Ashwin Jeyaseelan

Test Plan

Description:

The very first test I conducted was if the time counter was advancing properly and the lottery was occurring at the correct times. After I got the correct sequence of steps in the whole lottery working, I tested the progress of the lottery whenever it needed to be repeated. I purposely had the draw method return a negative method to see if the code reached the correct if statement and then looped back to a new event. To make the sequence simpler, I put the rest of the lottery steps in the else statement as long as the lottery number was not negative. Therefore the lottery only progressed with the awards as long as the lottery number was shifted to -1. Lastly, I had to test the exceptions. The match not found exception worked easily, since I created a lottery array (with existing lottery numbers) to be shared by the lottery super class. The repeat number exception was tested by retrieving the counter to count the exceptions. Finally, the output file was tested for by checking if it was generated successfully.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Number | Test Input Condition | Functionality Checked | Expected Output | Pass or Fail |
| 1 | output time counter after lottery drawn | check if lottery occurs when time counter reaches time | Pass | Pass |
| 2 | change output of lottery number | sequence of events | Pass | Pass |
| 3 | Are print statements at beginning happening again | Check if outermost loop working | Pass | Pass |
| 4 | Add or subtract numbers in lottery array to see if exceptions caught | MatchNotFoundException working | Pass | Pass |
| 5 | output messages of exceptions | exceptions working | Pass | Pass |
| 6 | File generated? | Output file generated | Pass | Pass |