance should equal starting balance + winnings |
| **Test Output** | Final balance remains as at starting balance (see failure below) and does not increase by winnings amount |



**Step 3: Tracing**

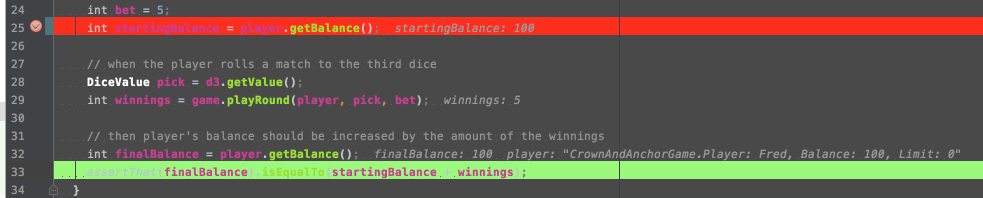
|  |  |
| --- | --- |
| **Debugging Log** | For Bug1: using TestBug1 |

**Debugging Preparation:** Observation of TestBug1

|  |  |
| --- | --- |
| **Null Hypothesis** | The test works! |
| **Prediction** | The player’s balance (after a winning round with one match) should increase by the winning amount. |
| **Observation** | The player’s balance did not increase (see screenshot above from TestBug1 output). |
| **Conclusion** | The hypothesis is rejected. |

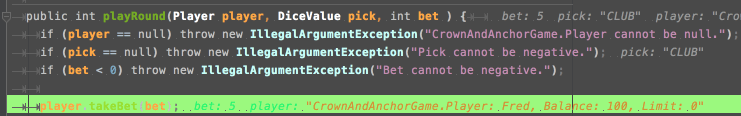
**Hypothesis 1**

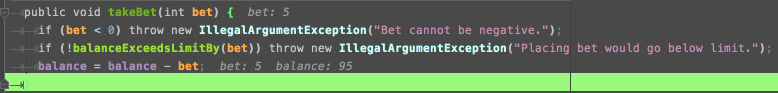
|  |  |
| --- | --- |
| **Null Hypothesis** | The action (line 29) in the test [ int winnings = game.playRound(player, pick, bet); ] does not return the correct winning amount |
| **Prediction** | winnings will equate to zero after execution of line 29 |
| **Observation** | winnings equates to 5 (the correct amount)  See screenshot below:  Before origin (action at line 29) starting balance (line 25) is sane (== 100)  After origin final balance (line 32) is not sane (remains == 100) although winnings is sane (== 5) |
| **Conclusion** | The hypothesis is rejected. |



**Hypothesis 2**

|  |  |
| --- | --- |
| **Null Hypothesis** | The invocation of the action game.playRound() at line 29 deducts an incorrect amount from the player’s balance (more than the actual bet). |
| **Prediction** | player.takeBet(); in game.playRound()will result in balance equating to 90. |
| **Observation** | Player’s balance is correct (== 95) after invocation of player.takeBet()  First screenshot below shows player’s starting balance is sane (== 100)  Second screenshot below shows player’s balance is also sane (== 95) after the bet has been taken |
| **Conclusion** | Hypothesis is rejected |





**Hypothesis 3**

|  |  |
| --- | --- |
| **Null Hypothesis** | The invocation of the action game.playRound() at line 29 returns the initial bet (that was deducted in the step above) along with the winnings. |
| **Prediction** | player.recieveWinnings(); in game.playRound()will result in balance equating to 105. |
| **Observation** | Player’s balance is incorrect (== 100) after invocation of player.recieveWinnings()  First screenshot below shows player’s balance is sane (== 95) after the bet has been taken AND that bet == 5, winnings == 5  Second screenshot shows player’s balance is not sane (== 100) after the winnings have been returned  Third screenshot simply shows there are no more changes to player’s balance after this invocation. |
| **Conclusion** | Hypothesis is rejected and that player.recieveWinnings() must return the initial bet in addition to the winnings (see line 44). |



