Activity 1. [PROBLEM SIZE]

YOU ARE REQUESTED TO: Make a table reflecting the execution times of the PythonA1.py module for the exposed values of n (10000, 20000, 40000, 80000, 160000, 320000, 640000). If for any n it takes more than 60 seconds you can indicate OoT ("Out of Time"), both in this section and in subsequent sections.

n	Time(ms)
10000	2655
20000	10882
40000	44591
80000	OoT
160000	OoT
320000	OoT
640000	OoT

Activity 2. [COMPUTER PERFORMANCE]

YOU ARE REQUESTED TO: Make a table that reflects, at least for two computers to which you have access, the execution times of the PythonA1.py module for the exposed values of n (10000,20000, …, 640000). Clearly indicate which CPU and RAM memory you are using in each test.

Computer	CPU	RAM
Computer 1	AMD Ryzen 5 2600 Six-Core Processor	16GB
Computer 2	11th Gen Intel(R) Core(TM) i5-11400H @ 2.70GHz	16GB

n	Time(ms)	
	Comp.1	Comp.2
10000	2655	5231
20000	10882	15163
40000	44591	27990
80000	OoT	OoT
160000	OoT	OoT
320000	OoT	OoT
640000	OoT	OoT

Activity 3. [IMPLEMENTATION ENVIRONMENT]

YOU ARE REQUESTED TO: Program a class named JavaA1.java, which uses the same A1 algorithm to find out if a number is a primer number, as in PythonA1.py. Then, a table must be made reflecting the execution times of JavaA1.java for the same values of n (10000, 20000, ..., 640000). Finally, compare these times with those obtained in Python (in a previous section) for that same algorithm A1.

n	Time(ms)	
	Python	Java
10000	2655	419
20000	10882	1646
40000	44591	6574
80000	OoT	26274
160000	OoT	OoT
320000	OoT	OoT
640000	OoT	OoT

Activity 4. [ALGORITHM THAT IS USED]

YOU ARE REQUESTED TO: Make a table reflecting the execution times of the modules PythonA1.py, PythonA2.py and PythonA3.py, for the same values of n (10000, 20000, …, 640000). Codify those same algorithms A2 and A3 in Java, in two classes named respectively JavaA2.java and JavaA3.java. Make a table that reflects the execution times of the classes JavaA1.java, JavaA2.java and JavaA3.java, for the values of n (10000, 20000, ..., 640000) WITHOUT OPTIMIZATION of the Java program. Make a table that reflects the execution times of the classes JavaA1.java, JavaA2.java and JavaA3.java, for the values of n (10000, 20000, ..., 640000) WITH OPTIMIZATION of the Java program. Finally, draw final conclusions by comparing the times previously obtained: with Python, with Java WITHOUT OPTIMIZATION and with Java WITH OPTIMIZATION. Optionally, an A4 algorithm can be implemented in Java, which should improve the previous algorithms.

n	Time(ms)					
	PythonA1	JavaA1	PythonA2	JavaA2	PythonA3	JavaA3
10000	2655	419	301	49	146	30
20000	10882	1646	1163	180	595	110
40000	44591	6574	4405	654	2230	401
80000	OoT	26274	16737	2454	8345	1474
160000	OoT	OoT	OoT	9147	31495	5470
320000	OoT	OoT	OoT	34501	OoT	20748
640000	OoT	OoT	OoT	OoT	OoT	OoT

n	Time(ms)					
	JavaA1 (No OPT.)	JavaA1 (OPT)	JavaA2 (No OPT.)	JavaA2 (OPT)	JavaA3 (No OPT.)	JavaA3 (OPT)
10000	419	203	49	27	30	14
20000	1646	765	180	87	110	43
40000	6574	2970	654	306	401	153
80000	26274	11945	2454	1088	1474	580
160000	OoT	47614	9147	4137	5470	2183
320000	OoT	OoT	34501	15525	20748	8248
640000	OoT	OoT	OoT	58804	OoT	31101

After analyzing the obtained data, some conclusions can be drawn.

- In both cases, Java and Python, it is clear that the third approach is the best of them all, overperforming the previous two in Java and Python. The worst approach, again in both languages, is the first one. The gap is especially big between the first and the second approach, with a smaller difference between the second and the third.
- When comparing head-to-head both languages, Java is by far the fastest of the two. The gap in time decreases as the overall execution time decreases, however it is still a very important gap. It's worth noting that the difference in performance leads to more test cases executing in time.
- Finally, comparing the performance of the program with and without JIT dissipates any doubt about the performance increase. When compiling with JIT, the execution time is remarkably reduced with almost a 50% reduction in every algorithm. The reduction is specially notorious in the second and third algorithms, where the execution time is more than halved.