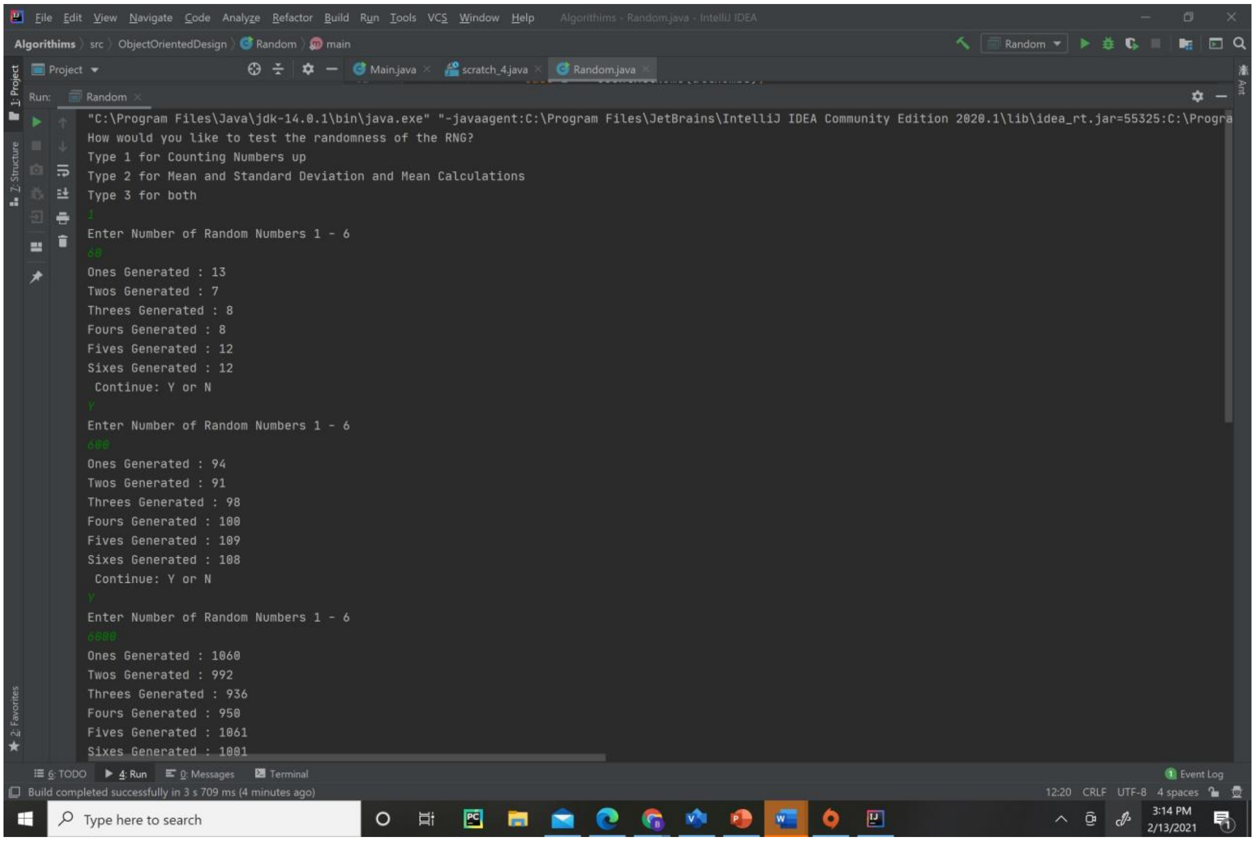
Ben Sottile

February 13, 2021

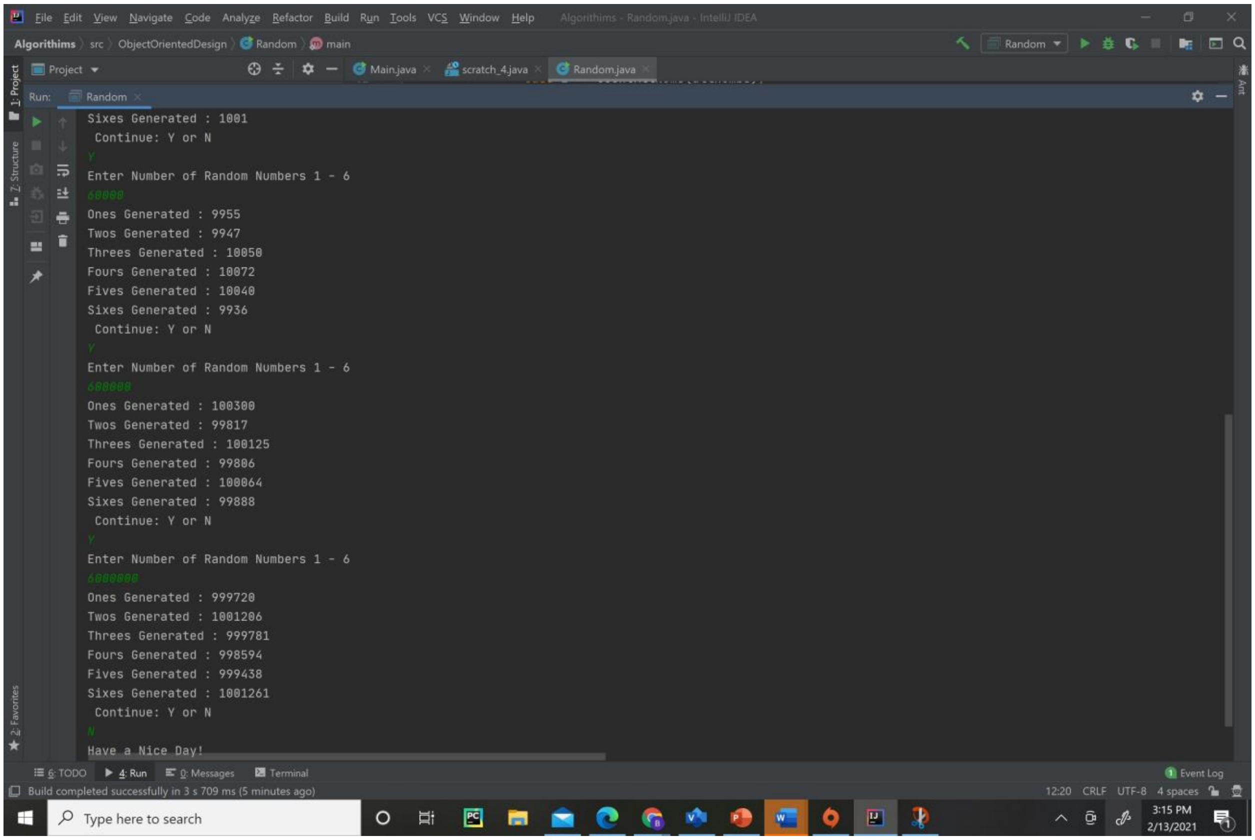
Assignment 2: CS 310 – Random Number Generator

Screenshot(s) of Counting Test Results



File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Algorithims - Random.java - IntelliJ IDEA
Algorithims 〉 src 〉 ObjectOrientedDesign 〉 Random 〉 main
Project
Main.java
scratch_4.java
Random.java
Run: Random
6: TODO
4: Run
0: Messages
Terminal
Event Log
Build completed successfully in 3 s 709 ms (4 minutes ago)
12:20 CRLF UTF-8 4 spaces
Type here to search
3:14 PM
2/13/2021

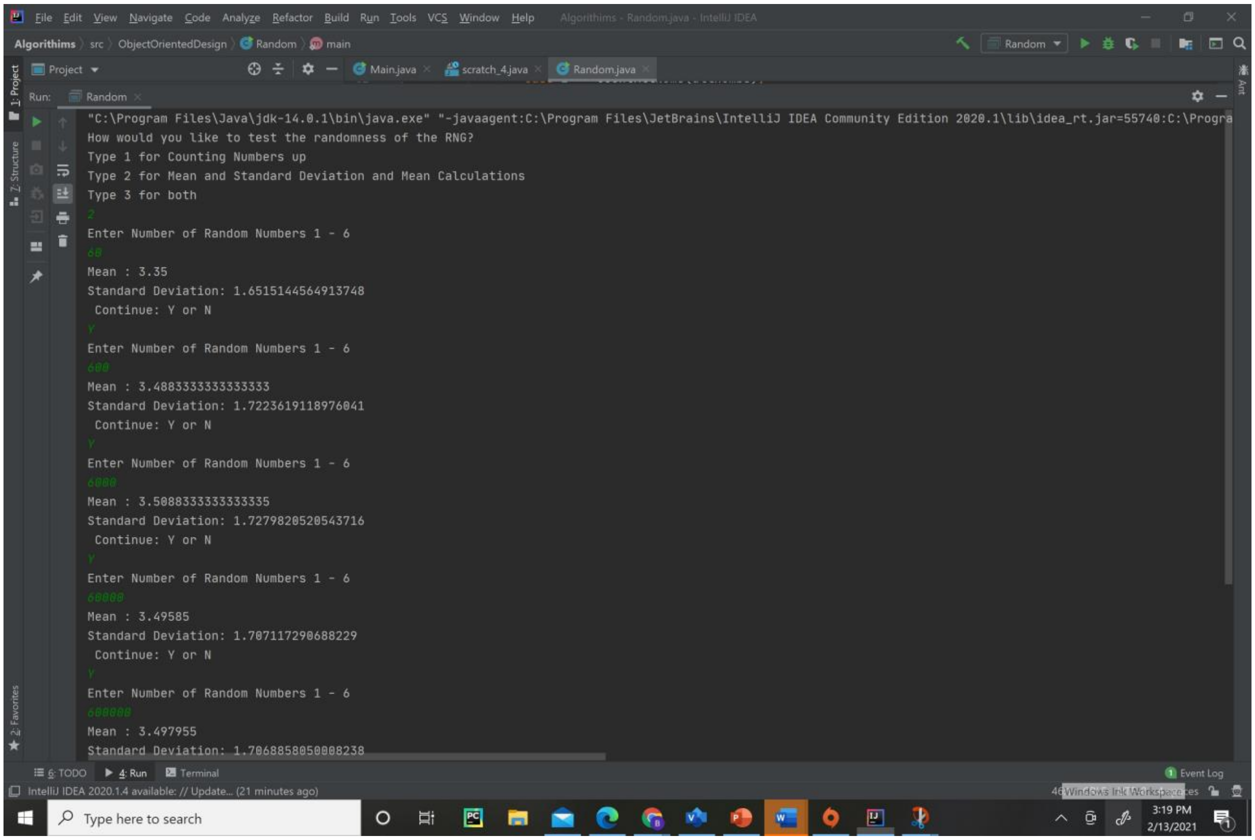
```
"C:\Program Files\Java\jdk-14.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2020.1\lib\idea_rt.jar=55325:C:\Progra
How would you like to test the randomness of the RNG?
Type 1 for Counting Numbers up
Type 2 for Mean and Standard Deviation and Mean Calculations
Type 3 for both
1
Enter Number of Random Numbers 1 - 6
60
Ones Generated : 13
Twos Generated : 7
Threes Generated : 8
Fours Generated : 8
Fives Generated : 12
Sixes Generated : 12
 Continue: Y or N
Y
Enter Number of Random Numbers 1 - 6
600
Ones Generated : 94
Twos Generated : 91
Threes Generated : 98
Fours Generated : 100
Fives Generated : 109
Sixes Generated : 108
 Continue: Y or N
Y
Enter Number of Random Numbers 1 - 6
6000
Ones Generated : 1060
Twos Generated : 992
Threes Generated : 936
Fours Generated : 950
Fives Generated : 1061
Sixes Generated : 1001
```


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Algorithims - Random.java - IntelliJ IDEA
Algorithims 〉 src 〉 ObjectOrientedDesign 〉 Random 〉 main
Random
Project
Main.java
scratch_4.java
Random.java
Run: Random
1: Project
7: Structure
2: Favorites
Ant
6: TODO
4: Run
0: Messages
Terminal
Event Log
12:20 CRLF UTF-8 4 spaces
Type here to search
3:15 PM
2/13/2021

```
Sixes Generated : 1001
 Continue: Y or N
Y
Enter Number of Random Numbers 1 - 6
60000
Ones Generated : 9955
Twos Generated : 9947
Threes Generated : 10050
Fours Generated : 10072
Fives Generated : 10040
Sixes Generated : 9936
 Continue: Y or N
Y
Enter Number of Random Numbers 1 - 6
600000
Ones Generated : 100300
Twos Generated : 99817
Threes Generated : 100125
Fours Generated : 99806
Fives Generated : 100064
Sixes Generated : 99888
 Continue: Y or N
Y
Enter Number of Random Numbers 1 - 6
6000000
Ones Generated : 999720
Twos Generated : 1001206
Threes Generated : 999781
Fours Generated : 998594
Fives Generated : 999438
Sixes Generated : 1001261
 Continue: Y or N
N
Have a Nice Day!
```

Build completed successfully in 3 s 709 ms (5 minutes ago)

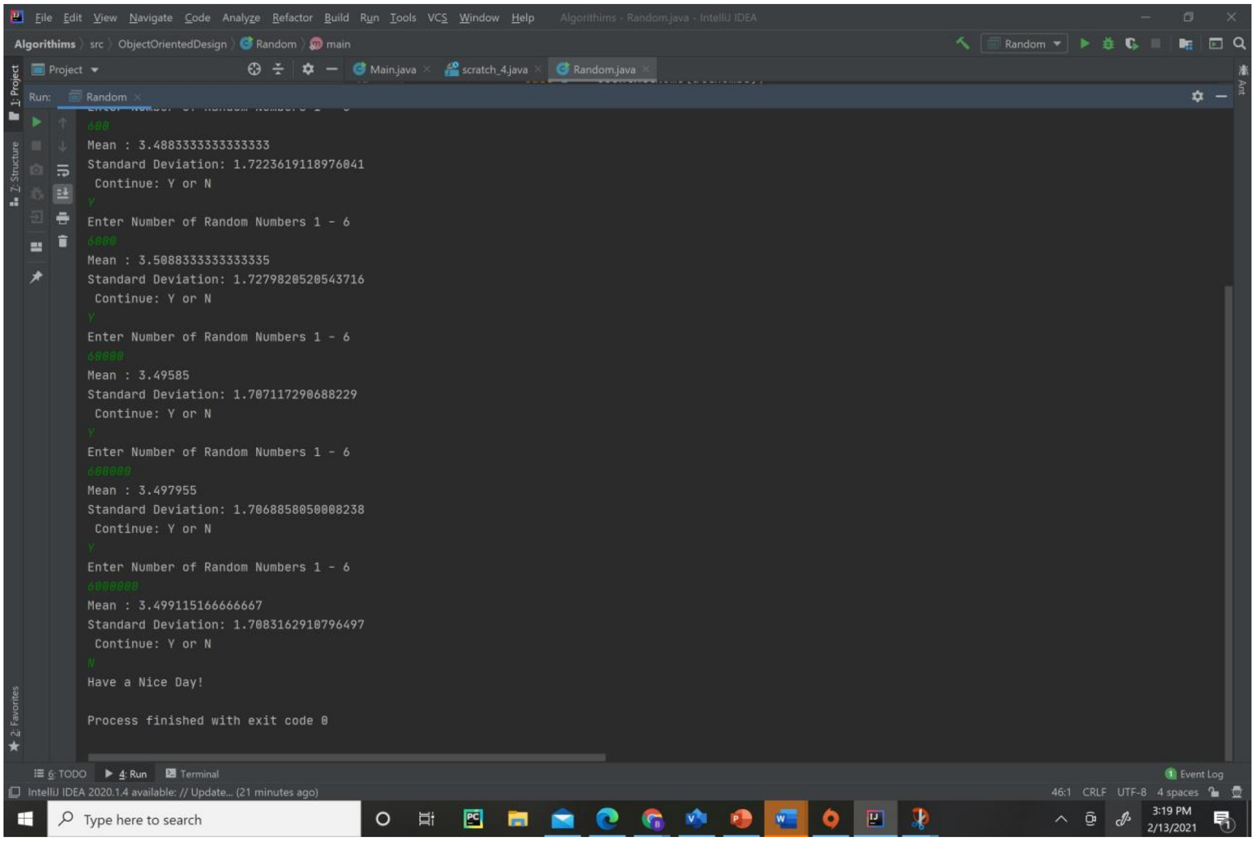
Screenshots(s) of Mean and SD Test Results



File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Algorithims - Random.java - IntelliJ IDEA
Algorithims › src › ObjectOrientedDesign › Random › main
Project
Main.java
scratch_4.java
Random.java
Run: Random
6: TODO
4: Run
Terminal
Event Log
Type here to search
3:19 PM
2/13/2021

```
"C:\Program Files\Java\jdk-14.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2020.1\lib\idea_rt.jar=55740:C:\Progra
How would you like to test the randomness of the RNG?
Type 1 for Counting Numbers up
Type 2 for Mean and Standard Deviation and Mean Calculations
Type 3 for both
2
Enter Number of Random Numbers 1 - 6
60
Mean : 3.35
Standard Deviation: 1.6515144564913748
 Continue: Y or N
Y
Enter Number of Random Numbers 1 - 6
600
Mean : 3.4883333333333333
Standard Deviation: 1.7223619118976041
 Continue: Y or N
Y
Enter Number of Random Numbers 1 - 6
6000
Mean : 3.5088333333333335
Standard Deviation: 1.7279820520543716
 Continue: Y or N
Y
Enter Number of Random Numbers 1 - 6
60000
Mean : 3.49585
Standard Deviation: 1.707117290688229
 Continue: Y or N
Y
Enter Number of Random Numbers 1 - 6
600000
Mean : 3.497955
Standard Deviation: 1.7068858050008238
```

IntelliJ IDEA 2020.1.4 available: // Update... (21 minutes ago)



File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Algorithims - Random.java - IntelliJ IDEA
Algorithims › src › ObjectOrientedDesign › Random › main
Project
Main.java
scratch_4.java
Random.java
Run: Random
6: TODO
4: Run
Terminal
Event Log
46:1 CRLF UTF-8 4 spaces
Type here to search
3:19 PM
2/13/2021

```
600
Mean : 3.4883333333333333
Standard Deviation: 1.7223619118976041
 Continue: Y or N
Y
Enter Number of Random Numbers 1 - 6
6000
Mean : 3.5088333333333335
Standard Deviation: 1.7279820520543716
 Continue: Y or N
Y
Enter Number of Random Numbers 1 - 6
60000
Mean : 3.49585
Standard Deviation: 1.707117290688229
 Continue: Y or N
Y
Enter Number of Random Numbers 1 - 6
600000
Mean : 3.497955
Standard Deviation: 1.7068858050008238
```

```
 Continue: Y or N
Y
Enter Number of Random Numbers 1 - 6
6000000
Mean : 3.499115166666667
Standard Deviation: 1.7083162910796497
 Continue: Y or N
N
Have a Nice Day!

Process finished with exit code 0
```

IntelliJ IDEA 2020.1.4 available: // Update... (21 minutes ago)

Reflective Essay:

A reflective essay on your successes, difficulties, and how you tested your code to ensure correctness

I’d say this was successful. I managed to generate an array of random numbers(allnumbs[userinput]) from one to six using a for loop and a modified Math.random call. (Add one, multiply by six, and cast to int). I tested the modified call by putting it into a scratch file and printing all a few numbers, when I found that it appeared to generate numbers from 1 to 6 in a uniform way, I worked on creating functions. The count all function and the mean and standard deviation function were created with no trouble. I believe my results are successful in proving that the Java math.random() call and psudo generator were random . For instance, In a perfectly random scenario all the outcomes are equal to each other. In this scenario there are 6 scenarios- (getting a 1,2,3,4,5,or 6). For this scenario to be random the probability of any one number being selected would be 1 for the scenario / 6 for the different outcomes or .166666667. In my simulation program, I generated these numbers and calculated the observed probability. As expected due to the law of large numbers, as the number of numbers generated grew so did the probability’s hone in on an observed probability at .166. Thus, the generator is proven to be random by those means. Similarly, if one looks at the mean generated you can see it approaches the number 3.499. In a perfectly random scenario all of the numbers generated would reach a mean of 1 + 2 + 3 + 4 + 5 + 6 = 21/6 = 3.5 – due to the fact all the numbers would bare equal weight on the mean in this formulation. The true value for standard deviation utilizing the mean of 3.5 is equal to Square Root of((1-3.5)^2+ (1-3.5)^2+(1-3.5)^2+(1-3.5)^2+(1-3.5)^2+(1-3.5)^2)/6) = 1.70785127659933. This true value also corresponds to the value I got at with my generator, so I can access that my generator can accurately mimic the truly random scenario represented by 6 discreet values. With the evidence shown here(counts, mean, and standard deviation) I can accurately say that this Java pseudo random number generator performs as specified to create random numbers.

N=60

Outcome	**Count**	**Observed probability**
1	13	0.216667
2	7	0.116667
3	8	0.133333
4	8	0.133333
5	12	.2
6	12	.2

N=600

Outcome	**Count**	**Observed probability**
1	94	94/600=0.1566667
2	91	91/600=0.1516667
3	98	98/600=0.1633333
4	100	100/600=0.166667
5	109	109/600=0.181667
6	108	108/600=0.18

N=600

Outcome	Count	Observed probability
1	1060	1060/6000=0.17667
2	992	992/6000=0.165333
3	936	936/6000=0.156
4	950	950/6000=0.158333
5	1061	1061/6000=0.17683
6	1001	1001/6000=0.16683

N=60000

Outcome	Count	Observed probability
1	9955	9955/60000= 0.165916
2	9947	9947/60000=0.165783
3	10050	10050/60000=0.1675
4	10072	10072/60000=0.16786667
5	10040	10040/60000=0.16733333
6	9936	9936/60000=0.1656

N=600000

Outcome	Count	Observed probability
1	100300	100300/600000=0.167166667
2	99817	99817/600000=0.1663616667
3	100125	100125/600000=0.166875
4	99806	99806/600000=0.16634333
5	100064	100064/600000=0.16677333
6	99888	99888/600000=0.16648

N=6000000

Outcome	Count	Observed probability
1	999720	0.16666
2	1001206	0.1668676667
3	999781	0.1666301667
4	998594	0.166432323
5	999438	0.166573
6	1001261	0.1668768333