Testing PRNG

Testing done

Visualization

The first test I did was to visualize the randomness of both implementations (Java.util.Random and MyRandom). I specified that I wanted 300x300 numbers with a bound of 2 (give 0 or 1). I then printed the output to a file and used matrixplot with a monochrome scale to visalize all of the the numbers to either white or black. If there were patterns in the visualization then that meant that the implementation would not be random. 0, 1, 0, 1, ... etc is not random as it follows a pattern.

Timing

I also measured the average time it took to generate 300x300 random integers with both of the implementations.
I simply recorded the time at start and the time at completion calculated the duration between them.
I then calculated the average execution time by adding all results from one implementation and dividing them by 10 as I did 10 tests for each implementation.

Misc

Key used for both tests: 123123123123123

Code

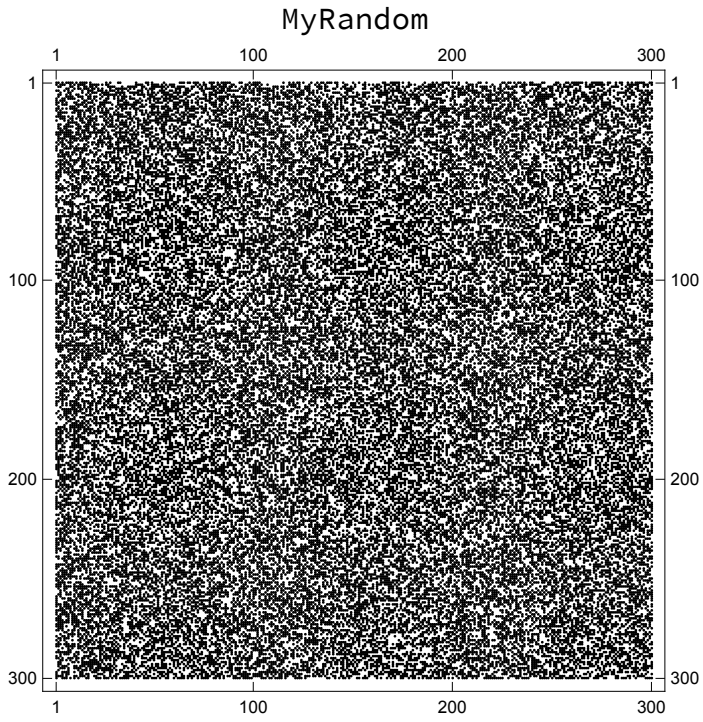
In[107]:=
```
myRandomImplementation := ReadString[
    "/Users/hampus.nilsson/Desktop/IV1013-Assignments/stream-cipher/part2/output
      "];
myRandomRes = ToExpression /@ Characters /@ StringSplit[output, "\n"];
m1 = Labeled[MatrixPlot[myRandomRes, ColorFunction → "Monochrome"],
    Style["MyRandom", 16], Top];
javaRandomImplementation := ReadString[
    "/Users/hampus.nilsson/Desktop/IV1013-Assignments/stream-cipher/part2/output
      -random"];
javaRandomRes = ToExpression /@ Characters /@ StringSplit[output, "\n"];
m2 = Labeled[MatrixPlot[javaRandomRes, ColorFunction → "Monochrome"],
    Style["JavaRandom", 16], Top];
```

Result

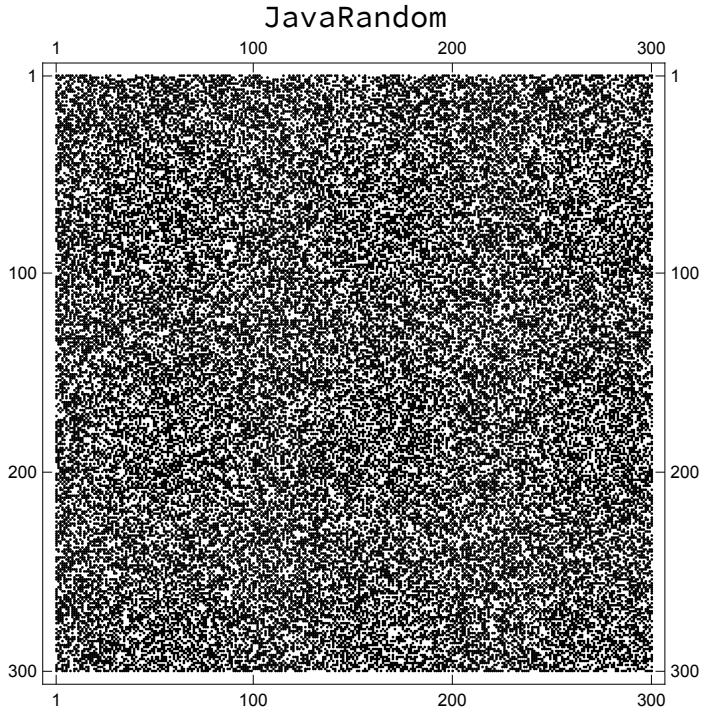
Visualization

In[113]:=
```
Show[m1]
Show[m2]
```

Out[113]=



Out[114]=



Timing

All tests generated 300x300 random numbers with a bound of 2.
Each implementation was tested 10 times each.

MyRandom

Test 1: 5439966 ns
Test 2: 4545603 ns
Test 3: 4525623 ns
Test 4: 5064743 ns
Test 5: 5365673 ns
Test 6: 4754044 ns
Test 7: 4580502 ns
Test 8: 4636537 ns
Test 9: 4676091 ns
Test 10: 5764642 ns
Average Execution Time: 4935342.4 ns

Java.util.Random

Test 1: 3022549 ns
Test 2: 2558573 ns
Test 3: 3047478 ns
Test 4: 2532810 ns
Test 5: 2625342 ns
Test 6: 2599770 ns
Test 7: 3003836 ns
Test 8: 2470043 ns
Test 9: 2636947 ns
Test 10: 2535281 ns
Average Execution Time: 2703262.9 ns

Analysis

Visualization

The visualization showed that the myRandom implementation did not follow a pattern. Java.util.Random did not follow a pattern either.
As the task implementation did not follow a pattern anymore than Java.util.Random did, it means that the myRandom implementation can be confidently used for atleast numbers up to 300x300

Timing

The execution time of the task implementation was on average 4935342.4 ns while the average execution time for Java.util.Random was 2703262.9 ns. The task iplementation took on average 82% more time to execute than using Java.util.Random.